TOOL PATH PLOTTING USING PLOT10

by

P. SARAVANA PRASAD

B.E. (Mechanical Engineering)
College of Engineering, Guindy
Madras, India, 1983

Advisor

Dr. MUTHURAJ VAITHIANATHAN

Assistant Professor
Kansas State University
Manhattan, Kansas

A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Industrial Engineering

KANSAS STATE UNIVERSITY

Manhattan, Kansas

1985

Approved by:

[illegible]

Major Professor

TABLE OF CONTENTS

		Page
ACKNOWLEDGEMENTS		i
LIST OF TABLES		ii
LIST OF FIGURES		iii
I.	INTRODUCTION	
	A. Preamble	1
	B. Numerical Control Technology	2
	C. History of Numerical Control Technology	4
	D. Tool Path Plotting	5
	E. Report Objectives	7
	F. Project Benefits	10
	G. Project Methodology	10
II.	HARDWARE AND SOFTWARE ENVIRONMENT	
	A. The Machine Tool	11
	B. The Hardware Resources	12
	C. The Software Resources	12
III.	SYSTEM STRUCTURE	
	A. Overall Structure	14
	B. Phase I	14
	C. Phase II	27
IV.	CONCLUSIONS AND RECOMMENDATIONS	35
V.	REFERENCES	36

VI. APPENDICES

A. Pratt and Whitney Machine Codes
B. PLOT10 Routines Used
C. Hierarchy of Subroutines
D. Flow Charts
E. Program Listing for Phase I
F. Program Listing for Phase II
G. Sample Program
H. PRAWTOPPS User's Manual
I. Exec Programs

VII. ABSTRACT

ACKNOWLEDGEMENTS

The author wishes to express his sincere appreciation and gratitude to Dr. Muthuraj Vaithianathan for his valuable guidance and encouragement throughout his study in this university.

The author is thankful to Professor Jacob J. Smaltz and Dr. Prakash Krishnaswami for serving on the graduate committee.

The author takes this opportunity to thank his friends and the students of the Industrial Engineering Department who helped in testing and developing this project.

Last but not the least, the author is indebted to his parents, brothers and sister without whose encouragement and support this graduate program may not have materialized. It is to them that this work is dedicated.

LIST OF TABLES

		Page
1.	Chronological development of Numerical Control technology	6

LIST OF FIGURES

		Page
1.	Present method of operation	8
2.	Proposed method of operation	9
3.	Overall structure of PRAWTOPPS	15
4.	Overall structure of Phase I	17
5.	Overall structure of Phase II	28

I. INTRODUCTION

A. Preamble

Due to the often high cost of resources (human, material and equipment) in a manufacturing environment, one does not have the luxury of experimenting with experimental decisions on the manufacturing floor to determine the outcome of such decisions. As a result, to the extent that is possible, floor related decisions are evaluated outside of the real manufacturing environment, thereby minimizing the detrimental impact that such experimental decisions may have on the floor. Examples of such decisions include the modification of plant layout, introduction of automated material handling etc.

The need to evaluate a decision outside of the real environment is not unique to the manufacturing environment. Such a need is felt in most real world systems. This need is often met both, in manufacturing and non-manufacturing environments, by simulating the decision in order to study the impact on the system.

There are several ways in which this simulation is carried out. These include physical simulation through use of mockups and scaled models, mathematical modelling and analysis, computer modelling and analysis, graphic simulation using computers etc.

In todays manufacturing environment, there is an increasing trend in the use of computer based graphics for simulation and decision making. One of the areas in manufacturing where computer graphics is increasingly used is in tool path plotting for numerical control machine tools. The intent here is to verify the accuracy and manufacturability of programmed instructions to the machine tool. One way of verifying programmed instructions is to actually input to the machine tool the instructions and manufacture the part. This obviously is disruptive to normal operations on the floor, time consuming and prohibitively expensive depending on the raw material cost. A better method is to graphically simulate on a computer the cutting path the cutting tool would have taken had the same instructions been input to the machine tool. Software systems that permit such simulation are called tool path plotting systems.

This report details the structure and operation of a tool path plotting system for the Pratt and Whitney 2 1/2 axis vertical milling machine. Hereon in this report, this system will be referred to by the acronym PRAWTOPPS (PRatt And Whitney TOol Path Plotting System).

B. Numerical Control Technology

A numerical control (NC) system is a system in which actions are controlled by the numerical data input by the users. The system must automatically interpret at least some portion of this data. NC technology can be defined as an extremely versatile means of automatically operating machines through the use of discrete numerical values introduced to the machine by some form of stored input medium such as a punched tape or directly from a computer.[1]

There are two types of numerical control systems, namely,

1) Point-to-Point or Positioning control systems

2) Continuous path or Contouring control systems

A point-to-point control system is a system in which the user has little control in the path taken by the machine tool between the start and end points. A contouring control system is a system which permits the user to be able to direct the path taken by the machine tool between the start and end points. Since contouring control systems can also perform point to point operations, and the cost of contouring systems has been greatly reduced, practically all systems now being offered are of the contouring type.

There are two ways of controlling the machine. The primary means of controlling is via a set of computer programmed instructions called machine code program. A secondary means of controlling the machine tool is manual, via the keys on the control panel. The latter method is generally used only in the case of exigencies.

Of the two means, machine code programming is the only practical means of instructing the numerical control machine tool. The operating system within the microcomputer that controls the machine tool understands only machine code programming. Compared to assembly level language it is more sophisticated and user friendly. However, compared to languages like APT, ADAPT, AUTOSPOT that permit part modelling, it is primitive and requires more detailed information from the user.

Using machine code programming the user can instruct to the machine tool all the functions necessary to control the machine tool. The user can specify the feed rate, automatic tool changing, spindle speed, start and stop of motor, tool paths for positioning and cutting, and repeated execution of a set of instructions.

As in any computer program, a machine code program to control a machine tool consists of lines of programming statements. Each line of a machine code program contains a set of individual instructions to the machine to do certain functions.

The general structure of every machine code program instruction consists of 2 fields. The first field specifies an alphabetic code and the second field a numeric value that further qualifies the alphabetic code. There are basically two types of codes. They are:

1) Movement or Position codes - These codes comprise linear and angular motion commands for the machine.[3]

2) Machine or Program control codes - These codes are preparatory functions used to describe specific types of movement, miscellaneous functions which control machine operation, sequence information, feeds and speeds, and tooling specification.[3]

The codes vary from machine to machine. In this report, the machine under consideration is the Pratt and Whitney machine, which is currently used at Kansas State University (KSU).

C. History of Numerical Control Technology

As with many inventions in machine tool technology, numerical control came into being because there came a need for manufacturing a product by a far simpler method than those that existed. The U.S. Air Force found itself in this position shortly after World War II when it was faced with the problem of time and difficulty in machining complex aircraft components and inspection fixtures to close accuracies on a repeatable basis. A proposal to develop a machine capable of manufacturing templates to inspect wing structures from numerical input was presented to the Air Force by the Parsons Corporation of Traverse City, Michigan. This resulted in a development contract in 1948. In 1949, Parsons was joined by MIT as a major subcontractor on the project. In 1951 MIT was awarded the prime contract and this resulted in the successful demonstration of a three-axis milling machine in 1952. An organization comprised of aerospace manufacturers, recommended to the Air Force that forthcoming machines be equipped with numerical controls and in 1955 the Air Force began awarding about 35 million dollars for the manufacture of approximately 100 numerically controlled machines. To accommodate numerical control equipment, some aircraft companies had to retrofit their machine tools, because at that time retrofitting was the only practical method of increasing the output of existing machines in the shortest period of time.

Later large numerical control machines began to be produced particularly for aircraft applications. Credit is undoubtedly due to the Air Force planners for their prophetic decision.

Point-to-point type machining soon followed the introduction of continuous types. The capability of numerical control machine tools was enhanced by the addition of automatic tool changers which was developed in the middle of 1956.

It was not until around 1960 that numerical control machine tools came to be accepted and therefore began to appear on a reasonably wide commercial scale. The growth in the number of numerically controlled machines has been accelrating rapidly ever since. It might also be added that the cost of numerical control systems has decreased to almost one-third of what they were a decade ago while the reliablity and capability of the systems have increased multifold.

Table-1 shows the chronological development of numerical control technology.

D. Tool Path Plotting

Under normal circumstances, the accuracy of the machine code program is not known till the program is loaded on the machine for a test run. A lot of time (man and machine hours) and money (scrap produced if program is incorrect) is wasted in testing the accuracy of the program. One way to overcome this problem is to plot the tool path that the program would produce.

A tool path plot is a plot of the movement of the tool on the part surface depending on the machine code program. By following the path taken by the tool, it can be verified with the desired path for accuracy. This method of verification reduces long hours spent in front of the machine trying to

Table - 1 Chronological development of Numerical Control technology

Timetable for Development of the Art of Numerical Control

Time	Development	Particulars
1949 to 1952	Early research	Pioneering experimentation. Technical management alerted.
1953 to 1955	Introductory and exploratory	NCMT in use. Progress recorded in technical and trade journals. Progressive managements investigating. NC systems developed. Orders placed.
1956 to 1957	Acceptance — First stage	Deliveries more general. Technical reports of machine performance reported and studied. Standardized terminology and compatibility studied.
1958 to 1960	Acceptance — Mature stage	Introduction of the first "Production Center" machine. September 1960 Machine Tool Show dominated by NCMT. Management aware of development. First true multi-purpose NCMT. Most are MT with NC — retrofits. 325 MT builders — over 10% show NCMT. Sales of standard conventional MT off. Some of the larger machine tool companies already have 40-50% of their total sales in NCMT.
1961 to 1965	Demand — First stage	Standard conventional MT recognized as obsolete. Sales of standard tools dropping. Highly competitive period. Retrofits diminish, new NCMT better. Acceptance of standard MT instruction terminology. Programming with established subroutines widespread.
1966 to 1970	Demand — Priority stage with heavy backlogs	Management demands NCMT. Publicity and training show results. Big sales period: 40-50% NCMT, mostly true NCMT. Not more than 150 important MT builders in business by 1970. Machine Tool Show of 1965 displays over 100 NCMT.
1971 to 1975	Necessity	Manufacturing techniques rapidly obsolete. Autofacturing concept accepted. Important MT builders less than 75 by 1975. Leaders of NCMT not more than handful; many traditional leaders of 1960 out of business; some new faces. NCMT orders backlogged. Cost of NCMT reduced because they are made on NCMT via autofacturing. Procrastinating user-managements severely punished by alert competition.

debug the machine code program. Thus both time and money can be saved.

There are basically two approaches to obtaining the tool path plotting programs. One approach is to buy a commercially available dedicated hardware and software package to do the plot. This approach is very user-friendly but expensive. The other approach is to write a program to do the plot using existing graphic software package and hardware. This approach is cheaper and may also utilize existing equipment.

There are many commercially available dedicated hardware and software packages to do tool path plotting. A few of them include Geo-stac, NUMERIPACT, NICAM, Compact I, Compact II and Toolplot. These software systems are menu-driven and data is entered interactively. According to the information provided by the user, the software system generates the part diagram on the plotter.

E. Report Objectives

The main objective of this project is to develop a software system to draw a tool path for any given machine code program for the Pratt and Whitney machine. The system is to be capable of checking for any syntax or logical errors in the machine code program. Given that the machine code program input by the user is error free, the system is to be capable of generating a final machine tool path that can be verified before the machine code program is actually loaded into the numerical control machine.

The present and the proposed mode of operation are shown in Figure-1 and Figure-2 respectively.

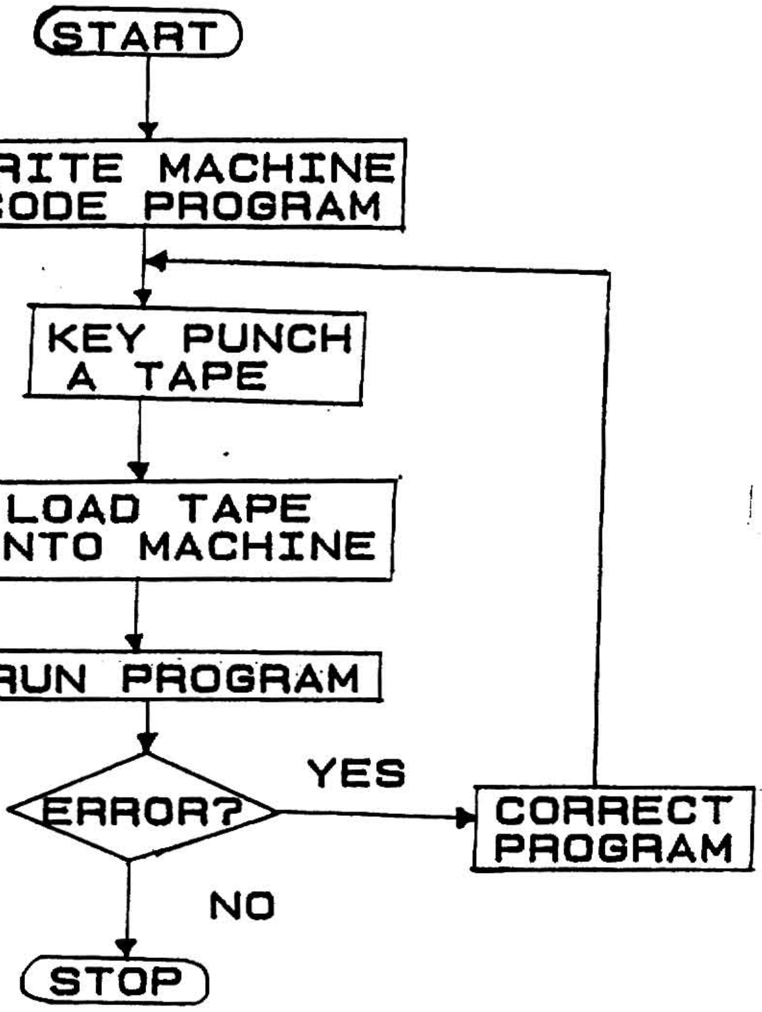


Figure - 1 Present method

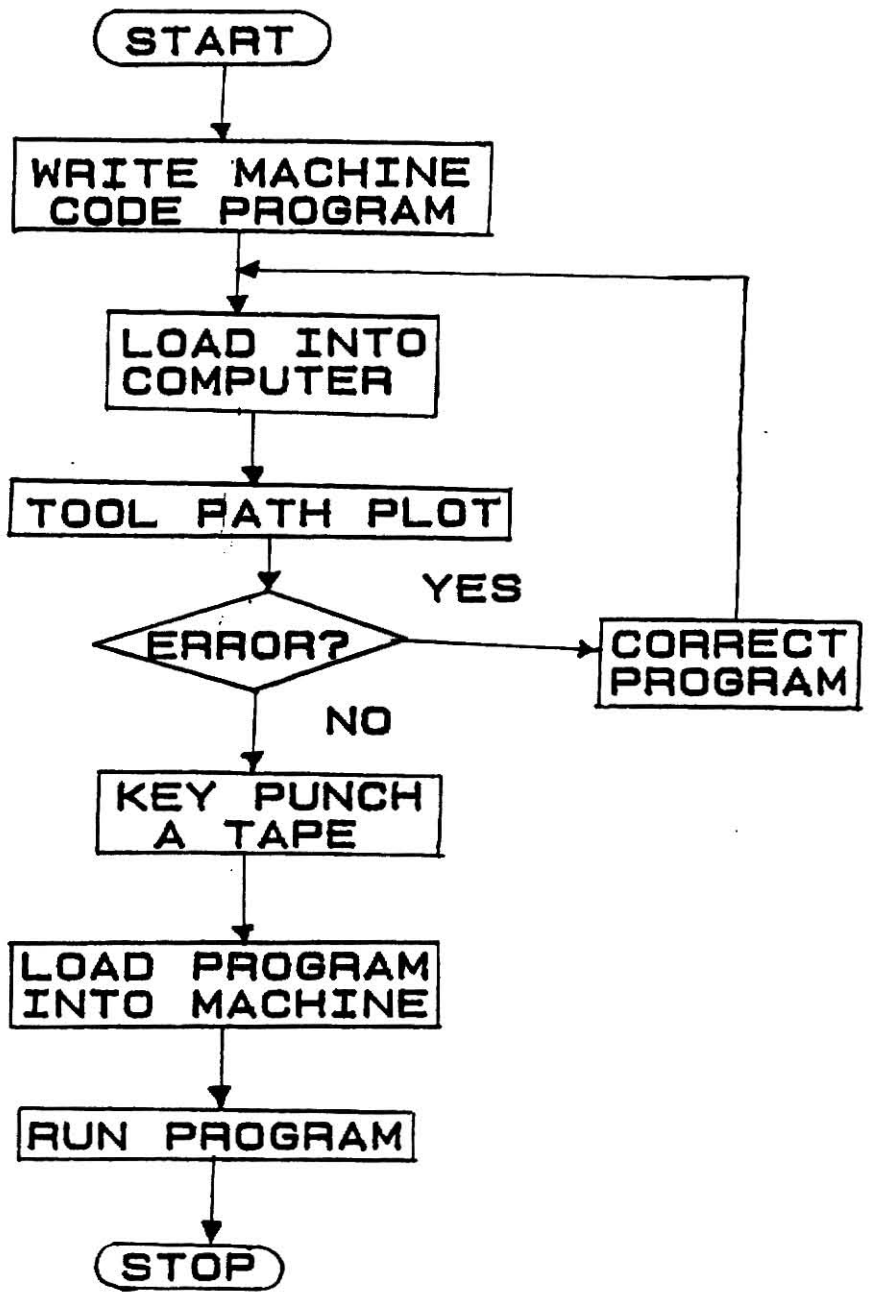


Figure - 2 Proposed method

F. Project Benefits

As stated before, the primary objective of the project is to develop a software system that can plot a tool path given a machine code program written for the part. With this system, one can verify the final machine tool path on a graphics terminal before the part program is loaded in the numerical control machine. By plotting the tool path before the actual machining, this program

1) reduces machine tool downtime,
2) improves the productivity of part programmers,
3) decreases tape proveout time,
4) reduces the risk of expensive machine tool crashes,
5) holds down the cost of part program verification,
6) and increases the overall productivity and profits.

G. Project Methodology

In order to develop this package, Plot10 (an existing basic graphics software package on the KSU mainframe) and a Selanar Hirez 100 (hardware that exists within the Industrial Engineering Department) are used. The software system developed consists of two segments or phases. The first segment reads in the machine code program and checks it for errors. If there are errors in the machine code program, appropriate error-messages are given. However if it is error free, a file containing the coordinate locations of the tool at the end of each machine code statement is outputted. The second segment takes these coordinates and plots the tool path on a graphics terminal according to the dimension of the tool.

II. HARDWARE AND SOFTWARE ENVIRONMENT

A. The Machine Tool

The Pratt and Whitney is the numerical control machine which is currently in use at Kansas State University. This is the machine used by the students at the university for whose benefit this software system has been developed.

This machine can perform machining operations such as milling, drilling, boring, and tapping economically and efficiently. The machine dimensions are 610 mm x 460 mm while the table dimensions are 500 mm x 380 mm. The positioning accuracy of the machine is +/- 0.05 mm / 300 mm. It uses a Fanuc DC servo motor (Model 0) to rotate the spindle and a Fanuc DC servo motor (Model 5) to move the table[5].

This numerical control machine can be controlled on the X, Y and Z-axis. But it is a 2 1/2 axis machine meaning that simultaneous movement is possible only in the X-Y and X-Z axis, but is not possible in the Y-Z axis. Both linear and circular movements are possible. However circular interpolation is possible only up to 90 degrees for an instruction. There are 7 positions in the tool turret and so 7 tools can be held at any given time. The machine also has the capability of storing up to 8 offset values in memory. These offsets are used to accomodate the different sizes of the various tools loaded in the turret.

Any numerical control machine has its own set of codes to distinguish the different functions to be performed. The machine codes that are used in the Pratt and Whitney are shown in Appendix A, along with a terse desciption of the codes.

B. The Hardware Resources

The computer graphics terminal that is available to the users is the Selanar Hirez 100. It is a combination alpha-numeric and high resolution graphic terminal. For alpha-numeric mode, the Selanar emulates the DEC VT102 and for the graphics mode, it emulates the Tektronix 4014. In addition to these two basic emulation modes, the device has native commands within the 4014 mode, local vector storage, local pan-zoom and a plotter/printer interface. The resolution of the machine is 1024 x 768[6].

Since this Selanar terminal is both an advanced alphanumeric as well as a high resolution graphics terminal, 2 planes of display memory are used. The first is used for ANSI and VT52 modes. It displays character data only. The second is a dot addressable graphics memory. In this memory each dot can be individually turned off and on under program control. The video from the 2 planes is OR'ed together to produce a composite alpha/graphic picture. Thus ANSI or VT52 mode data can be overlaid over graphics data[6].

C. The Software Resources

There are two graphics software packages available at KSU to plot the tool path. They are the Calcomp routines and the Plot10 Interactive Graphics Library (IGL).

Calcomp plots can be produced by programs running under Conversational Monitoring System (CMS) and using the Calcomp subroutines. The graphics produced can be previewed on Tektronix or Tektronix look-alike graphics terminals and a final copy produced on the Calcomp 1051 plotter which is also available.

The Plot10 IGL is a large set of FORTRAN subroutines providing graphics and text manipulation on Tektronix or Tektronix look-alike graphics devices.

IGL can be used from any language that can call FORTRAN routines such as FORTRAN, PL/1 and Pascal. Since IGL provides capabilities not available through the Calcomp routines, it was used to develop PRAWTOPPS. IGL provides the following advantages :

1. Interactive capabilities - Programs can receive input from a user indicating points on the screen with a mouse, joystick, graphics tablet or arrow keys.
2. Emulation for non-existent terminal features - Programs can be written for a top-of-the-line terminal with superior capabilities, such as color and local segment support. When a terminal without these capabilities is used, IGL will simulate the effect. For instance, on a monochrome terminal, colors will appear as varied patterns of crosshatching.
3. Extensive text manipulation - In addition to providing 16 special character fonts, text can be drawn in proportional mode, tilted, rotated, centered and justified. In addition, script letters can be joined to form a smooth line.
4. Three dimensional support - Pictures can be drawn in three dimensional coordinates and then projected onto a two dimensional screen.
5. Graphics segment - A portion of a picture can be put into a segment and thereafter be transformed, displayed, and manipulated individually. Segments provide a "building block" capability to develop a library of common picture components.
6. Line smoothing - Lines can be automatically smoothed using a combination quadratic and cubic spline technique.
7. The size of the picture can be varied by window/viewport transform.

The IGL has several FORTRAN subroutines, out of which a few were used in this software system. These subroutines are described in Appendix B.

III. SYSTEM STRUCTURE

A. Overall Structure

PRAWTOPPS basically has two phases or segments. One phase of the software reads the machine code program provided by the user as input, processes the data and checks for logical or syntax errors and outputs the coordinates of the tool center. The second phase of the software reads these coordinates from the first phase as input and produces the plot on the terminal.

This division of software was done in order to save computer time and money. The plotting procedure is both expensive and time consuming. Hence producing a plot of an incorrect program is wasteful. Therefore, it is suggested that phase I be run first and the machine code program checked for errors. The output produced from phase I can be checked at random points with the desired path for correctness. After completing phase I successfully, phase II is undertaken and the resulting tool path obtained.

Phase I was written in Pascal in order to utilize the character manipulation capabilities of the language. Furthermore, Pascal provides the use of records, which help in providing more structured data sets.

Phase II was written in FORTRAN 77 since it is easier to invoke the IGL routines from FORTRAN than from any other language.

Figure-3 shows the overall structure of PRAWTOPPS.

B. Phase I

The first phase of PRAWTOPPS reads the machine code program provided by the user as input, processes the data and checks for logical and syntax errors and outputs the coordinates of the tool center. This phase was written in Pascal.

The input for this phase is the machine code program written by the user

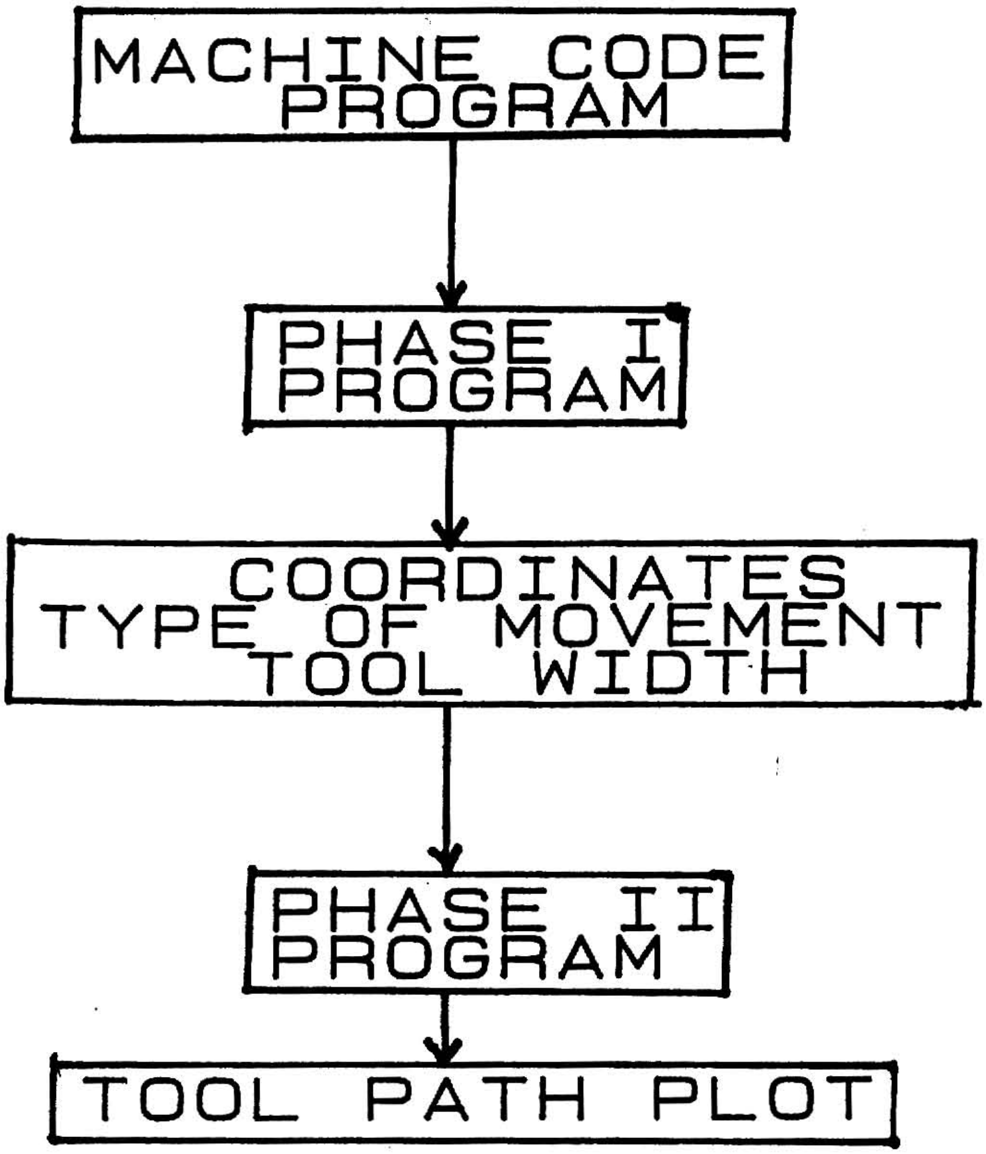


Figure – 3 Overall structure of PRAWTOPPS and stored in a file called NC DATA. The output from this phase is the tool center coordinates and this is stored in a file called POSITION FILE.

The overall structure of Phase I is shown in Figure-4.

This phase of the software system consists of a main program and a number of procedures to perform different functions. Each of these are discussed below individually. The control hierarchy of procedures for this phase of PRAWTOPPS, along with a terse description of the functions, is shown in Appendix C. The flow charts for the main program and the procedures are shown in Appendix D.

Main Program:

The main program consists of calls to three different procedures, namely Read_data, Inititalize, and Process_data. Each of these procedures performs the three basic functions of this phase - reading the data, initialize variables used in the program, and process the input data and thus produce the output.

Read data:

The Read_data procedure is invoked by the main program. It first opens the input and output files for data manipulation. It then reads the machine code program provided by the user. The machine code program should be under the filename °NC' and filetype °DATA'. The first column of each of the data lines should be a blank. Furthermore, a M02 code should be present at the end of the machine code main program.

As the machine code program is read, an index file is created by invoking the procedure Create_index. The statement number, the data line number, and the number of sets of machine code instructions in that line are stored in the index file.

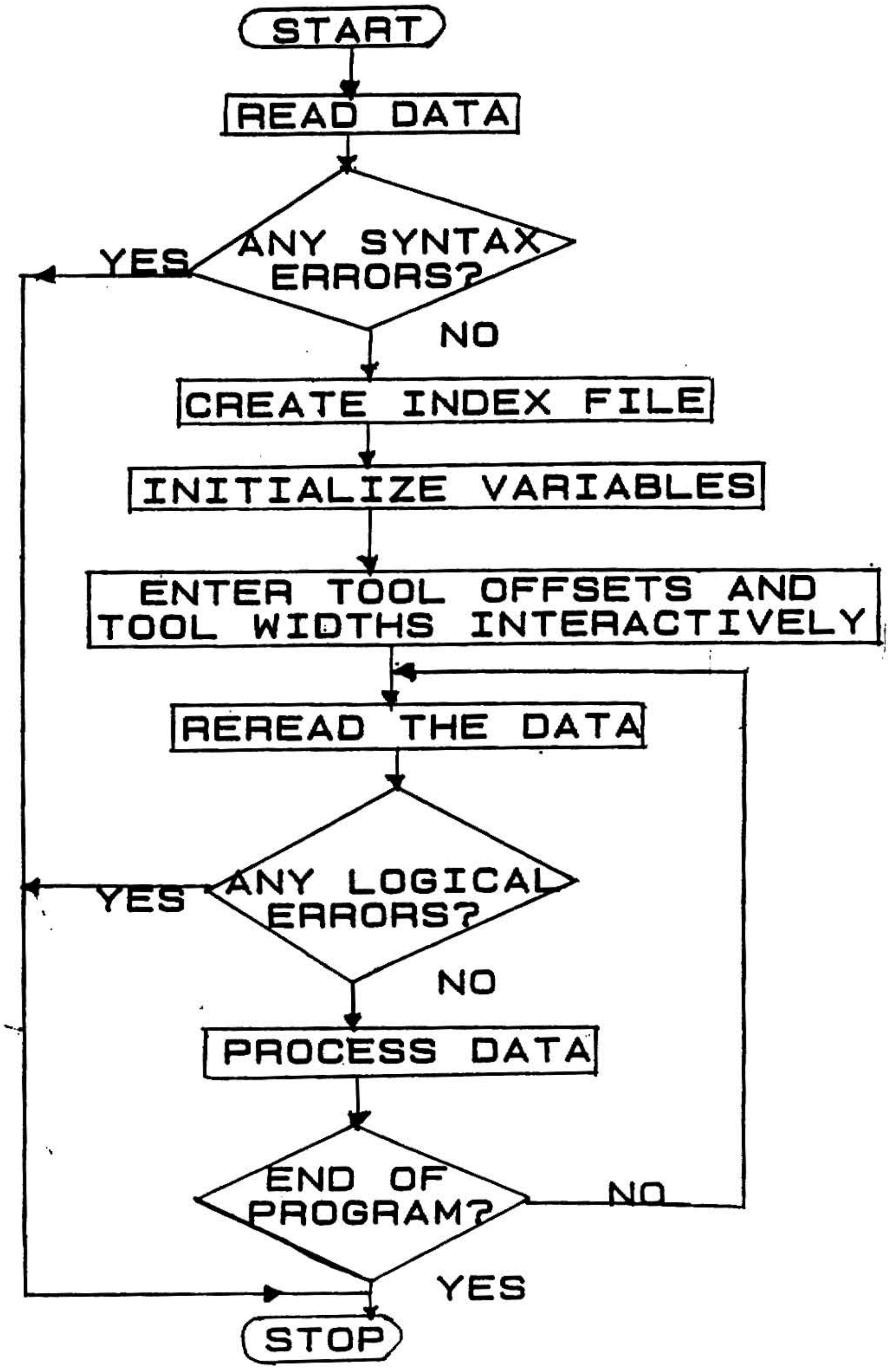


Figure - 4 Overall structure of Phase I

Create Index:

The Create_index procedure is invoked by the procedure Read_data. As the name indicates, this procedure creates an index file. Any index file consists of two parts - the key part and the information part. The key part contains a number on which the record is indexed and the information part has data associated with the key. Each line of instruction in the machine code program has a statement number. The index file is indexed on this statement number. The data that is stored in the information part of the index file is the position of the statement in the input data file and the number of sets of machine code instructions in that data line. This information is stored in a file called Index File. An example of the index file is shown in Appendix I.

Initialize:

The Initialize procedure is invoked by the main program. This procedure repositions the data pointer of the input file to the top. It also initializes some of the variables to their default values. The user is also asked to input the tool offsets and the toolwidths interactively when this procedure is executed. These values input by the user are stored in a file called OFFTOOL FILE. In order to prompt the user to enter the data, the following message is typed on the screen "Type Y/N to enter intial toolwidths and tool offsets". If the user is running the program for the first time or desires to change the existing toolwidths and tool offset values, then a Y should be typed. This allows the user to enter the initial toolwidth and tool offset values. However, if the user is rerunning the program and desires to retain the existing toolwidths and tool offsets, then a N should be typed. This option is provided to avoid retyping the initial toolwidths and the tool offsets each time the program is rerun.

This procedure also titles the final output obtained from this phase of the program.

Process data:

The Process_data procedure is invoked by the main program to process the machine code program. When the program reaches this stage, the input data is free of syntax errors. This procedures checks for logical errors and produces a tool center coordinate file.

The Process_data procedure invokes three other procedures (Reinitialize, Reread_check and Deviate) to perform its functions. The input data (the machine code program input by the user) is read once again from the beginning. Before each line of instruction is read, several flags are set. Then the line that was read is analyzed in order to interpret the instructions it contains. If the instruction is a call to a subroutine, then control is transferred to procedure Deviate. This procedure continually analyzes each data line till a M02 code or a logical error in the machine code program is encountered.

Reinitialize:

The Reinitialize procedure is invoked by procedures Process_data and Deviate. This procedure initializes variables that were not initialized by the Initialize procedure. The variables initialized by this procedure have to be reinitialized after analyzing each machine code statement.

Reread check:

The Reread_check procedure is invoked by procedures Process_data and Deviate. It reads the machine codes and invokes the procedure Check_codes to check the alphabetic part of the machine code. After checking the machine code, Reread_check invokes different procedures according to the function that machine code statement is intended to perform.

Deviate:

The Deviate procedure is invoked by Process_data to handle the subroutines of the machine code program. With the help of the Scroll procedure and the Index File, it locates the starting position of the subroutine and processes it with the help of the Reinitialize and Reread_check procedures.

Scroll:

The Scroll procedure which is invoked by Deviate, merely skips through a certain number of lines in the input data file before beginning to read from the file.

Lexical read:

The Lexical_read procedure, which is invoked by Reread_check, reads characters and converts them to integers if they are numeric characters. It invokes the function Numeric to do the verification.

Numeric:

Numeric is a boolean function invoked by Lexical_read. It returns a value TRUE if its argument is a numeric character, else it returns a value FALSE.

Check codes:

Check_codes is a procedure invoked by Reread_check. It checks the alphabetic part of the machine code and invokes different procedures depending on that character. If the letter is not legal, an error message is given and execution stops. The legal letters are F, G, M, L, R, D, H, X, Y, Z, I, J, and K.

Checkf:

The Checkf procedure is invoked by Check_codes and it checks the numbers associated with the F code. The numeric part of the F code is assigned as the feed for the normal or the fixed cycle operation, depending on whether the machine is in normal or fixed cycle mode.

Checkg:

The Checkg procedure is invoked by Check_codes and it checks the numbers associated with the G code. The valid numbers associated with the G code are 00 to 03, 28, 45 to 48, and 80 to 92. Depending on the number, variables are assigned corresponding values. If any other number is associated with the G code, an error message is outputted and execution is terminated.

Checkm:

The Checkm procedure is invoked by Check_codes and it checks the numbers associated with the M code. The valid numbers associated with the M code are 0, 2, 6, 30 to 33, 98, and 99. Depending on the number, variables are assigned corresponding values. If any other number is associated with the M code, an error message is outputted and execution is terminated.

Checkp:

The Checkp procedure is invoked by Check_codes and it checks the numbers associated with the P code. If the machine is in fixed cycle mode then the numeric part of the P code is assigned to be the dwell time. If the P code succeeds a subroutine call, then the numeric part is the starting statement number for the subroutine. If both these conditions are not satisfied, an error message is outputted and execution is terminated.

Checkl:

The Checkl procedure is invoked by Check_codes and it checks the number associated with the L code. The numeric part of the L code gives the number of times a particular machine code statement or a subroutine is to be repeated. Thus depending on this number, a set of instructions is executed repeatedly. The default value is 1.

Checkr:

The Checkr procedure is invoked by Check_codes and it checks the number associated with the R code. If the machine is in fixed cycle mode, then the numeric part of the R code gives the distance for rapid movement. However, if the machine is in normal cycle mode an error message is outputted and execution is terminated.

Checkd:

The Checkd procedure is invoked by Check_codes and it checks the number associated with the D code. The numeric part of the D code gives the offset code to be used for linear and circular offset movements on the X-Y plane. Depending on the code, the offset value is set. This offset is valid for movement in the X and Y directions, but not in the Z direction.

Checkh:

The Checkh procedure is invoked by Check_codes and it checks the number associated with the H code. The numeric part of the H code gives the offset code to be used for vertical offset movements. Depending on the code, the offset value is set. This offset is valid for movement in the Z direction, but not in the X and Y directions.

Checkxyz:

The Checkxyz procedure is invoked by Check_codes and it checks the number associated with the X, Y, and Z codes. This procedure gives the position of the tool center in the X, Y, and Z direction at the end of each machine code statement. The numeric part of X, Y, and Z codes gives the shift in the X, Y and Z directions respectively. If the machine is in reverse X or/and reverse Y mode, then the direction of the shift is changed, that is, the shift is multiplied by -1.

If the machine is in incremental mode, then the position of the tool center at the end of the statement is the sum of the present coordinates and the shift in the corresponding coordinate direction. However, if it is in absolute mode, the position of the tool center at the end of the statement is the sum of the shift in that coordinate and the shift in the position of the origin, if any. Usually, the initial origin position is the left hand corner of the workpiece on the X-Y plane and on the part surface on the Z plane.

During fixed cycle operations, at the end of the statement, there is no change in the position of the tool center in the Z direction. Thus, during the above condition, the position of the tool center in the Z direction remains unchanged.

Checkijk:

The Checkijk procedure is invoked by Check_codes and it checks the numbers associated with I, J, and K codes. These codes are used when the tool is moving in a circular path. The numeric part of the I, J and K codes gives the distance of the center of the circle from the current tool position (before the tool moves in the circular path) in the X, Y, and Z direction respectively. When these codes are used without the tool moving in a circular path, an error message is outputted and execution is terminated.

Change tool:

The Change_tool procedure is invoked by Reread_check whenever a M06 code is encountered. This procedure simulates the indexing of the turret. During this operation, the spindle is stopped and the tool is changed by indexing the turret once. The toolwidth is reset to the width of the current tool. The coordinates of the tool center at the end of this statement are written to the Position File by the Print procedure.

Change origin:

The Change_origin procedure is invoked by Reread_check whenever a G92 code is encountered. This procedure simulates the function of resetting the origin in the coordinate(s) specified. The X, Y, or Z values that are specified with the G92 code imply that the new origin is a certain distance (shown by the X, Y, or Z values) from the current tool position.

It is assumed that when the first G92 is encountered the machine is in the machine home position and the new origin is the left hand corner of the job for the X and Y axis and the part surface for the Z axis. This assumption was made to find the coordinates of the machine home position with respect to the origin. The coordinates of the tool center at the end of this statement are written to the Position File by the Print procedure.

Go_home:

The Go_home procedure is invoked by Reread_check whenever a G28 code is encountered. This procedure simulates the function of moving the cutting tool to the machine home position in the direction (X, Y or Z) specified. The coordinates of the tool center at the end of this statement are written to the Position File by the Print procedure.

Mid_stop:

The Mid_stop procedure is invoked by Reread_check whenever a M00 code is encountered. This procedure simulates the function of temporarily stopping further execution of the machine code program. This break is sometimes necessary to change tools or offset values and to change interpretation of the X and Y coordinates.

PRAWTOPPS requires the user to interactively input any changes that is desired at the time the machine is stopped. The coordinates of the tool center at the end of this statement are written to the Position File by the Print procedure.

Fixed_cycle:

The Fixed_cycle procedure is invoked by Reread_check whenever a fixed cycle (G81 to G89) code is encountered. This procedure simulates the condition when the machine is in fixed cycle mode. A subsequent call to G80, G01 or G00 will put the machine back into normal mode.

Operations like drilling, boring etc. are done in fixed cycle mode. During these operations, the tool moves vertically. But the position of the tool in the Z direction is unchanged at the end of the statement, since the tool comes up to the original position after performing the cutting operation.

When a dwell is encountered, the software system gives a prompt and waits for the user to respond. The coordinates of the tool center at the end of this statement are written to the Position File by the Print procedure.

Circular motion:

The Circular_motion procedure is invoked by Reread_check whenever a G02 or a G03 code is encountered. This procedure simulates the movement of the tool in a circular path. Before proceeding to move the tool, this procedure checks for the following error conditions:

1. A H offset code is used while moving in the X and/or Y direction.
2. A D offset code is used while moving in the Z direction.
3. The cutting tool is cutting the part without the feed rate and/or spindle speed values.
4. I, J or K not specified with the G02 or G03 code.
5. Circular movement of more than 90 degrees for an instruction.

Whenever an error is encountered, an error message is outputted and execution is terminated.

If an offset is used, the coordinates of the tool center and the coordinates of the circular path center are changed by the offset value. The coordinates of the tool center at the end of this statement (X, Y, and Z) along with the coordinates of the center of the circular path (I, J, and K) from the starting point of the circular path are written to the Position File by the Print procedure.

Go ahead:

The Go_ahead procedure is invoked by Reread_check whenever any of the the above 6 procedures (Change_tool, Change_origin, Go_home, Mid_stop, Fixed_cycle and Circular_motion) are not invoked. This procedure simulates the movement of the tool in a linear path. Before proceeding to move the tool, this procedure checks for the following error conditions:

1. A H offset code is used while moving in the X and/or Y direction.
2. A D offset code is used while moving in the Z direction.
3. The cutting tool is cutting the part without the feed rate and/or spindle speed values.
4. The tool is moving simultaneously in both the Y and Z directions.

Whenever an error is encountered, an error message is outputted and execution is terminated.

If an offset is used, the coordinates of the tool center are changed by the offset value. The coordinates of the tool center at the end of this statement are written to the Position File by the Print procedure.

Print:

The Print procedure is invoked by the procedures that write the position of the tool center to the Position file. This procedure writes the statement number of the machine code statement, width of the current cutting tool, the movement type code or G code (0 for positioning move, 1 for linear machining move, 2 for circular machining move in the clockwise direction, and 3 for circular machining move in the anti-clockwise direction), X, Y, and Z coordinates of the tool center at the end of the machine code statement and the I, J, and K values. For any other type of movement other than circular, the I, J, and K values are 0.

The instructions for using Phase I of PRAWTOPPS is detailed in Appendix H. The actual listing of the Phase I program is shown in Appendix E. Since this phase of the software system does not perform any graphic function, the program can be run on any (graphic or non-graphic) terminal available at KSU. Since the output of this phase (Position File) is used as the input of Phase II, it is recommended that the users check the coordinates calculated and written to the Position File with the desired coordinates before proceeding further. Appendix I contains a sample program to machine the letter "P". The input (machine code program) and the output (coordinate file) are shown, along with the Index File.

C. Phase II

The second phase of PRAWTOPPS reads the coordinate file from Phase I, processes the data and plots the path taken by the tool center on the terminal. This phase is written in FORTRAN 77. The PLOT10 IGL subroutines called from this phase of the software system draws the tool path plot, which is the ultimate goal.

The input for this phase is in a file called Position File.

The overall structure of Phase II is shown in Figure-5.

Phase II consists of a main program and a number of subroutines to do different functions. Each of these are discussed below individually. The control hierarchy of subroutines for this phase of PRAWTOPPS, along with a terse description of its function, is shown in Appendix C. The flow charts for the main program and the subroutines are shown in Appendix D.

Main Program:

The main program starts by prompting the user to enter the length and width of the work piece. Then default values are assigned to the options provided by the program. The IGL routines are initialized by calling the IGL function GRSTRT. The arguments of this function specify the graphics terminal that is being used.

Following the above IGL initialization, a call is made to subroutine Clear to draw the work piece. Then a prompt is given for the user to enter the command to perform any function. This software system can execute the

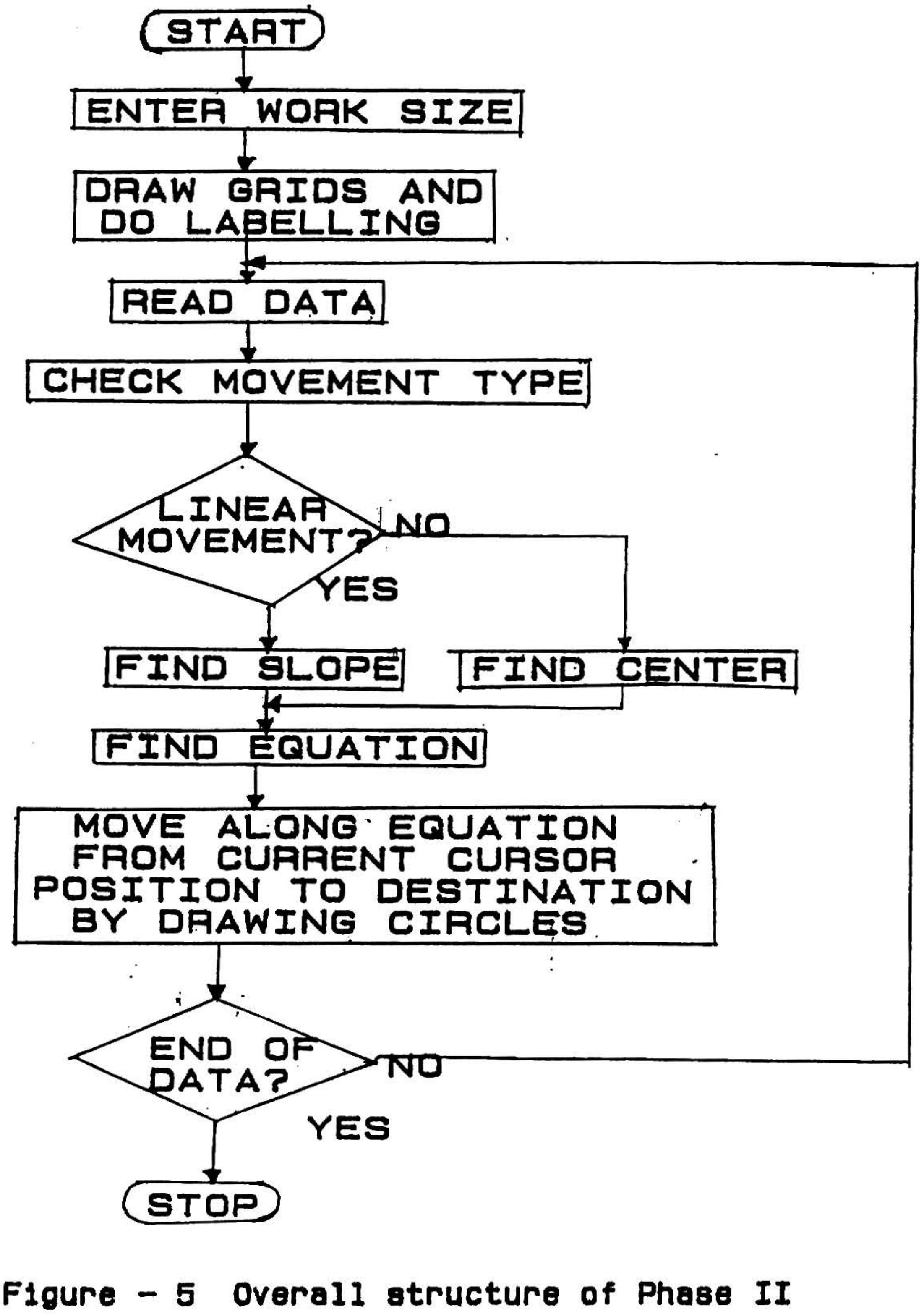


Figure - 5 Overall structure of Phase II following commands:

CLEAR	-	Clears the screen and draws the workpiece.
OPTIONS	-	Provides options for the user. The user can draw grids and fixtures by giving this command. The user can specify the data line at which the plotting should start and stop. The user can also choose to clear the screen each time the toolwidth changes.
DRAW	-	Draws the tool path according to the options specified by the user.
ZOOM	-	Enlarges a desired portion of the plot for better viewing.
QUIT	-	Quits the software system and returns back to CMS.

Different subroutines are called according to the command specified by the user. If a command other than the five specified above is given, an error message is outputted.

Finally a call is made to subroutine Sbansi to return the graphics terminal back to ANSI mode and thus end the graphics session.

Clear:

The Clear subroutine is called by the main program and the other subroutines to clear the screen and draw the work piece. It also writes the title for the display of the toolwidth and the Z-coordinate. The CLEAR command calls this subroutine to execute the command.

Sbansi:

The Sbansi subroutine is called by subroutine Option and the main program. This subroutine returns the Selanar Hirez back to the ANSI mode from the 4010 graphics mode.

Grid:

The Grid subroutine is called by subroutines Option, Draw and Zoom. This subroutine draws the grid and labels it. A grid line is drawn for every 2 Graphic Display Units (GDU). Every fifth grid line is a solid line, while the others are dotted lines. The Option subroutine calls Grid if the user desires to draw the fixtures. The Draw and the Zoom subroutines call Grid if the user desires the grid option.

Fixtur:

The Fixtur subroutine is called by the Draw subroutine if the user has specified the fixture option. This subroutine draws the fixture using hatched line at the locations already specified by the user in the Option command.

Sbdraw:

The Sbdraw subroutine is called by the main program whenever the user gives the DRAW command. This subroutine draws the tool plot according to the coordinates specified by Position File.

This subroutine starts by a call to subroutine Clear. This is followed by calls to subroutines Fixtur and Grid if the user has specifed the fixture and Grid options respectively.

The data file is reset to the top. The first two data lines are neglected since they contain the titles for the data file. Then the rest of the data are read and the values are converted to GDU in order to plot on the screen. A plot is made for the complete data file. However, the user is provided with the option of plotting only certain data lines by using the OPTIONS command.

Depending on whether the tool path is linear or circular, calls are made to subroutines Stline and Curve respectively. The subroutine Txtwrt is called to check for any changes in toolwidth or Z-coordinate values.

Option:

The Option subroutine is called by the main program whenever the user gives the OPTIONS command. First the terminal is brought back to ANSI mode in order to ease interactive input and output. The options specified by the user remain in effect till the current graphics session. The following options are provided to the user:

1. Specify the data line (excluding the titles) from which the plotting should commence. The default value is 1.
2. Specify the data line at which the plotting should stop. The default is till the end of the data file (Position File).
3. Specify the option of clearing the screen during toolwidth change. The default is to continue on the same screen.
4. Specify whether grids are required during the DRAW or ZOOM command. The default is not to draw the grids.
5. Specify whether fixtures are required during the DRAW command. The default is not to draw the fixtures. The user can draw upto 10 fixtures by moving the cross-hair cursor and specifying the four coordinates for each fixture.

Zoom:

The Zoom subroutine is called by the main program whenever the user gives the ZOOM command. The user marks one of the diagonal elements of the segment which is to be enlarged. This segment is converted to a square area with the length of the bigger side of the segment being the side of the square. This is done to overcome the distortion due to possible unequal scaling in the X and Y direction. The enlarged view is drawn on 80 x 80 GDU and does not depend on the size of the workpiece. This is also done to overcome distortion. The grids are drawn by a call to subroutine Grid, if the option is desired.

The data file is reset to the top. The first two data lines are neglected since they contain the titles for the data file. Then the rest of the data are read and the values are converted to GDU in order to plot on the screen. A plot is made for the complete data file. However, the user is provided with the option of plotting only certain data lines by using the OPTIONS command. Even though all the lines are read, only the desired segment is enlarged and displayed on the screen.

Depending on the whether the tool path is linear or circular, calls are made to subroutines Stline and Curve respectively.

Txtwrt:

The Txtwrt subroutine is called by the main program and it displays the toolwidth or the Z coordinate whenever a change in value in either one of them is encountered. A comparison is made between the toolwidth or Z coordinate of the current data line with that of the preceeding data line before the actual display.

If the user desires to plot on a clean screen after every tool change, a call to subroutine Clear is made. Then the grids and fixtures are redrawn if specified by the user. Otherwise, the plot is continued on the same screen and the new toolwidth is displayed below the previous value.

In the event of any changes in the Z-coordinate values, the new value is displayed below the previous value. Also the coordinate at which the change occurs is marked by the number of times the Z-coordinate has changed.

However if the end of the screen is encountered, all the previous values are removed the display again starts from the top of the screen.

Stline:

The Stline subroutine is called by subroutines Sbdraw and Zoom. It plots the tool path for linear movement of the tool center. Using the preceeding and the current X and Y coordinates, the equation for the linear path is determined. The tool path is drawn from the current cursor position to the final position by computing the coordinates based on the equation of the line. Depending on whether the G code (movement type code) value is 0 or 1, dotted circles or solid circles are drawn along the path. When it is a positioning move the G code value is 0 and when it is a linear machining move the G code value is 1.

A call is made to subroutine Tool to draw the circles along the path taken by the tool. This call is made till the tool reaches its destination.

Curve:

The Curve subroutine is called by subroutines Sbdraw and Zoom whenever the tool is moving in a circular path. From the coordinates of the center of the circle and the X and Y coordinates, the equation of the circular path is obtained. If the G code is 2, the tool moves in a clockwise direction and if the G code is 3, the tool moves in an anti-clockwise direction. Circles are drawn from the current cursor position to the final position along the circular path with the help of the equation. The circles are drawn by continuous calls to the subroutine Tool, until the tool reaches its destination. Since both the types of moves are machining moves, solid circles are drawn.

Tool:

The Tool subroutine merely draws a circle at the X and Y coordinates specified by the arguments of the subroutine. It draws a circle according to the width to the tool. Solid or dotted circles are drawn depending on the G code. If the G code is 0 then dotted circles are drawn, otherwise solid circles are drawn. Both the Stline and the Curve subroutines call this subroutine to draw the tool path.

Appendix H details the instructions for the user on using Phase II of PRAWTOPPS. The listing of the program used in this phase is shown in Appendix F. Since this phase of the software system does graphics, the program must be run on a graphics terminal. Since the plotting procedure involves a lot of computer time and money, it is recommended that the users check the coordinates calculated and recorded in the Position File with the desired coordinates before doing the plot. However when the plot is obtained on the screen, the user can verify it with the desired path and if it is correct, the program can be entered into the numerical control machine.

The tool path obtained from the machine code program to machine the letter "P" is shown in Appendix G.

IV. CONCLUSIONS AND RECOMMENDATIONS

PRAWTOPPS is a software system which has been developed to draw the tool path for any machine code program for the Pratt and Whitney machine at KSU. This plot can be viewed on a graphics terminal. The instructions for the user on using the software system is given in Appendix H. In order to make the PRAWTOPPS user-friendly, two exec programs were written - one each for Phase I and Phase II. These exec programs are shown in Appendix I.

This software system reduces some of the disadvantages of loading the machine code program directly onto the machine without verifying it on the computer. The time for diagnosing the errors is considerably reduced and this results in a saving of both man and machine hours. Further there is a significant reduction in scrap.

The system is not without its limitations.

This system can be used only for machine code programs written for the Pratt and Whitney machine at KSU. To use this system for codes written for other machines, modifications have to be made in Phase I of the software system.

Furthermore, in using the system the user is restricted to a single view of the tool plot. Only the X-Y coordinates are plotted on the screen. The Z coordinates are merely written on the side along with the toolwidth. This system can be further developed if other views such as an X-Z view is desired.

Another major drawback is that a hardcopy of the plot cannot be obtained on a plotter. To overcome this drawback, a study can be undertaken to interface the HP 7475A Plotter (currently available with the Industrial Engineering Department at KSU) and the Selanar Hirez 100 to obtain the plot.

V. REFERENCES

1) Principles of Numerical Control (Third Edition)
- James J. Childs

2) Introduction to Numerical Control in Manufacturing
- Raymond Howe

3) Numerical Control and Computer Aided Manufacturing
- Roger S. Pressman
- John J. Williams

4) Numerical Control (Laboratory Manual)
- Muthuraj Vaithianathan

5) Fanuc Tape Drill - Model C (Operator's Manual)
- Fujitsu Fanuc Ltd.

6) Hirez 100 Operator's Manual
- Selanar Corporation

7) PLOT10 Interactive Graphics Library (User's Manual)
- Tektronix

APPENDIX A

Pratt and Whitney Machine codes

Code	Modal	Function
G Codes		
G00	Yes	Point to point, positioning
G01	Yes	Linear interpolation
G02	No	Circular interpolation arc (CW)
G03	No	Circular interpolation arc (CCW)
G28	No	Goes to the machine home position
G45	No	Single positive offset
G46	No	Single negative offset
G47	No	Double positive offset
G48	No	Double negative offset
G80	Yes	Fixed cycle operation cancel
G81-G89	Yes	Fixed cycle operation start
G90	Yes	Movement in absolute mode
G91	Yes	Movement in incremental mode
G92	No	Redefines the origin
M Codes:		
M00	No	Stop in the middle of the program
M02	No	End of the program
M06	No	Change tool (indexing)
M30	Yes	Stop the spindle
M31	Yes	Rotate spindle at low speed
M32	Yes	Rotate spindle at medium speed
M33	Yes	Rotate spindle at high speed
M98	No	Call to a subroutine
M99	No	Return from a subroutine
F Codes:		
Fxxxx	Yes	Feed code for normal operation (or)
	Yes	Feed code for fixed cycle operation
P Codes:		
Pxxxx	Yes	Dwell time in seconds in fixed cycle operations (or)
	No	Statement where subroutine starts
L Codes:		
Lxxxx	No	Number of repetitions of statement(s)

Code	Modal	Function
R Codes:		
Rxxxx	Yes	Rapid movement in Z-axis in fixed cycle operations
D Codes:		
Dxxxx	Yes	Linear offset codes
H Codes:		
Hxxxx	Yes	Height offset codes
X, Y and Z Codes:		
Xxxxx	No	Distance to be moved or co_ordinate to be moved to in the X-direction
Yxxxx	No	Distance to be moved or co_ordinate to be moved to in the Y-direction
Zxxxx	No	Distance to be moved or co_ordinate to be moved to in the Z-direction
I, J and K Codes:		
Ixxxx	No	Distance of circle center from the current position in X-direction
Jxxxx	No	Distance of circle center from the current position in Y-direction
Kxxxx	No	Distance of circle center from the current position in Z-direction

APPENDIX B

PLOT10 Routines used[7]

CMCLOS

Category	:	System Environmental Routines
Purpose	:	Temporarily closes IGL communication with the terminal
Syntax	:	CMCLOS

CMOPEN

Category	:	System Environmental Routines
Purpose	:	Reestablishes IGL communication with the terminal
Syntax	:	CMOPEN

GRSTOP

Category	:	System Environmental Routines
Purpose	:	Terminates IGL.
Syntax	:	GRSTOP

GRSTRT

Category	:	System Environmental Routines
Purpose	:	Initializes IGL; directs output to a specified device.
Syntax	:	GRSTRT(idevic,iopt)
Paramters	:	
Idevic	:	Device on which output is to be displayed; usually the four-digit Tektronix product number.
Iopt	:	The device option code; further defines device by indicating its options.

DASHPT

Category	:	Graphic Environmental Routines
Purpose	:	Specifies pattern for dashed linesay of output.
Syntax	:	DASHPT(ipat)
Paramters	:	
Ipat	:	An integer indicating the desired dashed-line pattern.

FILPAN

Category	:	Graphic Environmental Routines
Purpose	:	Specifies way in which panels are filled.
Syntax	:	FILPAN(ipatno,qoutln)
Paramters	:	
Ipatno	:	Number of pattern used to fill panel (0-24).
Qoutln	:	Logical flag for outlining panel. .TRUE. - Outlines the panel in current vector color .FALSE.- Does not outline panel

SCALE

Category	:	Graphic Environmental Routines
Purpose	:	Specifies a scale factor applied to the coordinate system.
Syntax	:	SCALE(pxsc,pysc)
Paramters	:	
Pxsc	:	Positive scale factor applied to X-axis.
Pysc	:	Positive scale factor applied to Y-axis.

TRANSL

Category	:	Graphic Environmental Routines
Purpose	:	Applies specified translation (displacement) to coordinates.
Syntax	:	TRANSL(pxdisp,pydisp)
Paramters	:	
Pxdisp	:	Displacement along X-axis.
Pydisp	:	Displacement along Y-axis.

TRIDNT

Category	:	Graphic Environmental Routines
Purpose	:	Resets the modeling transform, the window/viewport transform, or both, to identity (intial values).
Syntax	:	TRIDNT(qfull)
Paramters	:	
Qfull	:	.FALSE. - Resets modeling transform (SCALE, TRANSL, ROTATE, MTRAN) to identity. .TRUE. - Resets modeling transform and window/viewport transform (WINDOW, VWPORT) to identity.

VWPORT

Category	:	Graphic Environmental Routines
Purpose	:	Defines location of output on the display surface.
Syntax	:	VWPORT(xmin,xmax,ymin,ymax)
Paramters	:	
Xmin	:	Minimum X coordinate of viewport.
Xmax	:	Maximum X coordinate of viewport.
Ymin	:	Minimum Y coordinate of viewport.
Ymax	:	Maximum Y coordinate of viewport.

WINDOW

Category	:	Graphic Environmental Routines
Purpose	:	Specifies the portion of the coordinate system to be viewed.
Syntax	:	WINDOW(xmin,xmax,ymin,ymax)
Paramters	:	
Xmin	:	Minimum X coordinate of window.
Xmax	:	Maximum X coordinate of window.
Ymin	:	Minimum Y coordinate of window.
Ymax	:	Maximum Y coordinate of window.

ARC

Category	:	Graphic Action Routines
Purpose	:	Draws an arc with a given radius from the starting angle to the ending angle as indicated; the current cursor position is the center point for the arc.
Syntax	:	ARC(prad,psara,penda)
Paramters	:	
Prad	:	The radius for the arc.
Pstara	:	Starting angle for the arc.
Penda	:	Ending angle for the arc.

DRAW

Category	:	Graphic Action Routines
Purpose	:	Draws a vector from the current location to a specified point.
Syntax	:	DRAW(px,py)
Paramters	:	
Px	:	X coordinate of point to which vector is drawn.
Py	:	Y coordinate of point to which vector is drawn.

HOME

Category	:	Graphic Action Routines
Purpose	:	Moves the cursor to the "home" position (left hand side of the viewport).
Syntax	:	HOME

LOCATE

Category	:	Graphic Action Routines
Purpose	:	Puts the terminal into graphic input (GIN) mode and stores the coordinates of points located by the graphic cursor.
Syntax	:	LOCATE(imaxpt,pxaray,pyaray,idat,igot)
Paramters	:	
Imaxpt	:	Maximum number of points to be located.
Pxaray	:	Array containing X coordinates of points located; must be dimensioned to at least the value of Imaxpt.
Pyaray	:	Array containing Y coordinates of points located; must be dimensioned to at least the value of Imaxpt.
Idat	:	An array containing device-dependent auxiliary Z-axis information to accompany the point digitized.
Igot	:	Number of points located.

MOVE

Category	:	Graphic Action Routines
Purpose	:	Moves cursor to a specified point without drawing a vector.
Syntax	:	MOVE(px,py)
Paramters	:	
Px	:	X coordinate of point to which cursor is moved.
Py	:	Y coordinate of point to which cursor is moved.

NEWPAG

Category	:	Graphic Action Routines
Purpose	:	Provides a clean surface for display of output.
Syntax	:	NEWPAG

PANEL

Category	:	Graphic Action Routines
Purpose	:	Displays a panel or an emulated panel on display screen.
Syntax	:	PANEL(icnt,pxaray,pyaray)
Paramters	:	
Icnt	:	Number of points defining the perimeter of panel.
Pxaray	:	An array containing X coordinates of points defining the perimeter of the panel; must be dimensioned atleast to the value of Icnt.
Pyaray	:	An array containing Y coordinates of points defining the perimeter of the panel; must be dimensioned atleast to the value of Icnt.

POLY

Category	:	Graphic Action Routines
Purpose	:	Draws a polygon.
Syntax	:	POLY(icnt,xarray,yarray)
Paramters	:	
Icnt	:	Size of Xarray and Yarray.
Xarray	:	X coordinates of points to be drawn; specified in world space.
Yarray	:	Y coordinates of points to be drawn; specified in world space.

TXICUR

Category	:	Text Environmental Routines
Purpose	:	Establishes the relationship of text output to the intial cursor position.
Syntax	:	TXICUR(ipos)
Paramters	:	
Ipos	:	Set to an integer from 1-9 to specify the position of text in relation to the intial cursor position.

INUMBR

Category	:	Text Action Routines
Purpose	:	Displays integer data as text.
Syntax	:	INUMBR(intval,imxchr)
Paramters	:	
Intval	:	The integer to be displayed.
Imxchr	:	Maxinum number of character to display.

RNUMBR

Category :	Text Action Routines
Purpose :	Displays a real number as text.
Syntax :	RNUMBR(pvalue,ipastd,imxchr)
Paramters :	
Pvalue :	The real number to be displayed.
Ipastd :	Maximum number of digits to the right of the decimal point; set to -1 to suppress the decimal point.
Imxchr :	Maximum number of character to be displayed.

TEXT

Category :	Text Action Routines
Purpose :	Displays a string of alphanumeric text.
Syntax :	TEXT(ilenst,ichray)
Paramters :	
Ilenst :	Number of characters in the string.
Ichray :	Text to be output; string format only;

APPENDIX C

Hierarchy of Subroutines

Phase I

MAIN PROGRAM		
	READ_DATA	Read the machine code and create an index file.
	INITIALIZE*	Initialize variables at the beginning. Read tool offsets and toolwidths.
	PROCESS_DATA	Process each machine code statement.
READ_DATA		
	CREATE_INDEX*	Create index for the position of the statement with respect to the statement number.
PROCESS_DATA		
	REINITIALIZE*	Initialize variables at end of processing each machine code statement.
	REREAD_CHECK	Reads the machine code again to process it.
	DEVIATE	Process subroutine calls of the machine code program.
DEVIATE		
	SCROLL*	Skips certain number of lines before reading.
	REINTIALIZE*	Initialize variables at end of processing each machine code statement.
	REREAD_CHECK	Reads the machine code again to process it.

REREAD_CHECK		
	LEXICAL_READ	Reads numeric characters and converts them to numbers.
	CHECK_CODES	Invokes procedures according to the alphabetic part of the machine code.
	CHANGE_ORIGIN	Changes origin for future coordinate calculations.
	CHANGE_TOOL	Indexes the tool holding device.
	GO_HOME	Takes the tool/machine to the machine home position.
	CIRCULAR_MOTION	Moves tool for circular movement.
	MID_STOP	Finds reason for stopping program in the middle and procedes accordingly.
	FIXED_CYCLE	Moves the tool during fixed cycle operations.
	GO_AHEAD	Moves tool during linear movement.
LEXICAL_READ		
	NUMERIC*	Checks whether a character is an integer or not.
CHANGE_ORIGIN		
	PRINT*	Prints coordinates for plotting.
CHANGE_TOOL		
	PRINT*	Prints coordinates for plotting.
GO_HOME		
	PRINT*	Prints coordinates for plotting.
CIRCULAR_MOTION		
	PRINT*	Prints coordinates for plotting.
MID_STOP		
	PRINT*	Prints coordinates for plotting.
FIXED_CYCLE		
	PRINT*	Prints coordinates for plotting.

GO_AHEAD		
	PRINT*	Prints coordinates for plotting.
CHECK_CODES		
	CHECKF*	Checks numbers that are associated with F codes.
	CHECKG*	Checks numbers that are associated with G codes.
	CHECKM*	Checks numbers that are associated with M codes.
	CHECKP*	Checks numbers that are associated with P codes.
	CHECKL*	Checks numbers that are associated with L codes.
	CHECKR*	Checks numbers that are associated with R codes.
	CHECKD*	Checks numbers that are associated with D codes.
	CHECKH*	Checks numbers that are associated with H codes.
	CHECKXYZ*	Checks numbers that are associated with X, Y and Z codes.
	CHECKIJK*	Checks numbers that are associated with I, J and K codes.

Phase II

MAIN PROGRAM		
	CLEAR*	Clears the screen, draws the workpiece and titles toolwidth and Z coordinate display.
	OPTION	Provides various options for the user.
	SBDRAW	Draws the tool path.
	ZOOM	Enlarge a certain segment of the plot.
	SBANSI*	Sets terminal back to ANSI mode.
OPTION		
	SBANSI*	Sets terminal back to ANSI mode.
	CLEAR*	Clears the screen, draws the workpiece and titles toolwidth and Z coordinate display.
	GRID*	Draws and labels the grid.
SBDRAW		
	CLEAR*	Clears the screen, draws the workpiece and titles toolwidth and Z coordinate display.
	FIXTUR*	Draws the fixtures.
	GRID*	Draws and labels the grid.
	TOOL*	Draws circles according to the toolwidth along the path of the tool center.
	STLINE	Draws tool path for linear movement.
	CURVE	Draws tool path for circular movement.
	TXTWRT	Displays the new value during any tool or height change.

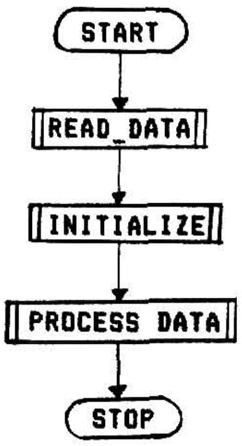
ZOOM		
	GRID*	Draws and labels the grid.
	TOOL*	Draws circles according to the toolwidth along the path of the tool center.
	STLINE	Draws tool path for linear movement.
	CURVE	Draws tool path for circular movement.
TXTWRT		
	CLEAR*	Clears the screen, draws the workpiece and titles toolwidth and Z coordinate display.
	FIXTUR*	Draws the fixtures.
	GRID*	Draws and labels the grid.
STLINE		
	TOOL*	Draws circles according to the toolwidth along the path of the tool center.
CURVE		
	TOOL*	Draws circles according to the toolwidth along the path of the tool center.

APPENDIX D

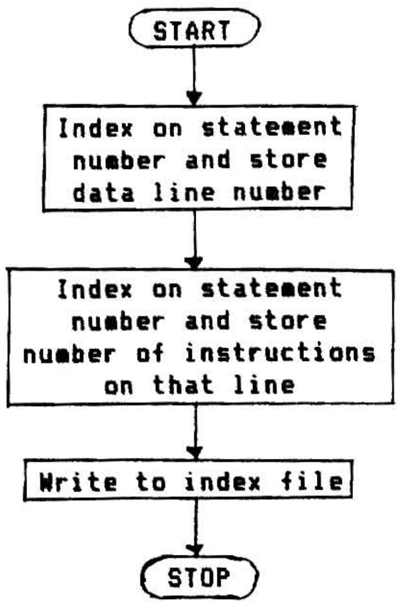
Flow Charts

PHASE I

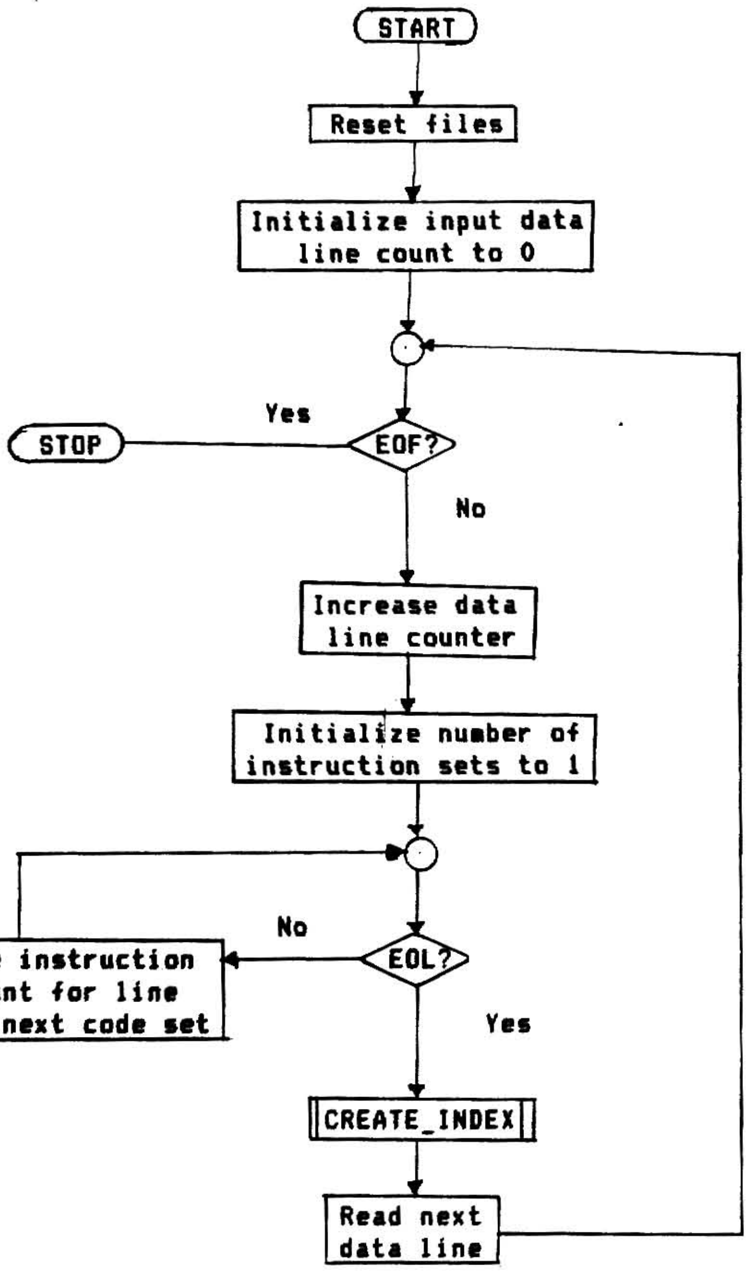
Main Program:



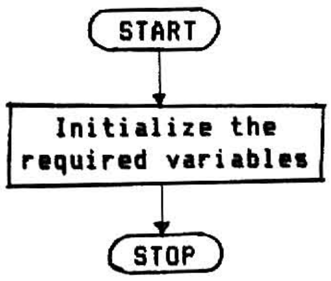
Create_index:


START
Index on statement number and store data line number
Index on statement number and store number of instructions on that line
Write to index file
STOP

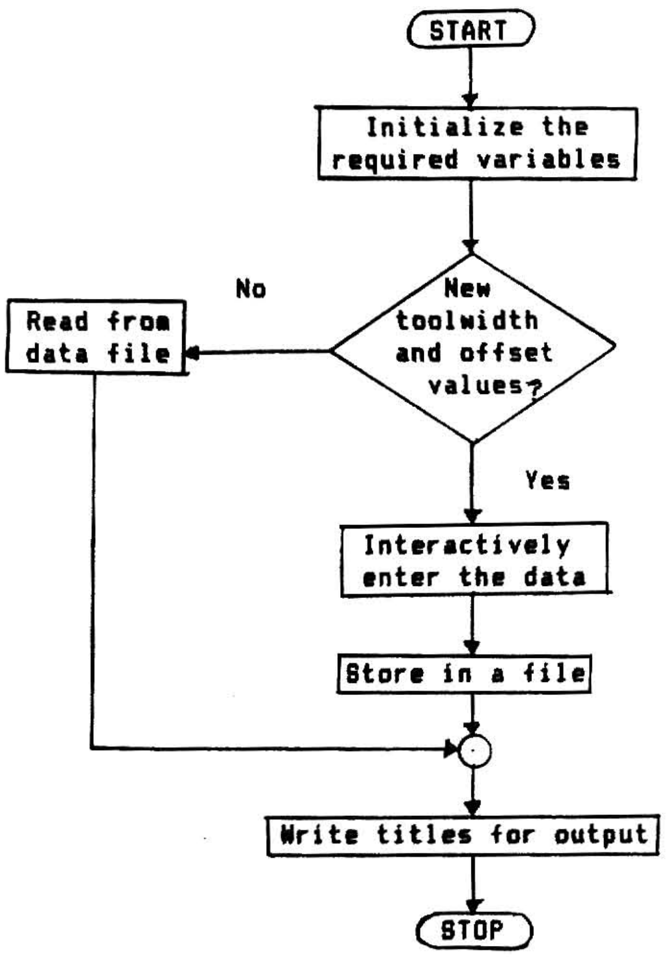
Read_data:


START
Reset files
Initialize input data line count to 0
Yes
STOP
EOF?
No
Increase data line counter
Initialize number of instruction sets to 1
No
Increase instruction set count for line and read next code set
EOL?
Yes
CREATE_INDEX
Read next data line

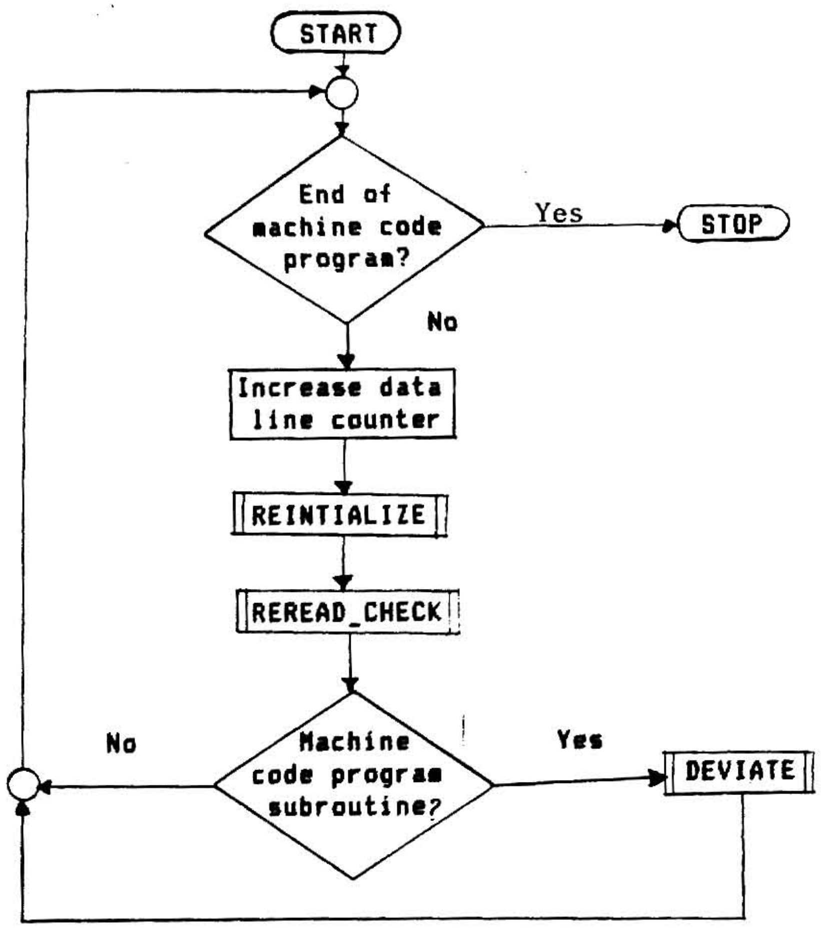
Reinitialize:



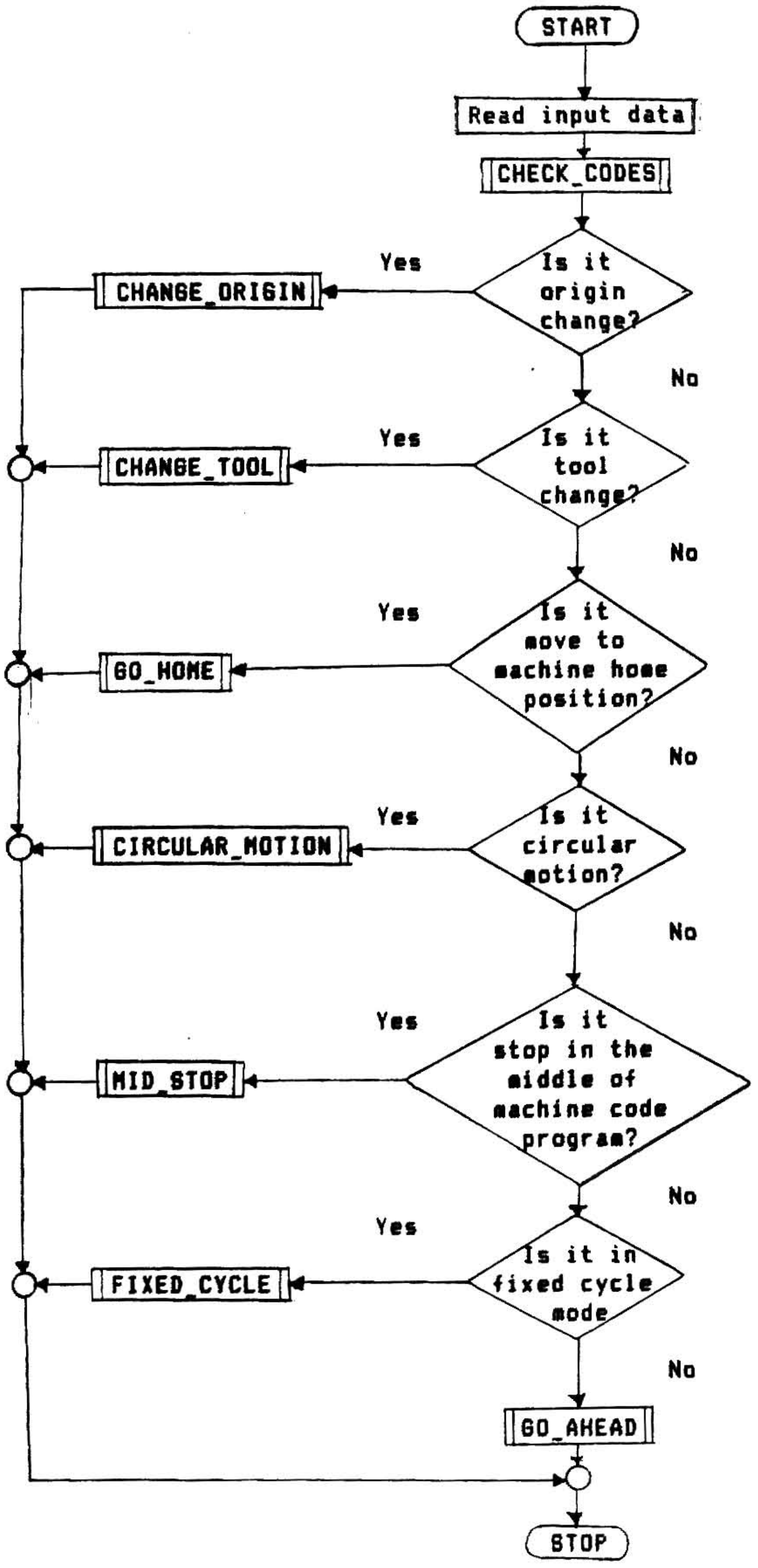
Initialize:


START
Initialize the required variables
New toolwidth and offset values?
No
Read from data file
Yes
Interactively enter the data
Store in a file
Write titles for output
STOP

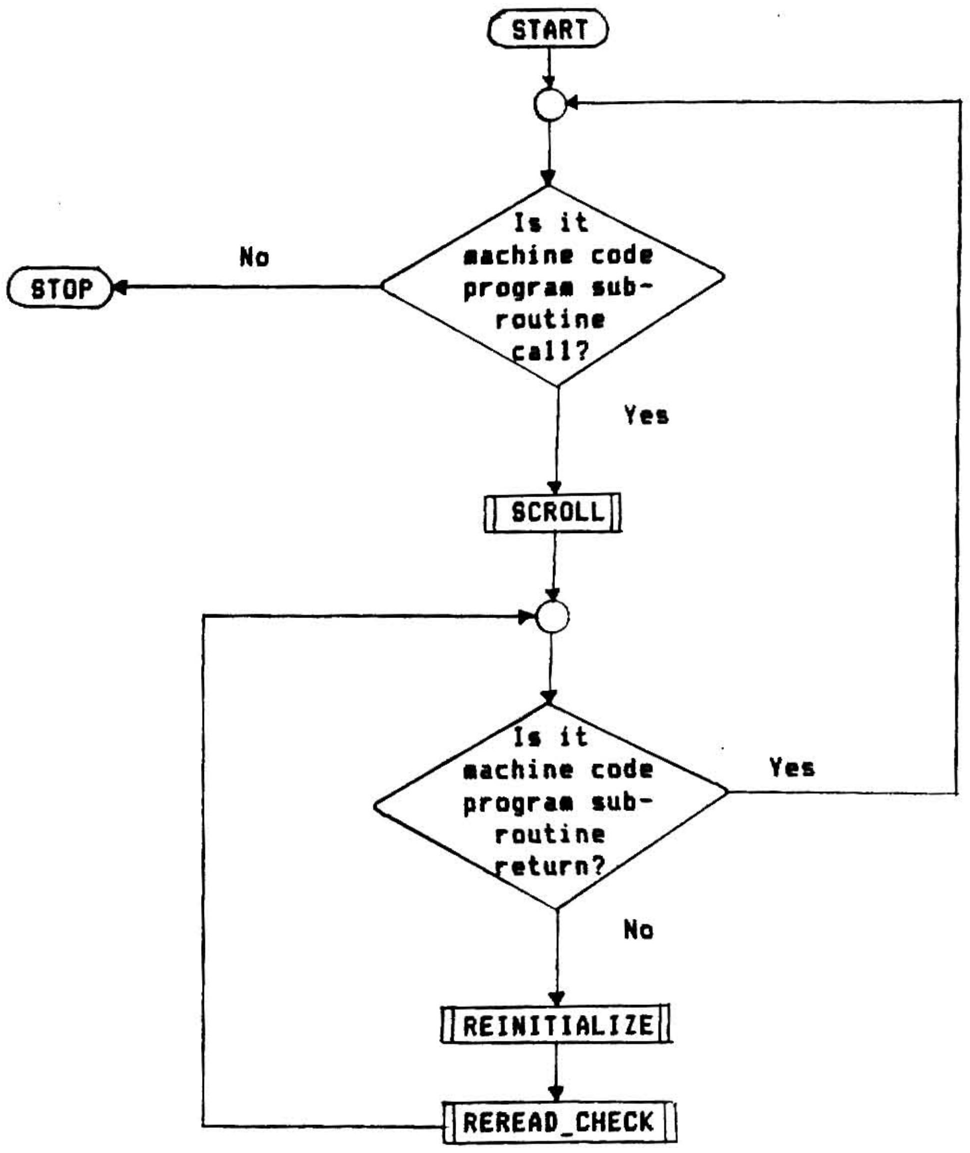
Process_data:


START
End of machine code program?
Yes
STOP
No
Increase data line counter
REINTIALIZE
REREAD_CHECK
Machine code program subroutine?
No
Yes
DEVIATE

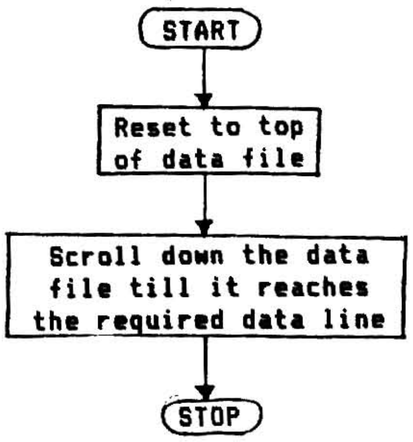
Reread_check:



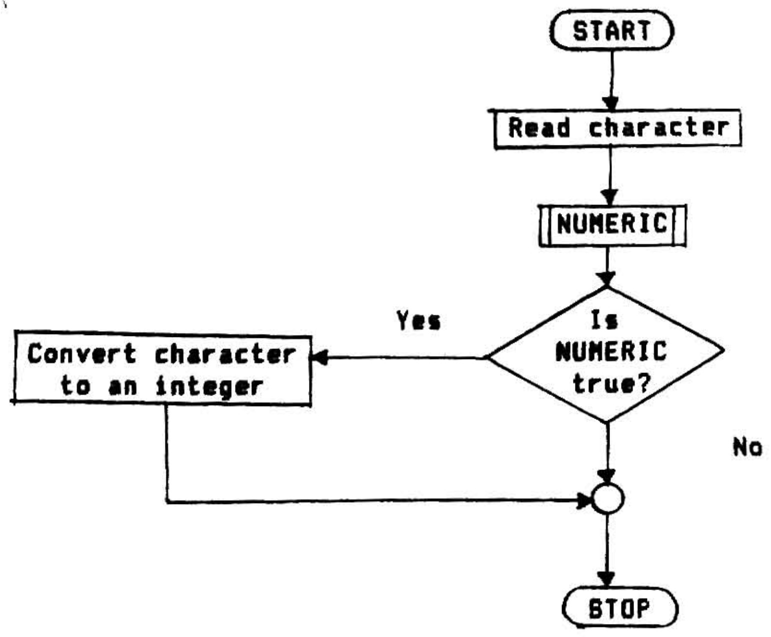
Deviate:


START
Is it machine code program sub-routine call?
No
STOP
Yes
SCROLL
Is it machine code program sub-routine return?
Yes
No
REINITIALIZE
REREAD_CHECK

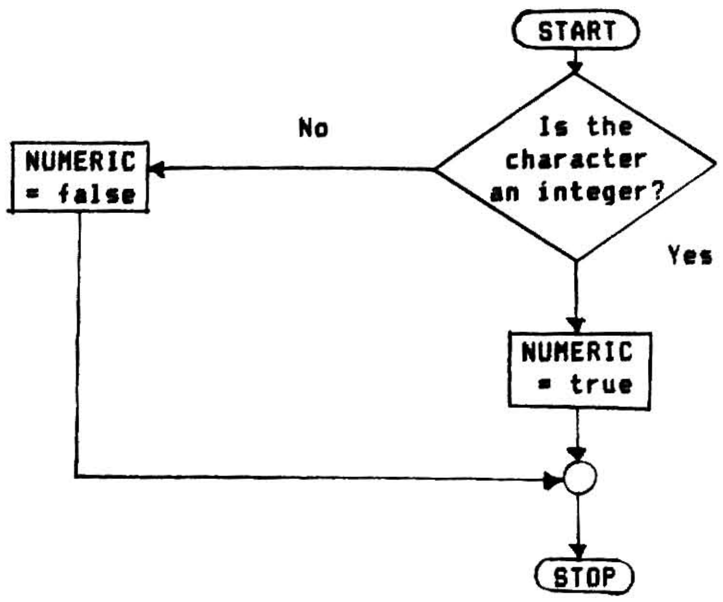
Scroll1


START
Reset to top of data file
Scroll down the data file till it reaches the required data line
STOP

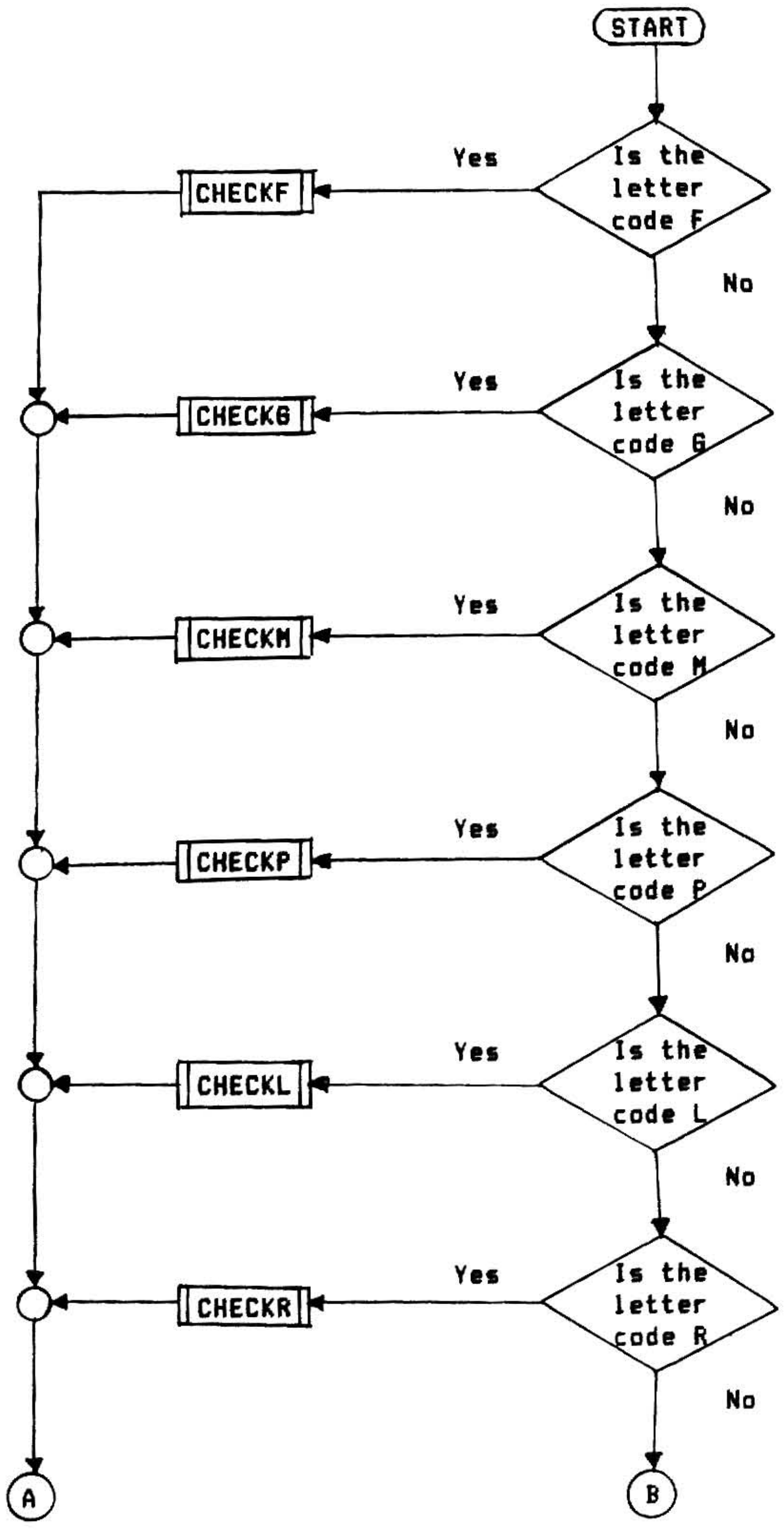
Lexical_read:

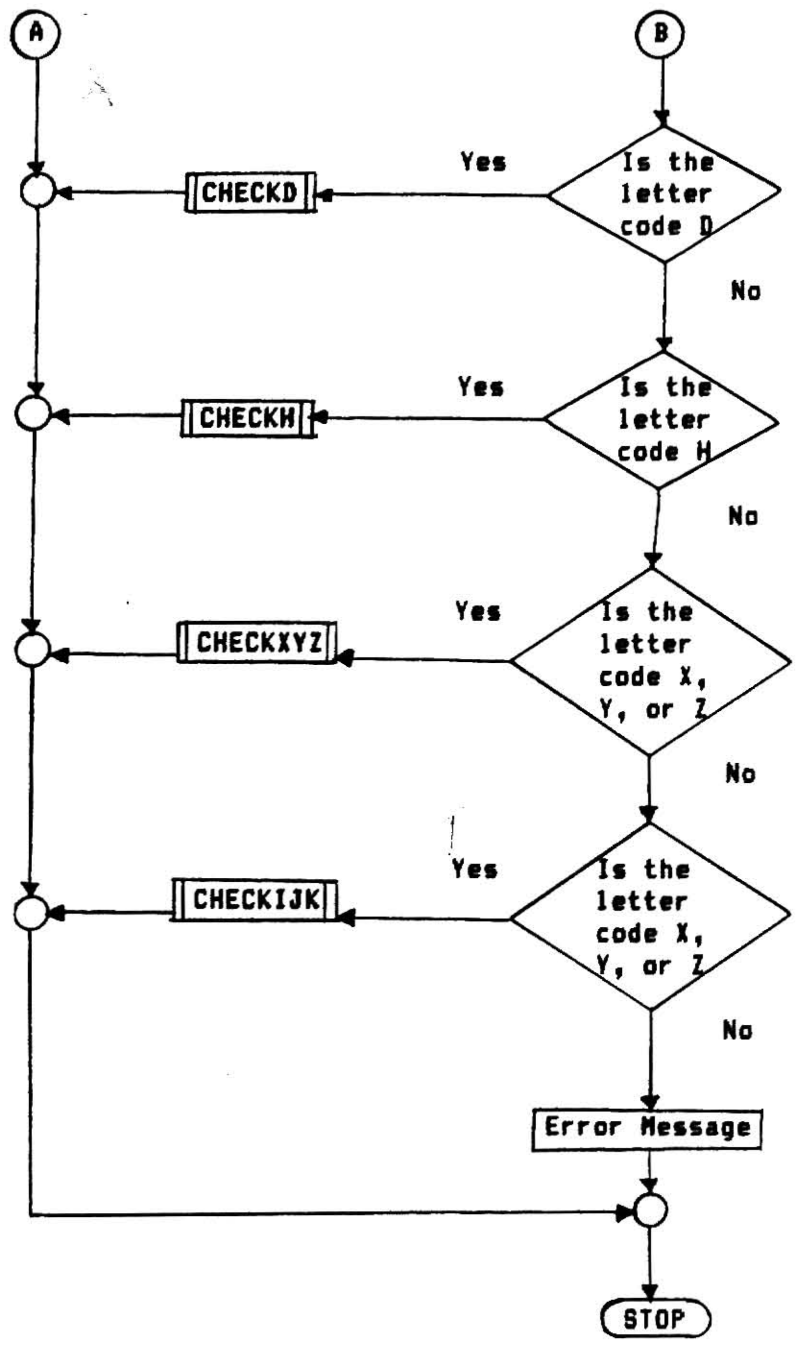


__Numeric:__

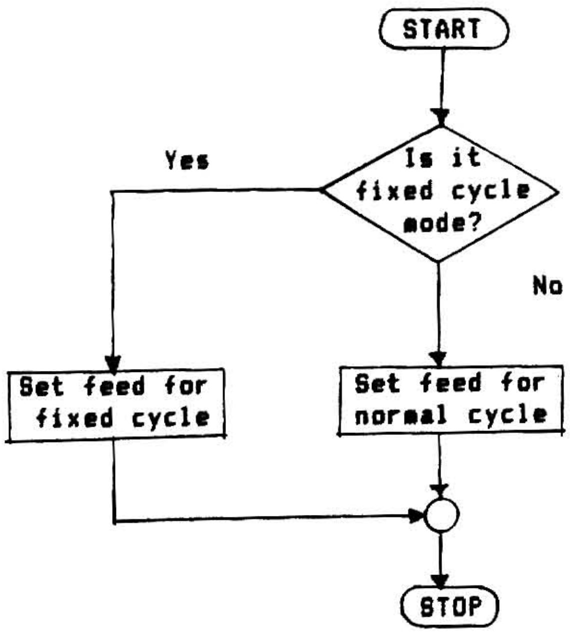

START
Is the character an integer?
No
NUMERIC = false
Yes
NUMERIC = true
STOP

Check_codes:

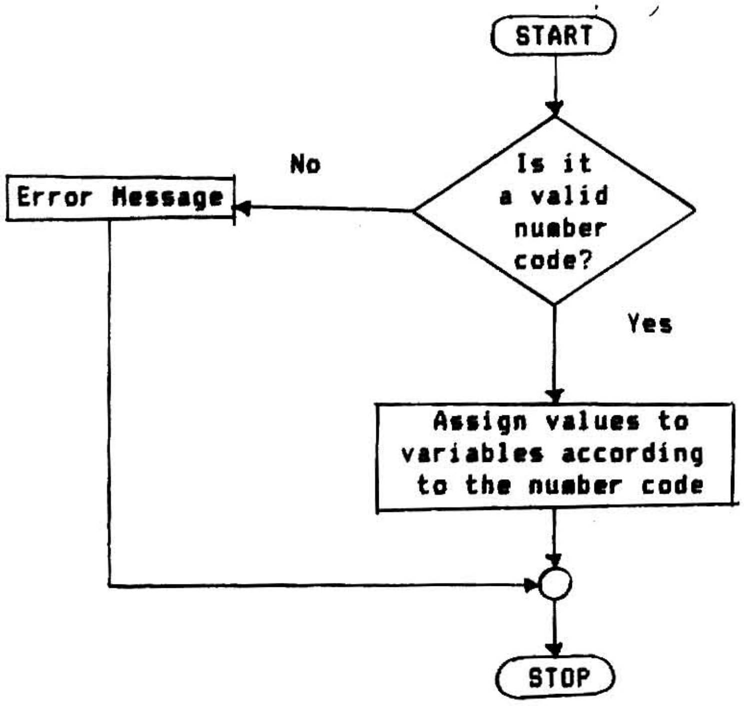

START
Is the letter code F
Yes
CHECKF
No
Is the letter code G
Yes
CHECKG
No
Is the letter code M
Yes
CHECKM
No
Is the letter code P
Yes
CHECKP
No
Is the letter code L
Yes
CHECKL
No
Is the letter code R
Yes
CHECKR
No
A
B


A
B
Is the letter code D
Yes
CHECKD
No
Is the letter code H
Yes
CHECKH
No
Is the letter code X, Y, or Z
Yes
CHECKXYZ
No
Is the letter code X, Y, or Z
Yes
CHECKIJK
No
Error Message
STOP

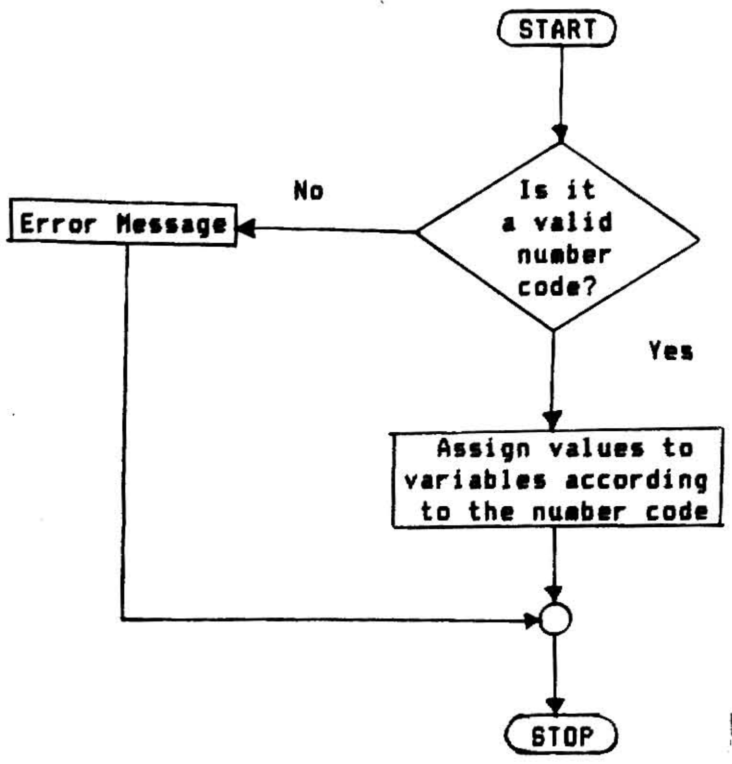
Checkf:


START
Is it fixed cycle mode?
Yes
No
Set feed for fixed cycle
Set feed for normal cycle
STOP

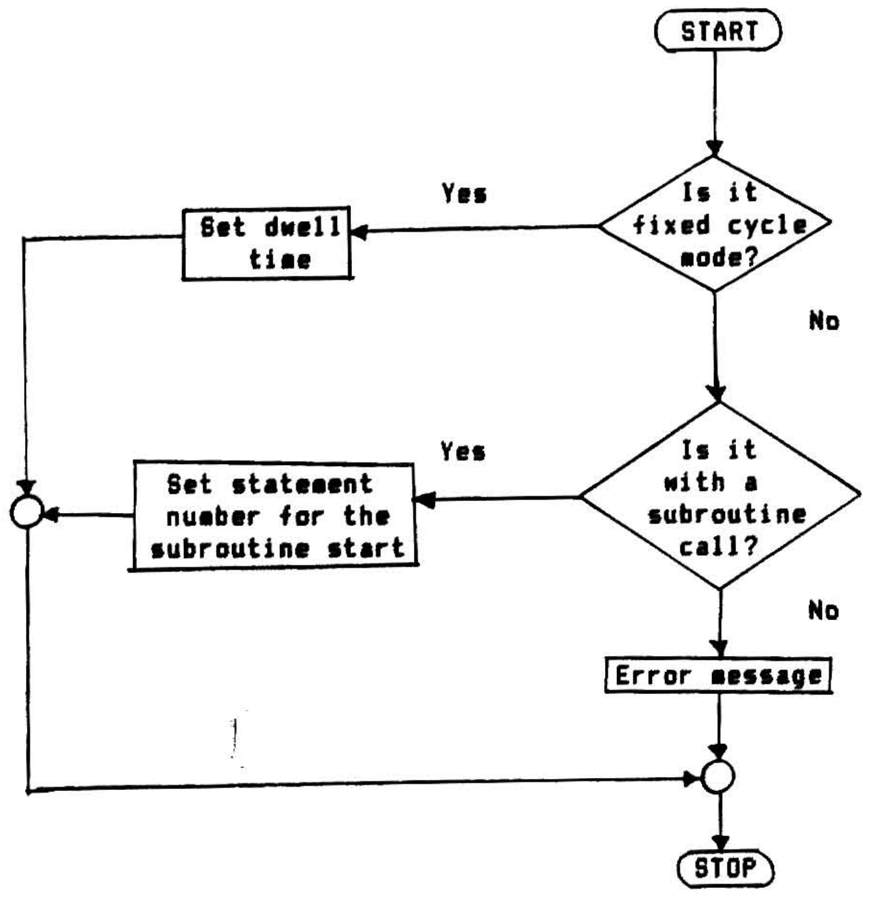
Checkgi


START
Is it a valid number code?
No
Error Message
Yes
Assign values to variables according to the number code
STOP

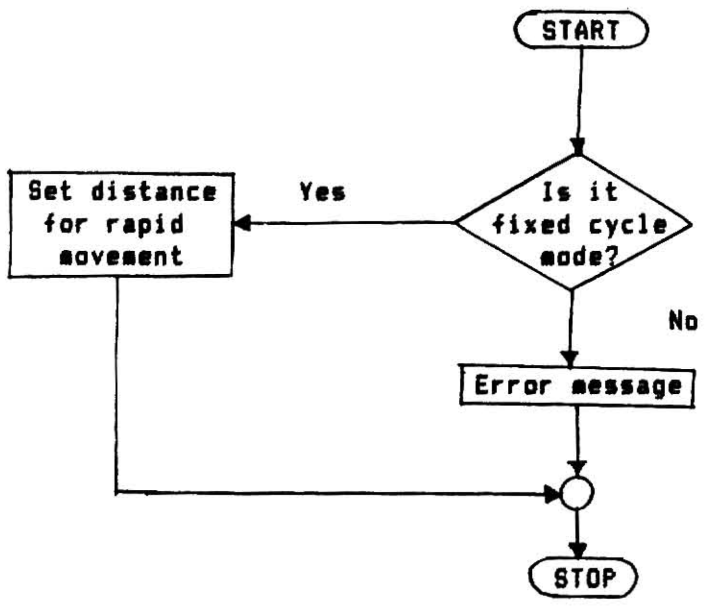
Checkm:


START
Is it a valid number code?
No
Error Message
Yes
Assign values to variables according to the number code
STOP

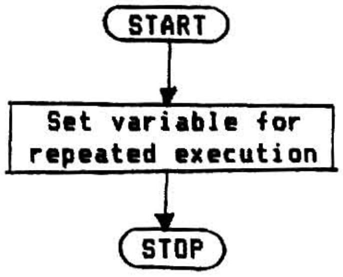
Checkei


START
Is it fixed cycle mode?
Yes
Set dwell time
No
Is it with a subroutine call?
Yes
Set statement number for the subroutine start
No
Error message
STOP

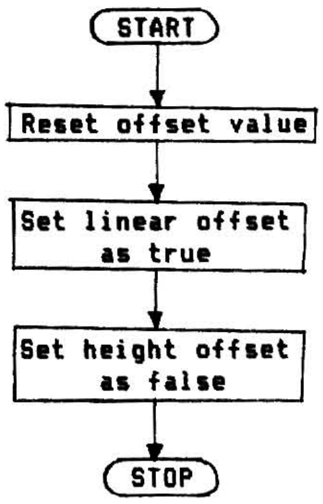
Checkr:


START
Is it fixed cycle mode?
Yes
Set distance for rapid movement
No
Error message
STOP

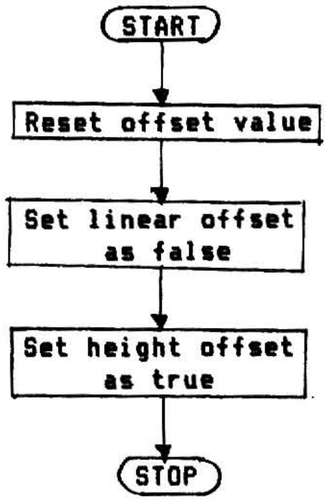
Checkli



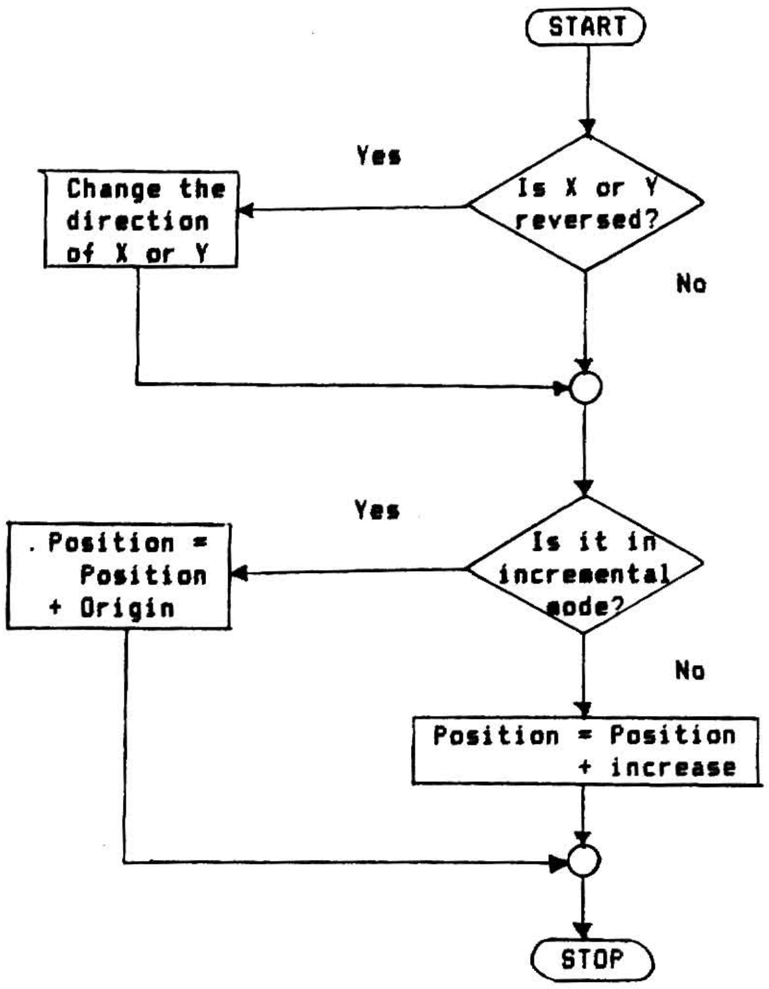
Checkd1


START
Reset offset value
Set linear offset as true
Set height offset as false
STOP

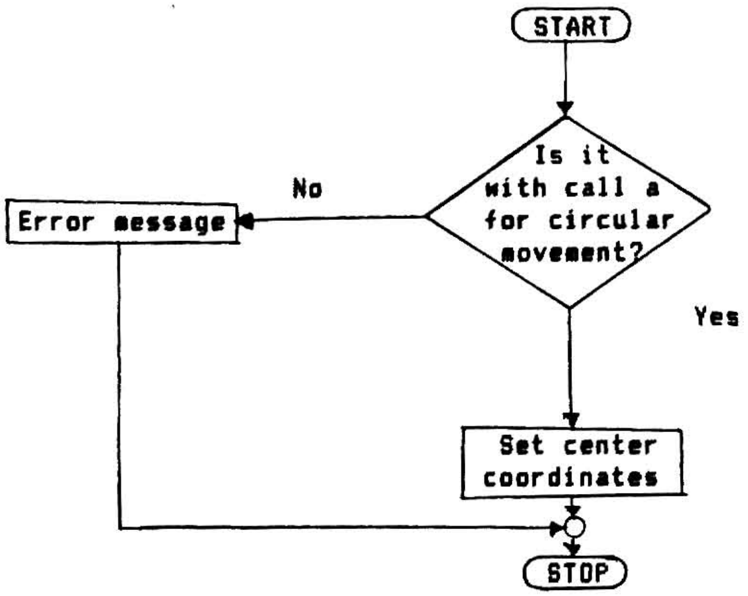
Checkhi


START
Reset offset value
Set linear offset as false
Set height offset as true
STOP

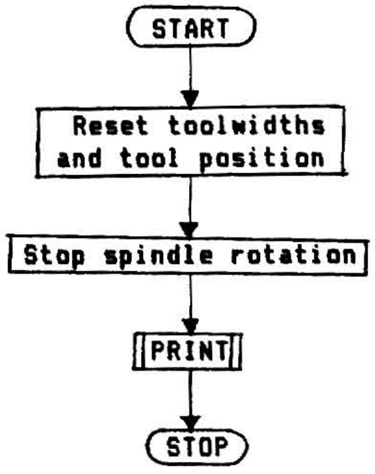
Checkxyz1


START
Is X or Y reversed?
Yes
Change the direction of X or Y
No
Is it in incremental mode?
Yes
. Position = Position + Origin
No
Position = Position + increase
STOP

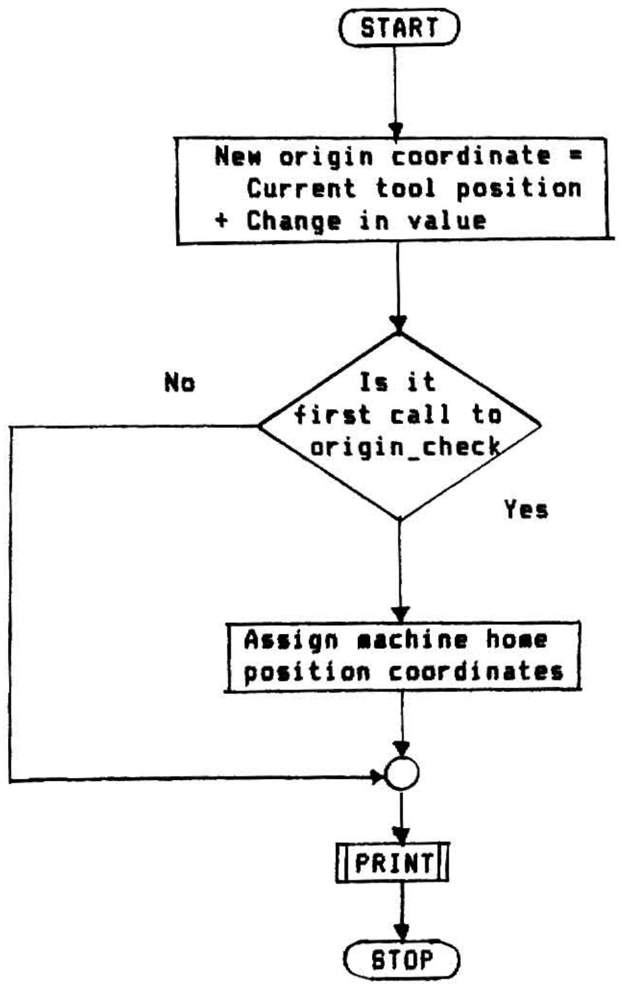
Checkijk:


START
Is it with call a for circular movement?
No
Yes
Error message
Set center coordinates
STOP

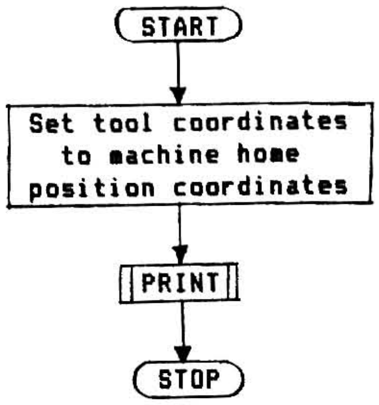
Change_tool:


START
Reset toolwidths and tool position
Stop spindle rotation
PRINT
STOP

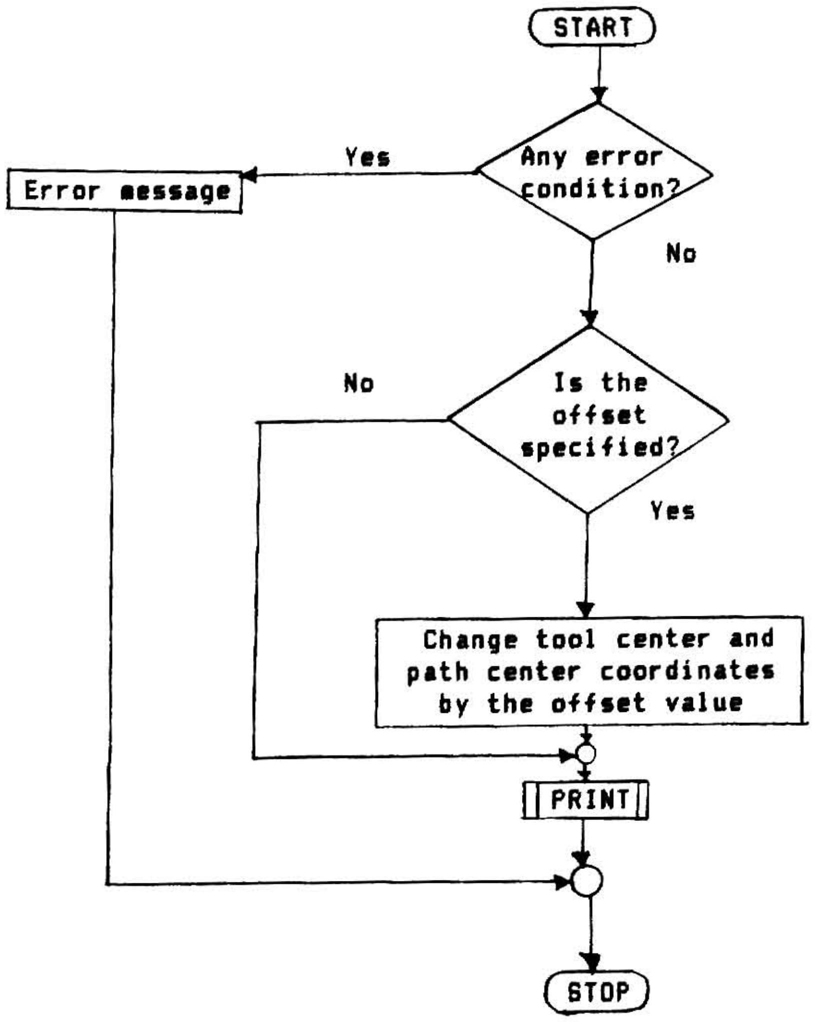
Change_origin:


START
New origin coordinate = Current tool position + Change in value
Is it first call to origin_check
No
Yes
Assign machine home position coordinates
PRINT
STOP

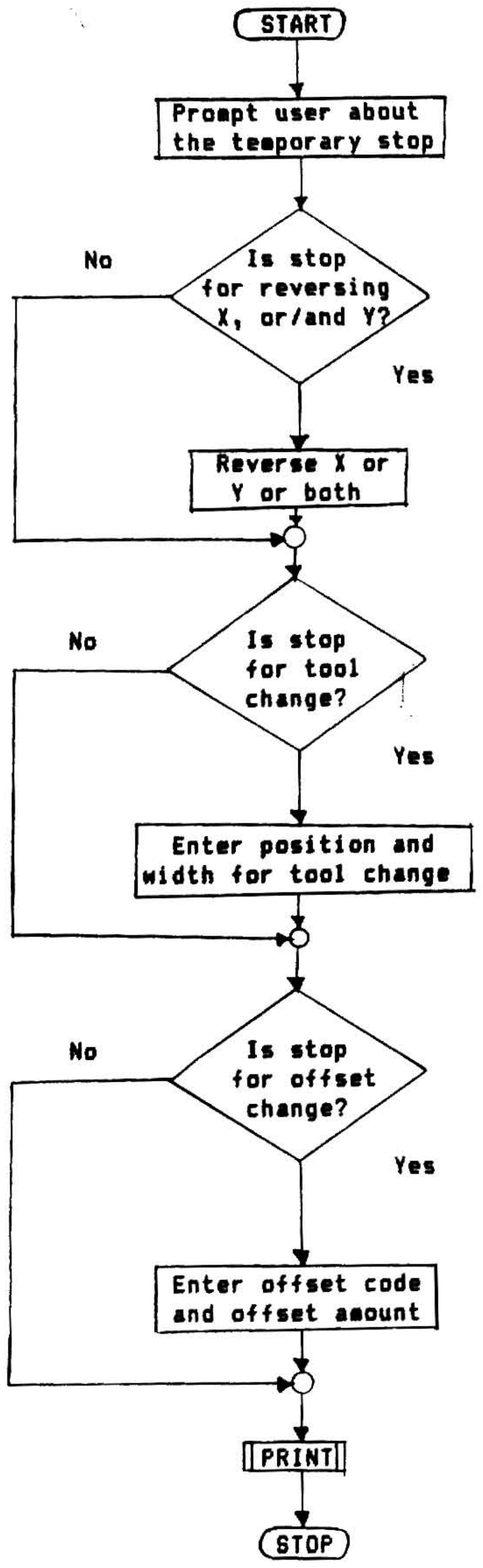
Go_home:


START
Set tool coordinates to machine home position coordinates
PRINT
STOP

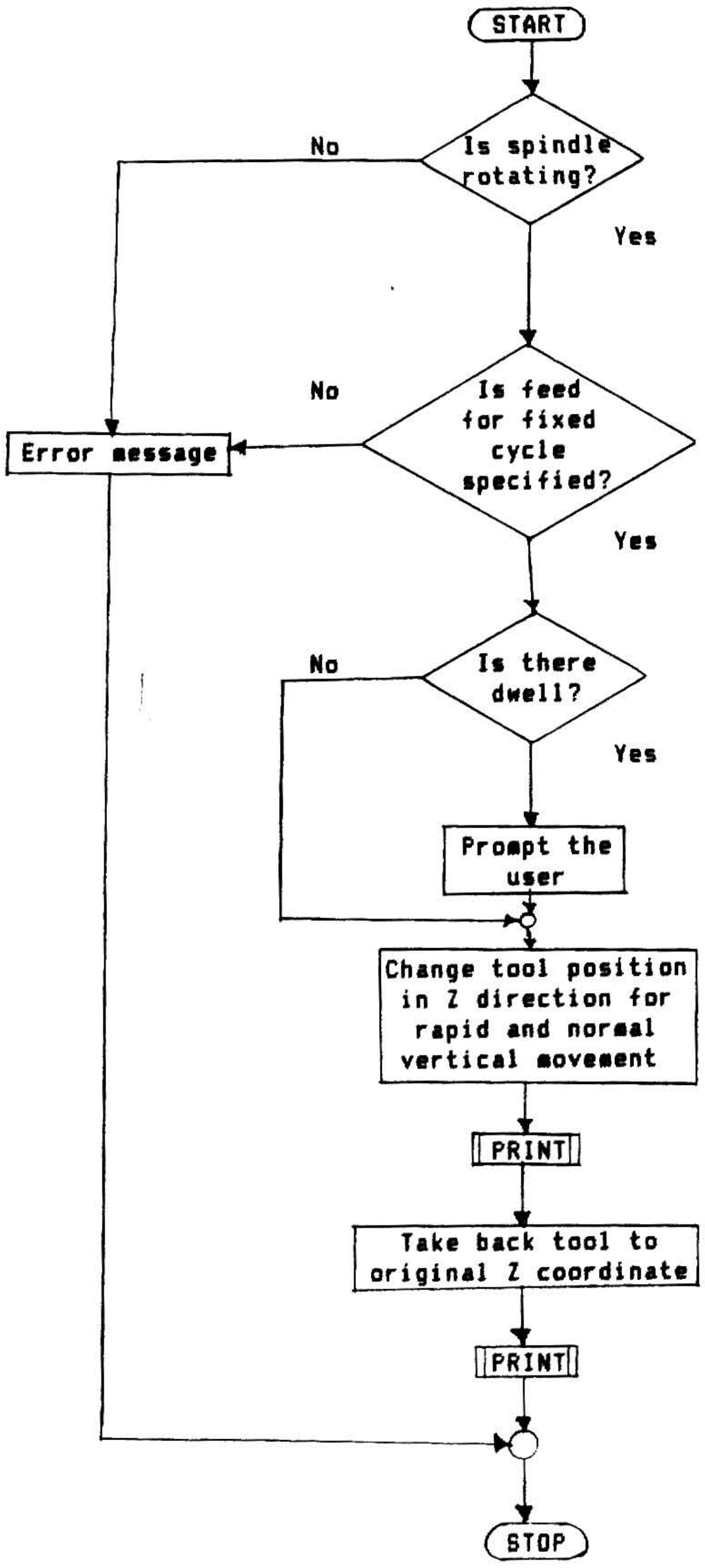
Circular motion:



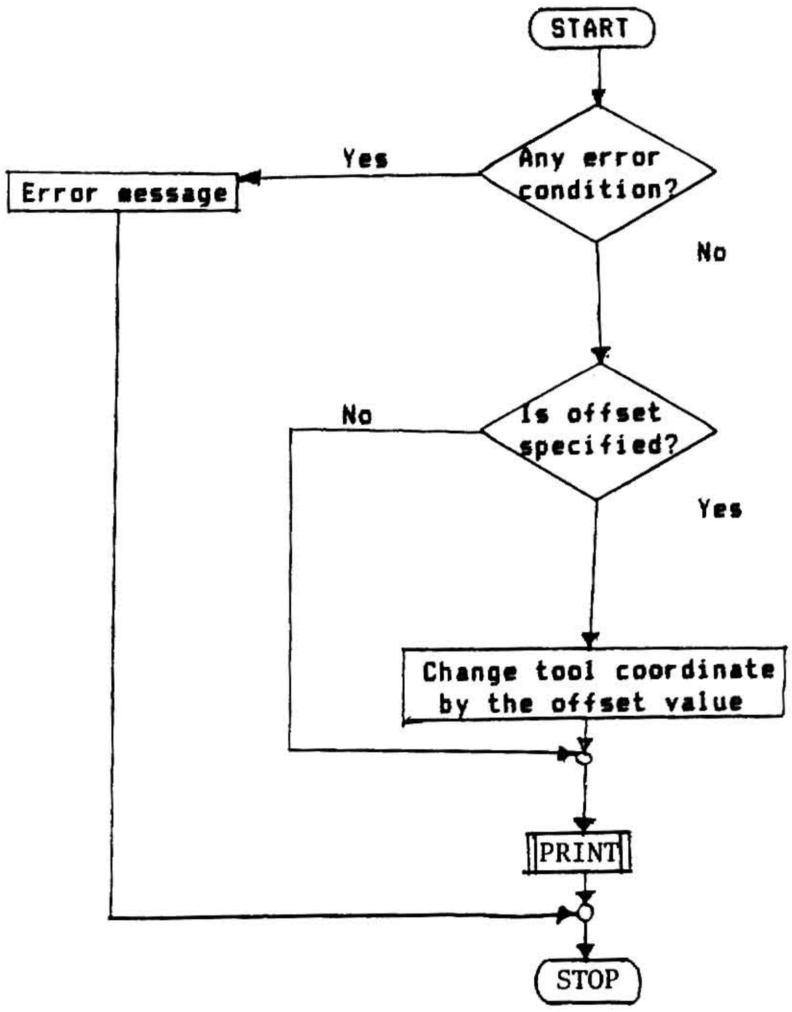
Mid_stop:


START
Prompt user about the temporary stop
Is stop for reversing X, or/and Y?
No
Yes
Reverse X or Y or both
Is stop for tool change?
No
Yes
Enter position and width for tool change
Is stop for offset change?
No
Yes
Enter offset code and offset amount
PRINT
STOP

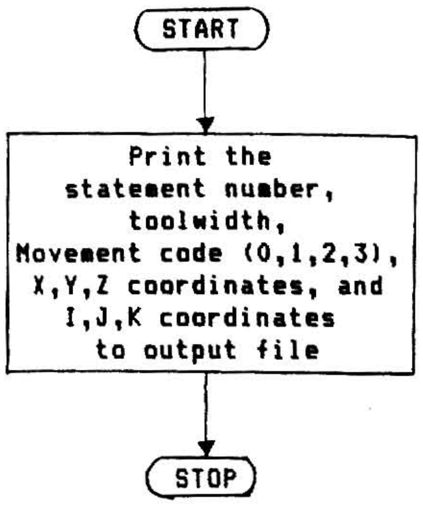
Fixed_cycle:


START
Is spindle rotating?
No
Yes
Is feed for fixed cycle specified?
No
Yes
Error message
Is there dwell?
No
Yes
Prompt the user
Change tool position in Z direction for rapid and normal vertical movement
PRINT
Take back tool to original Z coordinate
PRINT
STOP

Go_ahead:

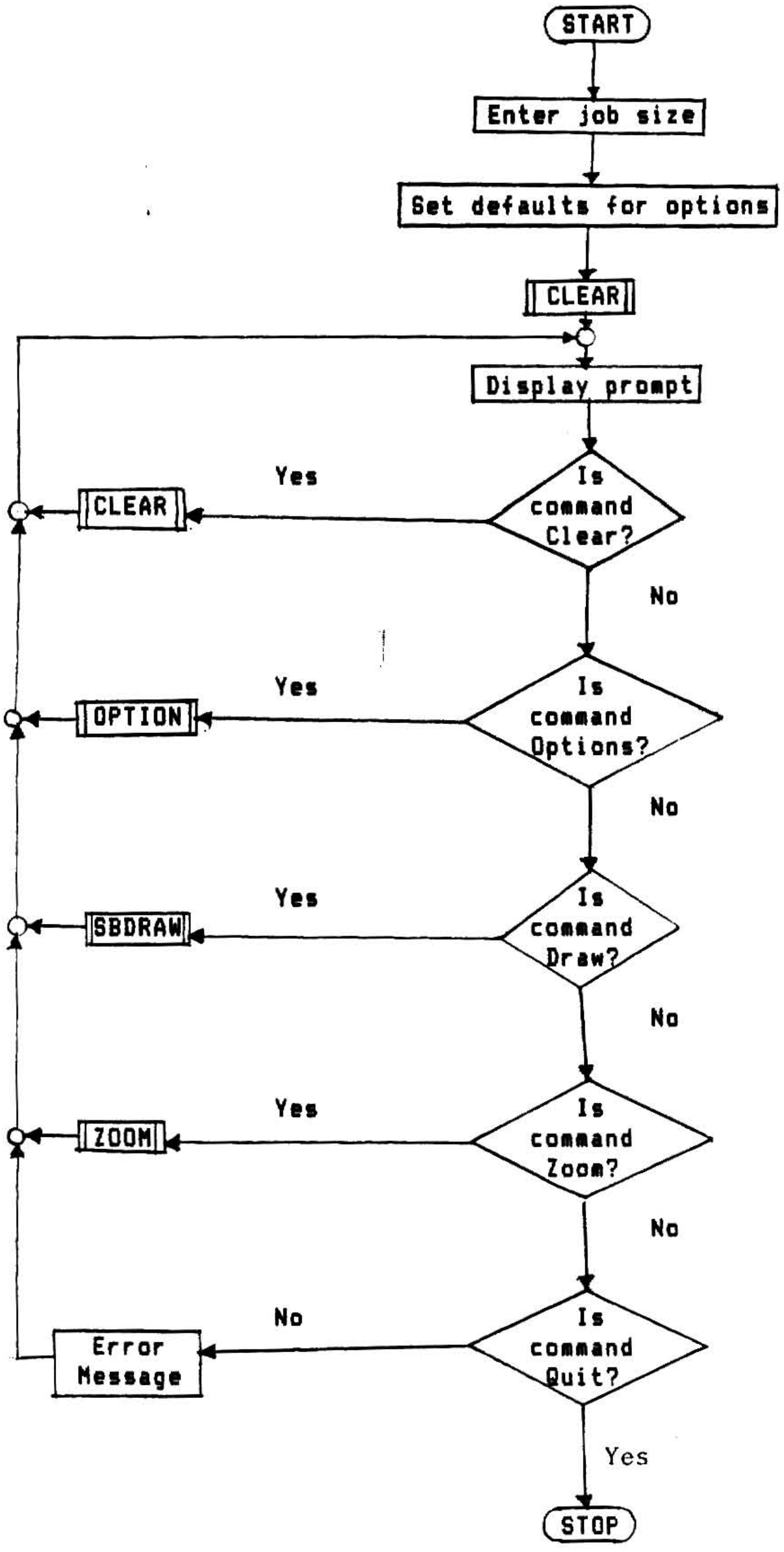

START
Any error condition?
Yes
No
Error message
Is offset specified?
No
Yes
Change tool coordinate by the offset value
PRINT
STOP

Print:

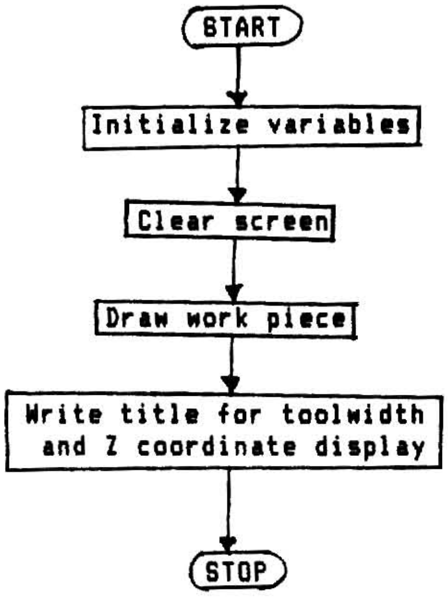

START
Print the statement number, toolwidth, Movement code (0,1,2,3), X,Y,Z coordinates, and I,J,K coordinates to output file
STOP

PHASE II

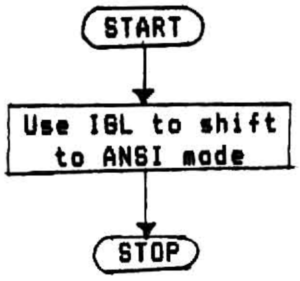
Main Program:



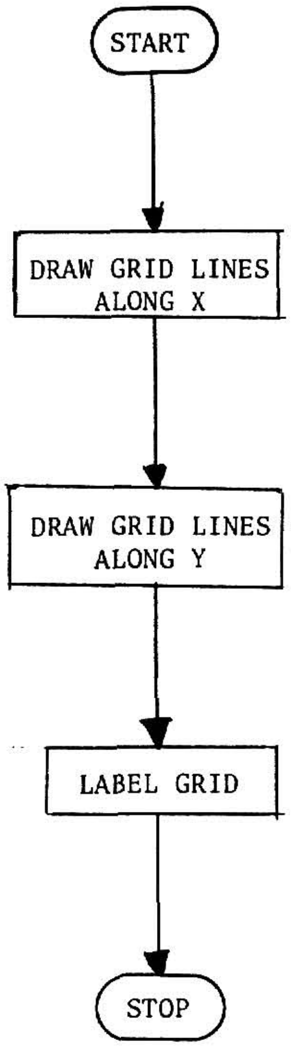
Clear:


START
Initialize variables
Clear screen
Draw work piece
Write title for toolwidth and Z coordinate display
STOP

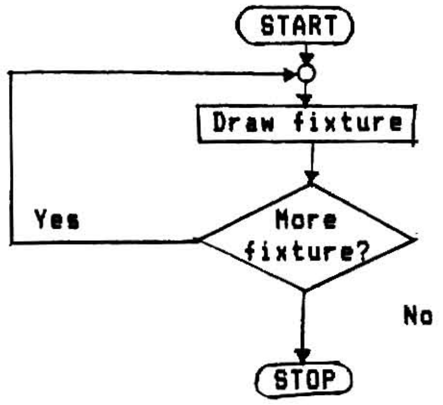
Sbansi:



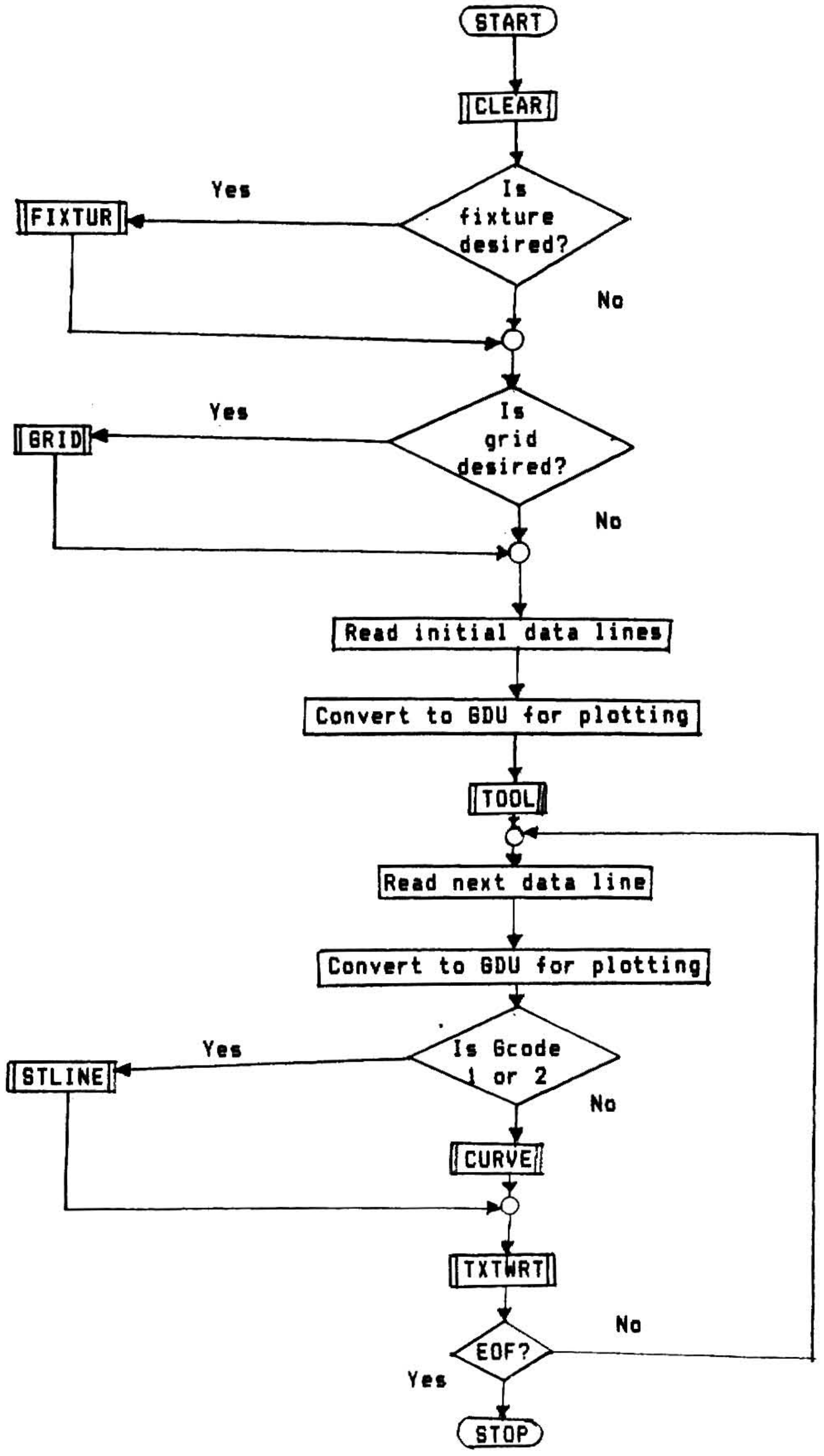
Grid:


START
DRAW GRID LINES ALONG X
DRAW GRID LINES ALONG Y
LABEL GRID
STOP

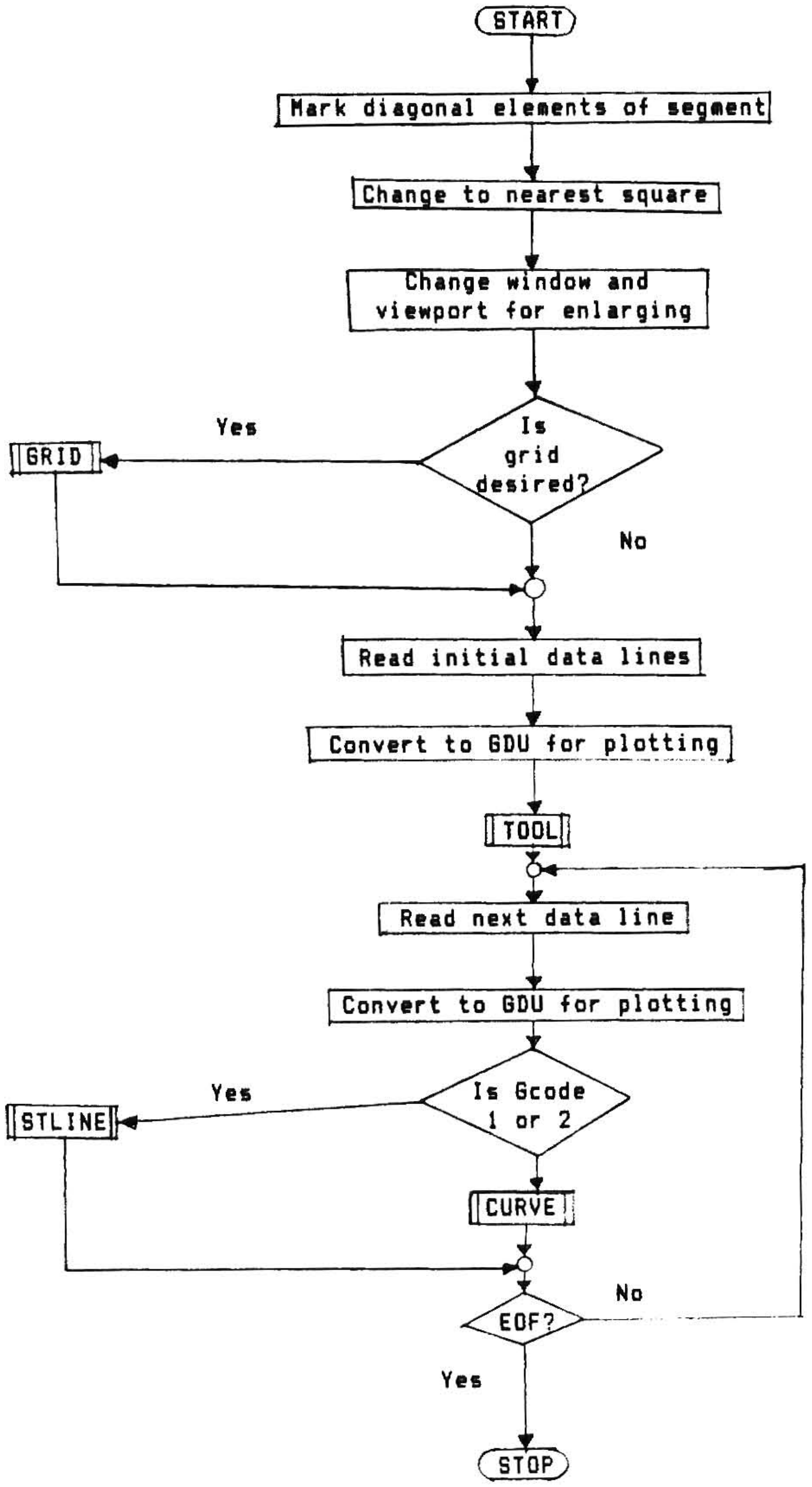
Fixture:



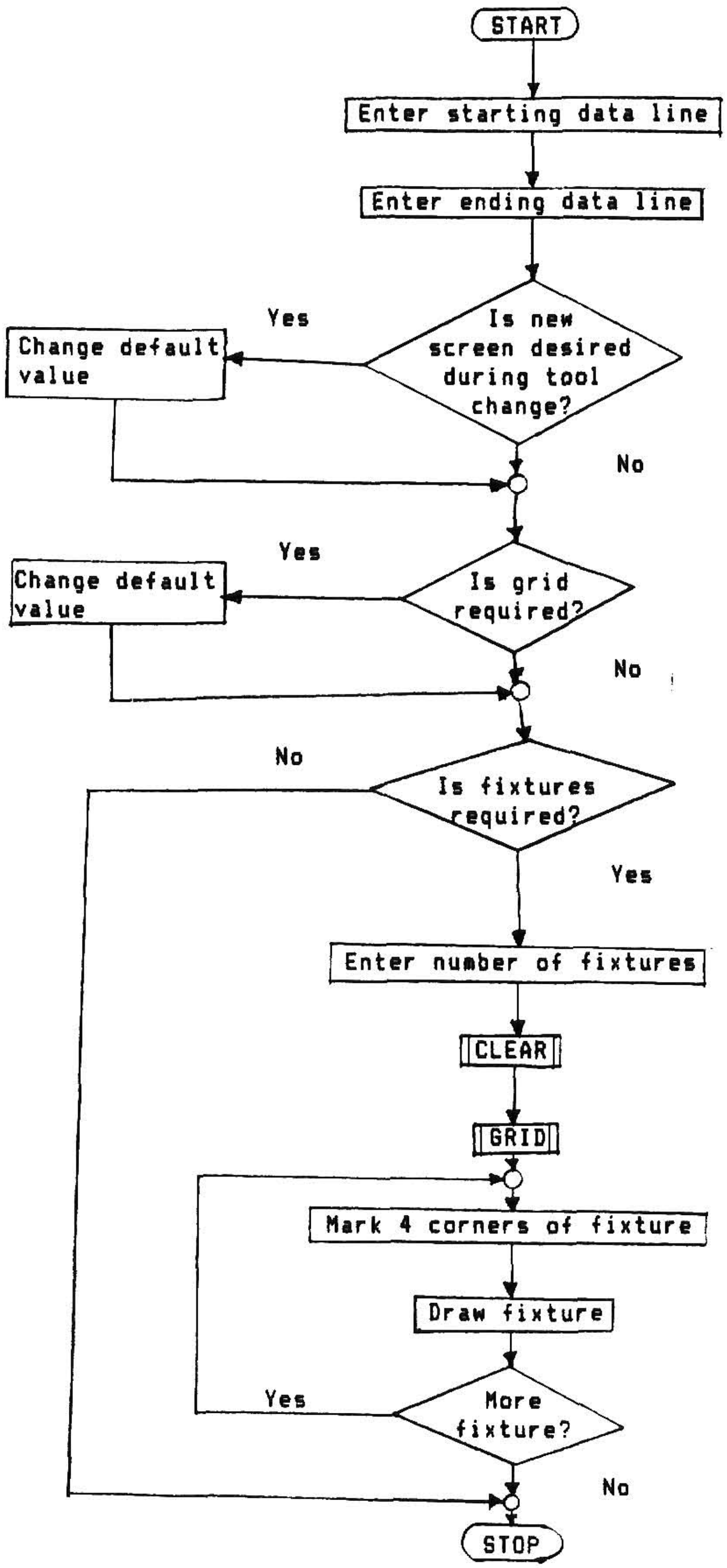
Sbdraw:


START
CLEAR
Is fixture desired?
Yes
FIXTUR
No
Is grid desired?
Yes
GRID
No
Read initial data lines
Convert to GDU for plotting
TOOL
Read next data line
Convert to GDU for plotting
Is Gcode 1 or 2
Yes
STLINE
No
CURVE
TXTWRT
EOF?
No
Yes
STOP

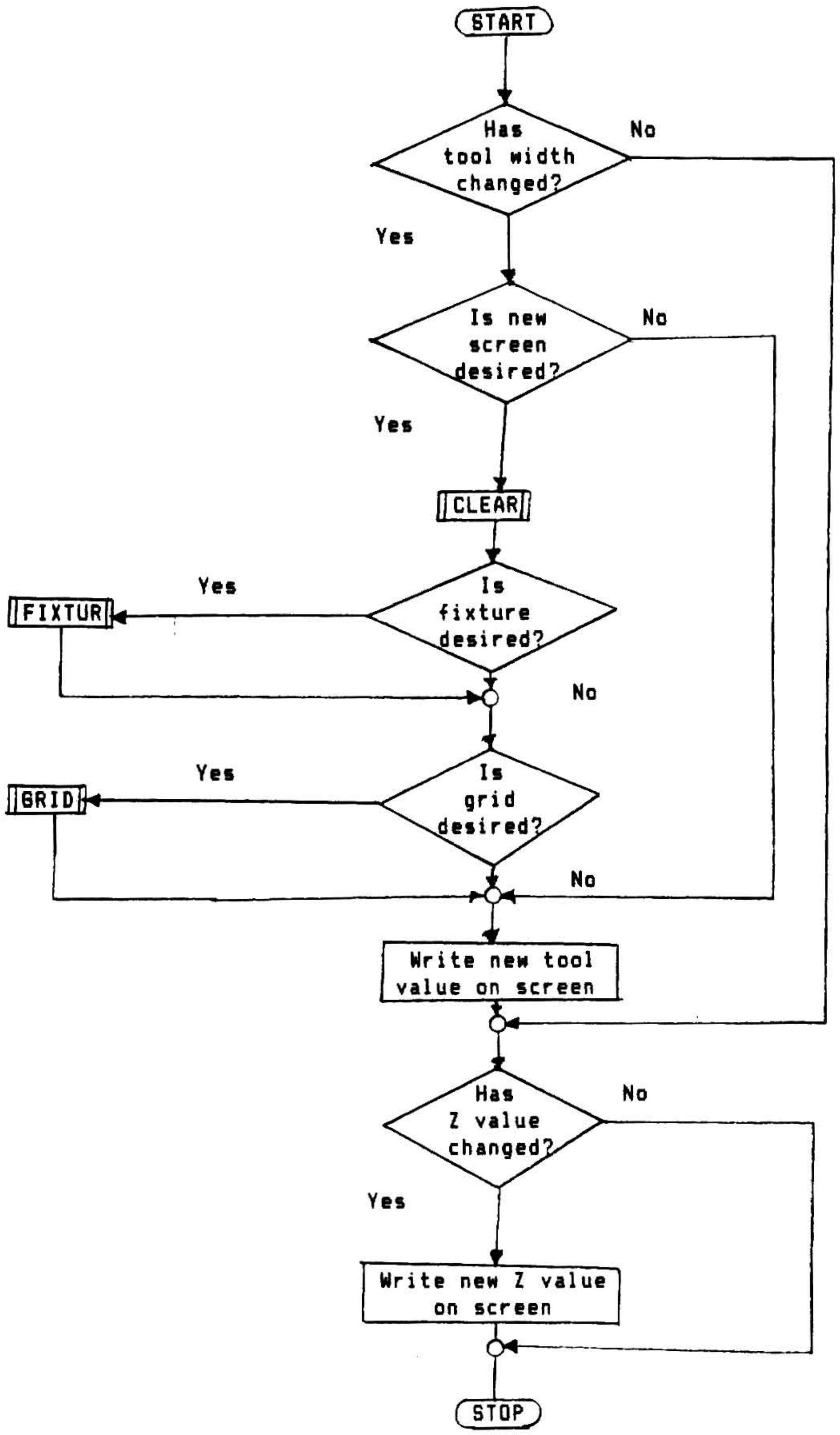
Zoom:


START
Mark diagonal elements of segment
Change to nearest square
Change window and viewport for enlarging
Is grid desired?
Yes
GRID
No
Read initial data lines
Convert to GDU for plotting
TOOL
Read next data line
Convert to GDU for plotting
Is Gcode 1 or 2
Yes
STLINE
CURVE
No
EOF?
Yes
STOP

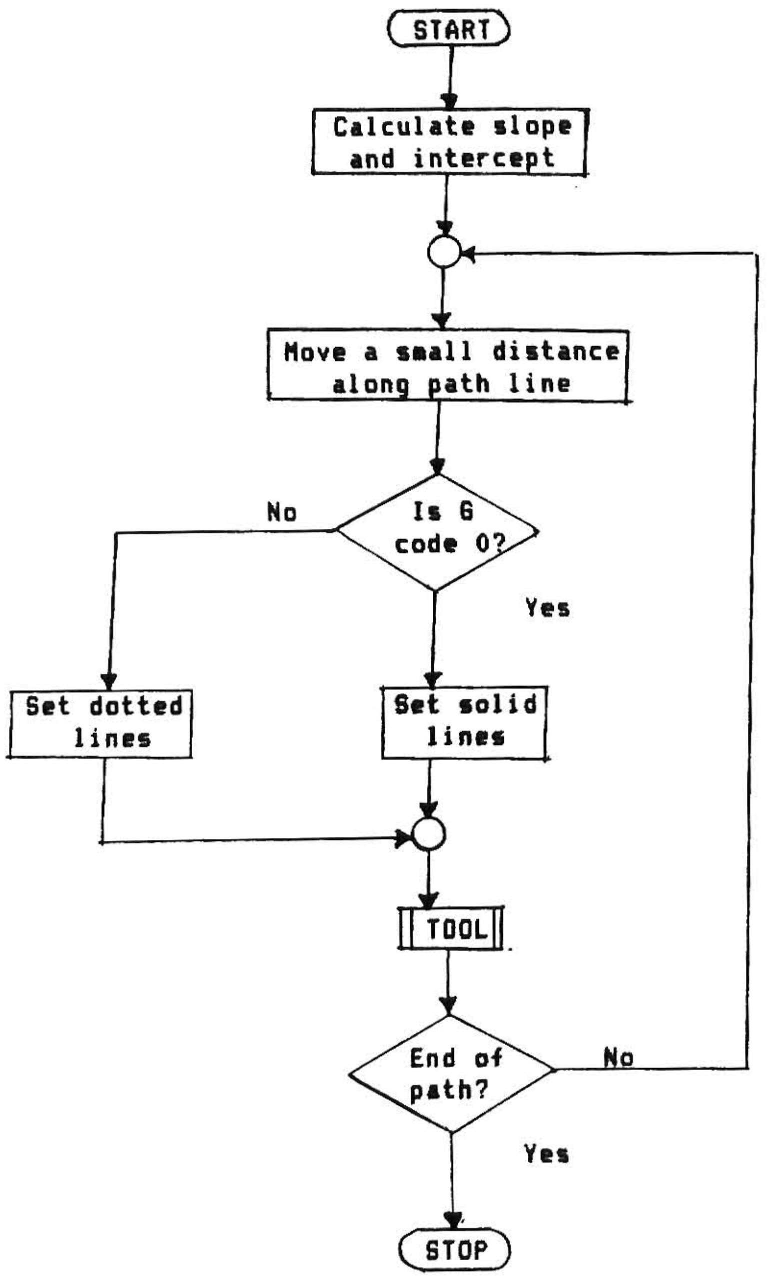
Option:


START
Enter starting data line
Enter ending data line
Is new screen desired during tool change?
Yes
Change default value
No
Is grid required?
Yes
Change default value
No
Is fixtures required?
No
Yes
Enter number of fixtures
CLEAR
GRID
Mark 4 corners of fixture
Draw fixture
More fixture?
Yes
No
STOP

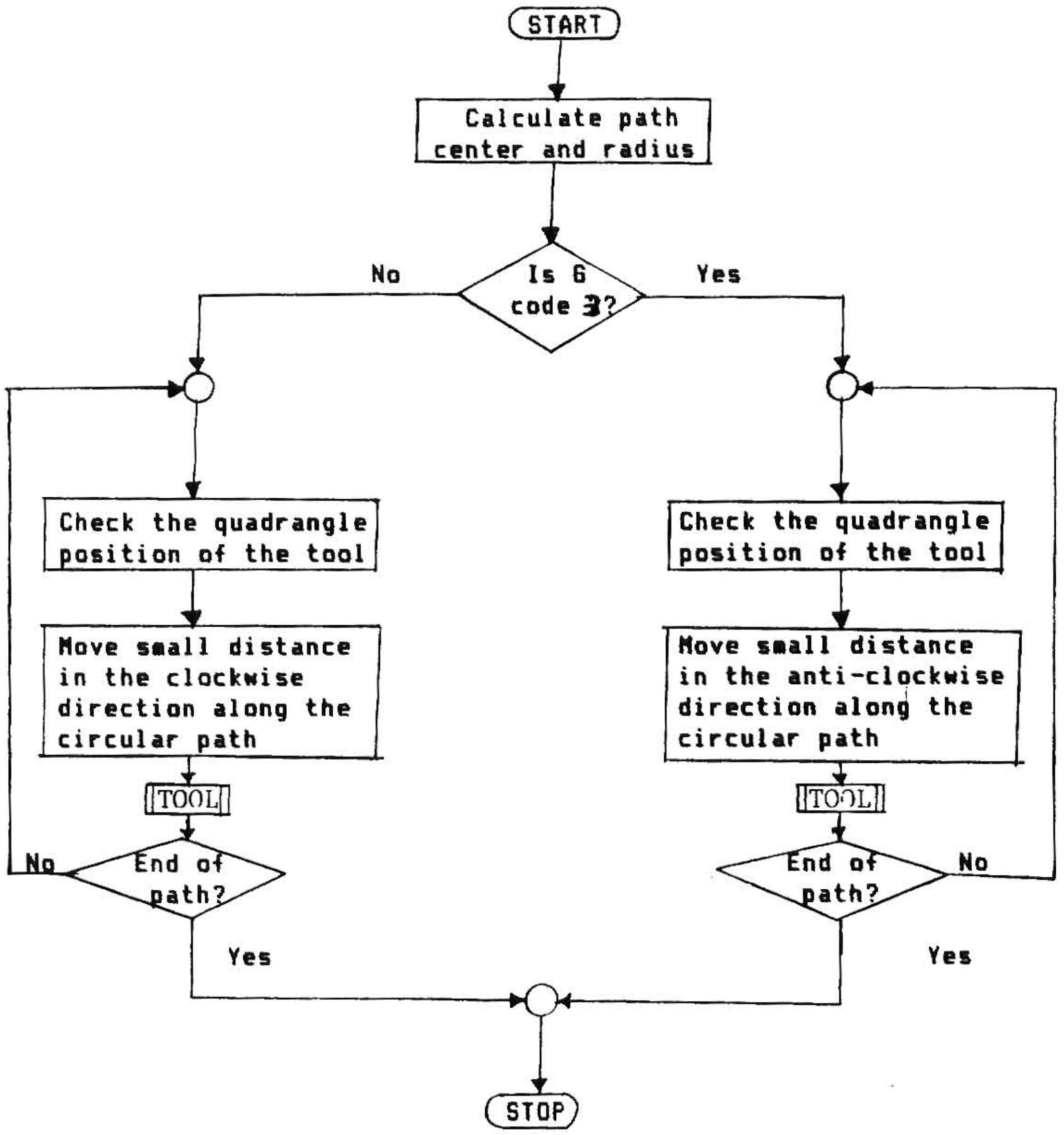
Txtwrt:


START
Has tool width changed?
No
Yes
Is new screen desired?
No
Yes
CLEAR
Yes
FIXTUR
Is fixture desired?
No
Yes
GRID
Is grid desired?
No
Write new tool value on screen
Has Z value changed?
No
Yes
Write new Z value on screen
STOP

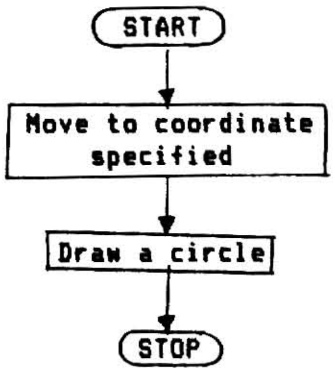
Stline:


START
Calculate slope and intercept
Move a small distance along path line
Is G code 0?
No
Yes
Set dotted lines
Set solid lines
TOOL
End of path?
No
Yes
STOP

Curve:


START
Calculate path center and radius
Is G code 3?
No
Yes
Check the quadrangle position of the tool
Move small distance in the clockwise direction along the circular path
TOOL
End of path?
No
Yes
Check the quadrangle position of the tool
Move small distance in the anti-clockwise direction along the circular path
TOOL
End of path?
No
Yes
STOP

Tool1



APPENDIX E

Program listing for Phase I

```
(*$B+*)
PROGRAM PLOT10 (INPUT,OUTPUT,TERMIN/,TERMOUT,OFFTOOL);

CONST
  MAX_IN_LINE = 10;          (* MAXIMUM SET OF CODES IN A LINE *)
  MAX_LINES   = 9999;        (* MAXIMUM NUMBER OF STATEMENTS *)
  MAX_OFFSET  = 7;           (* MAXIMUM NUMBER OF OFFSETS *)
  MAX_TOOL    = 7;           (* MAXIMUM NUMBER OF TOOL POSITIONS *)

TYPE
  ALPHABETS = 'A'..'Z';

  CODES = RECORD             (* RECORD TO READ IN EACH SET OF CODE *)
    LETTER : CHAR;           (* THE ALPHABET PORTION *)
    NUMBER : INTEGER         (* THE NUMERICAL PORTION *)
  END;

  COUNTER = RECORD           (* RECORD TO STORE THE INDEX *)
    LINE_COUNT : INTEGER;    (* STORES THE COUNT OF SET OF CODES/LINE *)
    STMT_COUNT : INTEGER     (* STORES THE STATEMENT POSITION *)
  END;

  REC1_ARRAY  = ARRAY (.1..MAX_IN_LINE.) OF CODES;
  REC2_ARRAY  = ARRAY (.1..MAX_LINES.) OF COUNTER;
  XYZ_ARRAY   = ARRAY (.'X'..'Z'.) OF INTEGER;
  IJK_ARRAY   = ARRAY (.'I'..'K'.) OF INTEGER;
  IZ_ARRAY    = ARRAY (.'I'..'Z'.) OF BOOLEAN;
  OFSET_ARRAY = ARRAY (.0..MAX_OFFSET.) OF INTEGER;
  TOOL_ARRAY  = ARRAY (.1..MAX_TOOL.) OF INTEGER;
```

```
VAR
  CALLTO        : INTEGER;      (* P VALUE DURING SUBROUTINE CALL *)
  CENTER        : IJK_ARRAY;    (* VALUE FOR I, J AND K CODES *)
  CODE          : RECT_ARRAY;   (* TO READ IN EACH CODE SET *)
  DWELL         : INTEGER;      (* P VALUE DURING FIXED CYCLE OPERATION *)
  ERROR         : BOOLEAN;      (* ERROR CHECKING *)
  FEED          : INTEGER;      (* THE FEED RATE DURING REGULAR OPERATION *)
  FIXED_FEED    : INTEGER;      (* THE FEED RATE DURING FIXED CYCLE OPERTN*)
  GOCODE        : INTEGER;      (* SHOWS WHETHER G00 OR G01 IS USED *)
  G23CODE       : INTEGER;      (* CIRCULAR MOVEMENT CODE - G02 OR G03 *)
  G4CODE        : INTEGER;      (* SHOW USE OF OFFSET - G45 TO G48 *)
  G8CODE        : INTEGER;      (* FIXED CYCLE OPERATION CODE - G80 TO G89*)
  G9CODE        : INTEGER;      (* SHOWS WHETHER G90 OR G91 IS USED *)
  HTOFFSET      : BOOLEAN;      (* OFFSET IS FOR HEIGHT *)
  HOMCOUNT      : XYZ_ARRAY;    (* NUMBER OF TIMES ORIGIN IS REDEFINED *)
  HOME          : XYZ_ARRAY;    (* MACHINE HOME POSITION W.R.T. ORIGIN *)
  HOMEPOS       : BOOLEAN;      (* BOOLEAN SET ON FOR G28 CODE *)
  INDEX         : REC2_ARRAY;   (* INDEX CONTAINING STMT # AND # OF CODES*)
  LETCODE       : CHAR;         (* TEMP ASSIGN FOR EACH LETTER CODE READ *)
  LINE          : INTEGER;      (* VALUE PART OF N *)
  LNROFFSET     : BOOLEAN;      (* OFFSET IS FOR LINEAR INTERPOLATION *)
  M3CODE        : INTEGER;      (* SPINDLE ROTATION - M30 TO M33 *)
  NEG_ZERO      : IZ_ARRAY;     (* BOOLEAN FOR NEGATIVE ZEROS *)
  NUMCODE       : INTEGER;      (* TEMP ASSIGN FOR EACH NUMBER CODE READ *)
  OFFCODE       : INTEGER;      (* STORE THE VALUE OF THE LAST OFFSET USED*)
  OFFSET        : OFSET_ARRAY;  (* CONTAINS THE OFFSET VALUES *)
  OFFTOOL       : TEXT;         (* STORES TOOLWIDTH AND OFFSET VALUES *)
  ORIGIN        : XYZ_ARRAY;    (* THE ORIGIN W.R.T. PART BOTTOM LHS *)
  ORIGIN_CHECK: BOOLEAN;        (* BOOLEAN SET ON FOR M92 CODE *)
  PAUSE         : BOOLEAN;      (* BOOLEAN SET ON FOR M00 CODE (MID STOP) *)
  POSITION      : XYZ_ARRAY;    (* POSITION OF TOOL IN X, Y AND Z AXIS *)
  RAPID         : INTEGER;      (* R VALUE DURING FIXED CYCLE OPERATION *)
  REVX          : BOOLEAN;      (* TRUE WHEN REVERSE X IS TRUE *)
  REVY          : BOOLEAN;      (* TRUE WHEN REVERSE Y IS TRUE *)
  ROUTINE       : BOOLEAN;      (* BOOLEAN SET ON FOR M98 AND OFF FOR M99 *)
  SHIFT         : XYZ_ARRAY;    (* THE X, Y AND Z VALUE FOR THAT LINE *)
  STMT          : INTEGER;      (* KEEPS TRACK OF THE STATEMENT COUNT *)
  STOP          : BOOLEAN;      (* BOOLEAN FOR M02 CODE FOR END OF PROGRAM*)
  TERMIN        : TEXT;         (* INTERACTIVE INPUT FROM TERMINAL *)
  TERMOUT       : TEXT;         (* OUTPUT TO TERMINAL *)
  TIMES         : INTEGER;      (* STORES THE L VALUE *)
  TOOL          : TOOL_ARRAY;   (* CONTAINS THE TOOLWIDTH OF ALL TOOLS *)
  TOOL_INDEX    : BOOLEAN;      (* BOOLEAN SET ON FOR M06 CODE *)
  TOOLPOS       : INTEGER;      (* CURRENT TOOLS POSITION IN MACHINE *)
  TOOLWIDTH     : INTEGER;      (* TOOLWIDTH OF THE CURRENT TOOL *)
```

```
(*****************************************************************
*                       PROCEDURE CREATE_INDEX                   *
*****************************************************************)

(*   THIS PROCEDURE CREATES THE INDEX FOR THE POSITION OF THE STATEMENT
      WITH RESPECT TO THE 'N' VALUE IN EACH LINE                      *)


PROCEDURE CREATE_INDEX (VAR ONE,STMT,LINE : INTEGER);

BEGIN

  IF (NOT ERROR) THEN
  BEGIN
    INDEX(.ONE.).STMT_COUNT := STMT;  (* UPDATE THE STATEMENT POSITION *)
    INDEX(.ONE.).LINE_COUNT := LINE;  (* STORE NUMBER OF CODES IN LINE *)
  END

END;
```

```
(**********************************************************************
*                          FUNCTION NUMERIC                          *
**********************************************************************)

(* THIS BOOLEAN FUNCTION CHECKS WHETHER A CHARACTER IS A NUMBER OR NOT *)


FUNCTION NUMERIC

(VAR
  CH   : CHAR) : BOOLEAN;

BEGIN

  CASE CH OF
    '0'..'9' : NUMERIC := TRUE;
    OTHERWISE  NUMERIC := FALSE;
  END

END;
```

```
(***********************************************************************
*                         PROCEDURE LEXICAL_READ                       *
***********************************************************************)

PROCEDURE LEXICAL_READ

(* THIS PROCEDURE READS NUMERIC CHARACTERS AND CONVERTS THEM TO NUMBERS*)

(VAR
  NXTLET   : CHAR);

VAR
  CH: CHAR;
  NEGATIVE : BOOLEAN;

BEGIN

  IF (NOT ERROR) THEN
  BEGIN
    NEGATIVE := FALSE;
    READ (CH);
    IF (CH = '-') THEN
    BEGIN
      NEGATIVE := TRUE;
      READ (CH)
    END;
    IF (NUMERIC (CH)) THEN
    BEGIN
      NUMCODE := 0;
      REPEAT
        NUMCODE := 10 * NUMCODE + ORD(CH) - ORD('0');
        READ (CH);
      UNTIL (NOT NUMERIC (CH));
      IF ((NUMCODE = 0) AND (NEGATIVE)) THEN
        NEG_ZERO(.NXTLET.) := TRUE
      ELSE IF (NEGATIVE) THEN
        NUMCODE := -NUMCODE;
      NXTLET := CH
    END
    ELSE
    BEGIN
      WRITELN (TERMOUT,' ERROR ** ENCOUNTERED IN STATEMENT NUMBER '
      ,CODE(.1.).NUMBER:4);
      WRITELN (TERMOUT,' DIGIT EXPECTED BUT ',CH:1,' FOUND INSTEAD');
      WRITELN (' ERROR ** ENCOUNTERED IN STATEMENT NUMBER ',CODE(.1.).NUMBER:4);
      WRITELN (' DIGIT EXPECTED BUT ',CH:1,' FOUND INSTEAD');
      ERROR := TRUE
    END
 END

END;
```

```
(************************************************************************
*                      PROCEDURE READ_DATA                              *
************************************************************************)

(* THIS PROCEDURE READS THE CODES, CHECKS FOR ERRORS AND MAKES AN INDEX*)

PROCEDURE READ_DATA;

VAR
  BLANK : CHAR;                (* READ INTIAL BLANK AND EOF MARKER *)
  NUMBR : INTEGER;
  NXTLET : CHAR;

BEGIN
  REWRITE (TERMOUT);
  RESET (TERMIN);
  STMT := 0;                   (* INITIALIZE THE STATEMENT COUNT *)
  ERROR := FALSE;              (* INITIALIZE THE DATA ERROR TO FALSE *)
  WHILE ((NOT EOF) AND (NOT ERROR)) DO    (* MARKER FOR EOF *)
  BEGIN
    LINE  := 1;                (* INITIALIZE THE CODE COUNT FOR EACH LINE*)
    STMT  := STMT + 1;         (* UPDATE STATEMENT COUNT *)
    READ (BLANK);              (* READ THE FIRST BLANK IN EACH LINE *)
    IF (BLANK <> ' ') THEN
    BEGIN
      WRITELN (TERMOUT,' ERROR ** ENCOUNTERED IN STATEMENT ',STMT:4);
      WRITELN (TERMOUT,' FIRST COLUMN IS NOT BLANK');
      WRITELN (' ERROR ** FIRST COLUMN IS NOT BLANK');
      ERROR := TRUE
    END;
    READ (CODE(.LINE.).LETTER);  (* READ THE LETTER OF FIRST CODE*)
    IF (CODE(.LINE.).LETTER <> 'N') THEN
    BEGIN
      WRITELN (TERMOUT,' ERROR ** ENCOUNTERED IN STATEMENT ',STMT:4);
      WRITELN (TERMOUT,' SECOND COLUMN IS NOT A "N"');
      WRITELN (' ERROR ** SECOND COLUMN IS NOT A "N"');
      ERROR := TRUE
    END
    ELSE
    BEGIN
      WHILE ((CODE(.LINE.).LETTER <> ' ') AND (NOT ERROR)) DO
      BEGIN                    (* EOLN MARKER *)
        LEXICAL_READ (NXTLET);
            (* READ THE NUMBER PART OF CODE AND THE NEXT LETTER *)
        CODE (.LINE.).NUMBER := NUMCODE;
        LINE := LINE + 1;      (* UPDATE CODE COUNT FOR THAT LINE *)
        CODE(.LINE.).LETTER := NXTLET
      END;
      LINE := LINE - 1;       (* REDUCE CODE COUNT DUE TO OFF-BY-ONE ERROR*)
      CREATE_INDEX (CODE(.1.).NUMBER,STMT,LINE);
              (* MAKE AN INDEX OF THE POSITION OF EACH STATEMENT NO.*)
      READLN;                  (* GO TO NEXT LINE *)
    END
  END
END;
```

```
(***********************************************************************
*                         PROCEDURE CHECKF                            *
***********************************************************************)


(*     THIS PROCEDURE ASSIGNS THE FEED FOR NORMAL AND FIXED CYCLES     *)

PROCEDURE CHECKF;

BEGIN

  IF (NOT ERROR) THEN
    IF (G8CODE = 80) THEN
      FEED := NUMCODE
    ELSE
      FIXED_FEED := NUMCODE

END;
```

```
(*************************************************************************
*                          PROCEDURE CHECKG                              *
*************************************************************************)

(*   THIS PROCEDURE CHECKS THE NUMBERS ASSOCIATED WITH THE G CODES   *)

PROCEDURE CHECKG;

BEGIN

  IF (NOT ERROR) THEN
  BEGIN
    IF ((NUMCODE = 90) OR (NUMCODE = 91)) THEN
      G9CODE := NUMCODE
    ELSE IF ((NUMCODE = 0) OR (NUMCODE = 1)) THEN
    BEGIN
      G0CODE := NUMCODE;
      IF (G8CODE <> 80) THEN
      BEGIN
        WRITELN (TERMOUT, ' WARNING ** G00 OR G01 OVERCOMES FIXED OPERATION');
        WRITELN (' WARNING ** G00 OR G01 OVERCOMES FIXED OPERATION');
        G8CODE := 80;
        FIXED_FEED := 0;
        RAPID := 0;
        DWELL := 0
      END
    END
    ELSE IF ((NUMCODE = 2) OR (NUMCODE = 3)) THEN
      G23CODE := NUMCODE
    ELSE IF ((NUMCODE >= 45) AND (NUMCODE <= 48)) THEN
      G4CODE := NUMCODE
    ELSE IF (NUMCODE = 92) THEN
      ORIGIN_CHECK := TRUE
    ELSE IF ((NUMCODE > 80) AND (NUMCODE <= 89)) THEN
      G8CODE := NUMCODE
    ELSE IF (NUMCODE = 80) THEN
    BEGIN
      G8CODE := NUMCODE;
      FIXED_FEED := 0;
      RAPID := 0;
      DWELL := 0
    END
    ELSE IF (NUMCODE = 28) THEN
      HOMEPOS := TRUE
    ELSE
    BEGIN
      ERROR := TRUE;
      STOP  := TRUE;
      WRITELN (TERMOUT,' ERROR ** ENCOUNTERED IN STATEMENT NUMBER',LINE:4);
      WRITELN (TERMOUT,' NUMBER ',NUMCODE:6,' NOT ASSOCIATED WITH G CODE');
      WRITELN (' ERROR ** NUMBER ',NUMCODE:6,' NOT ASSOCIATED WITH G CODE ')
    END
  END
END;
```

```
(**********************************************************************
*                          PROCEDURE CHECKM                          *
**********************************************************************)

(*    THIS PROCEDURE CHECKS THE CODES ASSOCIATED WITH THE M CODES     *)

PROCEDURE CHECKM;
BEGIN
  IF (NOT ERROR) THEN
  BEGIN
    IF (NUMCODE = 6) THEN
      TOOL_INDEX := TRUE
    ELSE IF ((NUMCODE >= 30) AND (NUMCODE <= 33)) THEN
      M3CODE := NUMCODE
    ELSE IF (NUMCODE = 98) THEN
      ROUTINE := TRUE
    ELSE IF (NUMCODE = 99) THEN
      ROUTINE := FALSE
    ELSE IF (NUMCODE = 0) THEN
      PAUSE := TRUE
    ELSE IF (NUMCODE = 2) THEN
      STOP := TRUE
    ELSE
    BEGIN
      ERROR := TRUE;
      WRITELN (TERMOUT,' ERROR ** ENCOUNTERED IN STATEMENT NUMBER',LINE:4);
      WRITELN (TERMOUT,' NUMBER ',NUMCODE:6,' NOT ASSOCIATED WITH M CODE');
      WRITELN (' ERROR ** NUMBER ',NUMCODE:6,' NOT ASSOCIATED WITH M CODE ')
    END
  END
END;
```

```
(****************************************************************************
*                           PROCEDURE CHECKP                                *
****************************************************************************)

(* THIS PROCEDURE CHECKS FOR DWELL AND STATEMENT # FOR SUBROUTINE CALL *)

PROCEDURE CHECKP;
BEGIN
  IF (NOT ERROR) THEN
  BEGIN
    IF (G8CODE > 80) THEN
      DWELL := NUMCODE
    ELSE IF (ROUTINE) THEN
      CALLTO := NUMCODE
    ELSE
    BEGIN
      ERROR := TRUE;
      WRITELN (TERMOUT,' ERROR ** ENCOUNTERED IN STATEMENT NUMBER',LINE:4);
      WRITELN (TERMOUT,' P CODE USED W/O A SUBROUTINE OR FIXED CYCLE CALL');
      WRITELN (' ERROR ** P CODE USED WITHOUT A SUBROUTINE OR FIXED CYCLE CALL')
    END
  END
END;
```

```
(*********************************************************************
*                         PROCEDURE CHECKL                          *
*********************************************************************)

(*  THIS PROCEDURE FINDS THE NUMBER OF TIMES A STATEMENT IS REPEATED   *)

PROCEDURE CHECKL;

BEGIN
  IF (NOT ERROR) THEN
    TIMES := NUMCODE
END;
```

```
(*****************************************************************************
*                              PROCEDURE CHECKR                             *
*****************************************************************************)

(* THIS PROCEDURE CHECKS THE RAPID MOVEMENT IN FIXED CYCLE OPERATIONS *)

PROCEDURE CHECKR;

BEGIN
  IF (NOT ERROR) THEN
  BEGIN
    IF (G8CODE > 80) THEN
      RAPID := NUMCODE
    ELSE
    BEGIN
      ERROR := TRUE;
      WRITELN (TERMOUT,' ERROR ** ENCOUNTERED IN STATEMENT NUMBER',LINE:4);
      WRITELN (TERMOUT,' R CODE BEFORE FIXED CYCLE WAS STARTED');
      WRITELN (' ERROR ** R CODE ENCOUNTERED BEFORE FIXED CYCLE WAS STARTED')
    END
  END
END;
```

```
(***********************************************************************
*                            PROCEDURE CHECKD                            *
***********************************************************************)

(*        THIS PROCEDURE CHECKS FOR LINEAR OFFSETSVALUES              *)

PROCEDURE CHECKD;

BEGIN
  IF (NOT ERROR) THEN
  BEGIN
    OFFCODE := OFFSET(.NUMCODE.);
    LNROFFSET := TRUE;
    HTOFFSET  := FALSE
  END
END;
```

```
(***********************************************************************
*                        PROCEDURE CHECKH                              *
***********************************************************************)

(*          THIS PROCEDURE CHECKS FOR HEIGHT OFFSETSVALUES           *)

PROCEDURE CHECKH;

BEGIN
  IF (NOT ERROR) THEN
  BEGIN
    OFFCODE := OFFSET(.NUMCODE.);
    HTOFFSET := TRUE;
    LNROFFSET := FALSE
  END
END;
```

```
(***********************************************************************
*                         PROCEDURE CHECKXYZ                           *
***********************************************************************)

(*   THIS PROCEDURE FINDS THE CHANGE IN THE X, Y OR Z COORDINATES      *)

PROCEDURE CHECKXYZ;

BEGIN
  IF (NOT ERROR) THEN
  BEGIN
    SHIFT(.LETCODE.) := NUMCODE;
    IF (NOT ORIGIN_CHECK) THEN
    BEGIN
      IF ((LETCODE = 'X') AND (REVX)) THEN
        SHIFT(.'X'.) := -SHIFT(.'X'.);
      IF ((LETCODE = 'Y') AND (REVY)) THEN
        SHIFT(.'Y'.) := -SHIFT(.'Y'.);
      IF (NEG_ZERO(.'X'.) AND REVX) THEN
        NEG_ZERO(.'X'.) := FALSE;
      IF (NEG_ZERO(.'Y'.) AND REVY) THEN
        NEG_ZERO(.'Y'.) := FALSE;
      IF ((LETCODE = 'X') OR (LETCODE = 'Y') OR
          ((G8CODE = 80) AND (LETCODE = 'Z'))) THEN
      BEGIN
        IF (G9CODE = 90) THEN
          POSITION(.LETCODE.) := SHIFT(.LETCODE.) + ORIGIN(.LETCODE.)
        ELSE IF (G9CODE = 91) THEN
          POSITION(.LETCODE.) := POSITION(.LETCODE.) + SHIFT(.LETCODE.)
      END
    END
  END
END;
```

```
(*************************************************************************
*                          PROCEDURE CHECKIJK                            *
*************************************************************************)

(* THIS PROCEDURE FINDS THE DIST. OF CENTER OF CIRCLE FROM CURRENT POS.*)

PROCEDURE CHECKIJK;

BEGIN
  IF (NOT ERROR) THEN
    IF ((G23CODE = 2) OR (G23CODE = 3)) THEN
      CENTER(.LETCODE.) := NUMCODE
    ELSE
    BEGIN
      ERROR := TRUE;
      WRITELN (TERMOUT,' ERROR ** ENCOUNTERED IN STATEMENT NUMBER',LINE:4);
      WRITELN (TERMOUT,LETCODE,' ENCOUNTERED WITHOUT G02 OR G03');
      WRITELN (' ERROR ** ',LETCODE,' ENCOUNTERED WITHOUT G02 OR G03')
    END
END;
```

```
(*****************************************************************************
*                           PROCEDURE CHECK_CODES                            *
*****************************************************************************)

(* THIS PROCEDURE GOES TO DIFFERENT PROCEDURES DEPENDING ON LETTER CODE*)

PROCEDURE CHECK_CODES;

BEGIN
  IF (NOT ERROR) THEN
  BEGIN
    IF (LETCODE = 'F') THEN
      CHECKF
    ELSE IF (LETCODE = 'G') THEN
      CHECKG
    ELSE IF (LETCODE = 'M') THEN
      CHECKM
    ELSE IF (LETCODE = 'P') THEN
      CHECKP
    ELSE IF (LETCODE = 'L') THEN
      CHECKL
    ELSE IF (LETCODE = 'R') THEN
      CHECKR
    ELSE IF (LETCODE = 'D') THEN
      CHECKD
    ELSE IF (LETCODE = 'H') THEN
      CHECKH
    ELSE IF ((LETCODE = 'X') OR (LETCODE = 'Y') OR (LETCODE = 'Z')) THEN
      CHECKXYZ
    ELSE IF ((LETCODE = 'I') OR (LETCODE = 'J') OR (LETCODE = 'K')) THEN
      CHECKIJK
    ELSE
    BEGIN
      ERROR := TRUE;
      WRITELN (TERMOUT,' ERROR ** ENCOUNTERED IN STATEMENT NUMBER',LINE:4);
      WRITELN (TERMOUT,LETCODE,' ENCOUNTERED; NOT PROPER CODE');
      WRITELN (' ERROR ** ',LETCODE,' USED; NOT PROPER CODE')
    END
  END
END;
```

```
(*************************************************************************
*                          PROCEDURE PRINT                               *
*************************************************************************)

(*   THIS PROCEDURE PRINTS THE COORDINATES FOR PLOTTING AND FOR USER  *)

PROCEDURE PRINT;

VAR
  I : INTEGER;                  (* LOCAL VARIABLE FOR LOOPING *)
  J : ALPHABETS;                (* LOCAL VARIABLE FOR LOOPING *)

BEGIN
  IF (G23CODE <> 0) THEN
    WRITE (CODE(.1.).NUMBER:4, TOOLWIDTH:6, G23CODE:2)
  ELSE
    WRITE (CODE(.1.).NUMBER:4, TOOLWIDTH:6, GOCODE:2);
  FOR J := 'X' TO 'Z' DO
    WRITE (POSITION(.J.):7);
  FOR J := 'I' TO 'K' DO
    IF (CENTER(.J.) = MAXINT) THEN
      CENTER(.J.) := 0;
  WRITELN (CENTER(.'I'.):7,CENTER(.'J'.):7,CENTER(.'K'.):7)
END;
```

```
(****************************************************************************
*                        PROCEDURE CHANGE_TOOL                               *
****************************************************************************)

(*         THIS PROCEDURE INDEXES THE TOOL HOLDING DEVICE              *)

PROCEDURE CHANGE_TOOL;

BEGIN
  IF (NOT ERROR) THEN
  BEGIN
    TOOLPOS := TOOLPOS + 1;
    M3CODE := 30;
    IF (TOOLPOS > MAX_TOOL) THEN
      TOOLPOS := 1;
    TOOLWIDTH := TOOL(.TOOLPOS.);
    PRINT
  END
END;
```

```
(*************************************************************************
*                        PROCEDURE CHANGE_ORIGIN                          *
*************************************************************************)

(* THIS PROCEDURE CHANGES THE ORIGIN FOR FURTHER CALCULATION OF COORD. *)

PROCEDURE CHANGE_ORIGIN;

VAR
  I : INTEGER;                (* LOCAL VARIABLE FOR LOOPING *)

BEGIN
  IF ((NOT ERROR) AND (G9CODE = 90)) THEN
  BEGIN
    ERROR := TRUE;
    WRITELN (TERMOUT,' ERROR ** ENCOUNTERED IN STATEMENT NUMBER',LINE:4);
    WRITELN (TERMOUT,' ORIGIN CHANGE IS POSSIBLE ONLY IN ABSOLUTE MODE');
    WRITELN (' ERROR ** ORIGIN CHANGE IS POSSIBLE ONLY IN ABSOLUTE MODE')
  END;
  IF (NOT ERROR) THEN
  BEGIN
    FOR I := 1 TO INDEX(.LINE.).LINE_COUNT DO
    BEGIN
      CASE CODE(.I.).LETTER OF
        'X': BEGIN
               IF (HOMCOUNT(.'X'.) = 1) THEN
               BEGIN
                 HOME(.'X'.) := CODE(.I.).NUMBER;
                 HOMCOUNT(.'X'.) := HOMCOUNT(.'X'.) + 1
               END;
               ORIGIN(.'X'.) := POSITION(.'X'.) - SHIFT(.'X'.)
             END;
        'Y': BEGIN
               IF (HOMCOUNT(.'Y'.) = 1) THEN
               BEGIN
                 HOME(.'Y'.) := CODE(.I.).NUMBER;
                 HOMCOUNT(.'Y'.) := HOMCOUNT(.'Y'.) + 1
               END;
               ORIGIN(.'Y'.) := POSITION(.'Y'.) - SHIFT(.'Y'.)
             END;
        'Z': BEGIN
               IF (HOMCOUNT(.'Z'.) = 1) THEN
               BEGIN
                 HOME(.'Z'.) := CODE(.I.).NUMBER;
                 HOMCOUNT(.'Z'.) := HOMCOUNT(.'Z'.) + 1
               END;
               ORIGIN(.'Z'.) := POSITION(.'Z'.) - SHIFT(.'Z'.)
             END;
        OTHERWISE
      END
    END;
    PRINT
  END
END;
```

```
(*************************************************************************
*                          PROCEDURE GO_HOME                             *
*************************************************************************)

(* THIS PROCEDURE TAKES THE TOOL/MACHINE TO THE MACHINE HOME POSITION *)

PROCEDURE GO_HOME;

VAR
  I : INTEGER;                (* LOCAL VARIABLE FOR LOOPING *)

BEGIN
  IF (NOT ERROR) THEN
  BEGIN
    FOR I := 1 TO INDEX(.LINE.).LINE_COUNT DO
      CASE CODE(.I.).LETTER OF
        'X'       : POSITION(.'X'.) := HOME(.'X'.);
        'Y'       : POSITION(.'Y'.) := HOME(.'Y'.);
        'Z'       : POSITION(.'Z'.) := HOME(.'Z'.);
        OTHERWISE
      END;
    PRINT
  END
END;
```

```
(*****************************************************************************
*                         PROCEDURE CIRCULAR_MOTION                          *
*****************************************************************************)

(*        THIS PROCEDURE MOVES THE TOOL FOR CIRCULAR MOVEMENT              *)

PROCEDURE CIRCULAR_MOTION;

VAR
  I : ALPHABETS;              (* LOCAL VARIABLE FOR LOOPING *)
  RADIUS : REAL;
  CHORD  : REAL;
  ANGLE  : REAL;
  PI_DIV_2 : REAL;        (* VALUE OF PI DIVIDED BY 2 *)

BEGIN
  IF (NOT ERROR) THEN
  BEGIN
    IF (((SHIFT(.'X'.) <> MAXINT) OR (SHIFT(.'Y'.) <> MAXINT))
    AND (HTOFFSET) AND (G4CODE <> 0)) THEN
    BEGIN
      ERROR := TRUE;
      WRITELN (TERMOUT, ' ERROR ** ENCOUNTERED IN STATEMENT NUMBER ',LINE:4);
      WRITELN (TERMOUT, ' HEIGHT OFFSET USED WITH X AND Y');
      WRITELN (' ERROR ** HEIGHT OFFSET USED WITH X AND Y')
    END
    ELSE IF ((SHIFT(.'X'.) = MAXINT) AND (SHIFT(.'Y'.) = MAXINT) AND (LNROFFSET)
    AND (SHIFT (.'Z'.) <> MAXINT) AND (G4CODE <> 0)) THEN
    BEGIN
      ERROR := TRUE;
      WRITELN (TERMOUT, ' ERROR ** ENCOUNTERED IN STATEMENT NUMBER ',LINE:4);
      WRITELN (TERMOUT, ' LINEAR OFFSET USED WITH Z');
      WRITELN (' ERROR ** LINEAR OFFSET USED WITH Z')
    END
    ELSE IF ((FEED = 0) OR (M3CODE = 30)) THEN
    BEGIN
      ERROR := TRUE;
      WRITELN (TERMOUT,' ERROR ** ENCOUNTERED IN STATEMENT NUMBER',LINE:4);
      WRITELN (TERMOUT,' SPINDLE NOT ROTATING OR FEED IS 0');
      WRITELN (' ERROR ** SPINDLE NOT ROTATING OR FEED IS 0')
    END
    ELSE IF ((CENTER(.'I'.)=MAXINT) AND (CENTER(.'J'.)=MAXINT) AND
            (CENTER(.'K'.)=MAXINT)) THEN
    BEGIN
      ERROR := TRUE;
      WRITELN (TERMOUT,' ERROR ** ENCOUNTERED IN STATEMENT NUMBER',LINE:4);
      WRITELN (TERMOUT,' CIRCLE WITHOUT A VALUE FOR I, J OR K');
      WRITELN (' ERROR ** CIRCLE CALL WITHOUT A VALUE FOR I, J OR K')
    END
```

```
ELSE       (* CHECK WHETHER ARC IS > 90 DEGREES *)
BEGIN
  PI_DIV_2 := 2.0 * ARCTAN(1.0);
  RADIUS := 0.0;
  FOR I := 'I' TO 'K' DO
    IF (CENTER(.I.) <> MAXINT) THEN
      RADIUS := RADIUS + SQR(CENTER(.I.));
  RADIUS := SQRT(RADIUS);
  CHORD := 0.0;
  FOR I := 'X' TO 'Z' DO
    IF (SHIFT(.I.) <> MAXINT) THEN
      CHORD := CHORD + SQR(SHIFT(.I.));
  CHORD := SQRT(CHORD);
  IF ((2.0*RADIUS) = CHORD) THEN
    ANGLE := 4.0*ARCTAN(1.0)
  ELSE
    ANGLE := 2.0 * ARCTAN(CHORD/SQRT(4.0*SQR(RADIUS) - SQR(CHORD)));
  IF (ANGLE > PI_DIV_2) THEN
  BEGIN
    ERROR := TRUE;
    WRITELN (TERMOUT,' ERROR ** ENCOUNTERED IN STATEMENT NUMBER',LINE:4);
    WRITELN (TERMOUT,' CIRCULAR ARC GREATER THAN 90 DEGREES');
    WRITELN (' ERROR ** CIRCULAR ARC GREATER THAN 90 DEGREES')
  END
ND;
F (NOT ERROR) THEN
EGIN
 FOR I := 'X' TO 'Z' DO
 BEGIN
   IF (SHIFT(.I.) <> MAXINT) THEN
   BEGIN
     IF (((SHIFT(.I.) < 0) OR (NEG_ZERO(.I.))) AND (G4CODE <> 0)) THEN
       OFFCODE := -OFFCODE;
     CASE (G4CODE) OF
       45        : POSITION(.I.) := POSITION(.I.) + OFFCODE;
       46        : POSITION(.I.) := POSITION(.I.) - OFFCODE;
       47        : POSITION(.I.) := POSITION(.I.) + (2*OFFCODE);
       48        : POSITION(.I.) := POSITION(.I.) - (2*OFFCODE);
       OTHERWISE
     END
   END;
   OFFCODE := ABS(OFFCODE)
 END;
```

```
      FOR I := 'I' TO 'K' DO
      BEGIN
        IF (CENTER(.I.) <> MAXINT) THEN
        BEGIN
          IF (((CENTER(.I.)<0) OR (NEG_ZERO(.I.))) AND (G4CODE<>0)) THEN
            OFFCODE := -OFFCODE;
          CASE (G4CODE) OF
            45         : CENTER(.I.) := CENTER(.I.) + OFFCODE;
            46         : CENTER(.I.) := CENTER(.I.) - OFFCODE;
            47         : CENTER(.I.) := CENTER(.I.) + (2*OFFCODE);
            48         : CENTER(.I.) := CENTER(.I.) - (2*OFFCODE);
            OTHERWISE
          END
        END;
        OFFCODE := ABS(OFFCODE)
      END;
      PRINT
    END
  END
END;
```

```
********************************************************************************
                         PROCEDURE MID_STOP                                   *
*******************************************************************************)

  THIS PROCEDURE FINDS REASON FOR STOPPING IN THE MIDDLE OF PROGRAM  *)

)CEDURE MID_STOP;

R
:NTER : CHAR;
IUM_OF_CHANGE : INTEGER;
'OS    : INTEGER;
'ALUE  : INTEGER;
       : INTEGER;


IN
F (NOT ERROR) THEN
EGIN
 M3CODE := 30;
 WRITELN (TERMOUT, ' STOP ENCOUNTERED IN THE MIDDLE OF THE PROGRAM');
 WRITELN (TERMOUT, ' DO YOU WANT TO REVERSE X OR/AND Y?   TYPE Y/N');
 READLN (TERMIN);
 READ (TERMIN, ENTER);
 IF (ENTER = 'Y') THEN
 BEGIN
   WRITELN (TERMOUT, ' TYPE X AND HIT ENTER FOR REVERSING X');
   WRITELN (TERMOUT, ' TYPE Y AND HIT ENTER FOR REVERSING Y');
   WRITELN (TERMOUT, ' TYPE B AND HIT ENTER FOR REVERSING BOTH');
   READLN (TERMIN);
   READ (TERMIN, ENTER);
   CASE (ENTER) OF
     'X' : REVX := NOT REVX;
     'Y' : REVY := NOT REVY;
     'B' : BEGIN
             REVX := NOT REVX;
             REVY := NOT REVY
           END;
     OTHERWISE
   END
END;
WRITELN (TERMOUT, ' DO YOU WANT TO CHANGE TOOLS?   TYPE Y/N');
READLN (TERMIN);
READ (TERMIN, ENTER);
IF (ENTER = 'Y') THEN
BEGIN
  WRITELN (TERMOUT, ' ENTER THE NUMBER OF TOOL CHANGES');
  READLN (TERMIN);
  READ (TERMIN, NUM_OF_CHANGE);
```

```
        FOR I := 1 TO NUM_OF_CHANGE DO
        BEGIN
          WRITELN (TERMOUT, ' ENTER TOOL POSITION AND WIDTH FOR CHANGE', I:2);
          READLN (TERMIN);
          READ (TERMIN,POS,VALUE);
          TOOL(.POS.) := VALUE
        END;
        TOOLWIDTH := TOOL(.TOOL_POS.)
      END;
      WRITELN (TERMOUT, ' DO YOU WANT TO CHANGE OFFSETS?   TYPE Y/N');
      READLN (TERMIN);
      READ (TERMIN, ENTER);
      IF (ENTER = 'Y') THEN
      BEGIN
        WRITELN (TERMOUT, ' ENTER THE NUMBER OF OFFSET CHANGES');
        READLN (TERMIN);
        READ (TERMIN, NUM_OF_CHANGE);
        FOR I := 1 TO NUM_OF_CHANGE DO
        BEGIN
          WRITELN (TERMOUT, ' ENTER OFFSET CODE AND VALUE FOR CHANGE', I:2);
          READLN (TERMIN);
          READ (TERMIN,POS,VALUE);
          OFFSET(.POS.) := VALUE
        END
      END;
      PRINT
    END
  END;
```

```
(**********************************************************************
*                         PROCEDURE FIXED_CYCLE                       *
**********************************************************************)

(*    THIS PROCEDURE MOVES THE TOOL DURING FIXED CYCLE OPERATIONS     *)

PROCEDURE FIXED_CYCLE;

VAR
  ENTER : CHAR;

BEGIN
  IF (NOT ERROR) THEN
  BEGIN
    IF (FIXED_FEED = 0) THEN
    BEGIN
      ERROR := TRUE;
      WRITELN (TERMOUT,' ERROR ** ENCOUNTERED IN STATEMENT NUMBER',LINE:4);
      WRITELN (TERMOUT,' FEED REQUIRED FOR FIXED CYCLE OPERATION');
      WRITELN (' ERROR ** FEED REQUIRED FOR FIXED CYCLE OPERATION')
    END
    ELSE IF (M3CODE = 0) THEN
    BEGIN
      ERROR := TRUE;
      WRITELN (TERMOUT,' ERROR ** ENCOUNTERED IN STATEMENT NUMBER',LINE:4);
      WRITELN (TERMOUT,' SPINDLE NOT ROTAING');
      WRITELN (' ERROR ** SPINDLE NOT ROTATING')
    END
    ELSE IF (G9CODE = 90) THEN
    BEGIN
      ERROR := TRUE;
      WRITELN (TERMOUT,' ERROR ** ENCOUNTERED IN STATEMENT NUMBER',LINE:4);
      WRITELN (TERMOUT,' FOR FIXED CYCLE OPERATIONS USE INCREMENTAL MODE');
      WRITELN (' ERROR ** FOR FIXED CYCLE OPERATIONS USE INCREMENTAL MODE');
    END;
    IF (NOT ERROR) THEN
    BEGIN
      IF (DWELL > 0) THEN
      BEGIN
        WRITELN (TERMOUT,' DWELL ENCOUNTERED. HIT ENTER TWICE TO CONTINUE');
        READLN (TERMIN);
        READ (TERMIN, ENTER)
      END;
      POSITION (.'Z'.) := POSITION(.'Z'.) + RAPID;
      IF (SHIFT(.'Z'.) <> MAXINT) THEN
      BEGIN
        POSITION(.'Z'.) := POSITION(.'Z'.) + SHIFT(.'Z'.);
        PRINT;
        POSITION(.'Z'.) := POSITION(.'Z'.) - SHIFT(.'Z'.)
      END;
      POSITION (.'Z'.) := POSITION(.'Z'.) - RAPID;
      PRINT
    END
  END
END;
```

```
(**********************************************************************
*                         PROCEDURE GO_AHEAD                          *
**********************************************************************)

(*     THIS PROCEDURE MOVES THE TOOL DURING ANY LINEAR MOVEMENT       *)

PROCEDURE GO_AHEAD;

VAR
  I : ALPHABET;

BEGIN
  IF (NOT ERROR) THEN
  BEGIN
    IF ((SHIFT(.'Y'.) <> MAXINT) AND (SHIFT(.'Z'.) <> MAXINT)) THEN
    BEGIN
      ERROR := TRUE;
      WRITELN (TERMOUT, ' ERROR ** ENCOUNTERED IN STATEMENT NUMBER ',LINE:4);
      WRITELN (TERMOUT, ' SIMULTANEOUS MOVEMENT IN Y AND Z DIRECTIONS');
      WRITELN (' ERROR ** SIMULTANEOUS MOVEMENT IN Y AND Z DIRECTIONS');
    END
    ELSE IF ((SHIFT(.'Z'.) <> MAXINT) AND (G4CODE <> 0) AND (LNROFFSET)
         AND ((SHIFT(.'X'.) <> MAXINT) OR (SHIFT(.'Y'.) <> MAXINT))) THEN
    BEGIN
      ERROR := TRUE;
      WRITELN (TERMOUT, ' ERROR ** ENCOUNTERED IN STATEMENT NUMBER ',LINE:4);
      WRITELN (TERMOUT, ' LINEAR OFFSET USED WITH Z');
      WRITELN (' ERROR ** LINEAR OFFSET USED WITH Z')
    END
    ELSE IF ((SHIFT(.'Z'.) = MAXINT) AND (G4CODE <> 0) AND (HTOFFSET)
         AND ((SHIFT(.'X'.) <> MAXINT) OR (SHIFT(.'Y'.) <> MAXINT))) THEN
    BEGIN
      ERROR := TRUE;
      WRITELN (TERMOUT, ' ERROR ** ENCOUNTERED IN STATEMENT NUMBER ',LINE:4);
      WRITELN (TERMOUT, ' LINEAR OFFSET USED WITHOUT Z');
      WRITELN (' ERROR ** LINEAR OFFSET USED WITHOUT Z')
    END
    ELSE IF ((FEED = 0) AND (M3CODE = 30) AND (GOCODE = 1)) THEN
    BEGIN
      ERROR := TRUE;
      WRITELN (TERMOUT,' ERROR ** ENCOUNTERED IN STATEMENT NUMBER',LINE:4);
      WRITELN (TERMOUT,' SPINDLE NOT ROTATING OR FEED IS 0');
      WRITELN (' ERROR ** SPINDLE NOT ROTATING OR FEED IS 0')
    END;
```

```
      IF (NOT ERROR) THEN
      BEGIN
        FOR I := 'X' TO 'Z' DO
        BEGIN
          IF (SHIFT(.I.) <> MAXINT) THEN
          BEGIN
            IF ((G4CODE <> 0) AND ((SHIFT(.I.) < 0) OR (NEG_ZERO(.I.)))) THEN
              OFFCODE := -OFFCODE;
            CASE (G4CODE) OF
              45         : POSITION(.I.) := POSITION(.I.) + OFFCODE;
              46         : POSITION(.I.) := POSITION(.I.) - OFFCODE;
              47         : POSITION(.I.) := POSITION(.I.) + (2*OFFCODE);
              48         : POSITION(.I.) := POSITION(.I.) - (2*OFFCODE);
              OTHERWISE
            END
          END;
          OFFCODE := ABS(OFFCODE)
        END;
        PRINT
      END
    END
  END;
```

```
(**********************************************************************
*                         PROCEDURE SCROLL                           *
**********************************************************************)

(*    THIS PROCEDURE SKIPS THROUGH LINES IN A FILE BEFORE READING     *)

PROCEDURE SCROLL

(VAR
   LINE_NUMBER:INTEGER);      (* NUMBER OF LINES TO SKIP BEFORE READ *)

VAR
  I : INTEGER;                (* LOCAL VARIABLE FOR LOOPING *)

BEGIN
  RESET(INPUT);
  FOR I := 1 TO LINE_NUMBER DO
    READLN
END;
```

```
(************************************************************************
*                          PROCEDURE INITIALIZE                         *
************************************************************************)

(*  THIS PROCEDURE INITIALIZES VARIABLES AT THE BEGINNING OF THE RUN   *)

PROCEDURE INITIALIZE;

VAR
  I : ALPHABETS;               (* LOCAL VARIABLE FOR LOOPING *)
  J : INTEGER;                 (* LOCAL VARIABLE FOR LOOPING *)
  ENTER : CHAR;                (* VARIABLE TO READ FROM TERMINAL *)

BEGIN
  IF (NOT ERROR) THEN
  BEGIN
    RESET(INPUT);
    STMT := 0;
    STOP := FALSE;
    FEED := 0;
    DWELL := 0;
    RAPID := 0;
    FIXED_FEED := 0;
    G9CODE := 0;
    GOCODE := 0;
    OFFCODE := 0;
    G8CODE := 80;
    CALLTO := 0;
    M3CODE := 30;
    ROUTINE:= FALSE;
    REVX := FALSE;
    REVY := FALSE;
    HTOFFSET := FALSE;
    LNROFFSET := FALSE;
    FOR I := 'X' TO 'Z' DO
    BEGIN
      HOMCOUNT(.I.) := 1;
      HOME(.I.) := 0
    END;
    FOR I := 'X' TO 'Z' DO
    BEGIN
      POSITION(.I.) := 0;
      ORIGIN(.I.) := 0
    END;
    WRITELN (TERMOUT, ' TYPE Y/N TO ENTER INITIAL TOOLWIDTHS AND OFFSETS');
    READLN (TERMIN);
    READ (TERMIN, ENTER);
    IF (ENTER = 'Y') THEN
    BEGIN
      REWRITE (OFFTOOL);
```

```
          FOR J := 0 TO MAX_OFFSET DO
          BEGIN
            WRITELN (TERMOUT,' ENTER OFFSET FOR OFFSET CODE', J:2);
            READLN (TERMIN);
            READ (TERMIN,OFFSET(.J.));
            WRITE (OFFTOOL, OFFSET(.J.):7)
          END;
          WRITELN (OFFTOOL);
          OFFCODE := 0;
          FOR J := 1 TO MAX_TOOL DO
          BEGIN
            WRITELN(TERMOUT,' ENTER TOOLWIDTH FOR POSITION',J:2);
            READLN (TERMIN);
            READ (TERMIN,TOOL(.J.));
            WRITE (OFFTOOL, TOOL(.J.):7)
          END;
          WRITELN (OFFTOOL);
        END
        ELSE
        BEGIN
          RESET (OFFTOOL);
          FOR J := 0 TO MAX_OFFSET DO
            READ (OFFTOOL, OFFSET(.J.));
          READLN (OFFTOOL);
          FOR J := 1 TO MAX_TOOL DO
            READ (OFFTOOL, TOOL(.J.))
        END;
        TOOLWIDTH := TOOL(.1.);
        TOOLPOS := 1;
        WRITE ('ST #','TL_WD':6,'G':2,'X_POS':7,'Y_POS':7,'Z_POS':7);
        WRITELN ('I':7,'J':7,'K':7);
        WRITELN
      END
    END;
```

```
(**************************************************************************
*                          PROCEDURE REINITIALIZE                          *
**************************************************************************)

(* THIS PROCEDURE INITIALIZES VARIABLES AFTER EACH M/C CODE STATEMENT*)

PROCEDURE REINITIALIZE;

VAR
  A : ALPHABETS;              (* LOCAL VARIABLE FOR LOOPING *)

BEGIN
  IF (NOT ERROR) THEN
  BEGIN
    PAUSE := FALSE;
    TOOL_INDEX := FALSE;
    ORIGIN_CHECK := FALSE;
    HOMEPOS := FALSE;
    G23CODE := 0;
    G4CODE := 0;
    TIMES  := 1;
    FOR A := 'I' TO 'K' DO
    BEGIN
      CENTER(.A.) := MAXINT;
      NEG_ZERO(.A.)  := FALSE
    END;
    FOR A := 'X' TO 'Z' DO
      NEG_ZERO(.A.) := FALSE;
    FOR A := 'X' TO 'Z' DO
      IF (((A = 'Z') AND (G8CODE = 80)) OR (A = 'X') OR (A = 'Y')) THEN
        SHIFT(.A.) := MAXINT
  END
END;
```

```
(*************************************************************************
*                          PROCEDURE REREAD_CHECK                        *
*************************************************************************)

(*   THIS PROCEDURE READS THE CODES AGAIN FOR PROCESSING THEM           *)

PROCEDURE REREAD_CHECK;

VAR
  NUM  : INTEGER;              (* SUBSCRIPT VARIABLE FOR ARRAY *)
  I    : INTEGER;              (* LOCAL VARIABLE FOR LOOPING *)
  J    : INTEGER;              (* LOCAL VARIABLE FOR LOOPING *)
  BLANK: CHAR;                 (* VARIABLE TO READ INITIAL BLANK IN INPUT *)
  NEXT_LETTER : CHAR;
  NUMBR : INTEGER;

BEGIN
  IF (NOT ERROR) THEN
  BEGIN
    READ (BLANK,CODE(.1.).LETTER,CODE(.1.).NUMBER);
    LINE := CODE(.1.).NUMBER;
    READ (NEXT_LETTER);
    FOR NUM := 2 TO INDEX(.LINE.).LINE_COUNT DO
    BEGIN
      CODE(.NUM.).LETTER := NEXT_LETTER;
      LETCODE := CODE(.NUM.).LETTER;
      IF ((NEXT_LETTER='X') OR (NEXT_LETTER='Y') OR (NEXT_LETTER='Z') OR
          (NEXT_LETTER='I') OR (NEXT_LETTER='J') OR (NEXT_LETTER='K')) THEN
        LEXICAL_READ (NEXT_LETTER)
      ELSE
        READ (NUMCODE,NEXT_LETTER);
      CODE(.NUM.).NUMBER := NUMCODE;
      CHECK_CODES
    END;
    READLN;
    FOR I :=  1 TO TIMES DO
    BEGIN
      IF ((I > 1) AND (NOT ERROR)) THEN
        FOR J := 2 TO INDEX(.LINE.).LINE_COUNT DO
        BEGIN
          LETCODE := CODE(.J.).LETTER;
          NUMCODE := CODE(.J.).NUMBER;
          CHECK_CODES
        END;
```

```
   IF (ORIGIN_CHECK) THEN
     CHANGE_ORIGIN
   ELSE IF ¬(TOOL_INDEX) THEN
     CHANGE_TOOL
   ELSE IF ¬(HOMEPOS) THEN
     GO_HOME
   ELSE IF ((G23CODE = 2) OR (G23CODE = 3)) THEN
     CIRCULAR_MOTION
   ELSE IF (PAUSE) THEN
     MID_STOP
   ELSE IF (G8CODE <> 80) THEN
     FIXED_CYCLE
   ELSE
     GO_AHEAD
  END
 END
END;
```

```
(****************************************************************************
*                          PROCEDURE DEVIATE                               *
****************************************************************************)

(*       THIS PROCEDURE TAKES CARE OF SUBROUTINE CALLS               *)

PROCEDURE DEVIATE

(VAR
   STMT   : INTEGER;          (* STMT FROM WHICH ROUTINE CALL MADE *)
   CALLTO : INTEGER );        (* STMT # BEING CALLED *)

VAR
  ROUT_TIMES : INTEGER;       (* # OF TIMES THE ROUTINE IS REPEATED *)
  I          : INTEGER;       (* LOCAL VARIABLE FOR LOOPING *)
  NUM        : INTEGER;       (* LOCAL VARIABLE FOR LOOPING *)
  LOCATE     : INTEGER;       (* POSITION PRECEDING THE STMT # CALLED *)

BEGIN
  IF (NOT ERROR) THEN
  BEGIN
    LOCATE := INDEX(.CALLTO.).STMT_COUNT - 1;
    ROUT_TIMES := TIMES;
    FOR I := 1 TO ROUT_TIMES DO
    BEGIN
      ROUTINE := TRUE;
      SCROLL(LOCATE);

      WHILE ((ROUTINE) AND (NOT ERROR)) DO
      BEGIN
        REINITIALIZE;
        REREAD_CHECK;
      END
    END;
    SCROLL(STMT)
  END
END;
```

```
(***********************************************************************
*                        PROCEDURE PROCESS_DATA                        *
***********************************************************************)

(*              THIS PROCEDURE ANALYSIS THE DATA                      *)

PROCEDURE PROCESS_DATA;

BEGIN
  IF (NOT ERROR) THEN
  BEGIN
    WHILE ((NOT STOP) AND (NOT ERROR)) DO
    BEGIN
      STMT := STMT + 1;
      REINITIALIZE;
      REREAD_CHECK;
      IF (ROUTINE) THEN
        DEVIATE (STMT,CALLTO)
    END
  END
END;
```

```
(**************************************************************************
*                              MAIN PROGRAM                               *
**************************************************************************)

(*   THE MAIN PROGRAM READS, INITIALIZES AND PROCESSES THE M/C CODE   *)

BEGIN
  READ_DATA;
  INITIALIZE;
  PROCESS_DATA
END.
```

APPENDIX F

<u>Program listing for Phase II</u>

```
C$JOB
C***********************************************************************
C                                                                      *
C                        TOOL PATH PLOTTING PROGRAM                    *
C                                                                      *
C                                   BY                                 *
C                                                                      *
C                            P. SARAVANA PRASAD                        *
C                                                                      *
C   This program is Phase II of the tool path plotting program. It     *
C   plots the tool path according to the coordinates specified by      *
C   POSITION FILE. This data file is generated in Phase I. This is an  *
C   interactive plotting program. It can perform different functions.  *
C   The main program calls different subroutines according to function *
C   specified by the user.                                             *
C   The functions that can be performed are:                           *
C                                                                      *
C     CLEAR   -     SUBROUTINE CLEAR                                   *
C                   This command clears the screen and draws the work  *
C                   piece.                                             *
C     OPTIONS -     SUBROUTINE OPTION                                  *
C                   This command provides other options for the user.  *
C                   The user can specify the data line from which the  *
C                   plotting should start (default = 1).               *
C                   The user can specify the data line at which the    *
C                   plotting should stop (default = End of data).      *
C                   The user can clear the screen each time the tool is *
C                   changed (default = Screen not cleared).            *
C                   The user can draw the grids during the DRAW or ZOOM *
C                   command (default = Grids not drawn).               *
C                   The user can draw upto 10 fixtures at desired place *
C                   by moving the cross-hair cursor and specifying the *
C                   corners for each fixture (default=Fixture not drawn)*
C     DRAW    -     SUBROUTINE SBDRAW                                  *
C                   This command draws the tool path according to the  *
C                   options specified by the user.                     *
C     ZOOM    -     SUBROUTINE ZOOM                                    *
C                   This command enlarges the portion of the drawing   *
C                   desired by the user. The two coordinates of one of *
C                   the diagonal elements of the area are specified by *
C                   the user by moving the cross-hair cursor. This area *
C                   is converted to the nearest square area and enlarged*
C     QUIT    -     MAIN PROGRAM                                       *
C                   This command returns the user back to CMS.         *
C                                                                      *
C***********************************************************************
```

```
C***********************************************************************
C                                                                      *
C                             MAIN PROGRAM                             *
C                                                                      *
C***********************************************************************
C                                                                      *
      REAL PREVX,PREVY,PREVZ,PRESX,PRESY,PRESZ,IVAL,JVAL,KVAL,TLWD
      REAL LENGTH,WIDTH,SCLFAC,PREVTL,XFIX(10,4),YFIX(10,4)
      REAL XLIMIT,YLIMIT
      INTEGER STMT,GCODE,NUMFIX,STRTLN,ENDLIN
      CHARACTER*1 TLCHCE,GRCHCE,FXCHCE
      CHARACTER*50 COMAND,BLANK
      COMMON/CHOICE/TLCHCE,GRCHCE,FXCHCE
      COMMON/VALUES/PREVX,PREVY,PREVZ,PRESX,PRESY,PRESZ,IVAL,JVAL,KVAL,
     +TLWD,STMT,GCODE,PREVTL
      COMMON/SCALES/LENGTH,WIDTH,SCLFAC,XLIMIT,YLIMIT
      COMMON/MISL/NUMFIX,XFIX,YFIX,STRTLN,ENDLIN
      DATA BLANK/' '/
C
C     Obtain the length and width of the workpiece and convert it to
C     to Graphic Display Units (GDU). Choose a scaling factor such that
C     the bigger of the two sides is 80 GDU.
C
50      CONTINUE
        REWIND 9
        WRITE (9,*) ' Enter the LENGTH and WIDTH of material in INCHES'
        READ (9,*,END=50) LENGTH,WIDTH
        LENGTH = LENGTH * 10.0
        WIDTH = WIDTH * 10.0
        SCLFAC = 80.0/MAX(LENGTH,WIDTH)
        XLIMIT = LENGTH * SCLFAC
        YLIMIT = WIDTH * SCLFAC
        XLIMIT = MIN (XLIMIT,80.0) + 10.0
        YLIMIT = MIN (YLIMIT,80.0) + 10.0
C
C     Set the default values for the options
C
        STRTLN = 1
        ENDLIN = 0
        TLCHCE = 'N'
        GRCHCE = 'N'
        FXCHCE = 'N'
C
C     Initialize IGL and draw the workpiece. Prompt for user command.
C
        CALL GRSTRT (4010,1)
        CALL CLEAR
10      CONTINUE
        CALL MOVE (5.0,3.0)
        CALL TXICUR(4)
        CALL TEXT (2,'=>')
        CALL CMCLOS
```

```
C
C     Read and process the user command.
C
        READ (9,20,END=60) COMAND
 60     CONTINUE
        REWIND 9
        CALL CMOPEN
        IF (INDEX(COMAND,'CLEAR').NE.0) THEN
          CALL CLEAR
        ELSEIF (INDEX(COMAND,'OPTIONS').NE.0) THEN
          CALL OPTION
        ELSEIF (INDEX(COMAND,'DRAW').NE.0) THEN
          CALL SBDRAW
        ELSEIF (INDEX(COMAND,'ZOOM').NE.0) THEN
          CALL ZOOM
        ELSEIF (INDEX(COMAND,'QUIT').NE.0) THEN
          GO TO 30
        ELSEIF (COMAND.NE.' ') THEN
          CALL TXICUR(4)
          CALL MOVE (5.0,3.0)
          CALL TEXT (50,BLANK)
          CALL MOVE (5.0,3.0)
          CALL TEXT (40,'Unknown Command - Hit RETURN to continue')
          CALL CMCLOS
          READ (9,20,END=40) COMAND
 40       CONTINUE
          REWIND 9
          CALL CMOPEN
          CALL MOVE (5.0,3.0)
          CALL TEXT (50,BLANK)
        ENDIF
        CALL MOVE (5.0,3.0)
        CALL TXICUR(4)
        CALL TEXT (50,BLANK)
        COMAND = ' '
        GO TO 10
 30     CONTINUE
        CALL NEWPAG
        CALL SBANSI
        CALL GRSTOP
      STOP
 20   FORMAT (A50)
      END
```

```
C***********************************************************************
C                                                                      *
C                        SUBROUTINE CLEAR                              *
C                                                                      *
C     This subroutine draws the workpiece and labels the toolwidth and *
C     Z-coordinate positions.                                          *
C                                                                      *
C***********************************************************************
C
      SUBROUTINE CLEAR
      REAL XPANEL(5),YPANEL(5),XLIMIT,YLIMIT,LENGTH,WIDTH,SCLFAC
      COMMON/SCALES/LENGTH,WIDTH,SCLFAC,XLIMIT,YLIMIT
C
C     Assign values to variables according the size of the workpiece
C     and draw the workpiece
C
        XPANEL(1) = 10.0
        XPANEL(2) = 10.0
        XPANEL(3) = XLIMIT
        XPANEL(4) = XLIMIT
        XPANEL(5) = 10.0
        YPANEL(1) = 10.0
        YPANEL(2) = YLIMIT
        YPANEL(3) = YLIMIT
        YPANEL(4) = 10.0
        YPANEL(5) = 10.0
        CALL NEWPAG
        CALL DASHPT(0)
        CALL MOVE (10.0,10.0)
        CALL POLY (5,XPANEL,YPANEL)
C
C     Label for the text display on the side of the plot
C
        CALL TXICUR (4)
        CALL MOVE (95.0,90.0)
        CALL TEXT (14, ' TOOL   Z_CORD')
      RETURN
      END
```

```
C***********************************************************************
C                                                                      *
C                          SUBROUTINE SBANSI                           *
C                                                                      *
C    This subroutine returns the terminal back to ANSI mode from the   *
C    graphics mode.                                                    *
C                                                                      *
C***********************************************************************
C
      SUBROUTINE SBANSI
      INTEGER IASRAY(2),IAMRAY(1)
      DATA IASRAY,IAMRAY/27,50,0/
        CALL KAS2AM (2,IASRAY,IAMRAY)
        WRITE (9,10) IAMRAY
      RETURN
10    FORMAT (' ',A4)
      END
```

```
C***********************************************************************
C                                                                     *
C                          SUBROUTINE GRID                            *
C                                                                     *
C    This subroutine draws the grid and labels it. Every 10th line is *
C    drawn by a different type of line.                               *
C                                                                     *
C***********************************************************************
C
      SUBROUTINE GRID
      INTEGER I,DIVCNT
      REAL XPOS,YPOS,DSPLYX,DSPLYY,LENGTH,WIDTH,SCLFAC,XLIMIT,YLIMIT
      COMMON/SCALES/LENGTH,WIDTH,SCLFAC,XLIMIT,YLIMIT
C
C     Draw the grid lines along X-axis, with every 5th line solid.
C
        DIVCNT = 0
        DO 10 I = 2,80,2
          XPOS = I + 10.0
          DIVCNT = DIVCNT + 1
          IF (DIVCNT.EQ.5) THEN
            DIVCNT = 0
            CALL DASHPT(0)
          ELSE
            CALL DASHPT(1)
          ENDIF
          CALL MOVE (XPOS,10.0)
          CALL DRAW (XPOS,90.0)
10      CONTINUE
C
C     Draw the grid lines along Y-axis, with every 5th line solid.
C
        DIVCNT = 0
        DO 20 I = 2,80,2
          YPOS = I + 10.0
          DIVCNT = DIVCNT + 1
          IF (DIVCNT.EQ.5) THEN
            DIVCNT = 0
            CALL DASHPT(0)
          ELSE
            CALL DASHPT(1)
          ENDIF
          CALL MOVE (10.0,YPOS)
          CALL DRAW (90.0,YPOS)
20      CONTINUE
```

```
C
C     Label the grid every 5th line along the X-axis
C
        CALL TXICUR(8)
        DSPLYX = 0.0
        CALL MOVE (10.0,8.0)
        CALL RNUMBR (DSPLYX,1,5)
        DO 30 I = 10,80,10
          XPOS = I + 10.0
          CALL MOVE (XPOS,8.0)
          DSPLYX = DSPLYX + MAX(LENGTH,WIDTH)/80.0
          CALL RNUMBR (DSPLYX,1,5)
30      CONTINUE
C
C     Label the grid every 5th line along the Y-axis
C
        CALL TXICUR(6)
        DSPLYY = 0.0
        CALL MOVE (8.0,10.0)
        CALL RNUMBR (DSPLYY,1,5)
        DO 40 I = 10,80,10
          YPOS = I + 10.0
          CALL MOVE (8.0,YPOS)
          DSPLYY = DSPLYY + MAX(LENGTH,WIDTH)/80.0
          CALL RNUMBR (DSPLYY,1,5)
40      CONTINUE
      RETURN
      END
```

```
C***********************************************************************
C                                                                      *
C                         SUBROUTINE SBDRAW                            *
C                                                                      *
C    This subroutine draws a X-Y plot according to the coordinates     *
C    specified by POSITION FILE. Circles are drawn according to the    *
C    toolwidth, along the path taken by the tool. The toolwidth and    *
C    the Z-coordinate are written on the side of the screen. The grids *
C    and fixtures are drawn if desired by the user.                    *
C                                                                      *
C***********************************************************************
C
      SUBROUTINE SBDRAW
      REAL PREVX,PREVY,PREVZ,PRESX,PRESY,PRESZ,XFIX(10,4),YFIX(10,4)
      REAL IVAL,JVAL,KVAL,TLWD,PREVTL,LENGTH,WIDTH,SCLFAC,ZCOORD
      REAL XLIMIT,YLIMIT,TLCORD
      INTEGER GCODE,CHECK,STMT,STRTLN,ENDLIN,ZCOUNT,NUMFIX
      CHARACTER*1 TLCHCE,GRCHCE,FXCHCE,FIXMRK
      CHARACTER*50 DUMMY,BLANK
      COMMON/CHOICE/TLCHCE,GRCHCE,FXCHCE
      COMMON/SCALES/LENGTH,WIDTH,SCLFAC,XLIMIT,YLIMIT
      COMMON/VALUES/PREVX,PREVY,PREVZ,PRESX,PRESY,PRESZ,IVAL,JVAL,KVAL,
     +TLWD,STMT,GCODE,PREVTL
      COMMON/MISL/NUMFIX,XFIX,YFIX,STRTLN,ENDLIN
      DATA BLANK /' '/
C
C     Clear the screen and draw the workpiece. Initialize variables for
C     writing changes in the Z-coordinates and toolwidth.
C
        CALL CLEAR
        ZCOUNT = 1
        ZCOORD = 75.0
        TLCORD = 75.0
C
C     Draw the fixture and the grid if desired
C
        IF (FXCHCE.EQ.'Y') CALL FIXTUR
        CALL WINDOW (0.0,XLIMIT,0.0,YLIMIT)
        CALL VWPORT (0.0,XLIMIT,0.0,YLIMIT)
        IF (GRCHCE.EQ.'Y') CALL GRID
        CALL TRIDNT (.TRUE.)
C
C     Reset to top of data file and omit the first two data lines having
C     labels for the data file (POSITION FILE). Read the third line.
C
        REWIND 1
        READ (1,*,END=30)
        READ (1,*,END=30)
        DO 10 I = 1,STRTLN
          READ (1,20,END=30) STMT,TLWD,GCODE,PRESX,PRESY,PRESZ,
     +    IVAL,JVAL,KVAL
10      CONTINUE
```

```
C
C      Convert to GDU and draw the starting positions and values
C
          CALL TXICUR(4)
          CALL MOVE (95.0,75.0)
          CALL RNUMBR (TLWD,-1,5)
          CALL MOVE (105.0,75.0)
          CALL RNUMBR (PRESZ,-1,7)
          TLWD = TLWD/2000.
          PRESX = PRESX/1000.
          PRESY = PRESY/1000.
          IVAL = IVAL/1000.
          JVAL = JVAL/1000.
          KVAL = KVAL/1000.
          CALL TRANSL (10.0,10.0)
          CALL SCALE (SCLFAC,SCLFAC)
          CALL DASHPT(0)
          IF (GCODE.EQ.0) CALL DASHPT(3)
          CALL TOOL (TLWD,PRESX,PRESY)
          CALL TXICUR(5)
          CALL MOVE (PRESX,PRESY)
          CALL INUMBR (ZCOUNT,2)
          CHECK = STRTLN
C
C      Check to the number of data lines to be read
C
40        CONTINUE
          IF (CHECK.NE.ENDLIN) THEN
C
C      Set the viewing transformations
C
            CALL TRIDNT(.FALSE.)
            CALL TRANSL (10.0,10.0)
            CALL SCALE (SCLFAC,SCLFAC)
C
C      Store the previous cursor position
C
            PREVX = PRESX
            PREVY = PRESY
            PREVZ = PRESZ
            PREVTL = TLWD
C
C      Read the rest of the data and convert to GDU
C
            READ (1,20,END=30) STMT,TLWD,GCODE,PRESX,PRESY,PRESZ,
     +      IVAL,JVAL,KVAL
            CHECK = CHECK + 1
            TLWD = TLWD/2000.
            PRESX = PRESX/1000.
            PRESY = PRESY/1000.
            IVAL = IVAL/1000.
            JVAL = JVAL/1000.
            KVAL = KVAL/1000.
```

```
C
C     Depending on the code, decide whether the movement is linear or
C     circular.
C
          IF ((GCODE.EQ.1).OR.(GCODE.EQ.0)) THEN
            CALL STLINE
          ELSE
            CALL CURVE
          ENDIF
C
C     Check for any changes in the Z-coordinate or toolwidth
C
          CALL TRIDNT (.FALSE.)
          CALL TXTWRT (ZCOUNT,ZCOORD,TLCORD)
          GO TO 40
        ELSE
        ENDIF
C
C     Complete the plotting procedure
C
30      CONTINUE
        CALL TRIDNT (.FALSE.)
        CALL TXICUR(4)
        CALL MOVE (5.0,3.0)
        CALL TEXT (50,BLANK)
        CALL MOVE (5.0,3.0)
        CALL TEXT (38,'Plotting done - HIT RETURN TO CONTINUE')
        CALL CMCLOS
        READ (9,50,END=60) DUMMY
60      CONTINUE
        REWIND 9
        CALL CMOPEN
        CALL MOVE (5.0,3.0)
        CALL TEXT (50,BLANK)
      RETURN
20    FORMAT (I4,F6.0,I2,6F7.0)
50    FORMAT (A50)
      END
```

```
C***********************************************************************
C                                                                      *
C                       SUBROUTINE OPTION                              *
C                                                                      *
C     This subroutine provides the user with certain plotting options. *
C     The data line from which the plotting starts, the data line at   *
C     which the plotting stops, drawing the fixtures, drawing the grids *
C     and clearing the screen when the tool changes are the options    *
C     provided.                                                        *
C                                                                      *
C***********************************************************************
C
      SUBROUTINE OPTION
      REAL XFIX(10,4),YFIX(10,4),TMPXFX(4),TMPYFX(4)
      REAL LENGTH,WIDTH,SCLFAC,XLIMIT,YLIMIT
      INTEGER STRTLN,ENDLIN,NUMFIX,I,J,IDAT,IGOT
      CHARACTER*1 TLCHCE,GRCHCE,FXCHCE,DUMMY
      CHARACTER*50 BLANK
      COMMON/SCALES/LENGTH,WIDTH,SCLFAC,XLIMIT,YLIMIT
      COMMON/CHOICE/TLCHCE,GRCHCE,FXCHCE
      COMMON/MISL/NUMFIX,XFIX,YFIX,STRTLN,ENDLIN
      DATA BLANK /' '/
C
C     Take cursor to the top of the screen and clear the screen. Then go
C     to ANSI mode to prompt for user options.
C
        CALL HOME
        CALL NEWPAG
        CALL SBANSI
C
C     Specify different user options. Variables are set according to the
C     option desired by the user.
C
        WRITE (9,*) 'All the options remain ineffect until changed'
        WRITE (9,*)
        WRITE (9,*)
C
C     Enter the data line (excluding titles) from which the plotting
C     should commence.
C
        WRITE (9,*) 'Enter STARTING DATA LINE for drawing'
        WRITE (9,*) '(Hit RETURN for default value 1)'
        READ (9,*,END=10) STRTLN
10      CONTINUE
        REWIND 9
C
C     Enter the data line at which plotting should stop
C
        WRITE (9,*) 'Enter ENDING DATA LINE for drawing'
        WRITE (9,*) '(Hit RETURN for all data lines)'
        READ (9,*,END=20) ENDLIN
20      CONTINUE
        REWIND 9
```

```
C
C     Provide the option for either clearing the screen whenever the
C     toolwidth changes or to continue on the same screen and write the
C     new value on the side of the X-Y plot.
C
        WRITE (9,*) 'Type Y/N for clearing screen during TOOL CHANGE'
        READ (9,60,END=30) TLCHCE
30      CONTINUE
        REWIND 9
C
C     Choose whether you need the grids to be drawn along with the plot
C
        WRITE (9,*) 'Type Y/N for drawing the grid'
        READ (9,60,END=40) GRCHCE
40      CONTINUE
        REWIND 9
C
C     Choose whether you need to draw the fixtures along with the plot.
C     If yes, enter the number of fixtures (upto 10) and mark the four
C     corners for each fixture.
C
        WRITE (9,*) 'Type Y/N for marking the fixtures'
        READ (9,60,END=50) FXCHCE
50      CONTINUE
        REWIND 9
        IF (FXCHCE.EQ.'Y') THEN
          WRITE (9,*) 'Enter the number of FIXTURES (upto 10)'
          READ (9,*,END=50) NUMFIX
          CALL CLEAR
          CALL WINDOW (0.0,XLIMIT,0.0,YLIMIT)
          CALL VWPORT (0.0,XLIMIT,0.0,YLIMIT)
          CALL GRID
          CALL TRIDNT (.TRUE.)
          DO 70 I = 1,NUMFIX
            CALL TXICUR(4)
            CALL MOVE (5.0,3.0)
            CALL TEXT (50,BLANK)
            CALL MOVE (5.0,3.0)
            CALL TEXT (26,'Mark location for Fixture ')
            CALL MOVE (49.0,3.0)
            CALL INUMBR (I,2)
            DO 80 J = 1,4
              CALL LOCATE (1,XFIX(I,J),YFIX(I,J),IDAT,IGOT)
              CALL TXICUR (5)
              CALL MOVE (XFIX(I,J),YFIX(I,J))
              CALL TEXT (1,'x')
              TMPXFX(J) = XFIX(I,J)
              TMPYFX(J) = YFIX(I,J)
80          CONTINUE
            CALL FILPAN (6,.TRUE.)
            CALL PANEL (4,TMPXFX,TMPYFX)
            CALL FILPAN (7,.TRUE.)
            CALL PANEL (4,TMPXFX,TMPYFX)
70        CONTINUE
```

```
          CALL MOVE (5.0,3.0)
          CALL TEXT (50,BLANK)
          CALL MOVE (5.0,3.0)
          CALL TXICUR(4)
          CALL TEXT (41,'Fixture(s) drawn - Hit RETURN to continue')
          CALL CMCLOS
          READ (9,60,END=90) DUMMY
90        CONTINUE
          REWIND 9
          CALL CMOPEN
        ELSE
        ENDIF
        CALL CLEAR
      RETURN
60    FORMAT (A1)
      END
```

```
C*********************************************************************
C                                                                    *
C                         SUBROUTINE ZOOM                            *
C                                                                    *
C    This subroutine provides the user with the option of focusing on *
C    a certain portion of the plot. The area specified is converted to *
C    the nearest square and this portion is enlarged for viewing.     *
C                                                                    *
C*********************************************************************
C
      SUBROUTINE ZOOM
      REAL PREVX,PREVY,PREVZ,PRESX,PRESY,PRESZ,XFIX(10,4),YFIX(10,4)
      REAL IVAL,JVAL,KVAL,TLWD,PREVTL,LENGTH,WIDTH,SCLFAC,YPANEL(5)
      REAL XCOORD(2),YCOORD(2),MINMUM,MAXMUM,XLIMIT,YLIMIT,XPANEL(5)
      INTEGER GCODE,CHECK,STMT,STRTLN,ENDLIN,ZCOUNT,NUMFIX
      CHARACTER*1 TLCHCE,GRCHCE,FXCHCE,FIXMRK
      CHARACTER*50 BLANK,DUMMY
      COMMON/CHOICE/TLCHCE,GRCHCE,FXCHCE
      COMMON/SCALES/LENGTH,WIDTH,SCLFAC,XLIMIT,YLIMIT
      COMMON/VALUES/PREVX,PREVY,PREVZ,PRESX,PRESY,PRESZ,IVAL,JVAL,KVAL,
     +TLWD,STMT,GCODE,PREVTL
      COMMON/MISL/NUMFIX,XFIX,YFIX,STRTLN,ENDLIN
      DATA XPANEL /10.0,10.0,90.0,90.0,10.0/
      DATA YPANEL /10.0,90.0,90.0,10.0,10.0/
      DATA BLANK /' '/
C
C     Locate the diagonal elements of the desired area
C
        CALL TXICUR (4)
        CALL MOVE (5.0,3.0)
        CALL TEXT (50,BLANK)
        CALL MOVE (5.0,3.0)
        CALL TEXT (42,'Mark any diagonal coordinates of zoom area')
        CALL LOCATE (2,XCOORD,YCOORD,IDAT,IGOT)
        CALL MOVE (5.0,3.0)
        CALL TEXT (50,BLANK)
C
C     Choose the smallest and largest coordinate value to make the
C     chosen area a square. This is done in order to remove any distor-
C     tion in the plot.
C
        MINMUM = MIN(XCOORD(1),XCOORD(2),YCOORD(1),YCOORD(2)) - 10.0
        MAXMUM = MAX(XCOORD(1),XCOORD(2),YCOORD(1),YCOORD(2)) - 10.0
C
C     Change the window and viewport coordinates in order enlarge the
C     desired area. The plot is shown on a 80 x 80 GDU and does not
C     vary with the size of the workpiece
C
        CALL NEWPAG
        CALL DASHPT(0)
        CALL MOVE (10.0,10.0)
        CALL POLY (5,XPANEL,YPANEL)
        CALL WINDOW (MINMUM,MAXMUM,MINMUM,MAXMUM)
        CALL VWPORT (10.0,90.0,10.0,90.0)
```

```
C     Draw the grid within the area if specified
C
        IF (GRCHCE.EQ.'Y') CALL GRID
C
C     Set scaling factor. Go to top of data file. Omit first two data
C     lines and read the third one for initializing.
C
        CALL SCALE (SCLFAC,SCLFAC)
        REWIND 1
        READ (1,*,END=30)
        READ (1,*,END=30)
        DO 10 I = 1,STRTLN
          READ (1,20,END=30) STMT,TLWD,GCODE,PRESX,PRESY,PRESZ,
     +    IVAL,JVAL,KVAL
10      CONTINUE
C
C     Convert to GDU and draw the intial position and value
C
        TLWD = TLWD/2000.
        PRESX = PRESX/1000.
        PRESY = PRESY/1000.
        IVAL = IVAL/1000.
        JVAL = JVAL/1000.
        KVAL = KVAL/1000.
        CALL DASHPT(0)
        IF (GCODE.EQ.0) CALL DASHPT(3)
        CALL TOOL (TLWD,PRESX,PRESY)
        CALL TXICUR(5)
        CALL MOVE (PRESX,PRESY)
        CALL INUMBR (ZCOUNT,2)
        CHECK = STRTLN
C
C     Check to the number of data lines to be read
C
40      CONTINUE
        IF (CHECK.NE.ENDLIN) THEN
C
C     Store previous cursor position
C
          PREVX = PRESX
          PREVY = PRESY
          PREVZ = PRESZ
          PREVTL = TLWD
C
C     Read the next data line and convert to GDU
C
          READ (1,20,END=30) STMT,TLWD,GCODE,PRESX,PRESY,PRESZ,
     +    IVAL,JVAL,KVAL
          CHECK = CHECK + 1
          TLWD = TLWD/2000.
          PRESX = PRESX/1000.
          PRESY = PRESY/1000.
          IVAL = IVAL/1000.
          JVAL = JVAL/1000.
          KVAL = KVAL/1000.
```

```
C
C     Depending on the machining code, decide whether the movement is
C     linear or circular
C
          IF ((GCODE.EQ.1).OR.(GCODE.EQ.0)) THEN
            CALL STLINE
          ELSE
            CALL CURVE
          ENDIF
          GO TO 40
        ELSE
        ENDIF
30      CONTINUE
C
C     End of the plotting
C
        CALL TRIDNT (.FALSE.)
        CALL TRIDNT (.TRUE.)
        CALL TXICUR(4)
        CALL MOVE (5.0,3.0)
        CALL TEXT (50,BLANK)
        CALL MOVE (5.0,3.0)
        CALL TEXT (38,'Plotting done - HIT RETURN TO CONTINUE')
        CALL CMCLOS
        READ (9,50,END=60) DUMMY
60      CONTINUE
        REWIND 9
        CALL CMOPEN
        CALL MOVE (5.0,3.0)
        CALL TEXT (50,BLANK)
      RETURN
20    FORMAT (I4,F6.0,I2,6F7.0)
50    FORMAT (A50)
      END
```

```
C*********************************************************************
C                                                                    *
C                        SUBROUTINE FIXTUR                           *
C                                                                    *
C    This subroutine draws the fixture, whose location has already   *
C    been specified in SUBROUTINE OPTION.                            *
C                                                                    *
C*********************************************************************
C
      SUBROUTINE FIXTUR
      CHARACTER*1 TLCHCE,GRCHCE,FXCHCE
      INTEGER NUMFIX,STRTLN,ENDLIN
      REAL XFIX(10,4),YFIX(10,4),TMPXFX(4),TMPYFX(4)
      COMMON/CHOICE/TLCHCE,GRCHCE,FXCHCE
      COMMON/MISL/NUMFIX,XFIX,YFIX,STRTLN,ENDLIN
C
C     Draw the fixture by drawing hatched lines
C
        DO 10 I = 1,NUMFIX
          DO 20 J = 1,4
            TMPXFX(J) = XFIX(I,J)
            TMPYFX(J) = YFIX(I,J)
20        CONTINUE
          CALL FILPAN (6,.TRUE.)
          CALL PANEL (4,TMPXFX,TMPYFX)
          CALL FILPAN (7,.TRUE.)
          CALL PANEL (4,TMPXFX,TMPYFX)
10      CONTINUE
      RETURN
      END
```

```
C***********************************************************************
C                                                                      *
C                           SUBROUTINE TOOL                            *
C                                                                      *
C    This subroutine draws a circle according to the toolwidth and at  *
C    coordinates specified by the arguments.                           *
C                                                                      *
C***********************************************************************
C
      SUBROUTINE TOOL (TLWD,X,Y)
      REAL TLWD,X,Y
        CALL MOVE (X,Y)
        CALL ARC (TLWD,0.0,360.0)
      RETURN
      END
```

```
C*******************************************************************
C                                                                  *
C                      SUBROUTINE TXTWRT                           *
C                                                                  *
C    This subroutine writes the toolwidth and the Z-coordinate at the  *
C    side of the X-Y plot. These values are written whenever there is  *
C    a change in either of their values. The values are written one    *
C    below the other. When the end of the screen is reached, it starts *
C    again from the top. Depending the number of times the Z-coordinate*
C    changes, a number is marked on the X-Y plot at the place where the*
C    value changes.                                                    *
C                                                                  *
C*******************************************************************
C
      SUBROUTINE TXTWRT (ZCOUNT,ZCOORD,TLCORD)
      INTEGER NUMFIX,ZCOUNT,STRTLN,ENDLIN
      REAL XLIMIT,YLIMIT,TLCORD
      REAL XFIX(10,4),YFIX(10,4),PREVTL,TLWD,LENGTH,WIDTH,SCLFAC
      REAL PREVX,PREVY,PREVZ,PRESX,PRESY,PRESZ,IVAL,JVAL,KVAL,ZCOORD
      CHARACTER*50 BLANK,DUMMY
      CHARACTER*1 TLCHCE,GRCHCE,FXCHCE
      COMMON/VALUES/PREVX,PREVY,PREVZ,PRESX,PRESY,PRESZ,IVAL,JVAL,KVAL,
     +TLWD,STMT,GCODE,PREVTL
      COMMON/CHOICE/TLCHCE,GRCHCE,FXCHCE
      COMMON/SCALES/LENGTH,WIDTH,SCLFAC,XLIMIT,YLIMIT
      COMMON/MISL/NUMFIX,XFIX,YFIX,STRTLN,ENDLIN
      DATA BLANK /' '/
        CALL TXICUR(4)
C
C     Check if the toolwidth has changed
C
        IF (PREVTL.NE.TLWD) THEN
          CALL MOVE (5.0,3.0)
          CALL TEXT (36,'Tool change - Hit RETURN to continue')
          CALL CMCLOS
          READ (9,10,END=20) DUMMY
20        CONTINUE
          REWIND 9
          CALL CMOPEN
          CALL MOVE (5.0,3.0)
          CALL TEXT (50,BLANK)
```

```
C
C     Check if the user has specified the option of clearing screen
C     during tool change.
C
          IF (TLCHCE.EQ.'Y') THEN
C
C     Clear screen and redraw grid and fixture. Write last Z-value also
C
            CALL CLEAR
            CALL WINDOW (0.0,XLIMIT,0.0,YLIMIT)
            CALL VWPORT (0.0,XLIMIT,0.0,YLIMIT)
            IF (GRCHCE.EQ.'Y') CALL GRID
            CALL TRIDNT (.TRUE.)
            CALL TXICUR(4)
            IF (FXCHCE.EQ.'Y') CALL FIXTUR
            ZCOORD = 75.0
            CALL MOVE (105.0,75.0)
            CALL RNUMBR (PREVZ,-1,7)
          ELSE
C
C     Continue on same screen. Check to see if bottom of screen has been
C     reached.
C
            IF (TLCORD.LT.6.0) THEN
C
C     Erase the previous toolwidths and start from the top.
C
              TLCORD = 75.0
              DO 40 I = 1,15
                YPOS = 75.0 - (I-1) * 5.0
                CALL MOVE (95.0,YPOS)
                CALL TEXT (5,BLANK)
40            CONTINUE
            ELSE
C
C     Continue to write below the old value
C
              TLCORD = TLCORD - 5.0
            ENDIF
          ENDIF
C
C     Write the new toolwidth
C
          CALL MOVE (95.0,TLCORD)
          TLWD = TLWD*2000.
          CALL RNUMBR (TLWD,-1,5)
          TLWD= TLWD/2000.
        ELSE
        ENDIF
```

```
C
C      Check if the Z-value has changed
C
          IF (PREVZ.NE.PRESZ) THEN
            ZCOUNT = ZCOUNT + 1
            CALL MOVE (5.0,3.0)
            CALL TEXT (44,'Z-coordinate change - Hit RETURN to continue')
            CALL CMCLOS
            READ (9,10,END=30) DUMMY
 30         CONTINUE
            REWIND 9
            CALL CMOPEN
            CALL MOVE (5.0,3.0)
            CALL TEXT (50,BLANK)
C
C      Check to see if bottom of the screen has been reached
C
            IF (ZCOORD.LT.6.0) THEN
C
C      Erase the previous Z-values and start from the top.
C
              ZCOORD = 75.0
              DO 50 I = 1,15
                YPOS = 75.0 - (I-1) * 5.0
                CALL MOVE (105.0,YPOS)
                CALL TEXT (7,BLANK)
 50           CONTINUE
            ELSE
C
C      Continue to write below the old value
C
              ZCOORD = ZCOORD - 5.0
            ENDIF
C
C      Write the new Z-coordinate value
C
            CALL MOVE (105.0,ZCOORD)
            CALL RNUMBR (PRESZ,-1,7)
C
C      Write the number of times the Z-value has changed at the position
C      where the Z-value changes
C
            CALL TRANSL (10.0,10.0)
            CALL SCALE (SCLFAC,SCLFAC)
            CALL TXICUR(5)
            CALL MOVE (PRESX,PRESY)
            CALL INUMBR (ZCOUNT,2)
            CALL TRIDNT(.FALSE.)
          ELSE
          ENDIF
        RETURN
 10     FORMAT (A50)
        END
```

```
C*********************************************************************
C                                                                    *
C                       SUBROUTINE STLINE                            *
C                                                                    *
C    This subroutine finds the slope and equation of the line when the *
C    tool moves in a linear path. Circles are drawn from the current  *
C    cursor position till the destination along the equation of the  *
C    line. If it is a machining move, solid circles are drawn. If it  *
C    is a positioning move, dotted circles are drawn.                 *
C                                                                    *
C*********************************************************************
C
      SUBROUTINE STLINE
      COMMON/VALUES/PREVX,PREVY,PREVZ,PRESX,PRESY,PRESZ,IVAL,JVAL,KVAL,
     +TLWD,STMT,GCODE,PREVTL
      REAL DELTAX,DELTAY,SHIFTX,SHIFTY,FACTOR,TLWD,PRESX,PRESY,PRESZ
      REAL PREVX,PREVY,PREVZ,SLOPE,INTCPT
      INTEGER GCODE
      LOGICAL QUIT
C
C     INITIALIZE VARIABLES
C
        FACTOR = 1.75
        QUIT   = .FALSE.
        SHIFTX = PREVX
        SHIFTY = PREVY
10      CONTINUE
C
C     Continue to move along the linear path till the destination
C
        IF (QUIT) GO TO 20
C
C     If the present and the previous X-coordinate are not the same,
C     then find the slope and intercept of the linear path
C
          IF (PREVX.EQ.PRESX) GO TO 50
            SLOPE = ((PREVY-PRESY)/(PREVX-PRESX))
            INTCPT = PREVY - (SLOPE * PREVX)
            IF (PRESX.LT.PREVX) GO TO 30
C
C     Move in short increments along the equation, in the increasing
C     X-direction.
C
              DELTAX = SHIFTX + FACTOR * TLWD
              DELTAY = SLOPE * DELTAX + INTCPT
              IF (DELTAX.LT.PRESX) SHIFTX = DELTAX
              IF (DELTAX.LT.PRESX) SHIFTY = DELTAY
              IF (DELTAX.GE.PRESX) SHIFTX = PRESX
              IF (DELTAX.GE.PRESX) SHIFTY = PRESY
              GO TO 40
30          CONTINUE
```

```
C
C     Move in short increments along the equation, in the decreasing
C     X-direction.
C
            DELTAX = SHIFTX - FACTOR * TLWD
            DELTAY = SLOPE * DELTAX + INTCPT
            IF (DELTAX.GT.PRESX) SHIFTX = DELTAX
            IF (DELTAX.GT.PRESX) SHIFTY = DELTAY
            IF (DELTAX.LE.PRESX) SHIFTX = PRESX
            IF (DELTAX.LE.PRESX) SHIFTY = PRESY
            GO TO 40
 50       CONTINUE
C
C     Move in short increments along the equation, in the increasing
C     Y-direction.
C
          IF (PRESY.LT.PREVY) GO TO 60
            DELTAY = SHIFTY + FACTOR * TLWD
            SHIFTX = PRESX
            IF (DELTAY.LT.PRESY) SHIFTY = DELTAY
            IF (DELTAY.GE.PRESY) SHIFTY = PRESY
            GO TO 40
 60         CONTINUE
C
C     Move in short increments along the equation, in the decreasing
C     Y-direction.
C
            DELTAY = SHIFTY - FACTOR * TLWD
            SHIFTX = PRESX
            IF (DELTAY.GT.PRESY) SHIFTY = DELTAY
            IF (DELTAY.LE.PRESY) SHIFTY = PRESY
 40       CONTINUE
C
C     Draw solid circles or dotted circlesaccording to the machining
C     code.
C
          IF (GCODE.EQ.1) CALL DASHPT(0)
          IF (GCODE.EQ.0) CALL DASHPT(3)
          CALL TOOL (TLWD,SHIFTX,SHIFTY)
C
C     Quit drawing when the final position has been reached
C
          IF ((SHIFTX.EQ.PRESX).AND.(SHIFTY.EQ.PRESY)) QUIT = .TRUE.
          GO TO 10
 20     CONTINUE
 70   FORMAT (3F9.3)
      RETURN
      END
```

```
C*************************************************************************
C                                                                        *
C                         SUBROUTINE CURVE                               *
C                                                                        *
C     This subroutine finds the center and equation of the circle when   *
C     the tool moves in a circular path. Circles are drawn from the      *
C     current cursor position till the destination along the equation   *
C     of the circle. Depending on the machining code, the tool moves in *
C     the clockwise or in the anti-clockwise direction. For both types   *
C     of moves, solid circles are drawn.                                 *
C                                                                        *
C*************************************************************************
C
      SUBROUTINE CURVE
      COMMON/VALUES/PREVX,PREVY,PREVZ,PRESX,PRESY,PRESZ,IVAL,JVAL,KVAL,
     +TLWD,STMT,GCODE,PREVTL
      REAL CENTRX,CENTRY,RADIUS,IVAL,JVAL,PREVX,PREVY,PRESX,PRESY
      REAL TOP,BOTTOM,ONE,SHIFTX,SHIFTY,DELTAX,DELTAY,FACTOR
      INTEGER GCODE
      LOGICAL QUIT
C
C     Initialize variables
C
        FACTOR = 0.5
        SHIFTX = PREVX
        SHIFTY = PREVY
        ONE = 1.
C
C     Find the center and radius of the circular path. Find the
C     coordinates of the top and the bottom most point of the circle.
C     This is required in order to find which quadrant the tool is
C     moving, which in turn decides whether the coordinates increase
C     or decrease.
C
        CENTRX = PREVX + IVAL
        CENTRY = PREVY + JVAL
        RADIUS = SQRT (IVAL**2 + JVAL**2)
        BOTTOM = CENTRY - RADIUS
        TOP = CENTRY + RADIUS
        QUIT = .FALSE.
C
C     Plot for movement in clockwise direction
C
        IF (GCODE.EQ.3) GO TO 10
C
C     Find the quadrant in which tool is going to move
C
          IF ((PREVY.EQ.BOTTOM).OR.(PREVX.LT.CENTRX)) ONE = -1.
```

```
 20        CONTINUE
 C
 C     Move in short increments along circular path till destination
 C
           IF (QUIT) GO TO 30
           DELTAY = SHIFTY - FACTOR * TLWD * ONE
           IF (RADIUS**2.GT.(DELTAY-CENTRY)**2) GO TO 40
             DELTAX = CENTRX
             DELTAY = CENTRY - RADIUS * ONE
             SHIFTY = DELTAY
             SHIFTX = DELTAX
             ONE = -ONE
             GO TO 50
 40        CONTINUE
             DELTAX = CENTRX + ONE*SQRT(RADIUS**2 - (DELTAY-CENTRY)**2)
             IF (ONE.LT.0.) GO TO 60
               IF (DELTAY.GT.PRESY) GO TO 70
                 SHIFTX = PRESX
                 SHIFTY = PRESY
                 GO TO 50
 70            CONTINUE
                 SHIFTX = DELTAX
                 SHIFTY = DELTAY
                 GO TO 50
 60          CONTINUE
               IF (DELTAY.LT.PRESY) GO TO 80
                 SHIFTX = PRESX
                 SHIFTY = PRESY
                 GO TO 50
 80            CONTINUE
                 SHIFTX = DELTAX
                 SHIFTY = DELTAY
 50        CONTINUE
 C
 C     Draw solid circles along path
 C
           CALL TOOL (TLWD,SHIFTX,SHIFTY)
           IF ((PRESX.EQ.SHIFTX).AND.(PRESY.EQ.SHIFTY)) QUIT = .TRUE.
           GO TO 20
 C
 C     Plot for movement in anti-clockwise direction
 10      CONTINUE
 C
 C     Find the quadrant in which tool is going to move
 C
           IF ((PREVY.EQ.TOP).OR.(PREVX.LT.CENTRX)) ONE = -1.
 120        CONTINUE
 C
 C     Move in short increments along circular path till destination
 C
            IF (QUIT) GO TO 30
            DELTAY = SHIFTY + FACTOR * TLWD * ONE
            IF (RADIUS**2.GT.(DELTAY-CENTRY)**2) GO TO 140
              DELTAX = CENTRX
              DELTAY = CENTRY + RADIUS * ONE
              SHIFTY = DELTAY
              SHIFTX = DELTAX
              ONE = -ONE
              GO TO 150
 140        CONTINUE
              DELTAX = CENTRX + ONE*SQRT(RADIUS**2 - (DELTAY-CENTRY)**2)
              IF (ONE.LT.0.) GO TO 160
                IF (DELTAY.LT.PRESY) GO TO 170
                  SHIFTX = PRESX
                  SHIFTY = PRESY
                  GO TO 150
 170            CONTINUE
                  SHIFTX = DELTAX
                  SHIFTY = DELTAY
                  GO TO 150
 160          CONTINUE
                IF (DELTAY.GT.PRESY) GO TO 180
                  SHIFTX = PRESX
                  SHIFTY = PRESY
                  GO TO 150
 180            CONTINUE
                  SHIFTX = DELTAX
                  SHIFTY = DELTAY
 150        CONTINUE
 C
 C     Draw solid circles along path
 C
            CALL TOOL (TLWD,SHIFTX,SHIFTY)
 100        FORMAT (3F9.3)
            IF ((PRESX.EQ.SHIFTX).AND.(PRESY.EQ.SHIFTY)) QUIT = .TRUE.
            GO TO 120
 30    CONTINUE
       RETURN
       END
```

APPENDIX G

Sample Program

INPUT FOR PHASE I

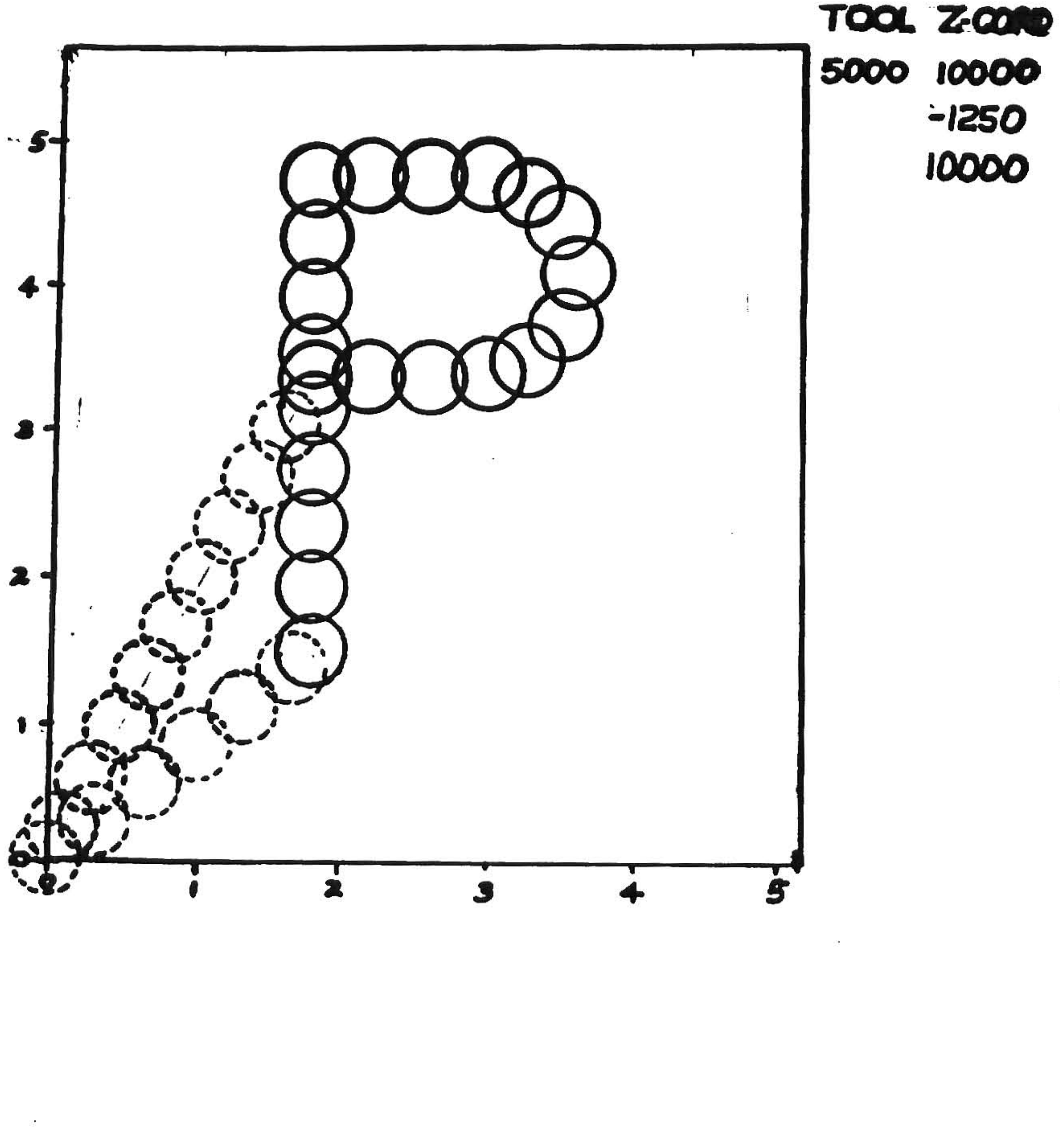
```
N0010G91F886M33
N0020G92X0Y0Z-10000
N0030G00G90X0Y0
N0040G91X17500Y15000
N0050M98P100L1
N0060G00G90X0Y0
N0070M02
N0100G01G91Z-1250
N0110Y42500
N0120X12500
N0130G02X6000Y-6000J-6000
N0140G02X-6000Y-6000I-6000
N0150G01X-12500
N0160Z11250
N0170M99
```

INTERNAL FILE FOR PHASE I

STMT NUMBER	POSITION	NUMBER OF CODES
1		
2		
⋮		
10	1	4
11		
⋮		
20	2	5
21		
⋮		
30	3	5
⋮		
80		
⋮		
100	8	4
111		
110	9	2
⋮		
170	15	2
⋮		
9999		

COORDINATE AND TOOL FILE

STMT	TLWD	G	X	Y	Z	I	J	K
10	2500	0	0	0	10000	0	0	0
20	2500	0	0	0	10000	0	0	0
30	2500	0	0	0	10000	0	0	0
40	2500	0	17500	15000	10000	0	0	0
50	2500	0	17500	15000	10000	0	0	0
100	2500	1	17500	15000	-1250	0	0	0
110	2500	1	17500	57500	-1250	0	0	0
120	2500	1	30000	57500	-1250	0	0	0
130	2500	2	36000	51500	-1250	0	-6000	0
140	2500	2	30000	45500	-1250	-6000	0	0
150	2500	1	17500	45500	-1250	0	0	0
160	2500	1	17500	45500	10000	0	0	0
170	2500	1	17500	45500	10000	0	0	0
60	2500	1	0	0	10000	0	0	0
70	2500	1	0	0	10000	0	0	0


TOOL Z-CORD
5000 10000
-1250
10000
0
1
2
3
4
5

APPENDIX H

PRAWTOPPS Users Manual

```
*    PASCAL PLOT10

*    RENAME NC LISTING A1 NC DATA A1

*    GO

=>   TYPE Y/N TO ENTER TOOL WIDTH AND OFFSET
*    Y    -   If running program for first time
     N    -   If rerunning program

*    X POSITION FILE

*    FORTVS IGL (NOMAP

*    PLOT

=>   ENTER LENGTH AND WIDTH OF WORK PIECE IN INCHES
*    ##.# ##.#

=>   Machine Prompt to enter command
*    CLEAR    -  Clears screen and draws work piece
     DRAW     -  Draw the tool path plot
     OPTIONS  -  Provides options for the user
     ZOOM     -  Enlarges certain segment of plot
     QUIT     -  Returns the user back to CMS


NOTE    => = MACHINE PROMPT        * = USER INPUT
```

APPENDIX I

Exec Programs

Profile Exec

```
RESTOR 512K
CP TERM CHARDEL
CP TERM LINEND  %
GLOBAL TXTLIB IGLSTUBS IGL PASCAL PLILIB CMSLIB VFORTLIB
```

Go Exec

```
ERASE POSITION FILE
FILEDEF SYSIN DISK NC DATA
FILEDEF SYSPRINT DISK POSITION FILE
FILEDEF TERMIN TERMINAL
FILEDEF TERMOUT TERMINAL
FILEDEF OFFTOOL DISK OFFTOOL FILE
LOAD PLOT10 (START
```

Plot Exec

```
FILEDEF 1 DISK POSITION FILE (RECFM V
FILEDEF 9 TERMINAL (RECFM F
LOAD IGL (START
```

TOOL PATH PLOTTING USING PLOT10

by

P. SARAVANA PRASAD

B.E. (Mechanical Engineering)
College of Engineering, Guindy
Madras, India, 1983

ABSTRACT

for a

MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Industrial Engineering

KANSAS STATE UNIVERSITY

Manhattan, Kansas

1985

ABSTRACT

The project undertaken is to develop a software system to draw a tool path for any given machine code program for the Pratt and Whitney machine at Kansas State University.

At present, after a machine code program is written to machine a part on the Pratt and Whitney, it is loaded onto the machine and run. This process involves wastage of time and material resources if the program has errors. In order to reduce the long hours spent at the machine, a software system which plots the cutting path the cutting tool would have taken, had the same instructions been input to the machine tools, has been developed. Software systems that permit such simulation are called tool path plotting systems.

There are two approaches to obtaining this tool path plot. One approach is to buy commercially available dedicated software and hardware packages to plot the path. The other approach is to develop a new software system which utilizes the existing hardware and software packages. Since the latter approach is cheaper, it was chosen for this project.

The software system developed checks the machine code program provided by the user for syntax and logical errors. If there are any errors in the program, the user is informed via a report. When it is free of errors, a two-dimensional plot of the tool movement is obtained on a graphics terminal. This plot can be compared with the desired path for validity before loading the program in the numerical control machine.

This software system can be improved in future by providing more options such as a three-dimensional viewing option. Also an interface can be designed with the HP 7475A Plotter available in the Department of Industrial Engineering to obtain a hard copy of the tool path plot.