COMPUTER ASSISTED PART-PROGRAMMING FACILITIES FOR NUMERICAL CONTROL

COMPUTER ASSISTED PART-PROGRAMMING FACILITIES FOR NUMERICALLY CONTROLLED MACHINE TOOLS

by

ERIC V. KLAASSEN, B. Eng.

Thesis

Submitted to Faculty of Graduate Studies

in Partial Fulfilment of the Requirements

For the Degree

Master of Engineering

McMaster University

November, 1971

MASTER OF ENGINEERING (1971)
(Production Engineering)

McMASTER UNIVERSITY
Hamilton, Ontario

TITLE: COMPUTER ASSISTED PART-PROGRAMMING FACILITIES FOR NUMERICALLY CONTROLLED MACHINE TOOLS

AUTHOR: Eric V. Klaassen, B. Eng. (McMaster)

SUPERVISOR: Professor M. C. de Malherbe

NUMBER OF PAGES: 187

PREFACE

Numerical Control part-programming consists of organizing technological and kinematic data into a format which may be read and processed by a machine control unit, the purpose of which is to control the movements of a machine tool cutter relative to a workpiece. There are basically two approaches to part-programming, manual and computer-assisted. Manual part-programming infers that all data is generated and organized by the programmer. In computer assisted part-programming the part dimensions are inserted into a processor which then generates data and puts it into the proper format for control tape generation.

In this thesis three areas of the NC technology are investigated for the purposes delineated below:

1) A survey of current NC computer-assisted part-programming languages.

- this study was undertaken to acquaint the author with the computer programs presently available to industrial users, to evalute them in terms of uniqueness, capability, application and availability, and to investigate their modes of operation.

2) A tape format translator program to convert control tapes encoded in Universal or Word Address formats into the MOOG fixed block format.

- this was a project, the aim of which was to restructure the data from the more conventionally encoded tapes (Universal, Word Address) to the proprietary MOOG format, using the mini-computer.

Problems requiring a solution such as the one outlined here could arise in a shop where two or more different machines must handle the same work load.

3) A time sharing post-processor system for the MOOG 3-Axis machining center in the Engineering Machine Shop at McMaster.

- presently this machine is programmed either manually or by special-purpose programs which generate mathematically defined contours. It is desirable therefore to have a simple language which is easily and quickly learned, so that regular point-to-point work, for which the machine is best suited, may be prepared for by computer. A time-sharing system is offered by CGE in Toronto which is presently on-line to the terminal in the Mechanical Engineering Department. This system was modified so that all the facilities available on the MOOG may now be employed for regular point-to-point work.

In terms of content, the thesis has been organized into six sections. In the first section the history and development of NC technology have been discussed to establish a background for subsequent work.

In section 2 the background of the APT system is discussed briefly. This discussion goes on to include the system structure of the updated version, its operation and use, and its present status as the "de facto" standard in the area of NC part-programming languages.

The investigation of the other languages ensues with AD-

APT, EXAPT, SPLIT and AUTOSPOT probably sharing the bulk of the non-APT workload. Some space is given to discussion of the time-sharing systems, acclaimed by some as the answer to the delays caused by batch processing and the expence of in-house software. With this in mind, a brief comparison of conventional post-processors and time sharing systems is given. Post-processors are the software items which translate all the switching functions and other non-movement functions from codes into the proper signals for machine tool operation.

The tape format translator and the work done in the development thereof is described in Section 3. The computer hardware used was a PDP 8/L data processor available on a time-rental basis from the Electrical Engineering department at McMaster. The software which was developed cannot be considered complete as the time and effort required to develop a completely interchangeable program were considered to be beyond the scope of this project. The program is written in PAL III, a proprietary assembler language which effectively reduces core requirements to about 1K 12 bit words.

The operation and organization of the time sharing post-processor is discussed in Section 4. An operating example is also given to illustrate the input and output requirements of the operating system.

Section 5 reviews some of the multiple applications of Numerical Control. This includes a brief discussion of an East

German display where a workpiece is handled and machines on several different worktables as well as an installation where a PDP 8/S computer directs the motions of 8 machine tools simultaneously. Both of these are applications of DNC or direct numerical control, which dispenses with the control tape entirely.

In Section 6 the work done in the thesis is evaluated in terms of its usefulness and some recommendations regarding the design of machine controller units are made, as well as some criterion for the purchase of present day NC systsms.

ACKNOWLEDGEMENTS

The author would like to express his appreciation for the assistance and direction of Professor M. C. de Malherbe throughout the duration of this project.

The assistance of Mr. J. R. Arrowsmith as a resource person was greatly appreciated, as was the technical and programming assistance of Mr. Ian Cuthbertson, of Canadian General Electric.

The author also is deeply grateful to his wife, Mrs. Mary Anne Klaassen, for her work in typing this manuscript.

TABLE OF CONTENTS

APPENDICES

LIST OF FIGURES

COMPUTER ASSISTED PART-PROGRAMMING FACILITIES FOR NUMERICAL CONTROL

1.0 INTRODUCTION

1.1 History.

The concept of controlling a mechanical device through the use of a digitally coded series or set of instructions stems back as far as 1650. At that time the application involved rotating drums on which were mounted pins so positioned as to ring chimes automatically, in a manner analagous to the music boxes found in children's toys today. Later on, around 1725 the European textile industry developed knitting machines which would produce patterns according to instructions supplied in the form of a pre-punched set of cards.

In 1863 the first automatic piano player was patented by one M. Fourneaux whose principle of operation was blowing air through a perforated roll of paper, ultimately actuating the piano's keys.

Numerical control of industrial machinery was confined to the textile industry until about 1930 when a patent was awarded Max Schenker for a control system employing the punch data concept whose function it was to control a machine tool.

These early control systems were exclusively mechanical or pneumatic, hence they were slow, occupied a great deal of space, and were not reliable. The whole concept therefore required an

electromechanical system in order that it could be more practically implemented. It was suspected that development to this end was being carried out in Germany during WW II. An almost parallel development evolved in the United States where John Parsons of Traverse City Michigan conceived a system for numerically controlling a jig borer to manufacture inspection templates for helicopter blades.

About this time the U.S. Air Force was encountering difficulty in meeting production schedules due to the extremely complex shapes which were required for modern aircraft and missile manufacture. A more flexible input for the production machinery was needed, and subsequent investigations brought to light the work that Parsons Corp had done a few years before. In 1948 a development contract with Parsons was signed to pursue the possibilities of numerical control. In 1949 MIT joined Parsons in the project and was awarded a development contract in 1951. The first successful demonstration of a 3-motion milling machine was observed in 1952.

1.2 Computer Assistance in N/C Programming.

The prime function of these early machines was to manufacture contours for the aerospace industry, contours which were difficult to define simply, hence the control unit often required a great deal of data to successfully execute the desired cuts. Early control systems used vacuum tubes in their electronics.

These caused excessive heat to be generated, were very bulky, and again, were not reliable. Monumental strides in computer hardware technology have had the effect of reducing the physical size requirements of the machine control units while multiplying their capabilities.

A prerequisite to making today's systems work is of course, rapid generation of precisely coded, correct data. The technology is already available in the form of modern, high-speed digital computers. The ability of these devices to process large amounts of data, perform involved calculations and generate controlling media is well known, and well suited to the requirements of the Numerical Control concept. Just how the computer accomplishes these tasks is, of course, subject to human direction as each of these machines can only execute when, and how it is told to do so. The result of this technology explosion has been an explosion in approach to computer assisted programming of NC machine tools. It is difficult to say how many languages exist today for part-programming workpieces on NC machines. Some of the more popular ones will be dealt with in detail while others may or may not be mentioned due to their popularity or lack thereof.

Not only are these variations in programming computers for NC machine tools, but the machine tools themselves differ widely according to the whims of the manufacturers, the task to which they are applied, and the in-house requirements of various shops. Differences also exist even in the input coding and for-

mats into the control system but these problems may be overcome again through a bit of recent and very adaptable technology, the mini-computer.

Finally, many computer languages, in order to remain as standard and flexible as possible, require an intermediate program to reprocess the data for a specific machine tool. Here again, various approaches have been taken, depending on complexity and requirements and great pains are taken to ensure that the capabilities of writing intermediate programs (called post-processors) remain in the hands of a select few for financial and prestige reasons. Systems do exist, however, which facilitate the compiling of such a program and require a minimum of computer programming skill.

2.0 PROGRAMMING FOR NUMERICAL CONTROL

2.1 The NC System.

In the technology as we know it today, piece-part manufacture invariably begins with a part drawing. Generally, NC technology also makes use of this approach although there are certain developments such as the use of the light pen which can be put to fairly effective use in the generation of data for piece-part manufacture. Even these developments however, only generate data refering to the size and shape of the part, and not to the way it is made.

In order to gain a perspective of the various areas of NC technology and how they relate to one another, consider first the basic system of manually coded and prepared tapes. Figure 2.1 shows the flow of information from print to data form, to tape punch and ultimately through the control unit, to the workpiece. What the figure does not show, however is the relative magnitudes of each of the designated tasks. The programming of the data form requires familiarity with a variety of considerations on the part of the programmer. Some of these considerations are:[32]

A. Familiarity of the characteristics of the machine tool-numerical control unit combination

B. Axis nomenclature.

C. Tape preparation equipment

D. Specific knowledge of the input requirements of the control

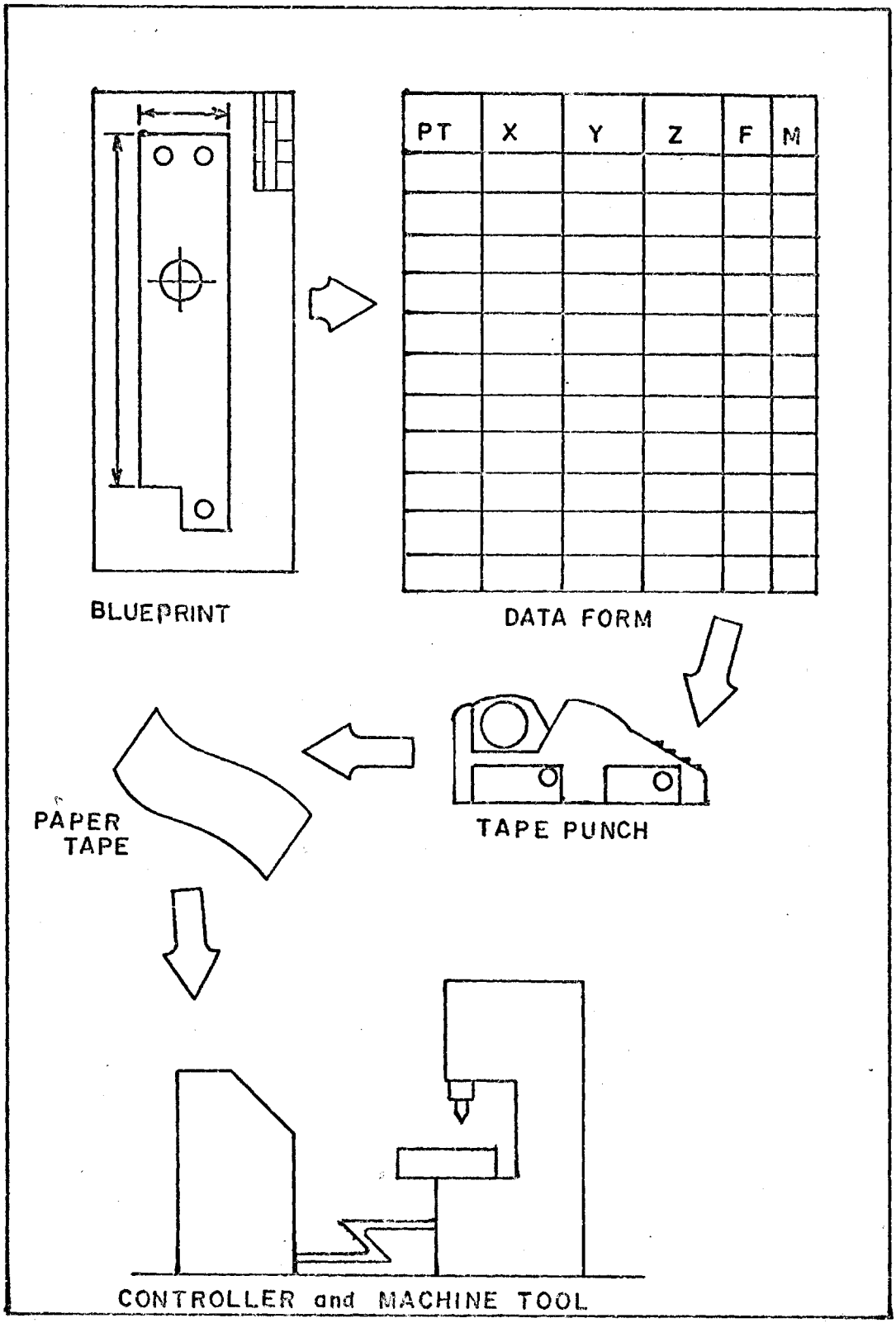


FIGURE 2.1 MANUAL TAPE PREPARATION.

unit. This would include such items as:

1. Tape Format
   a) word address/Tab sequential, etc.
   b) Maximum and minimum number of digits permissible for all codes.
   c) Maximum block length.
   d) Programming sequence within a block.
   e) Miscellaneous function coding and their specific functions.
   f) Preparatory function coding and their specific functions.
   g) Spindle speed coding.
   h) Feedrate coding.
2. Special Programming Parameters:
   a) Absolute or Incremental System.
   b) Linear or Circular interpolation.
   c) Pulse weight considerations.
   d) Acceleration/deceleration considerations.
   e) Tape reader limitations.
   f) Cutter path calculations.
   g) Other special features or options that may affect part programming and tape preparation.

It is easy to see that in manual tape preparation, the part programmer must have a full understanding of how the system works, what the inputs are and at the same time perform all the

calculations for tool offsets, tool path, and so on. The job is simplified to an extent by a certain degree of standardization with regard to machine activity functions as shown in fig. 2.2 and 2.3. These machine activity functions are Electronics Industries Association (EIA) and NASA standards which have been set so that relationships do exist between control systems.

The Miscellaneous functions are generally switching codes, used to turn an auxiliary part of the machine on or off or to select a mode of operation for these auxiliary parts. The Preparatory or "g" function on the other hand calls up a predetermined cycle of the operation of the machine itself which is built into the machine control unit.

From the listing of considerations given above, it is obvious that many of the tasks required of the programmer may be done with much more efficiency and speed if they are done by computer. The part programmer, in such an updated system, remains the key to the whole operation but his role has changed somewhat. Figure 2.4 shows a computer aided part programming system.

This is not to say however that the part programmer must now be a less knowledgeable individual. He now has the responsibility of communicating with the computer in a language that it too can understand, however knowledge of the details of the machine control unit is no longer a prerequisite. He must still have an appreciation for setup and be able to specify feedrates, spindle speeds, tooling and so on (although some developments in recent

PREPARATORY FUNCTION CODING

CODE	FUNCTION
G00	Point-to-point positioning
G01	Linear interpolation (normal dimensions)
G02	Circular interpolation ARC clockwise
G03	Circular interpolation ARC counterclockwise
G04	Dwell
G05	Hold
G06-07	Unassigned
G08	Acceleration
G09	Deceleration
G10	Linear interpolation (long dimensions)
G11	Linear interpolation (short dimensions)
G12	Unassigned
G13-16	Reserved for axis selection
G17	XY - plane selection
G18	ZX - plane selection
G19	YZ - plane selection
G20	Circular interpolation ARC CW (long dimensions)
G21	Circular interpolation ARC CW (short dimensions)
G22-29	Unassigned
G30	Circular interpolation ARC CCW (long dimensions)
G31	Circular interpolation ARC CCW (short dimensions)
G32	Unassigned
G33	Thread cutting, constant lead
G34	Thread cutting, increasing lead
G35	Thread cutting, decreasing lead
G36-39	Reserved for control use
G40	Cutter compensation cancel
G41	Cutter compensation - left
G42	Cutter compensation - right
G43-49	Cutter compensation if used, otherwise unassigned
G50-59	Unassigned

FIGURE 2.2

....continued....

G60-79	Reserved for positioning only
G80	Fixed cycle cancel
G81	Drill Cycle
G82	Peck or Dwell cycle
G83	Tap cycle - constant pitch
G84	Tap cycle - constant lead
G85	Bore cycle
G86-89	Reserved for cycle uses only
G90-99	Unassigned

FIGURE 2.2

MISCELLANEOUS FUNCTION CODING

CODE	FUNCTION
M00	Program stop
M01	Optional stop
M02	End of program
M03	Spindle on clockwise
M04	Spindle on counterclockwise
M05	Spindle off
M06	Tool change
M07	Mist coolant on
M08	Flood coolant on
M09	Coolant off
M10	Clamp
M11	Unclamp
M12	Unassigned
M13	Spindle clockwise and coolant on
M14	Spindle counterclockwise and coolant on
M15-16	Motion
M17-29	Unassigned
M30	End of tape
M31	Interlock bypass
M32-35	Constant cutting speed
M36-39	Unassigned
M40-45	Gear change if used
M46-49	Reserved for control use only
M50-59	Unassigned

FIGURE 2.3

years are making inroads into this area).

2.2 The APT System.

In perhaps more cases than not, the part programmer will communicate with the computer in a language called APT which is a foreshortened version of Automatically Programmed Tools. This is by far the best known of all the numerical control programs and forms the basis for most of the other computer assisted programming languages which are in use today.

2.2.1 The History of APT

The acronym APT stems back as far as 1944 to the work done by the Parsons Corporation in their efforts to develop a numerical control system for the manufacture of templates for propeller and helicopter blade inspection. At that time a prototype program was written for the M.I.T. Whirlwind computer to perform the highly involved and numerous calculations required to define the contours of the aerofoil sections in question. This program was called the "Automatically Programmed Tool" system and was used later in a modified form to program the early numerically-controlled machine tools developed at M.I.T.

The concept and structure of the APT system in its present form however was developed by M.I.T. under Air Force contract during 1956-57 [33]. The intent of the development was to provide a software system which could readily be extended into a variety of industrial applications. The magnitude of the task was such that the concerted efforts of twenty member companies of the Aerospace

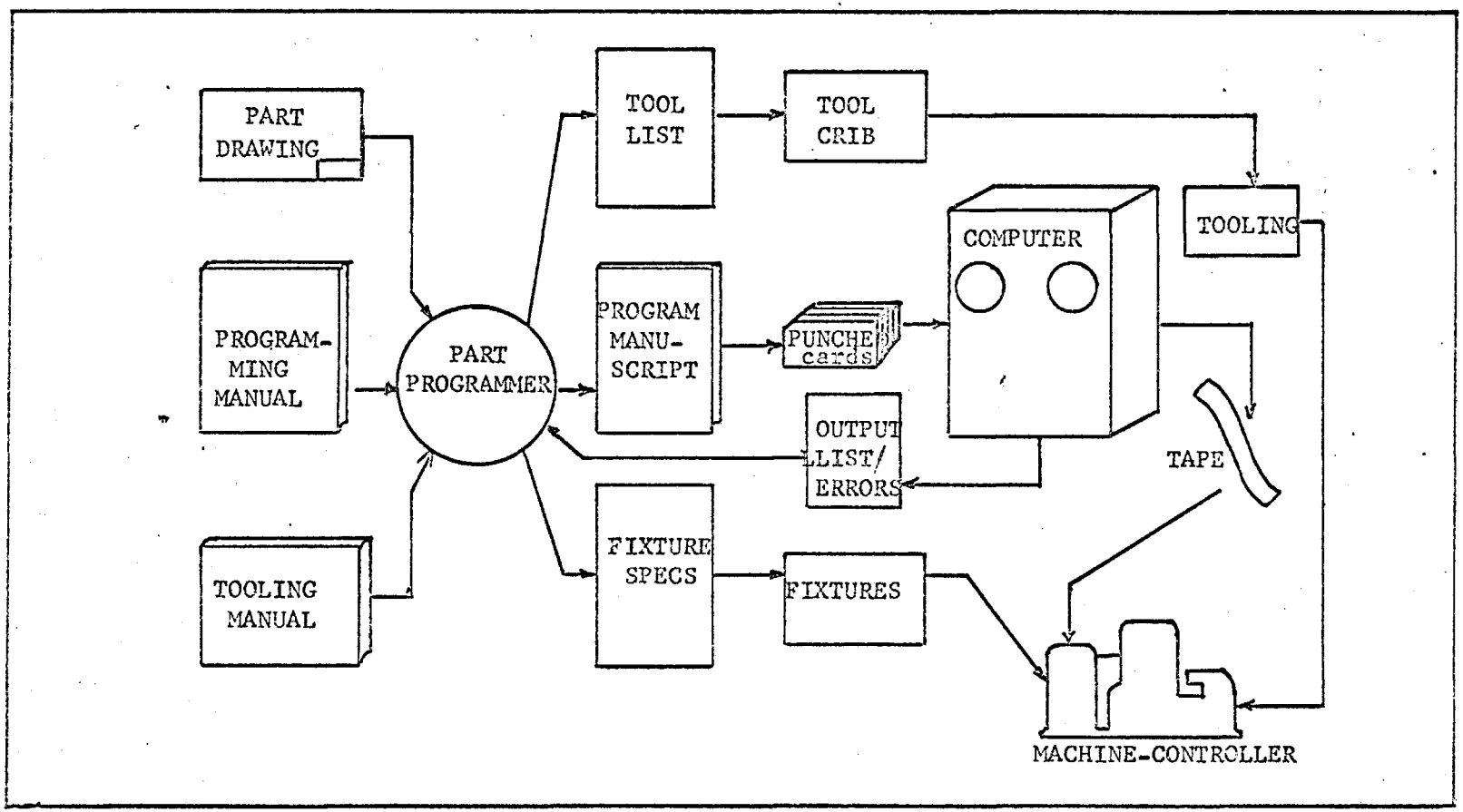


FIGURE 2.4 COMPUTER ASSISTED PART-PROGRAMMING SYSTEM.

Industries Association (AIA) were required, along with the guidance and direction of M.I.T. Nevertheless, in 1958 a second version (APT II) was released.

Continued development by this group resulted in the release of the updated APT III version in 1961. Also in 1961, the AIA decided to establish the APT Long Range Program in order to continue the necessary maintenance and further development of the system and also to extend the benefits of APT to other companies. The new administrator of the program was the Illinois Institute of Technology Research Institute (IITRI), which is still responsible for the updating and expansion of the APT language today. The program is sustained by annual fees paid by the participating members of the ALRP who in return decide the direction of future activities and have access to the most recent developments and modifications.

Since 1964 a policy of "staged release" was adopted by the ALRP wherein a new system is made available to the public two years after its release to use by members. Even without this, however, it is significant that present members of the ALRP include such corporations as Control Data Corporation, IBM, Honeywell, General Electric and others. Hence when computing facilities are leased from these companies, or alternatively, a time sharing agreement is made, many of the latest developments in the APT system may be available to the user.

At McMaster there are now two independent outlets where the

APT system may be used. One is the Multiple Access system available on a time-shared basis, and the other is the CDC 6400 in the university's own computer facility. This latter facility does not as yet have all the proper hardware for successful implementation.

2.2.2 The Structure of the APT System

At first, the APT system was written in an assembler language. This of course reduced the core requirements in the computer however it also required that the programmers have a full and intimate knowledge of the machine and severely restricted its computer independence.

IITRI took upon itself the task of restructuring the program and language in such a way that the system is now "open ended" [23], that is, technological improvements may now be easily incorporated into the program as they are developed. This is done by, in some cases, inserting empty subroutines which merely are filled in as the need arises. It is also signigicant that the bulk of the system is written in Fortran which, although it requires more core, is much more easily changed and is a more popular form of communication than are the assembler languages which change from one company's machine to another.

The 'new system' as it is called assembles and calculates simultaneously, hence if there is an error in the part program, a number of calculations will have been carried out, and time perhaps wasted. It is however more efficient than the interpretive approach where the program is first assembled, and then the subroutines are

called up as they are needed.

2.2.3 The APT System at McMaster

As of August 13, 1971 McMaster has its own in-house version of APT namely 6400/6600 APT. The system structure of 6400/6600 APT is similar to that of Standard Version APT III. By sections, the program operates as follows: [34]

(I) translates the part program statements and reduces geometric definitions to their canonical forms;

(II) calculates consecutive tool positions.

(III) prints, copies, and performs transformations on tool location data.

Section IV although included in the program is exclusively a calling subroutine for the post processor designated in the part-program, if the machine control tape is desired. The 6400/6600 APT system does not include post-processors; these must be supplied externally by either the machine-tool builder or the user.

The 6400/6600 APT system requires significantly less core than systems such as IBM 360 APT partly due to the larger word size of the 6400 (60 bits) and the extensive use of overlays [35]. Rather than the need for up to 256 K (IBM 360 APT) the CDC system can run with 65K. Execution times are compatible with Fortran programs of comparable complexity. In many cases the names given to the various subroutines describe their function.

The input to the APT system is the previously mentioned part program. Output is a cutter-location tape (CL tape), a file containing successive cutter location coordinates and other control information ready for post-processing. Printout consists of part program, diagnostics, and the edited CL-tape if requested. The format of the CL-tape is dictated by the ALRP so that the rules for post-processor writing need not be oriented to a specific computer, rather to a specific machine.

2.2.4 Programming in APT [34].

The APT language shares many elements with other high-level computer languages (e.g. Fortran, Algol, etc.), that is punctuation symbols, constants defined symbols, Hollerith information and APT vocabulary words are all used. APT vocabulary words in most cases resemble their English equivalents. Examples are:

GODWN	go down
GORGT	go right
ELLIPS	ellipse
CIRCUL	circular interpolation
JUMPTO	jump to

APT manuals generally differentiate between certain types of words, i.e. they may be Major or Modal. Major words specify the general function of the statement in which they occur. They appear in the first section of the statement and are usually accompanied by parameters completing their meaning. An example of

a major word is:

LINE/ x coord, y coord, z coord.

Modal words generally specify conditions to remain in effect until countermanded. These may or may not be major in format. Tolerances, cutter diameters and so on are modal as are instructions for positioning the tool on the right or left of a given line.

Other APT words may be classed as minor and "postprocessor". Minor words are used in conjunction with, and to assist in the definition of, major words. Postprocessor words are ignored by the APT system and are carried through to the system CL Tape where they will be used by the postprocessor. COOLNT/ON is major in format but is a postprocessor word instructing the machine to be turned on or off.
OFF

As in all computer programming, the correct punctuation must be employed at all times. The common punctuation symbols retain for the most part, their conventional English or Fortran usage. Departures from this norm are the Line Continuation symbol ($) and the End of Statement symbol ($$) the latter of which permits a remark or comment to be inserted on the same card with some regular programming.

The symbols used in APT may represent scalar variables, geometric entities, statement identifications, Macro names, or arrays. The concept of the scalar variable is generally known. The geometric entity refers to some point, line, matrix or any

other surface type recognized by the APT system. Some commonly used examples are:

P1 = POINT/1,2,3

C5 = CIRCLE/CENTER, P1, RADIUS, 2.5

A macro name refers to a sequence of APT statements. The macro is in its operation, similar to a function subroutine the difference being that it is found inside the part program. The use of statement identification symbols is characterized by separating the statement and the symbol by a right parenthesis. Arrays may also be grouped under one name.

All of the above symbols may be subscripted if necessary or desirable. FORTRAN rules for subscripted symbols apply in kind to use in the APT system.

The Part Program is a sequence of APT statements. The basic structure of a part program generally consists of the following:

Part identification
Environment and description
Geometric definitions
Macro definitions (if any)
Production commands including
    Motion statements
    Computing sequence
    Input/Output statements
    Postprocessor commands
Termination

Of these, the part identification and termination are fixed. The first statement of a part program must invariably begin with a PARTNO statement in the first 6 columns of the card.

Following this the programmer may identify the part in any suitable format. The program must always end with a FINI statement, which terminates processing and activates the output commands.

Looping capability is incorporated into APT, controlled by the words LOOPST and LOOPND. Some computational rules apply in the control and exit from these loops in the form of conditional transfers [IF (J) Z1, Z2, Z3,] where J may be less than, equal to, or greater than zero. Unconditional transfers (JUMPTO) do not have this flexibility. These statements (IF, JUMPTO) may only be used within a loop.

APT geometry consists of points, lines, circles and other types of surfaces. In cases such as points lines and circles, there may be ten or more variations in the definition form. Vectors are used for extablishing directions and defining surfaces. At present there are two classifications of APT surfaces, Analytic surfaces and Large Surfaces. The analytic surfaces are those which generally can be defined by a single equation. These are designated as follows:

LINE	line
PLANE	plane
CIRCLE	circle
CYLNDR	circular cylinder
ELLIPS	ellipse
HYPERB	hyperbola
CONE	cone
GCONIC	general conic
LCONIC	loft conic
SPHERE	sphere
QADRIC	general quadratic

In some instances simple mathematical equations are not adequate for the definition of a workpiece. When this is the case, the large surface capabilities in APT may be employed. They are represented by the words POLCON,TABCYL, and RLDSRF.

POLCON refers to a polyconic surface which is a continuous family of conic sections in parallel planes. The shape of the conic is regulated by a set of polynomial curves.

TABCYL (tabulated cylinder) is a surface generated by a moving line, parallel to some fixed base, and travelling along a space curve or directrix which is defined by a set of points known to lie on the surface. Interpolation formulae are employed to generate the directrix.

RLDSRF calls up a ladder type surface generated, again by a moving straight line which travels along two defined space curves.

Further geometric capabilities to assist in point-to-point programming, and also in continous path, are matrices and patterns. Associated with these is a REFSYS statement which is employed when a change in coordinate systems is required.

The matrix routine however is fixed and only applies to an array of numbers having three rows of four. Its general usage is in the transformation of coordinate systems. Ordered sequences of points are defined by the PATERN/statement. The patterns may be defined as linear, circular, parallelogram and random. Here again, a number of auxiliary tool motion commands are available

to assist in programming a desired sequence for the tool. Examples are OMIT, AVOID and THRU.

The MACRO sequence mentioned earlier is one of the most powerful tools that the APT programmer has at his disposal. Technically a macro definition is comprised of: MACRO/statement, macro sequence, and a TERMAC statement. If the program for the machine tool requires a cycling through some specified number of operations wherein one or more parameters change from cycle to cycle, the programmer set up the proper sequence of events inside the MACRO, leaving those parameters which vary cyclicly as constants. The MACRO statement could then have a format as given:

MA = MACRO/ a_1, a_2 , a_3 -----

In the above example MA denotes the macro name and the a_i are the variable parameters in the APT sequence of the macro. Having listed the entire macro, it may be called in this example as follows

CALL/MA, a_1 = 2.5, a_2 = P7, a_3 = C6 -----

Care must be taken when using and writing macros to ensure that: (a) macro definition precedes the CALL in a part program;

(b) definition in another part program requires use of a READ command to bring it into the current part program;

(c) the macro used in another part program is on the APT library tape.

From the prior discussion of APT geometry, it is obvious that the system is designed for continuous path milling, however, as also stated before, geometric facilities are also available for

point-to-point work. Motion instructions consist of a GO --- statement which initiates motion in a direction determined by the part programmer as if he were sitting astride the tool and driving it around the specified part geometry.

In principle the tool motion (in continuous motion work) is determined by three control surfaces, which may be real or mathematical. One of these is the part surface, generally the surface normal to the tool axis. The drive surface becomes the line or space curve the tool is directed to follow, and the check surface is the space curve at which the tool changes direction. Normal orientation of these curves is shown in Figure 2.5. Many variations of these three surfaces exist as may be seen in Figure 2.6.

Once the programmer understands these few concepts he is ready to begin writing trial programs. Self-teaching APT manuals exist which greatly accelerate the learning process. One example is reference 33, prepared by the ALRP.

Central to the concept of computerized numerical control is of course the capability of calculating coordinate locations and/or scalar values. Arithmetic operations may be designated in what amounts to a standard FORTRAN format and the heirarchical rules of FORTRAN also apply. To assist in this computational capability certain mathematical functions are recognized:

SINF	-sine of an angle expressed in degrees
COSF	-cosine of an angle expressed in degrees
ATANF	-angle expressed in degrees when tangent is given

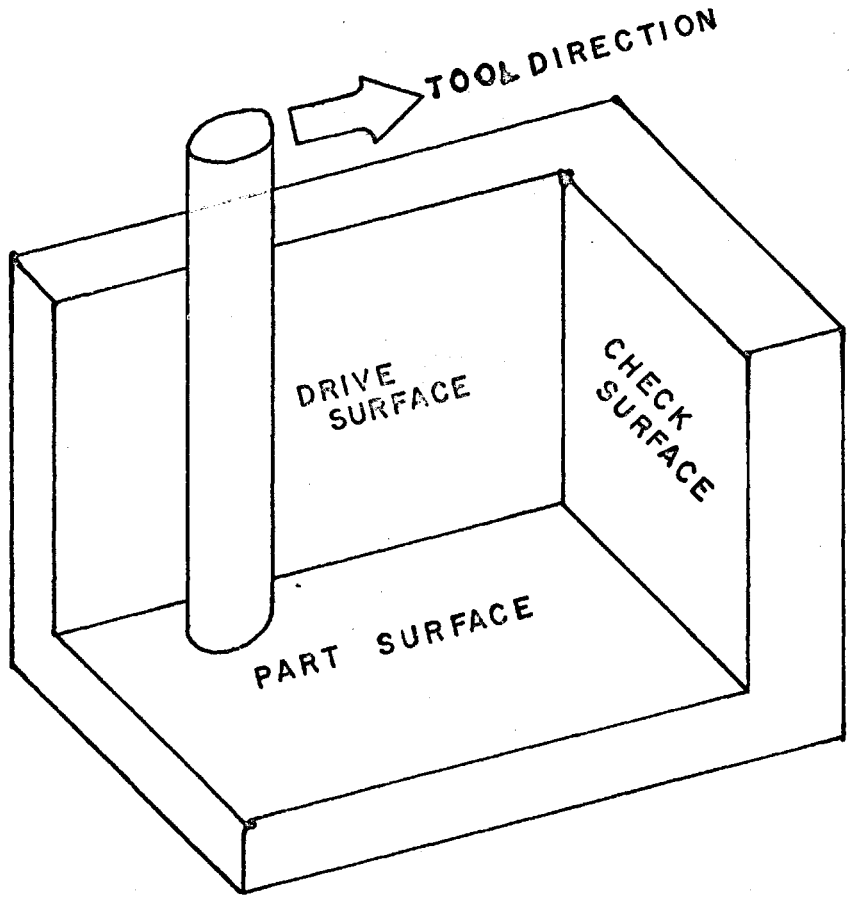
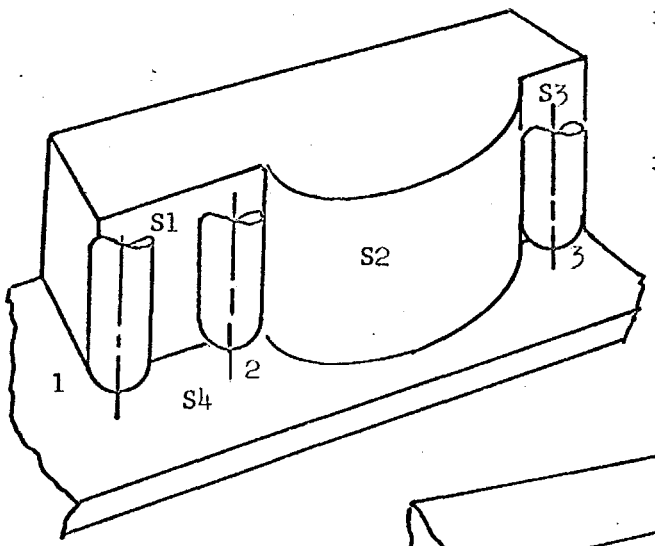
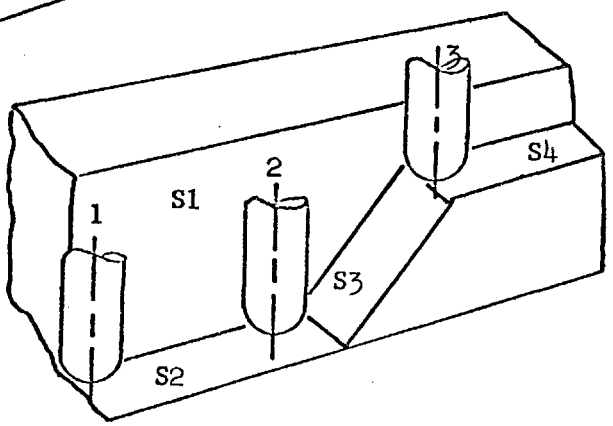


FIGURE 2.5 RELATIONSHIP OF PART, DRIVE, AND CHECK SURFACES IN THE APT SYSTEM



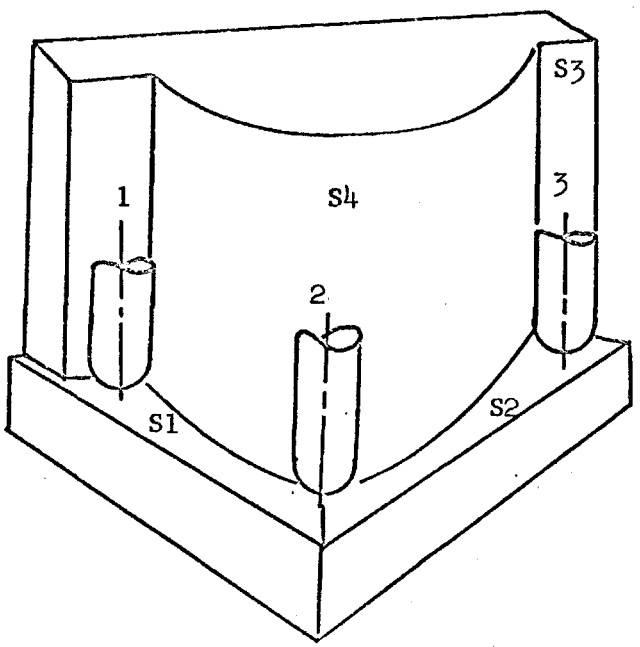
from 1 to 2
DS is S1
PS is S4
CS is S2

from 2 to 3
DS is S2
PS is S4
CS is S3



from 1 to 2
DS is S2
PS is S1
CS is S3

from 2 to 3
DS is S3
PS is S1
CS is S4



from 1 to 2
DS is S1
PS is S4
CS is S2

from 2 to 3
DS is S2
PS is S4
CS is S3

FIGURE 2.6 SOME VARIATIONS OF PART, CHECK AND DRIVE SURFACES IN THE APT SYSTEM

SQRTF	-square root of a non-negative number
LOGF	-natural logarithm of a positive number
EXPF	-value of raised to a power
ABSF	-absolute value of a number
LNTHF	-length of a vector
DOTF	-Scalar or dot product of two vectors

It is possible to incorporate an arithmetic operation into another APT statement by enclosing it in parenthesis in a manner similar to imbedding a geometric definition. This is called "Nested Computing".

A full range of input/output instructions are part of the APT system. These instruction statements enable the programmer to:

Punch geometric and macro definitions
Read geometric and macro definitions
Print geometric definitions and scalars
Print the contents of the CL-tape
Create a tape for use on a plotter.

The APT system also contains a fairly comprehensive list of diagnostics, and it is here that the internal system structure comes to light. In Section 1 of the system an error occurs if the part programmer violates a language rule, exceeds system capacity, programs an infinite loop, presents false logic in the part program, or calls for an unavailable macro. Failure to successfully execute Section 1 terminates job execution and prints an error number.

In Section 2 failure may be due to errors in a part program or inadequacies in the calculation algorithm used by the system. The part program errors may consist of: describing an invalid cutter, specifying non-intersecting control surfaces;

using a GOFWD before a forward direction has been established, inadequate start-up information, and many other types. A failure in Section 2 produces a printout of the error message number, the card number on which the error was detected, and the error message itself. Additionally a dump of certain internally stored information is given. An error in Section 2 generates two possibilities:

1) Cessation of further computation

2) Warning diagnostics followed by continued processing at the point where the system can reestablish a valid tool position and vector direction.

Section 3 diagnostics include language violations in COPY, INDEX, TRACUT, PLOT and VTLAXS statements, too many index markers on nested COPY commands or an illegal transfer into a COPY loop.

Other errors may be input/output errors in which case the local computer facility would supply its own system diagnostics. Programmed debugging aid are a part of the APT system. Debugging section 1 may be accomplished by a DEBUG/ON statement. Section 2 debugging may be facilitated by a COMDMP statement which produces an edited printout of the entire contents of the Section 2 common area.

Some examples of APT III diagnostics are shown in Figures 2.7 and 2.8

2.2.5 Strengths and Weaknesses of the APT System.

The APT system, as previously stated, is the oldest, probably the most sophisticated, and without a doubt, the most widely

PASS 1 ERROR DIAGNOSTICS

Error Number	Error Comment
1	Variable symbol contains more than 8 characters.
6	Statement identification doubly defined.
26	Entry following slash in ZSURF not a plane.
65	Only one vector in dot function argument.
89	Incorrect number of scalars used in definition.
132	Invalid punctuation or vocabulary word. Check syntax.
154	Number of names exceeds number of values in canonical form.
183	Too many unnamed nested definitions. Assign symbolic names.
201	Illegal macro definition in loop or macro.
231	Macro nesting exceeds system capacity.
251	Insufficient internal storage for array.
355	Vocabulary word used as synonym symbol.
379	More than 20 points given in POCKET statement.
401	First word of IF or JUMPTO statement not a vocabulary word.
504	Too many postprocessors called.
804	Number of points in pattern is negative.
2002	Not enough data cards to complete READ list.
	DEGREE OF EQUATION EQUAL TO OR LESS THAN ZERO.

FIGURE 2.7 Selected Pass 1 APT error messages.

SECTION II ERROR DIAGNOSTICS

Error Number	Error Comment
309	WARNING - TOLERANCE VALUES EXCEED ALLOWED NO. OF SURFACES. EXTRA VALUES IGNORED.
311	CUTTER NOT DEFINED PRIOR TO STARTUP CALCULATION.
414	CUTTER DEFINITION NOT VALID.
505	UNABLE TO DETERMINE INITIAL MOTION DIRECTION VECTOR OR ITS ORIENTATION.
704	DRIVE SURFACE AND CHECK SURFACE NOT TANGENT AS STATED.
802	CUT VECTOR ITERATION TO CHECK SURFACE FAILED.
2	WARNING - CUTTER IS MOVING AWAY FROM THE CHECK SURFACE.
843	CUT VECTOR ITERATION FAILED IN MOVE ALONG DS=PS PLANES RESTART COUNT EXHAUSTED.
882	WARNING-CANNOT FIND CRITICAL MIDDLE POINT. CUT VECTOR ACCEPTED.
903	5-AXIS ITERATION DID NOT CONVERGE. SEE COMPUTER PROGRAMMER.
1003	WARNING - CANNOT CALCULATE SATISFACTORY TOOL NORMAL.
1083	CAN NOT FIND DIRECTION VECTOR THAT INTERSECTS THE SURFACE.
1201	TRYING TO FIND A UNIT NORMAL TO A POINT, VECTOR, OR MATRIX.
3002	TOOL CANNOT BE ITERATED INTO CONTROL SURFACE.
3507	ANGLE BETWEEN TOOL AXIS AND POCKET PLANE NORMAL IS TOO LARGE.

FIGURE 2.8 Selected Section II APT error messages.

used part-programming language for numerical control. It is likely also the most complex, and herein lie some of the chief disadvantages of the system.

APT has routines for elliptical cylinders, ruled surfaces, and polyconic surfaces, among others, which occur rather infrequently in most work including the aerospace industry. Routines such as this require the use of large digital computers, in some cases a core in excess of 256K bytes is needed. When only two dimensional contouring is required, the APT system is redundant in its operation because of its size and complexity. Point-to-point work is handled very inefficiently when it comes to fixed machining cycles such as tap and counterbore. In some of these cases it is almost simpler to program manually than in APT, however the aforementioned MACRO systems may be used to aid in this regard.

The language itself is not really geared to an engineer's or machinist's "jargon", on the contrary its similarity to FORTRAN in syntax and logic infers that it is conceptually a mathematician's language. Hence part-programmers are effectively required to immerse themselves in a different technology level in order to be effective users of APT. The structure of the program (see Fig. 2.9), although it appears to be segmented, is distinctly non-modular, the only detachable sections being the postprocessor which are not really part of the system. Considering the system size, this large, interrelated fixed format does not easily facilitate changes. Indeed, throughout the system listing there are

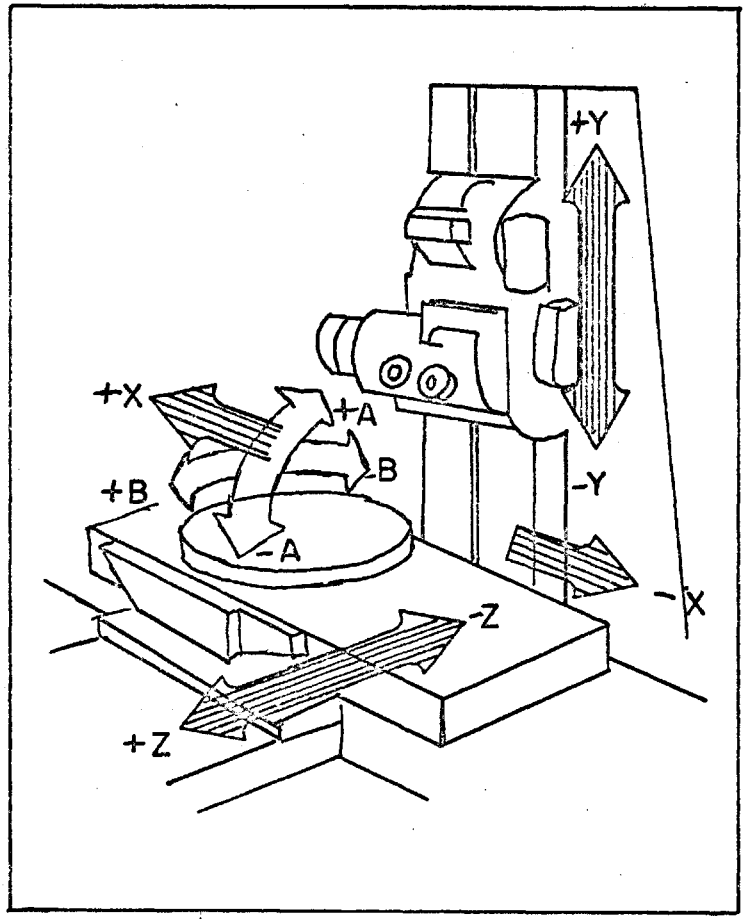


FIGURE 2.10 MACHINE TOOL AXIS DESIGNATIONS

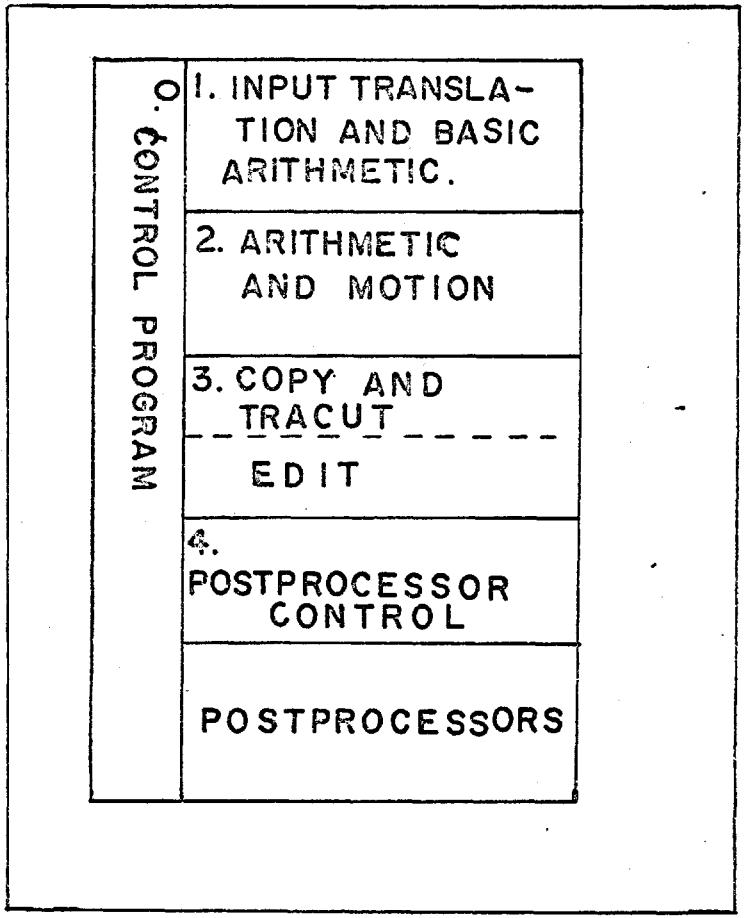


FIGURE 2.9 APT SYSTEM STRUCTURE

frequent warnings to refer back to the manual before making changes.

On the positive side though, it must be reiterated that the APT system conceptually was far-sighted, in some cases so much so that proper implementation was not realized until the advent of the third generation, integrated circuit computer hardware. Its capabilities are far reaching and it is perhaps best used when a stack of part programs is submitted in one run.

The language itself, a weakness because of its verbosity, and complexity, is nevertheless comprehensive. Through the use of the Synonym capability the programmer may if he chooses, substitute his own terminology for some of the geometric definitions. The strength and continued updating of the APT system is enhanced by the organization and support of the ALRP.

One of the objectives of the APT system was a machine-independent language. A penalty is incurred with a machine independent language in that another program, the postprocessor, is needed to interpret certain instructions which do not pertain to part geometry or continuous path tool motion. This effectively means that once the part program data has been successfully prepared by the APT processor, it must still be transformed completely into specific machine codes by another program.

An alternative approach to this problem is of course to structure the language for a particular range of machine tools which have similar coding requirements, resulting in a single-pass processor, one which needs no additional encoding for machine

utilization. This philosophy is generally termed "Single Processor" whereas the philosophy requiring a postprocessor is generally refered to as "General Processor". The problems of using a single processor language manifest themselves as soon as the user finds it advantageous to operate his shop with machine tools of more than one brand name. The problems of a general processor are primarily the initial cost of the software, which could be high.

The inadequacies of the APT language and its inherent disadvantages to some users of NC (due to size and complexity) have prompted many users to develop their own languages of lesser capability. These are in many cases derived in concept, and in vocabulary, from the APT system and hence bear the designation "APT-like" or "APT-compatible". Others are designed specifically for a particular range of machine-tool operations. This means that there will be languages of varying degrees of capability for handling point-to-point work, contouring work only, and some apply only to lathes.

This forms a basis for the study and comparison of other part-programming languages, what their capabilities are (geometrically speaking) where they are used, and if possible, where they may best be applied in industry.

2.3 The NC Language Explosion

As is frequently the case in instances where a problem emerges in different locations simultaneously, a considerable

variation in approach is likely to be realized from one location to another. This applies to the NC part programming languages as well which have grown in number until a complete review of all languages becomes a mere exercise in frustration. It should however be of some value to potential NC users to have at their disposal a list of the computer-assisted programming software available together with some indication of the relative capabilities and unique features of some of these systems.

In addition to catergorizing languages in terms of their philosophy, i.e. single/general processor, they may also be slotted according to capability. These capabilities may be segregated into:

a) 5 axis contouring,
b) 4 axis contouring,
c) 3 axis contouring with 4th axis positioning ($3\frac{1}{2}$),
d) 3 axis contouring,
e) 2 axis contouring with 3rd axis positioning ($2\frac{1}{2}$),
f) 2 axis contouring,
g) multi-axis point-to-point,
h) 3 axis point-to-point,
i) 2 axis point-to-point;

where an axis is defined as a degree of freedom in motion, (fig. 2.10)

2.3.1 Multi-Axis Contouring Languages

Multi-axis contouring languages are those which generally, have 4 and 5 axis contouring capability. These languages are highly sophisticated and require large computer systems for their implementation and execution. With a few exceptions, they are basically the variations of APT III as structured by the various computer hardware manufacturers for use in their large systems.

Thus the differences may be slight but the significance of some of the differences may be the deciding factor in a choice between systems.

2.3.1.1 6400/6600 APT

6400/6600 APT 2.2 Version 3.3 is the system described in some detail in section 2.2.2, supplied to users of Control Data Corporation computing hardware. It is fully compatible with APT III and, as CDC is a member of the ALRP, incorporates changes in the system through the use of its own internal update facility. This is the system presently available in McMaster's own Data Processing Center (See Appendix). It requires 65K words of core storage in the 6400 series computer.

2.3.1.2 360 APT

360 APT NC Processor (360A-CN-10X) Version 4 is the IBM proprietary system available to IBM users. It is perhaps the most up-to-date version available from any computing facility at the time of writing. The error diagnostics are the most comprehensive, the multi-axis capabilities are perhaps at a higher level of development and, due to IBM's extremely high market penetration, the system is perhaps more widely available than any other.

The 360 APT processor is generally well documented, (documentations available on request from IBM's local publications department) and its instructions for its use are given in detail, (see References 65, 66) core requirements of former systems of 360 APT were very high; 256-262K on a model 40 or larger. The new

system configuration recommended by IBM is: System 360 Model 2040 H (or 2050H or larger if high usage of the APT system is anticipated) with the appropriate arithmetic capabilities, storage facilities and input/output devices.

2.3.1.3 APT IV

Under development for some time, APT IV has at this time, not yet made any "public" appearance. It is supposedly going to be an advanced version of APT III. IITRI and the ALRP are engaged in its development.

2.3.2 Three Axis Contouring Processors.

Only a relatively small segment of the machine tool industry using numerical control needs the processor capability of the APT III system and its direct counterparts. Indeed, with a smaller vocabulary, programming a part may in some cases be simplified since the programmer now need only visualize one plane at a time as against 3-dimensional contouring capable on the other machine. Documentation on this class of processors is not easily available in every case, however sufficient information is available such that a few of them may be ably compared.

2.3.2.1 AD-APT

AD-APT is an acronym for Air Material Command Developed APT. This processor was written by IBM for the US Air Force when it was found that many suppliers simply could not comply with the demands of leasing or purchasing and maintaining a digital computer facility of the size required by the APT processor. Rather than being a

scaled-down version of APT, it has been redesigned so that it is in many ways completely compatible with APT, the difference being that it has a few teeth missing. These missing capabilities are the MULTAX or multi-axis capabilities. Hence there is room for a certain degree of interchangeability, AD-APT programs may be run on an APT system, however not all APT programs may be run on AD-APT. A more important reason for this flexibility rests on the interchangeability of postprocessors between APT and AD-APT, greatly reducing the software requirements for implementing a specific machine.

In operation AD-APT is similar to APT however the structure of AD-APT is much more flexible in terms of being able to expand or tailor the language capabilities to the particular needs of a manufacturer. This modular composition facilitates the use of smaller computer systems (IBM 360/2030) with up to 40K words of core.

In its present form offered by IBM, the processor includes the AUTOSPOT point-to-point positioning program, about which more will be discussed in a later section. It is also available from G.E., SDS, Honeywell, and other companies.

"The AD-APT language permits the specification of work operations involving points, circles, straight lines, and curves passing through sets of points. The calculating ability of the program includes arithmetic operations as well as the functions of sine, cosine, arc tangent, square root, exponentiation, tangents,

and logarithms." (36) Macros are used as in APT.

The scope of AD-APT is sufficient to handle most two dimensional contouring problems which account for the great majority of situations. It is thus perhaps more useful than APT due to its smaller requirements and in some of its versions, especially when coupled with AUTOSPOT, handles point-to-point work much more efficiently.

2.3.2.2 REMAPT

The REMAPT system is a version of AD-APT modified by the General Electric Corporation for use in their Mark II time sharing system. The REMAPT vocabulary has been altered and slightly reduced in size, however the facility of AD-APT still basically remains. Continued development on this package has expanded the point-to-point facility somewhat to the point where AUTOSPOT programs also may be run on this processor. As in AD-APT, a postprocessor is still needed in order to successfuly generate a machine control tape. The postprocessor may be for APT or AD-APT, with suitable modifications to match the minor changes in vocabulary in the point-to-point areas. The POCKET routine is not yet available on REMAPT although some moves toward including this facility exist.

Having noted some of the omissions in the REMAPT processor it must be emphasized that the package is nevertheless extremely desirable in terms of its availability. As with other time-sharing systems, the need for a dedicated in-house computer no longer

exists, the only requirements being a contract, acoustic coupler, teletype and telephone. The MARK II system is extremely comprehensive in its coverage and is considered by some to be superior to the batch processing systems of in-house computers. With time sharing, tapes can be made in minutes close to the shop floor. This flexibility may in many cases outweigh the limited vocabulary (with respect to AD-APT) for those who do not have computing facilities readily available.

In summary, REMAPT is a $2\frac{1}{2}$, axis contouring Ge al Processor language with 3 Axis Point-to-Point facility and is available on a time sharing basis.

2.3.2.3 UNIAPT

The United Computing Corporation (UCC) of California has taken a radically different approach in terms of the acutal hardware-software configuration. Recent developments in the data processing industry have resulted in a new, compact unit popularly dubbed as the "mini computer". These are extremely versatile units which generally do not require special air conditioned surroundings for their operation. UCC have restructured the APT processor and successfully implemented it on the Digital Equipment Corporation's PDP8 mini-computer. Other versions have been written for the IBM 1130 and the GA 18/30.

The core memory of these units is a mere 12K words as compared to the 256-262K required on large systems for conventional APT processing. This was acheived by extensive use of overlays,

and the use of the disk storage unit as an extension of the core memory.

UCC claims APT compatibility for all 2, 3 and some 4 and 5 axis programs. Some minor differences exist in terms of POCKET and PATERN definitions, areas in which APT III is not particularly impressive. The program does not have full 5-axis capability and is presently marketed as a 3-axis contouring system. MACROS are not yet available as in APT.

The advantages of this system may be cited as follows:

i) Immediate access to the company's own dedicated computer in close proximity to the work station;

ii) Low cost software and hardware compared to large systems;

iii) Potential use of the shop floor computer for Direct Numerical Control (DNC) completely bypassing the tape reader facility, and for other production control systems.

The cost of the system hardware is listed at $48,880 for a minimum configuration (as suggested by UCC) which includes a PDP8/I Computer with 12K memory and ASR 33 Teletype, Disk file (64K) and controller, High speed paper tape reader and Punch and the cabinet to house the hardware. This also includes the software at $12,000. Adding an arithmetic unit, card reader-punch and line-printer and a larger disk storage unit raises the cost to near $78,000. Leasing plans are also available. [11]

UNIAPT is a general processor program and still requires

a postprocessor for implementation. UCC claims to be able to supply these at a cost ranging from $500 to $1,500 as they are not compatible with other APT postprocessors. These systems are in use and significant productivity increases have been realized by some user firms.

2.3.2.4 SPLIT

A somewhat alternative approach to NC contour programming has been taken by the Sunstrand Machine Company. They have, in conjunction with IBM prepared a point-to-point contouring language called SPLIT (Sundstrand Processing Language Internally Translated). The rather major difference of the SPLIT processor is that it is individually machine oriented and does not require further processing in order to produce a machine control tape. The program is operative on the IBM 7090, 1620, 620 and 360. The language consists of 16 to 18 major statements and approximately 40 minor statements in a semi-pidgin English.

The SPLIT language is capable of handling up to five axes contouring but it is not as powerful as APT and is marketed as a $3\frac{1}{2}$ axis contouring language. It is, due to its machine oriented characteristics, a proprietary language basically restricted to Sundstrand's machine tool customers. However, Sundstrand have also prepared the system for use with Bendix Dynapath, G.E. Mark Century and Mark Series control systems, with the result that a machine shop using these controls may also use the SPLIT processor.

In terms of capability, SPLIT can handle canned cycles for

deep-hole drilling, boring and tapping. Matrices of holes may be specified with but a few commands in addition to the translation and rotation of coordinate system features available with most NC processors. Some typical major instructions are CUT, MOVE and CRINT (for circular interpolation) while minor operations are initiated by mnemonics such as: COF (Coolant off) TOL, and LT (for path of the table).

The real emphasis of the SPLIT language however is not contouring - rather it is designed as a multi-axis point-to-point language with accompanying contouring capabilities built in.

2.3.3 Two- to Three-Axis Contouring

As the capability of the language becomes more restricted, the syntax and vocabulary becomes invariably simpler, computer core requirements for system operation are reduced, and the spectrum of processors available increases drastically. Some of the languages here listed may no longer be in use, some are special purpose (i.e. applicable primarily to lathes) and as before, some are proprietary. Also in this section are listed some of the processors developed in Europe and Great Britain. These are not necessarily available for use by Canadian industry.

2.3.3.1 The EXAPT Family of Languages

The approach to NC-programming taken by the EXAPT Verein of Germany carries the concept of organizing technological data perhaps further than do most other part programming processors. This association has to date produced two operable languages and is executing final development of a third.

A) EXAPT 1

EXAPT 1 is essentially a point-to-point processor. Its language format is free, that is, all instructions could be major in that almost any combination of instructions is possible. The language definition capabilities are; straight line and circular point patterns, and the transformation, mirroring or gathering of these patterns. The power of the EXAPT concept is revealed in the next step where EXAPT 1 may be called upon to automatically calculate (1) feed and cutting speed
(2) Spindle withdrawal characteristics with deep holes
(3) Tools
(4) Work sequences (42)

The processor also ensures that the effective length of the tool is sufficient and that no interference is encountered during the tool-change cycle. Such calculations may of course only be executed with the aid of a tool file, a material file, and a machining file. Figure 2.11 shows graphically how the programming system operates. EXAPT 1 is a 3-Axis point-to-point processor.

B) EXAPT 2

The EXAPT 2 processor was designed with lathe programming in mind. The programmer is here called upon to describe the geometrics of the workpiece before and after machining. Programming involves definition of the upper-half cross section by a series of geometric elements connected by linking statements. [42] Machining is invoked by calling up a predefined machining definition and a part of the finished contour. Functions and movements of the tool

Automatic programming with EXAPT

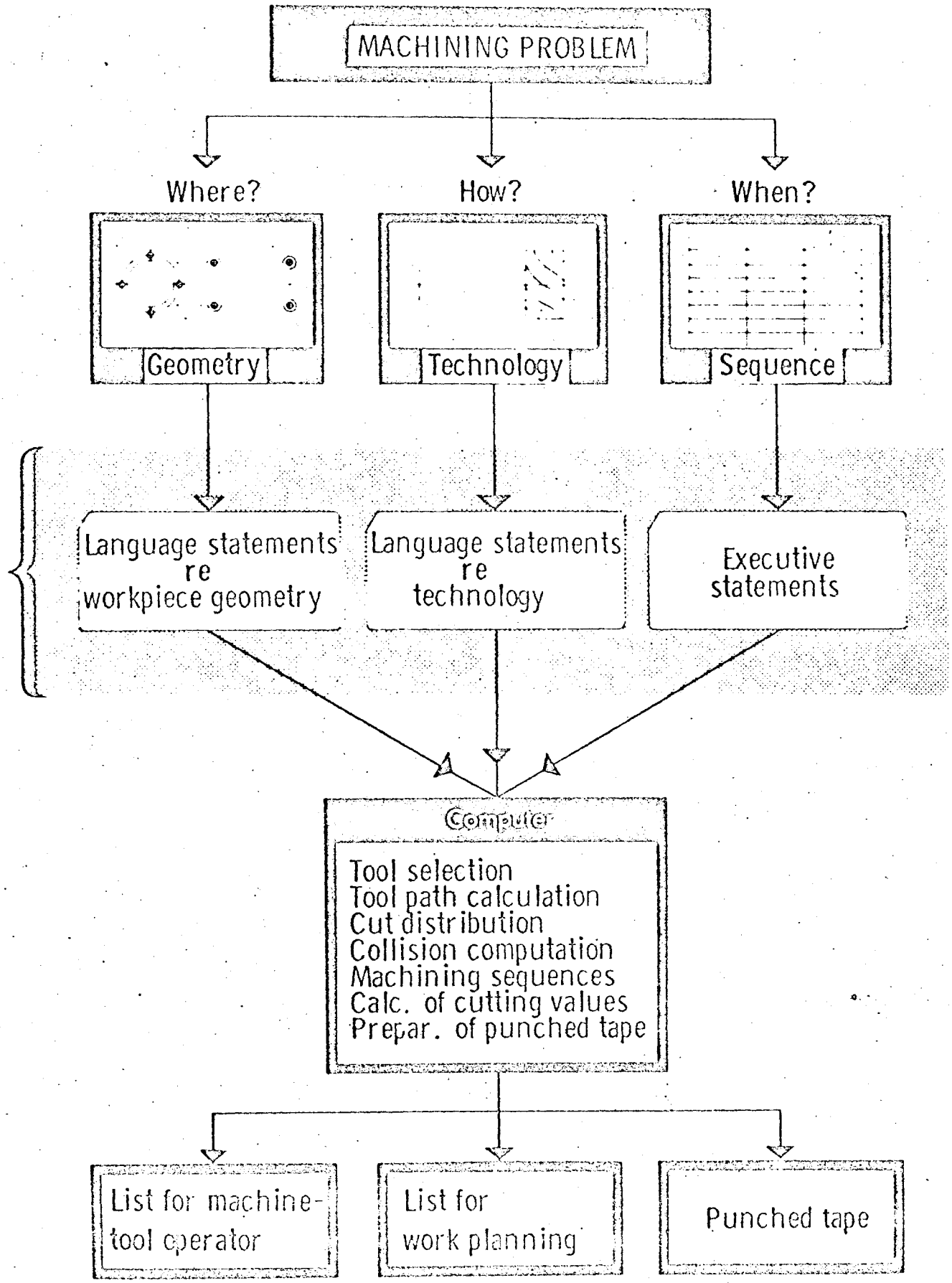


FIGURE 2.11

path are determined by the processor.

The features of EXAPT 2 are:

1) Cutting speed and feed calculations for most operations.
2) Depth of cut determination.
3) Cutting distribution with collision checking.

Figure 2.12 illustrates some of the programming and operational characteristics as applied to turning a shaft using EXAPT 2. EXAPT 2 is essentially a 2-axis contouring processor.

C) EXAPT 3

Under development at time of writing, the EXAPT 3 version is intended to be a $2\frac{1}{2}$ axis contouring program. It will include most of the capabilities of EXAPT 1 and 2 except for those which are specific to lathes, with the inclusion of a POCKET milling routine. [44]

As in all EXAPT processors, EXAPT 3 uses the basic APT vocabulary, but the language format again is free, thus significantly simplifying the required programming effort. Postprocessors are available for EXAPT 1, 2 and are being developed with and for EXAPT 3.

2.3.3.2 The NEL Family of Languages.

A parallel development to the EXAPT processors described above were the 2CL, 2PL, and 2C languages developed by the National Engineering Laboratories in Great Britain. The initial approach was one of nationalism, that is, the U.K. was to have its own standard NC language. The result was, in the case of 2CL, another version of AD-APT with a few modifications. Many of the APT

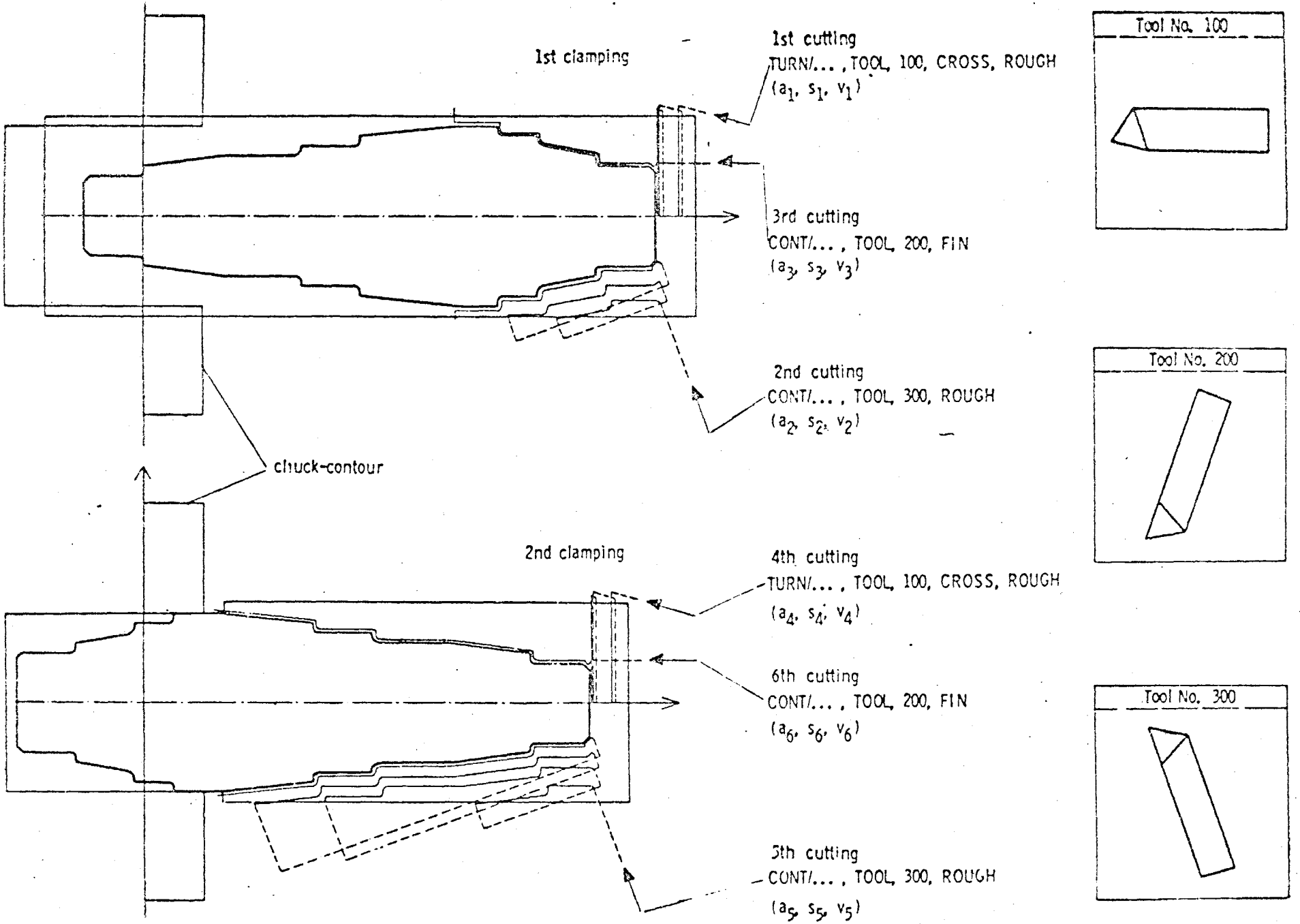


FIGURE 2.12 Determination of Tool Path and Cutting Conditions

names have been shortened from six letters to two for reduced programming effort.

The development however does not include the technology sections as in EXAPT, although attempts have been made recently in this direction. In concept the languages have been structured similarly to EXAPT however in this case the $2\frac{1}{2}$ AXIS 2CL program was the first processor in operation (1969). A brief description of each processor follows:

A) 2PL

2PL (for 2 axes positioning, one linear) is a $2\frac{1}{2}$ axis point-to-point program. It is a general processor program (as are its sister programs).

In structure the program is composed of four sections; 1) decoding, 2) geometry, 3) Tool library, 4) motion. Of these the tool library is perhaps significant in that tools are entered into a permanent file in the processor library and are subsequently called up by library number. This file may be updated as desired and required. Most of the syntax and vocabulary are taken from the APT system however it is a much smaller processor than its NEL counterparts.

B) 2C

A two-axis contouring processor, 2C, has been extracted from the 2CL system for use in lathes. It has been implemented on an ICL 1904 with 32K core store and four tape drives, although it has also been programmed for the Univac 1108. A feature of the

2C program is the external tool file, similar in concept and execution to that of the tool file for the 2PL processor.

Area clearance facilities are also built in which with a minimum of programming input remove a predetermined cylinder of metal from the workpiece, the computer performing the task of calculating the tool path.

C) 2CL

Work on the original 2CL (two axes contouring, one linear) system was begun in 1965 with a target date of 1967 however its actual implementation was not achieved until late in 1969. It is a $2\frac{1}{2}$ axis general processor, similar in concept to APT, however with a drastically reduced core requirement (32K of 24 bit words). It is a multi-pass program, similar to APT. A flow diagram of the 2CL processor is shown in fig 2.13.

These programs are all available at no cost to U.K. users, for whom they were designed. In light of the long development time and the inroads made by existing programs during this time however, their acceptance has been rather limited.

2.3.3.3 The General Electric Time Sharing Systems.

The philosophy at General Electric seems to be that the largest market in terms of potential users of computer assisted programming lies with the smaller firms who do a great deal of point-to-point programming. Judging from their successes of recent years it seems that this has been a timely and well-thought-out program.

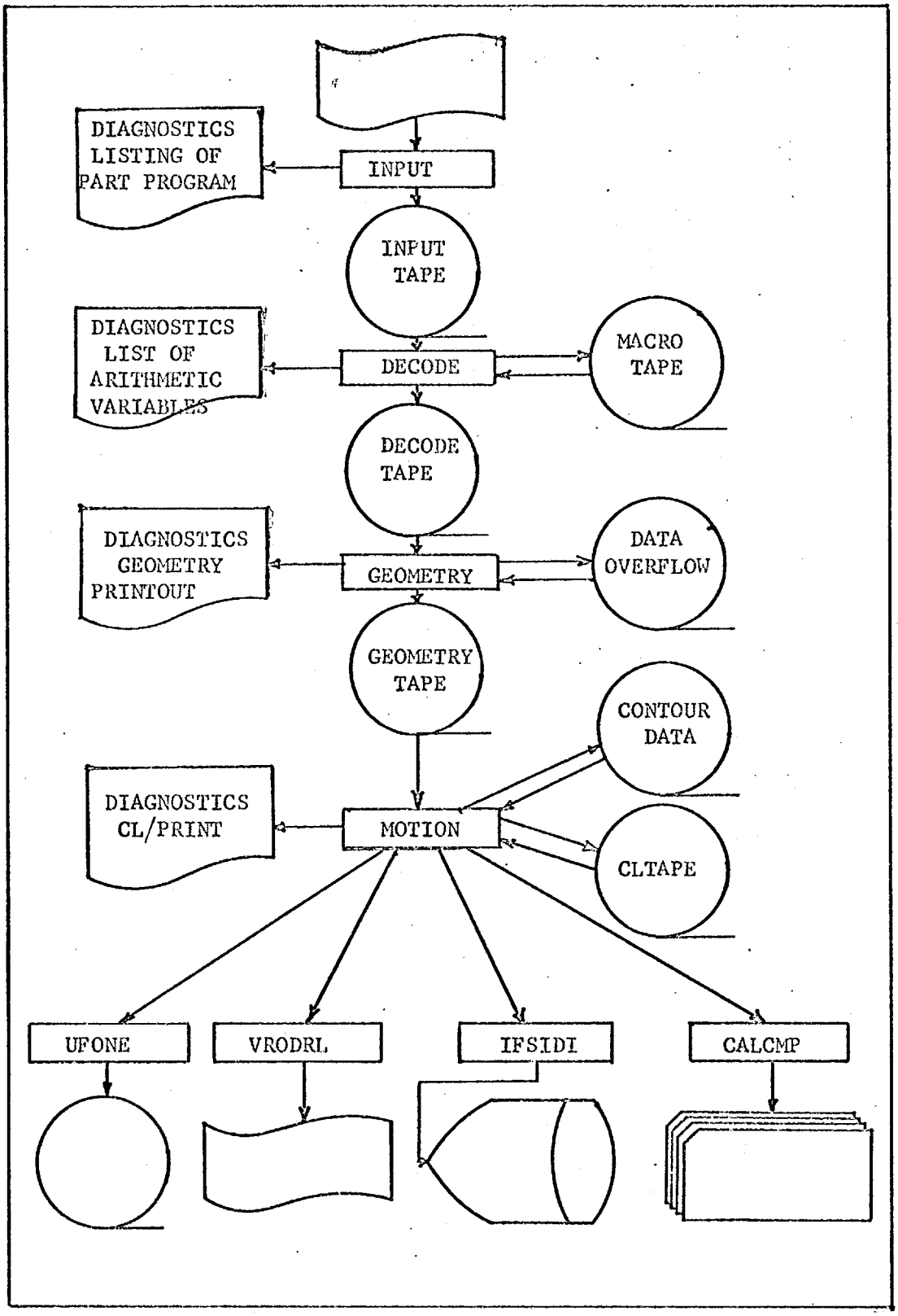


FIGURE 2.13 FLOW DIAGRAM FOR 2C,L PROCESSOR

A) The MARK I Point-to-Point System

Three independent computer programs may be used to generate control tapes for the MARK I system. An illustration of the system operation is shown in Figure 2.14. From the part drawing, the programmer describes the part geometry. He then uses the NCPTS$ program in the interactive mode to generate a point coordinate file. Mnemonic Codes such as C3P, PAL, and PAC assist the programmer in defining hole coordinates (see fig. 2.15). Having specified the coordinates of all the holes, the programmer saves the file. He then either prepares or calls up a previously prepared Machine Tool Description File (MTDF) wherein the parameters of the machine are listed in a sort of "plug in" type of program. This done, he prepares a part program using a vocabulary which is a free format language. Statements such as GOLINE and GOCIRC facilitate the generation of points not listed in the point coordinate file. Operating codes include those inserted into the MTDF, as well as controlling statements such as ORIGIN, which specifies the origin, and FINI which must appear at the end of a program. Cycle files may also be written to execute a repeatable sequence of operations.

Having done this, the programmer feeds these bits of information into the computer terminal and calls up the coordinating program NCPPX$ which then either supplies a tape listing or punches out an EIA coded tape as directed. The ASCII-EIA conversion is performed by another program NCEIA$. Capability of this series is

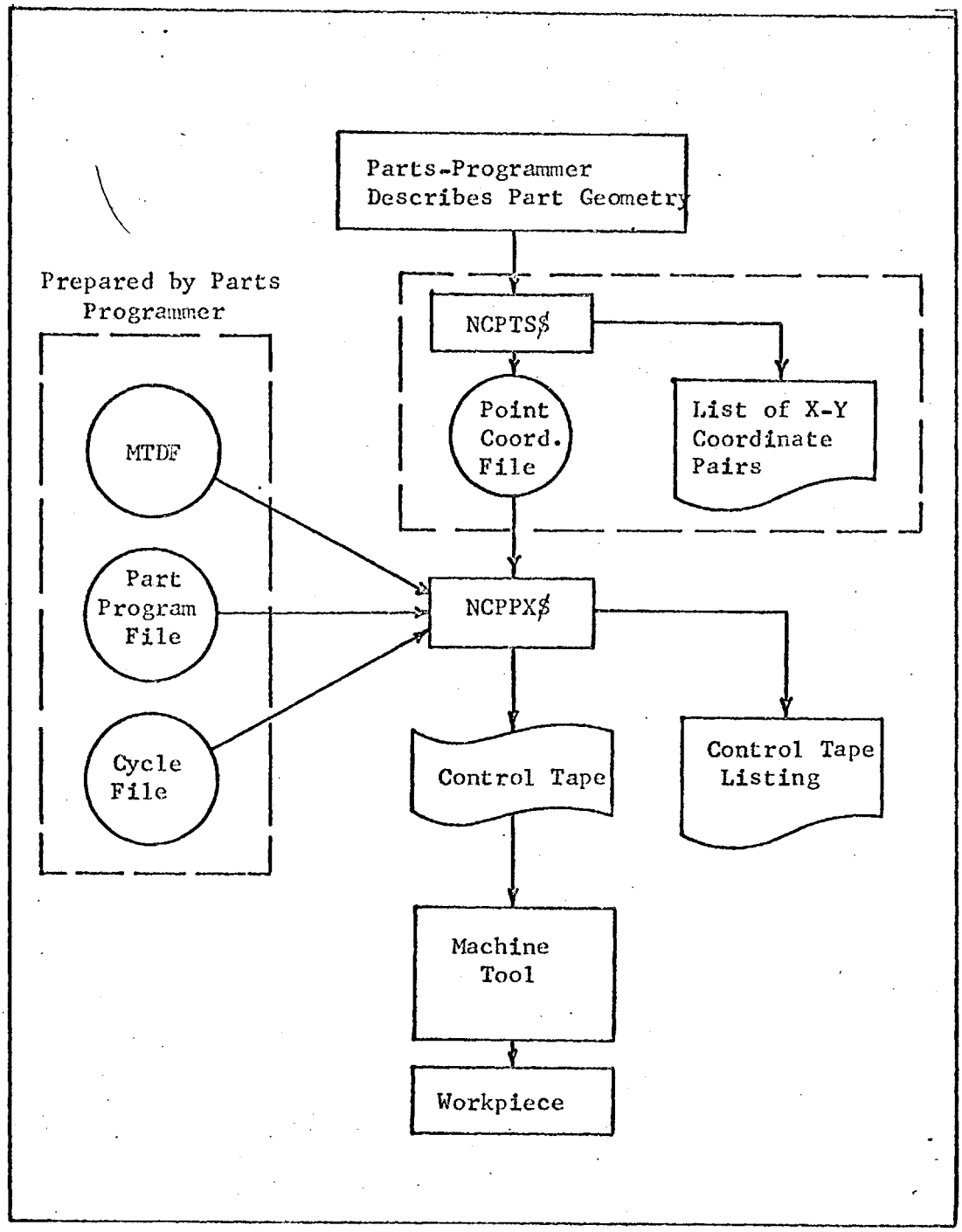


FIGURE 2.14 ILLUSTRATION OF MARKI SYSTEM OPERATION.

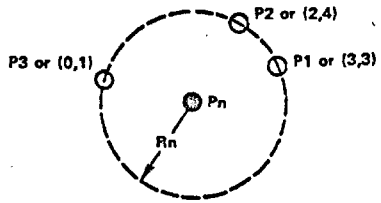
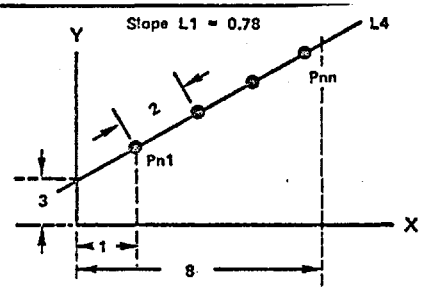
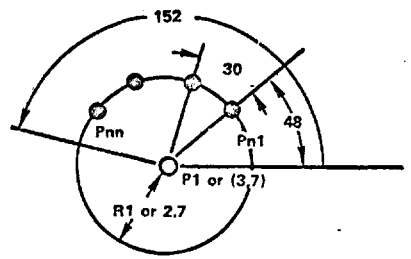
Mnemonic	Function		Input	Output
C3P	Defines the center point and radius of a circle, given three points on the circle.		+C3P, XP_{k1}, XP_{k1}, YP_{k2}, XP_{k3}, YP_{k3} -C3P, IP_{k1}, IP_{k2}, IP_{k3} +C3P, 3, 3, 2, 4, 0, 1 -C3P, 1, 2, 3 (Index of P3; Index of P2; Index of P1)	Pn, X, Y Rn, magnitude
PAL	Defines a series of equally spaced points on a line, given line, starting and stopping X-coordinates, and a fixed step size.		+PAL, 0.78, 3, 1, 8, 2, print option -PAL, 4, 1, 8, 2, print option (Index of L4)	Pn1, X, Y Pnn, X, Y
PAC	Defines a series of equally spaced points on a circle, given the circle, starting and stopping angles, and a fixed step angle.		+PAC, 3, 7, +2.7, 48, 152, 30, print option (Actual radius) or +PAC, 3, 7, -1, 48, 152, 30, print option (Index of R1) -PAC, 1, +2.7, 48, 152, 30, print option or -PAC, 1, -1, 48, 152, 30, print option (Index of P1)	Pn1, X, Y Pnn, X, Y

FIGURE 2.15 Mnemonics and their definitions in the NCPTS program.

limited to $2\frac{1}{2}$ axis point-to-point.

B) The MARK II Time-Sharing System

The MARK II System is a vast improvement over the system described above with capabilities ranging to 3 axis contouring (with the employment of REMAPT, Section 2.3.2.2) and an advertised 4 axis point-to-point capability.

In system configuration, three programs may be used, although only one is necessary. NCPTS$, referred to above is again offered, in a slightly improved version, for point coordinate calculations. A MTDF is prepared similarly to the MARK I system and, with the use of the NCPPL$ program, one is ready to generate tapes. Here the similarity ends and the power of the updated system comes to the fore. A third program, NCUTIL may be employed for one of four services. These are: [49]

i) To search any line-numbered ASCII file for imbedded non-printout characters which are not permissible in NCPPL$ processing.

ii) To convert a REMAPT CLFILE (cutter Location File) to a Points File which can be used as input to NCPPL$.

iii) Compile a permanent Cycle File with its MTDF in a binary mode for an NCPPL$ quick start.

iv) Convert a REMAPT CL file to a Part File which can be used as input to NCPPL$. NCPPL then becomes the REMAPT postprocessor.

The flexibility of this system is its main feature, facilitating adaptation to those machines which do not have postprocessors.

The details of the various possibilities have been well-documented (see Ref. 49).

2.3.3.4 AUTOMAP

In 1964 IBM implemented a program for the IBM 1620 computer. It was a two-axis, continuous path contouring program the function of which was to develop instruction sets for milling. Positioning capability may be had in 3 axes. The program is of the general processor type and is APT compatible. Circular interpolation is included.

The main source program may be altered slightly to suit user demands or requirements. It is available without charge to users of the IBM 1620.

2.3.3.5 ACTION

Developed by Numerical Control Computing Services (NCCS), ACTION is marketed as a 2 axis continuous path processor. It follows the general processor philosophy, the postprocessors being supplied by NCCS for a range of machine tools. Computer requirements are the IBM 360/30 with 64K or its equivalent.

2.3.3.6 PHILCON [51]

A special purpose language has been developed by the Philips Research Laboratories in the Netherlands to control cam-milling machines. No attempt was made to manually program such a machine since the computations involved in fitting a 7th order or 9th order polynomial and then subsequently generating coordinate points and vectors from it would be too complex.

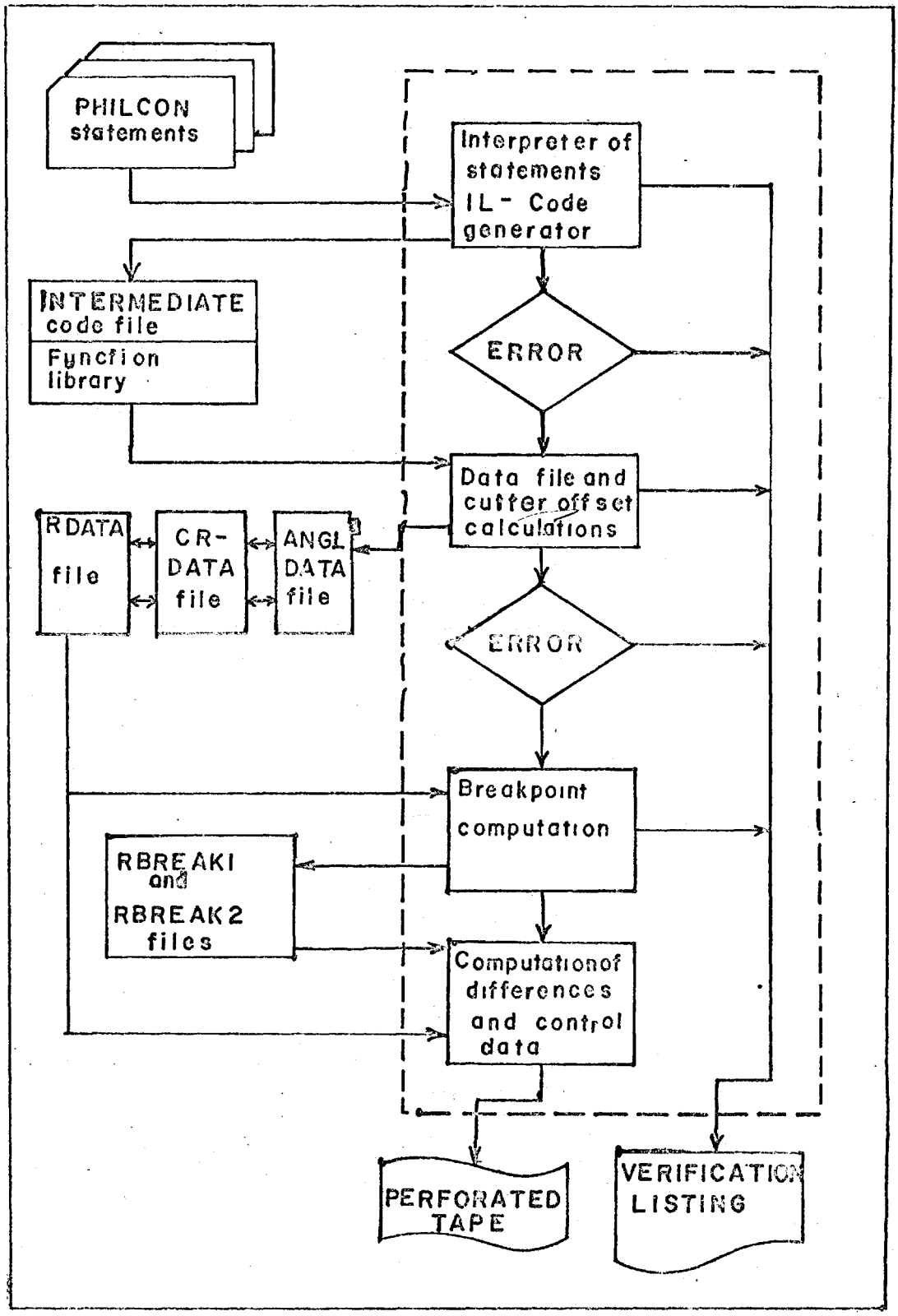


FIGURE 2.16 FLOWCHART OF PHILCON LANGUAGE

Final implementation of the program was carried out on the CDC 3600/3200 installation at the Philips Research facilities in Eindhoven in the Netherlands. The approach to program design employed exclusively the polar coordinate system, dealing only with R (radial distance) and θ (angular displacement).

The language is extremely compact, consisting of 23 symbols or statements, examples of which are: PP1 (referring to a specific machine controller); POLYNOM5 (5th order polynomial); and ENDFINI (the end-of-program controller) to mention but a few.

In operation, the program accepts the card deck of PHILCON statements and enters the section called MAIN, or the executive program. Intermediate code files are set up, function libraries are called which supply data for file and cutter offset calculations. These again are stored in separate files according to the nature of the displacement, with the purpose of supplying data for the penultimate coordinate data file and the final tape generation (See Fig. 2.3).

The program has been largely enveloped by the APT processor however it is still operable on a small COBRA computer. It must be noted that although it may have limited use, it performs a function that is virtually impossible to perform without a computer.

2.3.3.7 Other Contouring Languages.

The languages here described account for by far the most contouring work, indeed a recent survey in the U.S.A. showed the usage of APT to be as high as 44 per cent of all the NC languages. [30]

While this figure seems high it should be noted that the popularity of the APT system has made it a de facto standard, not only in the U.S.A. but throughout much of the industrial world.

Other contouring languages of course, exist. Names such as NUCOL, COCOMAT, AUTOPOL, IFAPT, and many others proliferate in the never-ending development of this area of control. Numerous attempts have been made to "standardize" in order to reduce development costs. Many processors have fallen from grace, some such as AUTOPROMPT have been entirely annexed and put to work inside the APT system. Hence a complete documentation of all contouring (and point-to-point) processors would be useless and uncertain in light of the constant flux of new and old ideas on the fringes of this area of concern.

2.3.4 POINT-TO-POINT PROCESSORS

2.3.4.1 AUTOSPOT

In much the same manner that APT and AD-APT have become the industry norms for contouring work, AUTO-SPOT has become the model for point-to-point computer assisted programming. It is perhaps the most comprehensive processor written for point-to-point work.

In terms of capability, AUTOSPOT is able to handle virtually all point-to-point work, including 2 axis continuous path contouring. It is applicable to machining centers as well as to simple drilling machines.

AUTOSPOT (AUTOmatic System for POsitioning Tools) includes in its repertoire allowances for all hole-making and finishing

operations as well as auxiliary machine functions such as automatic tool changing, table indexing and coolant control.

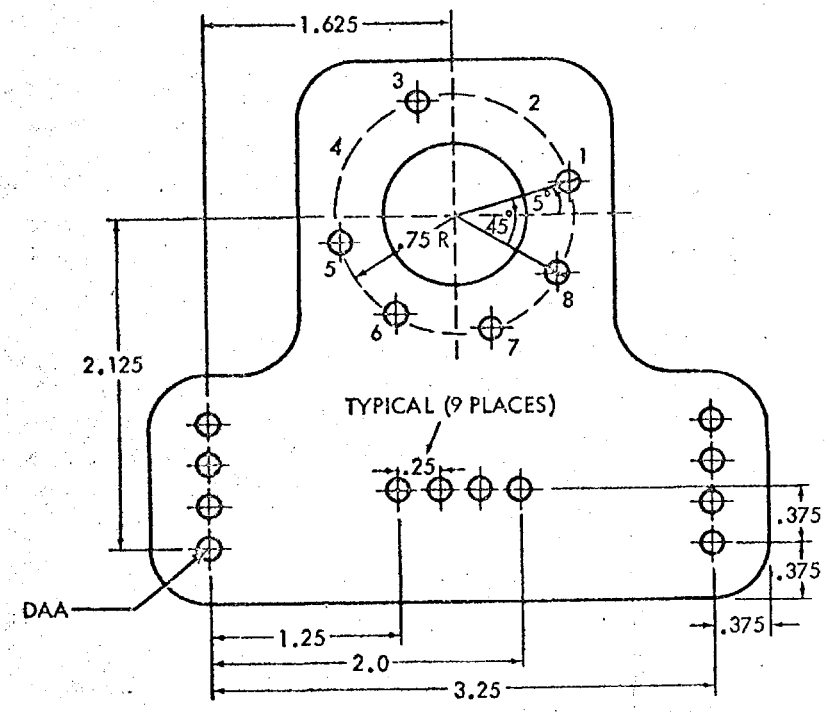
Language structure is made up of four sections the functions of which are: [52]

i) To define geometric entities,
ii) To describe machining operations,
iii) To describe auxiliary machine tool functions,
iv) To define the operation Modes.

The elements of the AUTOSPOT language is revealed especially when programming a matrix or pattern of holes. Having defined the tooling earlier, one calling sequence can initiate a toolchange, and perform a machining operation as well as manipulating (i.e. rotating/translating) a given predefined pattern (See Fig. 2.17). Single commands exist for the expansion or shrinking of a pattern. A POCKET milling routine is available which is applicable to triangles, four sided figures whose sides are parellel, or any shape whose sides are of equal length with angles less than 180°. Other polygonal shapes are also permissible as long as no more than 210 cutter notions are required per pocket or not more than 20 points are required to define the pocket outline. The modifier used is PMILL.

Other features of this processor are statements such as EXCEPT where a list of points is to be acted upon except one or more of the points. The Z avoidance feature facilitates programming to ensure clearance of a protrusion on the work-piece. These are but a few of the special features in this processor.

ENGINEERING DRAWING




```
PARTNO N/C 360 AUTOSPOT SAMPLE PART                                    ( 1)
REMARK USE TOOLS 1212, 1214                                            ( 2)
TOOL/SPDRL 1212 0.100 118.0 7.0000 6.9399 1200 7.0 ONA                 ( 3)
TOOL/DRILL 1214 0.045 118.0 7.0000 6.9730 1500 6.0 ONA                 ( 4)
    CL = 0.115                                                         ( 5)
   DAA = 10.0, 6.0, -18.75                                             ( 6)
BCIRC  = PATERN/ AT(1.625, 2.125) RADIUS (0.75) SA(5.0) $              ( 7)
            IA(45.0) NH(8) EXCEPT(2,4)                                 ( 8)
   PAT = PATERN/SX(1.25) SY(0.375) DX(0.25) NH(4)                      ( 9)
  PATX  = SPDRL, 1212/DI(0.06)/DAA,BCIRC,PAT(0,0) ATANGL(90) PAT, $    (10)
            PAT(3.25,0) ATANGL(90)                                     (11)
          DRILL,1214/DP(0.5)/PATX                                      (12)
FINI                                                                   (13)
```

FIGURE 2.17

Core requirements of AUTOSPOT are 32K on the IBM 360/30 or the IBM 1620 computers. Many postprocessors exist for AUTOSPOT among them Kearney and Trecker, Pratt and Whitney, Cintimatic and Hughes controls. Costs for lease of software are $3,000/month on an IBM 1620 and $7,200/month on the 360/30. [1]

2.3.4.2 AUTOPROPS

AUTOPROPS (for AUTOmatic PROgramming for Positioning Systems) is a general purpose program to handle point-to-point systems with X- and Y-axis control. It is a fixed format language developed for use on the IBM 1401 general-purpose computer at a monthly rental of about $2,000. [50] Information is entered into a manuscript in a predefined sequence. This sequence consists of:

i) list of operations to be performed
ii) mode number and operation sequence
iii) number of holes

Having done this, the setup point on the machine tool bed is recorded and the mode numbers are entered to execute the program, list the output, plot the output tool points, and generate a control tape. Very little calculating ability is available in AUTOPROPS, thus perhaps restricting its use. Its advantage lies in the extreme simplicity of its use and the attendant programming ease.

2.3.4.3 COMPACT

Another company working in the time sharing field with numerical control is Com-Share Corporation of Ann Arbor, Michigan. The computer in use is a Scientific Data Systems 940 which serves

eastern United States and Canada.

The COMPACT System (COM-Share's Program for Automatically Controlled Tools) is of the 3-axis point-to-point variety with the capability for some $2\frac{1}{2}$ axis contouring work. Continued development of this language should increase its capability to that of ADAPT. A fourth axis rotation routine is available for complex workpieces.

An important feature of COMPACT is its debugging routine QED which does not require the executive system for its operation, hence saving expensive core time. At almost any time during the compliation of the program, the programmer may call for a listing. Error diagnostics are returned to the teletype as the program is edited. Once on the system, an intermediate coordinate listing may be called for.

The system itself operates in a manner similar to the G.E. MARK systems described previously. A data file, or machine tool link converts geometric, motion statements and auxiliary function statements into a tape format applicable to a given machine.

Many of the pattern manipulation routines as described in AUTOSPOT are also offered in this system. Some users have described it as similar to SPLIT in concept and execution, with perhaps a little more attention given to contouring definitions and capabilities.

2.3.4.4 QUICKPOINT-8

Quickpoint 8 is a numerical control tape preparation system prepared by the Digital Equipment Corporation (DEC) for use on the PDP-8 series mini computers. All that is required for the execution of this system is a PDP-8 computer and a teletype unit. The total cost of this system should be about $6,000 including software. It is a complete system, requiring no attendant external hardware.

The program language is simple, easy to learn, performs the coordinate hole calculations through the use of a few selected mnemonic symbols and ultimately produces the control tape for the machine. The output processors are made available by DEC for a range of machine tools and in the opinion of the author, these could also be prepared by the user if the proper interfacing techniques were known.

The program is basically 2-axis point-to-point and does not really compare to some of the sophisticated computational giants as for instance AUTOSPOT. It offers a viable alternative to those whose work is restricted to point-to-point and who prefer a slightly higher overhead to a high monthly charge with low overhead.

Coordinate input facilities are available as are offset and permanent memory facilities. For short tapes, the input is read and processed and the whole tape is generated in one pass. Longer tapes, however are processed line by line, thus enabling the preparation of tapes using many blocks. The DEC software

group has consistently produced detailed documentation of their software acclaimed by some as the best in the industry.

2.3.4.5 SNAP

One of the few "in-house" or machine-oriented point-to-point languages has been prepared by Brown & Sharpe for their Turr-E-Tape turret drill and for two-axis sue on their Hydro-Tape machining centers. The control system used is G.E. Mark Century. The program is operational on an IBM 1401 computer which produces cards for the preparation of the control tape when the part program is submitted to it.

This processor handles four types of patterns: [50]

1 - a single hole with two coordinates (point)
2 - hole(s) on a radius (separated by a constant angle)
3 - holes on a line separated by a constant incremental di tance
4 - a matrix of holes, parallel to and equidistant from one another.

The numbers accompanying the above categories become the code numbers for the programmer when he wishes to define a series of patterns. Having defined the nature of the pattern, he then adds necessary data such as incremental angle (distance) and radius values in the case of a circle.

2.3.5 Summary

Much more could, and has been said about NC languages; listing them all would have little value here. The selection criterion of those languages listed were briefly:

1) Are they well known?

2) Is information available?

3) Are the languages available to Canadian and American users?

4) Is there anything unique in them?

Clearly not all four criterion could be applied in every case and it is quite possible that serious omissions have been made. However, it has not been the intention to include all the languages, if indeed this were possible. Background work done for this study and another one [41], resulted in the cataloging of 45 processors for numerical control tape preparation, the complete documentation of which would be a monumental task.

The organization of the languages into their categories has not been strictly adhered to in those cases where families of languages appear. The point-to-point processors of these categories are largely modules taken from one large comprehensive processor and for this reason, these were included in their respective groups, albeit not in their proper category.

Table 2.1 summarizes the languages and provides an "at a glance" comparison of some of the relevant properties of each program.

NUMERICAL CONTROL LANGUAGE COMPARISON

LANGUAGE NAME	CONTROL CAPABILITY	NUMBER OF AXES	TYPE OF PROGRAM	COMPUTER AVAILABILITY
APT III	CPC, PTP_1	5,6	GP_2	IBM,CDC,UNIVAC, and other large systems.
2CL	CPC, PTP	3	GP	UNIVAC,ICT,KDF9, ELLIOT 4100
EXAPT I	PTP	3	GP	GE,CDC,KDF9,ICT,IBM UNIVAC,Siemens
II	CPC	2	GP	
III	PTP,CPC	3	GP	
ADAPT	CPC	3	GP	IBM
AUTOSPOT	PTP	3,4	GP	IBM
AUTOMAP	CPC	3	GP	IBM 1620
$AUTOPROMPT_3$	CPC	5	GP	IBM 7090
SPLIT	CPC,PTP	2,5	IMO_4	IBM,Honeywell, and others.
2C	CPC	2	GP	Same systems as 2CL above.
AUTOPOL	CPC	2	GP	IBM 360/30
KIPPS	PTP	2	GP	KDF9
MILMAP	PTP	2	GP	-
2PL	PTP	$2\frac{1}{2}$	GP	Same systems as 2C,2CL.
COCOMAT	CPC	3	----	PL/1
CAMP IV	CPC, PTP	2, 5	GP	IBM, CDC, UNIVAC.
SNAP	PTP	2	IMO	IBM 1401
ROMANCE	PTP	2	-	IBM 1130
SYMPAC	CPC	3	GP	UNIVAC 80, 90.
ACTION	CPC	2	GP	IBM 360/30
UNIAPT	CPC	3	GP	PDP 8, IBM 1130, GA 18/30, GEPAC 4020.
AUOTPROPS	PTP	2	GP	IBM 1401.
PRONTO	PTP	3	GP IMO	GE 225 IBM 704, 7090.

TABLE 2.1

....continued....

NC LANGUAGE COMPARISON (continued)

NAME	CAPABILITY	AXES	TYPE	COMPUTERS
PHILCON	cam-milling	2	-	IBM 650, CDC 3600.
NUCOL	CPC	$2\frac{1}{2}$	GP	UNIVAC 1108
NCPPL	CPC, PTP	2, 3	MF_5	GE MARK II
NCPPX	PTP	$2\frac{1}{2}$	MF	GE 225 MARK I
COMPACT	CPC, PTP	$2\frac{1}{2}$, 3	MF	SDS 940
QUICKPOINT 8	PTP	2	-	PDP 8

Legend:

1) CPC refers to Continuous Path Contuoring, and PTP defines Point-to-point machining.

2) GP is here defined as General Processor.

3) No longer in common use, as many of its advanced features have been combined into the APT language.

4) IMO is defined as Individually Machine Oriented.

5) MF is defined as the Machine Tool File as used by some of the time-sharing systems.

TABLE 2.1

2.4 THE FUNCTION AND OPERATION OF THE POST-PROCESSOR

2.4.1 Introduction

Of the 22 languages cataloged in section 2.3 all but two are of the general processor variety, that is they are able to calculate lines, point, circle and other geometric definition orientations but they do not contain sufficient information, by themselves, to produce a control tape. It is therefore necessary to load the output information of the general processor onto a file for subsequent processing by yet another computer program, called the "post-processor". The post-processor contains the necessary logic to convert the auxiliary machine tool functions, and other statements not acted upon by the first program, into a code recognizable by a specific machine tool-controller combination. This done, it must organize the codes into a tape format determined by the construction of the reader-buffer on the controller unit, and then supply the proper interfacing with a tape punch unit to generate the control tape. (See fig. 2.18)

One of the problems of the post-processor from the industrial user point of view is that of cost. Software packages for complex machines using APT run into the thousands of dollars, in some cases significantly adding to the purchase price. To compound this problem, the industrial salesman, whose duty it is to ardently expound on the virtues of a machine, frequently is not too clear himself on the details of its computer interface. Besides, it is no doubt an extremely difficult task to convince a machine-

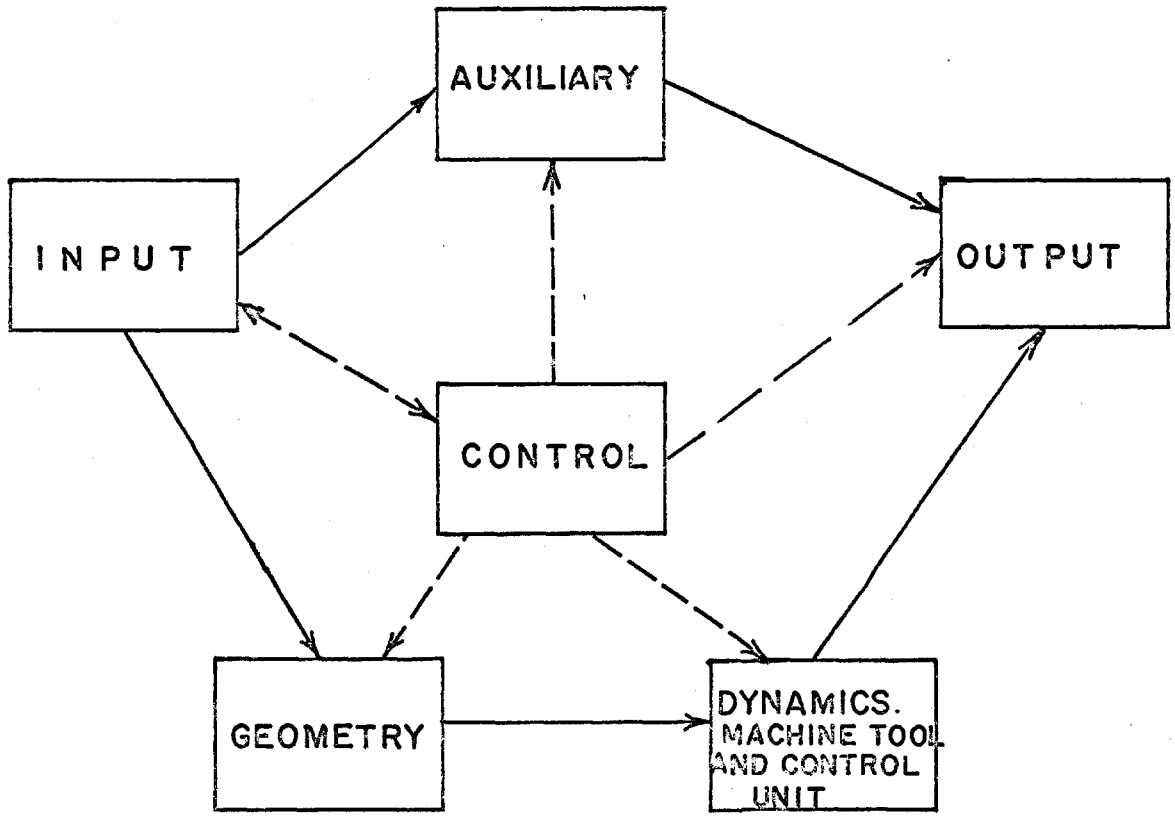


FIGURE 2.18 GENERALIZED POST-PROCESSOR STRUCTURE

tool buyer that he needs this five or six thousand dollar reel of magnetic tape for his newly purchased numerical control machine.

Of course, it goes much deeper than that; computer programmers, while they abound in an academic environment, are seldom brought face-to-face with the actual operation of the computer itself. The post-processor must be able to recognize each bit of information in the CL data file and it is at the interfacing level of the two programs where many simply do not bother. A knowledge of the machine assembler language and its use is a must, and since each family of computers has its own, the number of interfacing programs becomes the product of the number of computer assemblers and the number of machine tool controllers which must be interchangeable.

One solution of course is to market the software for a machine as an integral part of its purchase price. Many companies do this but unfortunately not all of them. A different approach is to set up modular skeleton processors which may be filled in by the local part programmer with very little formal knowledge of any computer language. The Machine Tool Description File of the General Electric System is an example of this concept. Most computer-programmed machines to date use "custom-made" postprocessors which are written in their entirety for a specific machine-computer language combination. This is a more classic approach, than the "kit" form available from G.E. and Com-Share, for instance.

2.4.2 Structure of Conventional Postprocessors

Post-processors written in the early stage of NC development were compiled in assembler language. As more computer manufacturers entered the market, each with their own assembler, it became obvious that a computer independent approach to postprocessor writing would be highly desirable. This would of course require a greater operating core.

In 1963 APT Postprocessor Standards were established by IITRI for members of the ALRP. This was to ensure that:

1. Postprocessors would be modular, with common modules in various postprocessors, lowering costs and increasing the inherent reliability,

2. Postprocessors would be thoroughly documented, for user's information and alteration, if necessary,

3. Postprocessors would be similarly structured across product lines to decrease user training time and costs. [56, 57, 58]

One manufacturer of NC machine controller units has followed these guidelines to the point where many different machines may be operated each using the same basic controller. The result is obviously lowered hardware and software costs.

In structure the standard postprocessor consists of two phases. The first phase reads the CL tape and processes it, sets flags and paramaters and may also store an output record in a special intermediate array. When the END of FINI statement is encountered the postprocessor begins the second phase which sequentially processes the array and produces a control tape. The

organization of the program is as follows:

i) Input and Control. CL tape records are read and partially processed. If no errors exist in the record type, those sections not processed are dealt with.

ii) Auxiliary All the auxiliary machine functions - spindle speed, feedrate, table rotation, and so on - are handled by the auxiliary section, which set the appropriate parameters in terms of an output code

iii) Motion. Part coordinates are converted to machine coordinates, type of interpolation is determined, and non-axial feedrates are established.

iv) Output (non-motion) - If the information transmitted to the machine deals only with auxiliary machine functions the output record is compiled and the tape is punched.

v) Output (motion) - total motion is broken into one or more tape blocks setting feedrates and making certain that the dynamics of the machine and controller have time to respond to input commands.

This is of course a vastly oversimplified algorithm in light of the myriad of features available on NC machines. It serves to illustrate however, the magnitude and nature of the tasks performed by the conventional postprocessor in the preparation of a machine control tape. It does not give details, which are generally classified information.

2.4.3 Postprocessor Construction Using the G.E. MARK II System.

Prior mention has been made of the time-sharing systems offer by General Electric Corporation. The new NCPPL$ language (for Numerical Control Part-Programming Language, $ denoted proprietary system) is in effect a complete processor in a modular form set up so that the user may make alterations to any part of his system as he wishes.

The processor structure is a rather major departure from the fixed format, fixed vocabulary languages generally available. A complete processor in NCPPL$ may consist of as many as four segments and the vocabulary may include terms specified and de- by the programmer himself. The four segments are named:

1. Part Program File
2. Machine Tool Description File
3. Cycle File
4. Points File.

Of these four, the first two are mandatory with items 3 and 4 offered as extremely desirable options. Each time the program is run, it asks the programmer to specify the files he is using by name. These files must have been compiled, entered and saved prior to calling NCPPL for successful execution of the program. Having been supplied with the relevant bits of information, the system returns with either an output listing or an output tape (if no errors have been found), in response to the appropriate command from the programmer.

Items 2 and 3 above together may be thought of as the

postprocessor in this system, depending on how the cycle file is to be used. The NCPPL Users Guide (Reference 49) contains comprehensive illustrations how cycle files may be prepared for specific purposes. Some samples listed illustrate the following:

i) Use of a Cycle File to mathematically relocate programmed points by transformation, rotation, or mirroring.

ii) Table Look-up logic for S-command value, and Feedrate calculation ability for F-codes (IPR, IPM).

iii) A technique for repetitive programming such that a pattern of points is repeated in a mesh or grid pattern of rows or columns

iv) Incremental Machine Cycle, including:

a) Start-up logic
b) Twin-head compensation
c) Automatic segmentation of movements greater than 9.999 inches
d) Absolute printout at desired stages
e) automatic M-code formatting.

v) Absolute Machine Cycle including:

a) M and G code determination
b) Automatic gage length, clearance plane and cut depth adjustments.
c) Internally generated comments.

The cycle files employ a facility called a storage register. This register is made up of 500 sequentially referenced locations which may be used to contain a spindle speed table, for example, and which are also used as output buffers for output "words" of information. These registers are therefore greater in size than a computer word. They may also be used in computational work as

the current location of a variable which is brought into the calculation cycle by its register address rather than its actual value.

The NCPPL system includes in its repertoire ten arithmetic operations. These are:

ADD	-	addition
SUB	-	subtraction
MPY	-	multiplication
DIV	-	division
POW	-	raise total to a power
SQR	-	square root
LOG	-	natural logarithm
NEG	-	reverse the existing sign
ABS	-	take the modulus of total
INT	-	truncate to a whole number

Trigonometric functions are also included (sine, cosine, e.g.). The register storage locations are entered and loaded by means of USE n and PUT m commands. USE n calls up a specific number n whereas USE/n refers to a previously located number in storage position n, entered into that position by the command PUT n. These facilities, in conjunction with the programmer's expertise, and the MTDF, facilitate the in-house writing of postprocessors, as they are traditionally called.

The MTDF of the NCPPL language is an expansion of the MARK I system file described earlier, permitting a much greater flexibility of input and output. This file provides the computer with resident information on a particular machine tool. The nature of the information called for follows:

i) Multi-valued vocabulary words; examples COOLNT, FEDRAT,

ii) Single valued words; DRILL, MILL, BORE,

iii) Standard vocabulary words (these are unalterable)

GOTO, DPT,

iv) Coded output format options,

- leading, trailing zeros, tab, no tab, nature of block codes,

v) Number of output registers,

vi) Output register address letters,

vii) Output register formats,

viii) Repeat codes for output registers,

- some machines require blocks of information to be repeated, others do not,

ix) Register storage locations for multi-valued words, single valued words,

x) Numerical values of single-valued words,

xi) Tape format.

When this file is completed and saved, the programmer is in a position to write his part program, with or without the aid of the NCPTS point generation program.

2.4.4 Summary

The traditional secrecy which blankets the postprocessor in industry has generated some controversy among potential users of NC equipment, and indeed may have turned some away from it altogether. An attempt has here been made, not to destroy this mystery, but to explain in part why it exists and to give one example of how it may be circumvented by the machine tool user himself.

3.0 TAPE FORMAT TRANSLATOR

3.1 Introduction

3.1.1 Existing Tape Formats

Four standard NC tape formats are presently in use. These are: 1) Fixed Block, 2) Tab Sequential, 3) Word Address, and 4) Universal.

The fixed block format requires that all the words or code sets, be arranged in a constant, predetermined order. If a number does not change it must still be entered, if no numerical value is assigned to a location (or series thereof) zeros must appear in those locations. Since a great deal of NC work requires that only one or two parameters change in a block, this format can result in excessively long tapes.

The tab sequential format is perhaps the first step in reducing the length of tape and the amount of information required for each block. The order in which the words appear remains fixed but these are now separated by a TAB, which has the same effect as the TAB button on a typewriter. Only that information which changes from one block to another must now be entered, however a tab must still be inserted for each possible word whether it appears or not. This also has the desirable effect of organizing columns of coordinates and feedrate or miscellaneous codes on a manuscript on the flexiwriter.

A more readable format is the word address wherein symbolic letters prefix each of the words of the block. If a change

is made in any of the parameters, the symbolic letter and its new numerical value appear, otherwise it is ignored entirely. This format stems from the block address which specifies by a numerical code, which words in the block are to appear. Block address format is extremely confusing to read, hence the need for word address.

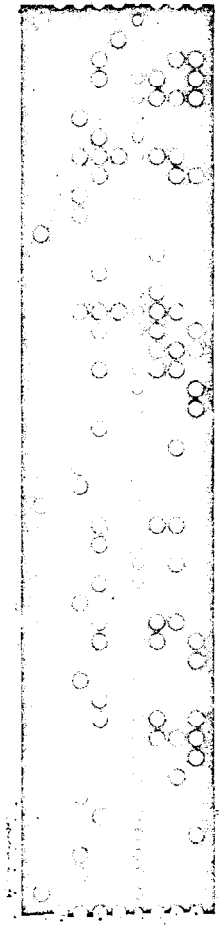
The Universal format combines the features of word address and tab sequential formats resulting in properly aligned columns of word addressed codes on the flexiwriter manuscript.

A comparison of the four basic control tape formats is shown in figure 3.1.

3.1.2 Tape Translator! Problem Definition.

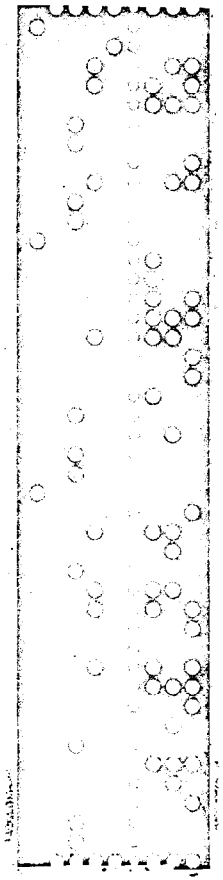
It is quite possible, due to the proliferation of NC machines on the market that a user firm may own two or more machines with similar capabilities but with different control systems. These may or may not read the same tape format. Some paper tape reader units claim the ability to read two or more formats but in actual practice this is quite rare.

It is furthermore a reality that at McMaster, the Engineering Machine Shop now has a MOOG 83-1000MC numerical control machining center. The details of the machine are not relevant to the subject presently under discuesion. The tape format, however, is a non-standard fixed block system which doubles up to an effective 9, or 10 word system to position and control the Z-axis. The format called for is extremely error-prone to manual-programming



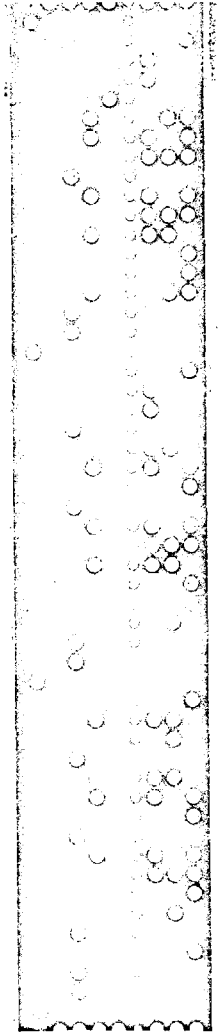
001 02175 01560 2 6
002 11675 4 4
003 07538

Tab Sequential



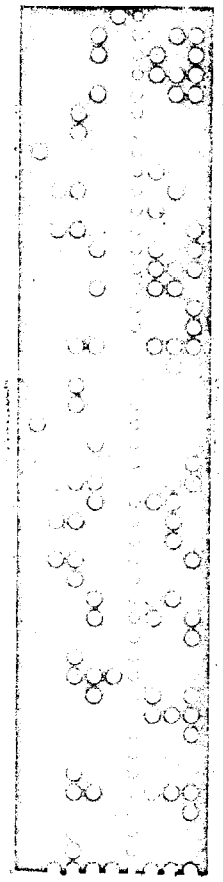
00127021750156026 1
00204116754 4
00310075 38

Block Address



00102175015602 61
00211675015604 41
00311675075384 41

Fixed Block



001X02175Y01560A2B6C1
002X11675A4B4
003Y07538

Word Address

FIGURE 3.1 Some Current Tape Formats

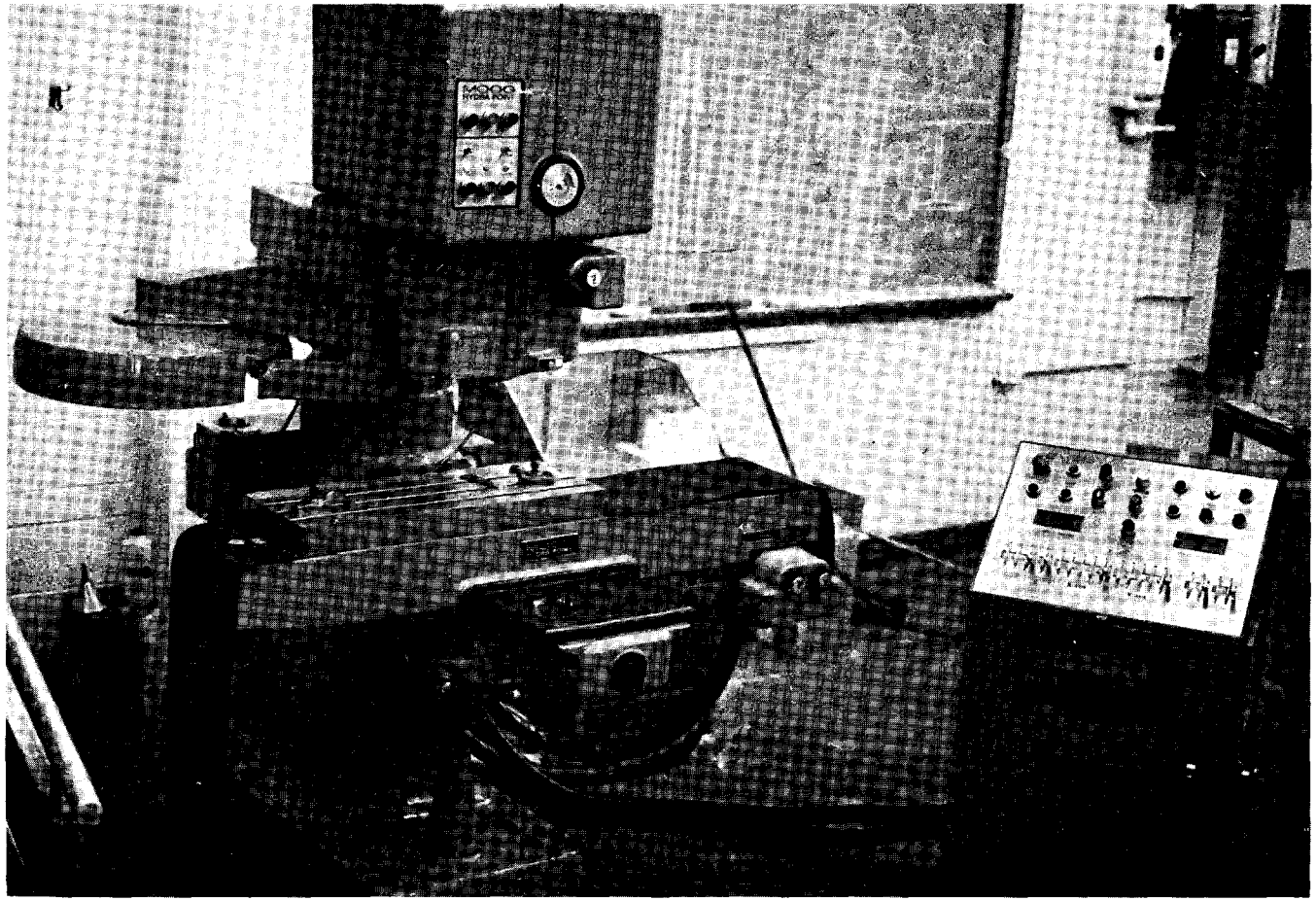


FIGURE 3.2 MOOG 83 1000 N/C Machine in the Engineering machine shop at McMaster.

and is not known to be used with any of the popular machines.

It is therefore a desirable thing to be able to translate paper machine control tapes from one format to another. Some conversions are simple, for instance word address to tab sequential to universal, or any variation of these three.

It was decided that the initial conversion would be to translate a tape encoded in the Universal format to that of the MOOG, since perhaps the major share of the problems would occur here. Situations may arise in which one block in Universal must be converted to as many as three in MOOG.

The objective of the Project then became: to design a software package for a PDP-8/L which will translate the data of an NC tape written in one of the standard formats into the format of the MOOG and subsequently produce the MOOG tape.

Some of the anticipated difficulties at the start of the project were:

1) The splitting up of the block into as many as three in order to introduce all the relevant information into the MOOG control system.

2) Many machines use more digits per word than the MOOG.

3) Various methods exist for coding feeds and speed.

4) A table selection routine should be instituted.

3.1.3 The PDP-8/L System

The machine on which this program was implemented was the PDP-8/L computer with a 4K, 12 bit word memory. The auxiliary

input/output devices include basically the ASR 33 or 35 teletype units, a Sykes Datatronics magnetic tape drive unit for cassette data recording, and a high-speed paper tape reader-punch assembly.

It was considered advantageous, both from the academic viewpoint, and that of saving core, to use the PDP machine assembler language, PAL III. This language consists of mnemonic codes which are directly translatable into machine language. There are seven basic instructions and a full list of operating microinstructions which help to control various operations in the computer. Figure 3.3 shows one of the installations used for the project.

The present versions of the PDP 8 utilize integrated circuitry, resulting in extremely rapid processing as long as core capacity is not exceeded. Some of the problems with working in machine language lie in the computational area, that is, whole routines must be written to work in floating point arithmetic. It is the preferred modus operandi, however, when data handling routines are required. The advantage lies in that the designated codes need not be in a form recognizable to the computer since only binary numbers are used.

The software system used for compiling and debugging the program consisted of:

1) A Symbolic Editor Program - to compile and correct the logic and generate the symbolic take.

2) A Utility Program - another form of editor.

3) The PAL III Assembler - a three pass assembler which

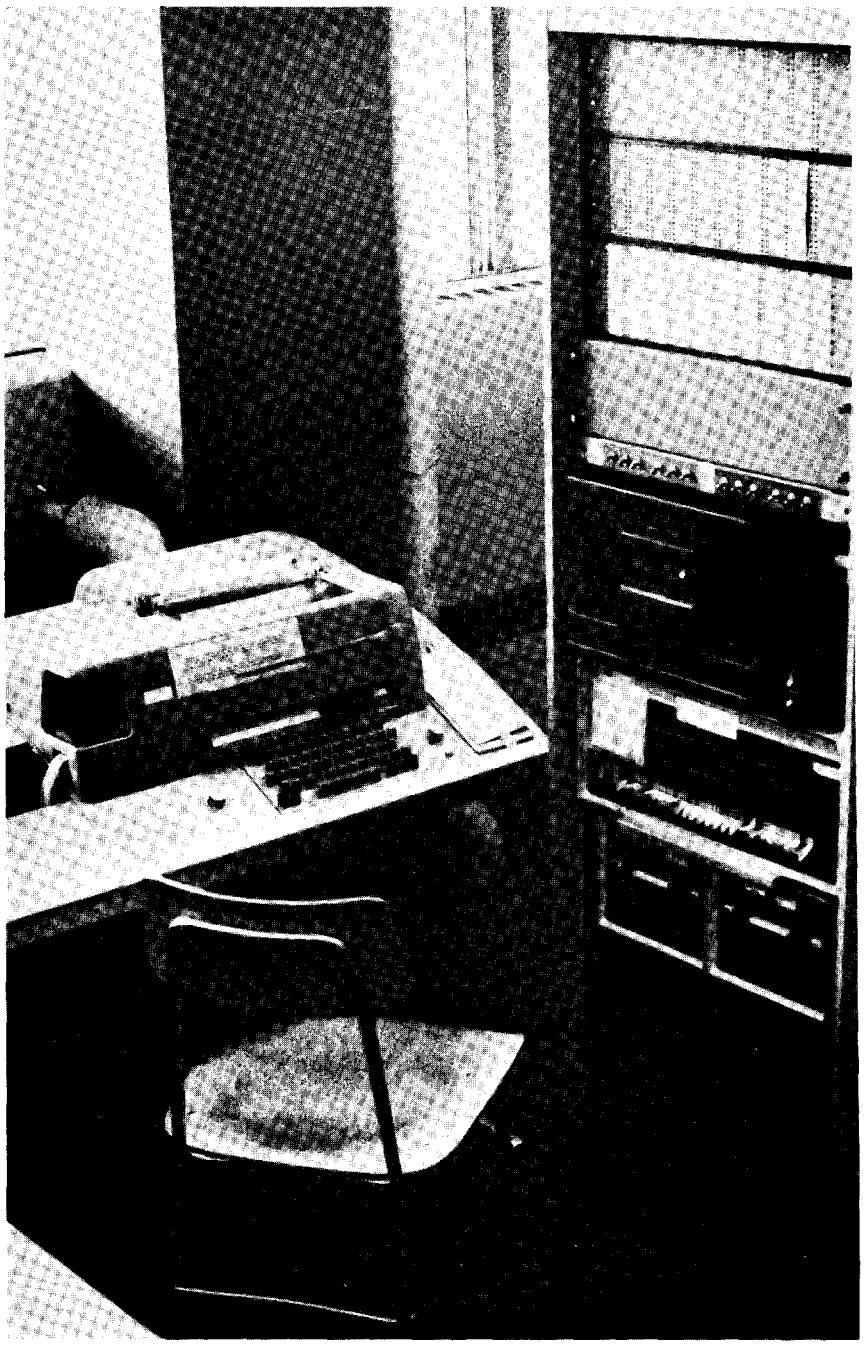


FIGURE 3.3 PDP 8/L installation used for the MOOG translator program development and operation.

sorts out logical errors, produces a binary operating tape, and supplies the user with complete cross-referenced documentation.

Extensive use was made of the DEC publication (reference 59) to assist in the programming and proper operation of the machine.

3.2 Tape Translator Development

3.2.1 The Details of the Problem

Many machine tool controller units presently use the word address or universal format for ease of reading and to facilitate control unit storage location referencing. In fact, these two formats, have become more or less industry standards due to their readability and flexibility. It was decided therefore to use the Universal format as input (although word address works equally well) to the computer. Figure 3.4 shows a coding sheet of the proposed input format. Each group of numeric symbols is preceded by a letter designating the function of the number or code. This alpha-numeric string is called a "word".

Furthermore, there may be as many as ten of these words in the standard Universal block. By registers, their functions are:

i) Sequence number, preceded by N or H depending on controller,

ii) Preparatory function code, preceded by G, which is a major instruction affecting the mode of machine operation (i.e. drill cycle, linear-circular interpolation, or other 'canned' cycles),

iii) X-axis register, preceded by X, to inform the machine

Sequence Number (N)	Preparatory Function (G)	X-Axis (X)	Y-Axis (Y)	Feed Point (R)	Final Depth (Z)	Feedrate (F)	Spindle Speed (S)	Tool Funct (T)	Misc. Funct (M)

FIGURE 3.4 Coding Sheet of Proposed Universal Input Format

of the desired X axis location,

iv) Y axis register, preceded by Y, to locate the desired Y location,

v) "Rapid-to" register, preceded by R, to inform the machine how far it can position at "rapid" positioning speed.

vi) Final depth register, preceded by Z, designates the maximum Z axis travel at a controlled working feedrate,

vii) Feedrate register, preceded by F, supplies the controller with a code which determines the X, Y, and Z feedrates,

viii) Spindle speed register, preceded by S, again in code informs the controller at what RPM the spindle is to turn,

ix) Tool function register, preceded by T, calls up non-sequenced tools by drum location or tool code,

x) Miscellaneous function, preceded by M, performs certain switching functions such as tape rewind, coolant (on, off, mist, flood) or end of program signals.

Having listed the input parameters, the attention must now be focused on the desired output, or rather, the coding required for MOOG operation. Figure 3.5 shows that no address letters are used and that there are but 7 registers per block, to handle the functions that require 10 in the universal format.

A unique system is employed in that many of the registers are able to "double up" when certain bits of information are called for. To emphasize this, a register-by-register function description is again employed.

Sequence No.	Prep Fct.	Feed Point	Final Dep.	Res. Fct.	Tool Fct. Code	Misc. Fct.
		X-AXIS 00.000	Y-AXIS 00.000			

FIGURE 3.5 MOOG Coding Blocks

i) Sequence number register, to give tape location,

ii) Preparatory function register, to determine cycle or operation made, and to control access to the Z or depth registers. These codes are unique to the MOOG, and are not in compliance with EIA standards.

iii) X registers, to give X-axis location and to double up as the "Rapid to" or Feedpoint register when so called for,

iv) Y register to locate the Y-position and inform the machine of the working Z depth when this is needed,

v) a Reserve function register, not used,

vi) a "Tool Function Code" register which handles all the codes for: a) tool drum locations,

b) X, Y, Z feedrates codes,

c) spindle speed code, in conjunction with the vii) "Miscellaneous Function Code" which, in addition to performing the switching functions as earlier described also determines which code is in the tool function code register.

The end-of-block code, common to all tape formats is the carriage return code generated by the flexiwriter. This code separates one block of information from another.

The codes available for use on the MOOG are shown in Figure 3.6. Some EIA Standard codes are shown in Figures 2.2 and 2.3.

It should be reiterated that the Universal or Word Address format does not repeat information which does not change, or con-

MOOG HYDRA-POINT PROGRAMMING DATA
Model 83-1000 Machining Center

Tool Funct. Code	X,Y Axis Feed in/min	Z Axis Feed in/min	Spindle Speed R. P. M.	Tool Funct. Code	Spindle Speed R. P. M.
01	1.0	1.0	65	21	525
02	1.9	2.1	75	22	625
03	2.7	3.6	91	23	760
04	4.0	5.3	105	24	875
05	5.5	8.1	120	25	1015
06	7.2	10.3	140	26	1140
07	9.1	13.3	155	27	1300
08	11.2	16.4	175	28	1450
09	13.3	20	195	29	1610
10	16.0	25	215	30	1800
11	18.6	28	240	31	2000
12	21.5	32	270	32	2225
13	24.7	37	300	33	2470
14	27.9	42	330	34	2740
15	31.5	48	375	35	3100
16	35.6	53	415	36	3450
17	39.7	59	465	37	3830
18	44.0	64	495	38	4090

PREPARATORY FUNCTIONS

- Position only
1 - Tap-feed control
4 - Peck
5 - Drill
6 - Mill
7 - Bore
8 - Tap-pitch control
9 - Read Z

MISCELLANEOUS FUUCTIONS

00 - Program stop
02 - End of program
06 - Tool change (manual)
07 - Flood coolant on
08 - Mist coolant on
09 - Coolant off
80 - No change
81 - Partial retract on
82 - Partiel retract off
83 - Indexing workpiece
84 - "X" Feed rate
85 - "Y" Feed rate
86 - Automatic tool change
88 - Spindle speed change
89 - "Z" Feed rate

FIGURE 3.6

versely, only those bits of information which change from one block to another are entered; those which remain constant from one block to another are not repeated. Fixed block formats demand exactly the opposite, that all parameters be listed each time (except in certain code registers) regardless of their variability from block to block.

3.2.2 The Requirements of the Conversion

3.2.2.1 Sequence Number

The sequence number is retained in the conversion, the only change being the deletion of the address letter N. The sequence number will repeat itself in cases where more than one block of information of MOOG format is generated by one block of universal format. This presents no difficulties for the MOOG reader unit.

3.2.2.2 Preparatory Function

The Preparatory (or "G") functions of the EIA standard codes in no way resemble those used by the MOOG. Hence, in addition to deleting the address letter, a different number must be summoned to activate the desired machine cycle.

Three other problems arise out of the Preparatory function translation and these are further delineated:

i) There is no common EIA standard for a "MILL" cycle to compare with the MOOG number 6 G function. Some programmers use an unassigned G79 or G78 depending on the type of cycle desired but this varies. The reason for this is that many machining

centers and NC milling machines assume milling to be the standard mode, with canned cycles taking over drilling, tapping and boring operations, among others. For this reason, the program in its present form is strictly point-to-point in that it does not include the straight-line milling capability. Another difficulty arises in the X and Y feedrate coding in this regard, which will be elucidated later.

ii) When the Z axis instructions are called for, the "9" must be inserted in this location. This should hence be determined, not by a "G" code, but rather by the existence of an "R" or a "Z" in input.

iii) Sometimes an extra block of information is required for one or more of the "M" functions, due to the rather limited flexibility of the fixed block, and the great number of T-M code combinations required to transmit the required information. When this happens, the "G" function is normally a positioning block, that is, only the switching function is called for in this particular block.

3.2.2.3 The X, Y, R, Z, Registers

Two modifications are required in the translation from Universal format to MOOG of the actual numerical values which supply the position informations for the X, Y, and Z axis movements. The first modification is the deletion of the address letter. The other is the truncation of the number from 6 digits to 5. A note of explanation may be in order here.

In most closed-loop control systems of NC machines, the actual tool position is compared continually with the position stored in the register. The closed-loop concept furthermore depends on an error signal for its operation. To maintain accuracy of say $\pm$ .001 inch, the electrical resolution in an electromechanical system must be $\pm$ .0001 inch. The MOOG system, dependent on varying oil pressure for activating the axis movements, needs no such .0001 resolution, hence the X, Y, R and Z inputs use only 3 decimal places instead of the 4 required by an electromechanical NC control system.

The result is that the last digit can be dropped in the conversion. It was initially proposed to insert a rounding up routine whenever the last digit had a value of 5 or more but the problem of developing such a routine was deferred to allow for initial debugging of the main logic. Hence the last digit is now merely dropped. This has the effect of increasing the positional error from an advertised $\pm$ .003 inches to $\pm$ .004 inches, which is still acceptable in NC machine tool technology. It was felt that this error was a small price to pay for the initial reduction in required programming effort.

Another problem, unrelated to the numerical values, arises out of the split block system used on the MOOG. Whenever a G code of 9 is used, the information following that block refers exclusively to Z axis movements, that is feed point, final depth and working feedrate of the tool. Herein also lies another problem

with milling, although this is easily circumvented, in that it is difficult to separate out to which axis the feedrate applies. Sophisticated machines do not require this sort of information as they are able to determine feedrates in non-axial directions as well as axial. The MOOG however requires separate coding for the feedrates in each of the coordinate directions. In its present form, the program only handles point-to-point work so this is not an issue at the present state of development.

3.2.2.4 The Reserve Function Register

A zero must be entered in this register. It has no function at the present.

3.2.2.5 The F, S, T, M Registers

These are here discussed as a group because in their implementation on the MOOG, various combinations of Tool function code and Miscellaneous function designate which machine function is to be controlled and what the input code of the controlled function is to be. Figure 3.6 lists the combinations which can be applied.

Herein also lies the greatest deficiency of the program. The problem definition loosely states that a table of values be consulted to relate the feedrate in inches per minute to tool function code. To properly execute this part of the program it must be able to recognize, and translate at least three different types of feedrate-spindle speed codes in addition to that of the MOOG. These are:

i) EIA Magic 3 code. - This code gives as its first digit the order of magnitude of the number (with its base as 10^{-3}). That is if the first digit of the code is a 1, the feedrate is in hundredths of an inch per minute. Each order of magnitude higher increases this digit by one. Thus a feedrate of 2 inches per minute would appear as 420 where the 4 indicates the magnitude of the number and the 20 its numerical value.

ii) EIA Inverse time feed code.

"Inverse time feedrate coding provides the control with a BCD feedrate code which is equal to the reciprocal of interpolation time in minutes". [56] The code is calculated according to the formula $FC = \frac{I}{T}$ or $\frac{V}{S}$ for linear interpolation and

$$FC = \frac{\theta}{T} \text{ or } \frac{V}{R} \text{ for cicular interpolation}$$

where V = Vector
S = Vector distance in inches
T = time in minutes
R = arc radius in inches
and θ = arc length, in radians

In this case a feedrate of 2 inches per minute and an axis distance of 5 inches would be carried out as follows:

$$FC = 2.0 \frac{in}{min} \times \frac{1}{5in} = \frac{2}{5min} = \frac{1}{2.5min} \text{ or } .4/min$$

iii) Bendix increased resolution;

The machines using Bendix increased resolution require a slight modification of the above formula which effectively increase the flexibility and accuracy of the coding.

For this sort of calculation, it is imperative that float-

ing point decimal numbers may be used. Since all of the calculations in the core ordinarily use BCD with 3 bit numbers, the routines required to execute the functions above described would be extremely complex. Here again, it was decided to forgo this whole area and simply code in directly, the MOOG feedrate codes for the F and S designations. Hence, the input to the program is removed from the ideal by the lack of this code selection facility.

Some approaches which could be taken to include this facility are:

i) use of a polynomial curve with coefficients such that when a given feedrate-spindle speed is inserted into it, the proper feed code is punched on the output tape. This is not really the answer since very few machines read IPM or RPM directly.

ii) a comprehensive routine to: (a) sort out the type of input code (Magic 3, Inverse Time, Bendix etc.), (b) relate it to speed in inches per minute or revolutions per minute and (c) select the proper tool function code from a table of values.

The additions outlined above would possibly require by themselves as much time as was necessary to develop the main program and its attendant subroutines. The problem in that case needs to be redefined so that either a general program, or a specific input is to be dealt with. Some questions remain as to whether the core storage is sufficient to handle these added capabilities.

3.3 Program Development

3.3.1 Basic Concept

The requirements of the MOOG basically are, in terms of block variations:

i) A repeatable register for X and Y values

ii) A non-repeatable register for the Z axis values

iii) A register for auxiliary functions

Item (iii) may be considered optional but in terms of program development it was included since in the Universal format certain M functions may appear in the same block as spindle speed and feedrate information. Due to the MOOG M and T function setup, this is not possible to duplicate unless an extra (positioning) block is programmed.

Since some bits of information in the Universal block could by themselves prompt the output of a whole block of information for the MOOG, it was considered necessary to process the incoming tape block by block rather than, perhaps bit by bit as do some translation routines. With the above variations in mind, the generalized alorithm for the program became:

i) Read a block of tape and store it in an input memory

ii) Process the information according to a predetermined priority scale for each information word.

iii) Set up an output array (or arrays) and type it (them).

The three output memories, each exactly 20 characters long (see fig. 3.7), when filled and punched each constituted one

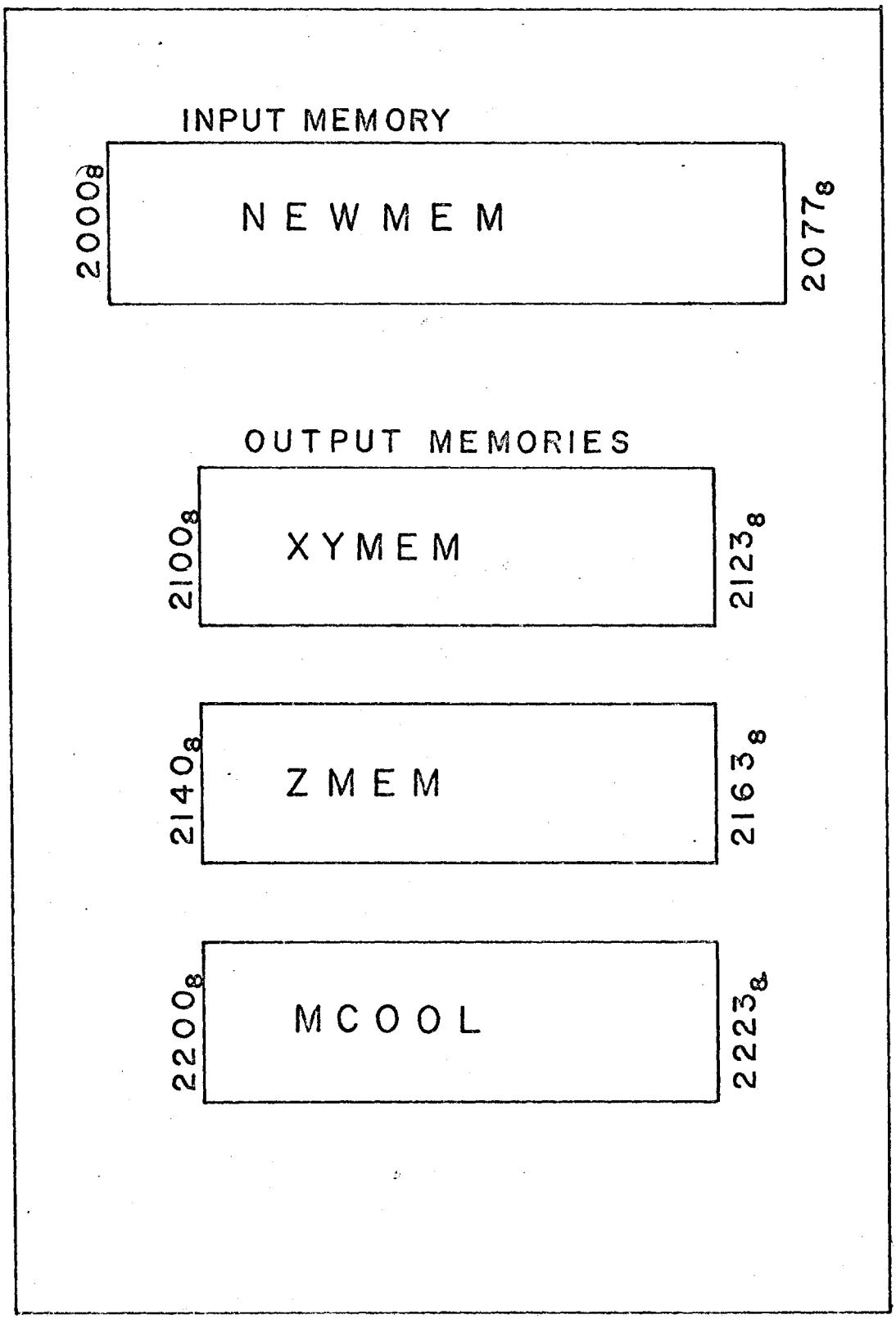


FIGURE 3.7 MEMORY STRUCTURE IN MOOG TRANSLATOR

block of MOOG information. Their contents are described above in terms of requirements. The input memory is simply a catch-all for all the information on the input tape. It must be cleared when a block of tape has finished processing.

Due to the continuous recycling nature of the program, there is no theoretical or actual limit to the physical length of the tapes which it may be called upon to process.

3.3.2 PAL III Implementation

A listing of the entire processor as developed is given in the appendix. Following is a listing of the symbols used and their role in the program. In-program addresses, whose only role is to locate a line (for a "Jump-to" statement for instance), are not included. The computer language used is PAL III. The symbols appear here in the order of their occurrence in the processor.

INDEX -denotes a special register (at address 10_8) which automatically increments a previously defined counter each time it is called. This register is used in the memory clearing subroutines.

TYPE -the call symbol for the subroutine which operates the teleprinter/punch unit.

EOB -call symbol for the punching of an end-of-block code when one block of information has been output.

COUNT -a location used to store and increment a cycle counter (for clearing arrays).

XYMEM -address 2100_8. This is the starting address of the X-Y output memory.

NEWMEM -address 2000_8. This is the starting address of the input memory.

MCOOL -address 2200_8. This is the starting address of the auxiliary (M) function memory.

ZMEM -address 2140_8. Starting address of the Z register memory.

XYCTR, MEMCTR, COOLCT, and ZCTR all hold counters which determine locations in the above memories.

CR, END -CR designates an octal 7600, which is the two's complement of END, an octal 200. The octal 200 is repeatedly used to determine when the end of an input block occurs, when an output memory should be typed, and to separate the blocks on the output tape.

M1, M2, M3, M5, M19, M18, -These symbols all refer to counters which zero the accumulator after 1, 2, 3, 5, 19, 18 cycles respectively.

STOP 1, STOP 2, STOP 3, STOP 5, STOP 50 -These are the locations which hold the counters referred to above.

K277, K243 - ASC II codes for use of the teleprinter in the error message subroutines.

SIX, NINE, ZERO -The EIA codes require that only an odd number of holes appear on tape for any given symbol. For this reason a parity check is added where the binary code would produce an even number. These numbers refer to their octal equivalents according to the EIA code.

MASK -used in a logical AND operation to ensure that no hole will be punched on the eighth track of the tape.

CHEX to AGAIN - with a few exceptions, these are links which enable the page-oriented computer to move from a core location on one "page" through a location which is accessible from all parts of the memory, to a predetermined location on another page.

Using the foregoing general vocabulary, and other local terms, the algorithm used is given below: The numbers on the program listing (see Appendix B) and the flowchart (fig. 3.8) corre-

spond to those here assigned to the sequence of operations.

*1. Clear locations 2000_8 to 2277_8 with an EIA zero
*2. Punch a leader
3. Deposit an octal 200 in the 20th location of each of the output memories
4. Set output memory counters to their respective first locations
5. Read a block of tape and deposit the contents in the input memory (NEWMEM)
6. Reset the input memory to 2000_8 and search for an N
7. Cannot find N, return to 5 and/or type out an error message number.
8. Found N, pick out the following three numbers and deposit in first three locations of XYMEM (Sequence numbers)
9. Reset input memory and search for an M
10. Found M, go to subroutine to process M and output a suitable tape block, return to main program at 5 or 12, depending on the type of input
11. No M, deposit an 80 in the last two valid locations of XYMEM (MOOG code for no change)
12 Reset input memory and search for G
13. Found a G, jump to a subroutine to generate the appropriate code
14. No G, continue
*15. Search for an X
*16. Found X, select next 5 characters and deposit in the assigned locations of XY memory
*17. No X; search for a Y
*18. Found Y, repeat 16 for Y
19. No Y, search for a spindle (S) code
*20. Found S, obtain code and transfer to XY memory register

location

21. No S, check for an R
22. No R, clear input memory and return to 5
23. Found an R, organize sequence and preparatory function registers.
*24. Allocate the R and Z values to their proper output memory locations
25. Search for an F, if none, go to 5
*26. Found F, insert F code into tool function register and type the contents of the Z memory (ZMEM)
*27. Clear input memory
28. Return to 5 (to repeat steps 5 - 28)
*29. Punch a trailer if a tape feed symbol (octal 177) is encountered

The search routines execute by comparing the two's complement of the octal code with the input character. When the character in question appears in the accumulator along with its two's complement, the accumulator becomes zero. This then prompts a call out of the loop to a subroutine or some other predefined location in the program. To protect the program an additional counter is built in to make sure that not more than 50 cycles are executed for any one search. When this number is reached a number of things may happen

a) the program moves on to the next sequence

b) an error message is typed out

c) a subroutine is called

The operations preceded by an asterisk indicate that the task is performed at a location outside of the main program (see

flowchart, fig. 3.8). Future additions to the program, if desired, could easily be incorporated by the insertion of more subroutines with their appropriate links. In this way some of the aforementioned deficiencies could be amended without altering the logic of the main program.

3.3.3 Implementation on the PDP8/L

The details of programming in the PAL III assembler language are well described in reference 59. The program was compiled with the aid of a "Symbolic Editor" which facilitates writing, correcting and editing of the symbolic tapes. These tapes are then processed by a three pass service program which translates the symbolic programs into binary programs. The functions of each pass are given below.

Pass 1. A symbol table is compiled, and a check is made for certain error modes.

Pass 2. The binary tape is generated using the symbol table of pass 1 and a check is made for an illegal reference error.

Pass 3. A complete assembly listing is either typed or punched which facilitates debugging and modification of the program.

No attempt was made to aim for peak efficiency in program execution since the output elements (i.e. the teleprinter/punch) were so many orders of magnitude slower than the operation of the program itself that if it worked at all, it was always waiting for

the punch unit.

A great deal of time was lost in dealing with the problems of equipment breakdowns, both in the input/output devices and the processor unit itself. The processor unit problems were mainly restricted to insufficient voltages and related problems. The I/O breakdowns were largely mechanical in natrue due to the over-use of this equipment.

3.3.4 Results

Approximately 80 hours of time were spent in the compiling and debugging of the package as it exists in its present form. Some early problems arose out of unfamiliarity with equipment and as the program development progressed the logic errors became prominent.

Many of the errors were detected by means of a "single step" switch which permits the operator to manually advance the computer through the sequence of operations while observing the contents of the memory address, memory buffer, and accumulator registers.

3.3.5 Observations

Fig. 3.9 shows a flexiwriter printout of (a) the input tape and (b) the processed tape from the computer ready for the MOOG. In accordance with the initial requirements, the following details should be noted:

a) All X and Y values are repeated on the output tape regardless whether they change or not.

b) Up to 3 blocks of information are generated in MOOG format from one block of Universal format. When this happens, the sequence register does not advance, rather it retains the number assigned to the block from which it orginates on the input tape. This was seen as an option but in this case a desirable one. It enables the machine operator to use the program manuscript from the Universal tape as a debugging aid without the need for examining the somewhat confusing MOOG format.

c) A 5 block leader and trailer is punched on the output tape.

d) The program begins its actual cycle when it comes across the first "N" character on the tape. It halts when a tape feed character from the flexiwriter appears in the reader.

e) The resolution of the 6 digit input values is reduced by a factor of ten. The numbers are truncated, not rounded off.

f) Error subroutines do appear in the program, but only one is presently operational. It should return a "? No. 2" when a toolchange is specified but no tool is given. An error subroutine to point out the lack of an "N" symbol was bypassed in the progressive development of the program however it was decided to leave it in there as some other use might be found for it later. If now there is no N symbol on the input tape, the program will recycle itself until it sees an octal 177 at which time it will halt.

g) The milling capability is not included in this program It does however have all the positioning cycle capabilities of the

MOOG.

h) Tool-function codes are merely F and S codes previously selected from the MOOG speed and feed selection table with the address letter deleted (also the tool number).

i) Separate blocks are punched for activating or de-activating spindle coolant.

j) The program presently operates only with a low-speed teleprinter/punch unit. It is possible however to convert it for use in the high speed system. This might be desirable for generating very long control tapes. The only modifications are that the TLS, TSF, KRB, and KSF micro instructions are replaced by their high speed equivalents wherever they occur.

k) The program is presently loaded on magnetic tape for convenience of handling and use.

3.3.6 How To Use the MOOG Translator

i) Turn PDP "power" switch to "on" position,

ii) Load cassette in tape reader,

iii) Set switch register to octal 7777, depress "LOAD ADDRESS", depress start.

iv) Teletype returns with "READY", type "L MOOG",

v) Set switch register to 200,

vi) Load input tape in tape reader, turn reader to "START", depress LSP "ON",

vii) Depress "LOAD ADDRESS", depress "START"

viii) Reader should read block by block while the MOOG tape

is generated. The program will halt itself.

ix) Remove tapes, turn POWER to "OFF".

x) (optional) mount output tape in flexiwriter to obtain a listing of MOOG format.

Certain precautions should be taken when preparing a tape for this program. Any symbol except a "TAPE FEED" character may appear on the leader of the input tape. One possibility is to type a string of zeros on the tape, perhaps 20 or so, and then begin the actual part program. Secondly, if the input tape is to physically halt the program, the last block of information must be followed by a carriage return on the flexiwriter and then the tape feed button must be used. When the tape feed symbol enters the processor it will cause a 5 block trailer to be punched and the processing will stop.

3.3.7 The Applications and Physical Requirements of the PDP - MOOG Translator.

If a translator program such as the one here developed is to be put to practical use certain basic requirements are necessrry. These are:

1) A Universal format control tape generating facility. This may be a computer, a Flexiwriter, or a time-sharing terminal. (It could also be Word Address format).

2) A PDP-8/L or 8/I computer with basic 4K core and an ASR33 Teleprinter/Punch unit. The program requires about 1.6K bits

of core from address 10_8 to 2223_8. Hence there is a large amount of unused core which may serve in some future developments.

3) The need for a translator facility, that is two or more machines with radically different control systems yet which may be called upon to manufacture similar workpieces.

The problems of actually accomplishing the translation to its fullest extent have already been discussed in an earlier section. It should be noted however that there are a great many different types and brand names of NC machine tools with perhaps an equally great variation in the coding required to fully realize the potential of each machine. This in itself may be considered a justification for not persuing further the task of a fully generalized speed, feed and tool selection facility. Indeed, if it were possible to include such a facility, the need for post-processors described in a previous section could be called into question.

A suggested operation mode of the system at McMaster would be to

1) Prepare a Machine Tool Description File for the MOOG, such that the tape generated would be of Universal or Word Address format, but that the auxiliary function codes would be MOOG compatible.

2) Use this tape as an input to the PDP 8 computer program here described to generate the control tape for the NC machine.

3.3.8 Suggestions for Future Development.

Some of the omissions or deficiencies of the present state of the software have already received considerable attention. It may be desirable however to increase the usefulness and facility of the system with the addition of the following capabilities:

a) The full incorporation of the MILL cycle (could be triggered by a G79) with its axial X and Y feedrate instructions,

b) A speed and feed selection routine relating the MOOG tables (fig. 3.6) to some standard code (such as the EIA Magic Three).

c) High speed tape reader and generator capabilities.

3.4 Summary

The infancy of the NC machine tool technology has been plagued with the same problems which have accompanied many other areas of rising technological development, that is, an uncoordinated, cancerous growth. One need only briefly survey the early history of the automobile, when it was possible to see or drive steam cars, electric cars as well as those powered by a wide variation of internal combustion engine. It seems that in the manufacturing industry, as in nature, those who are the most fit survive.

The evolution of NC technology has, in addition to providing the user with a rather confusing collection of machinery, consistently disregarded or overpowered the attempts of some to standardize the modes of input to these machines. In a few things

there are agreement , such as the use of a 1" wide eight track paper tape, the general use of the EIA code (in North America), and the application of a few standard codes for machine operation.

While this area of technology is sorting itself out, some aids are required to assist the programmer/operator to in some cases interface one machine with another. A sister technology, that of integrated circuit mini-computers lends itself well to this application. These units are relatively inexpensive and can, in addition to their roles as tape processing units, be used as problem solvers in the shop.

The project undertaken here of writing a tape translator program served to illustrate the following:

1) The use of the mini computer in its application to non-academic problems,

2) An introduction to software design and the use of the assembler language,

3) The need for a more cohesive, more binding set of standards if the NC concept is to survive in the face of rising costs. A counterargument may be raised with this point that standards hinder development. In the rather common areas of information handling this may be especially critical.

4) A coordinated approach to the development of a more universal information handling system for NC should be taken, failing the establishment of standards as in 3.

5) Some of the problems involved in postprocessor writing.

Some of the functions performed by the program here developed have their direct counterparts in the postprocessor concept.

3.5 Conclusion.

The MOOG Tape Translator is one attempt to bridge the gap between the various tape formats available for NC machine tools. In so doing, it must also try to cope with the machines themselves, their peculiarities, and their specialties. In its present form, the program is probably not too useful, and it would be most desirable if the need for such a translator would disappear entirely. It has however, been more than a mere academic exercise in that the potentialities of the software-hardware combinations available today need more publication, to gain more non-academic support.

Further development of this translator is better geared to mating two specific machines, such as the impending MARWIN and the MOOG, for simple parts. The translation of any of the standard formats to any other as given in the Project Definition will only multiply the difficulties brought to light in this section.

FLOWCHART OF MOOG TRANSLATOR PROGRAM

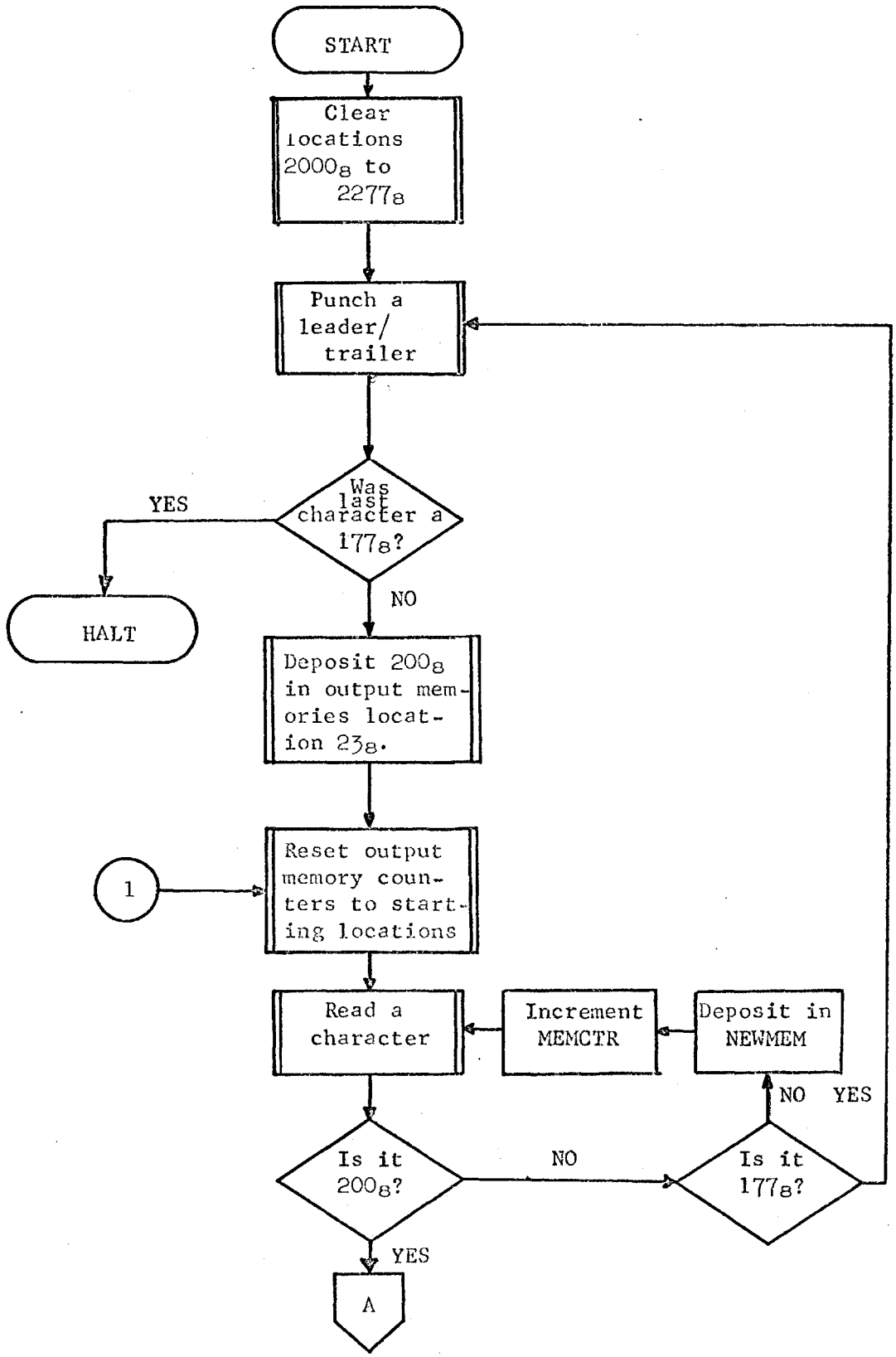


FIGURE 3.8

....continued....

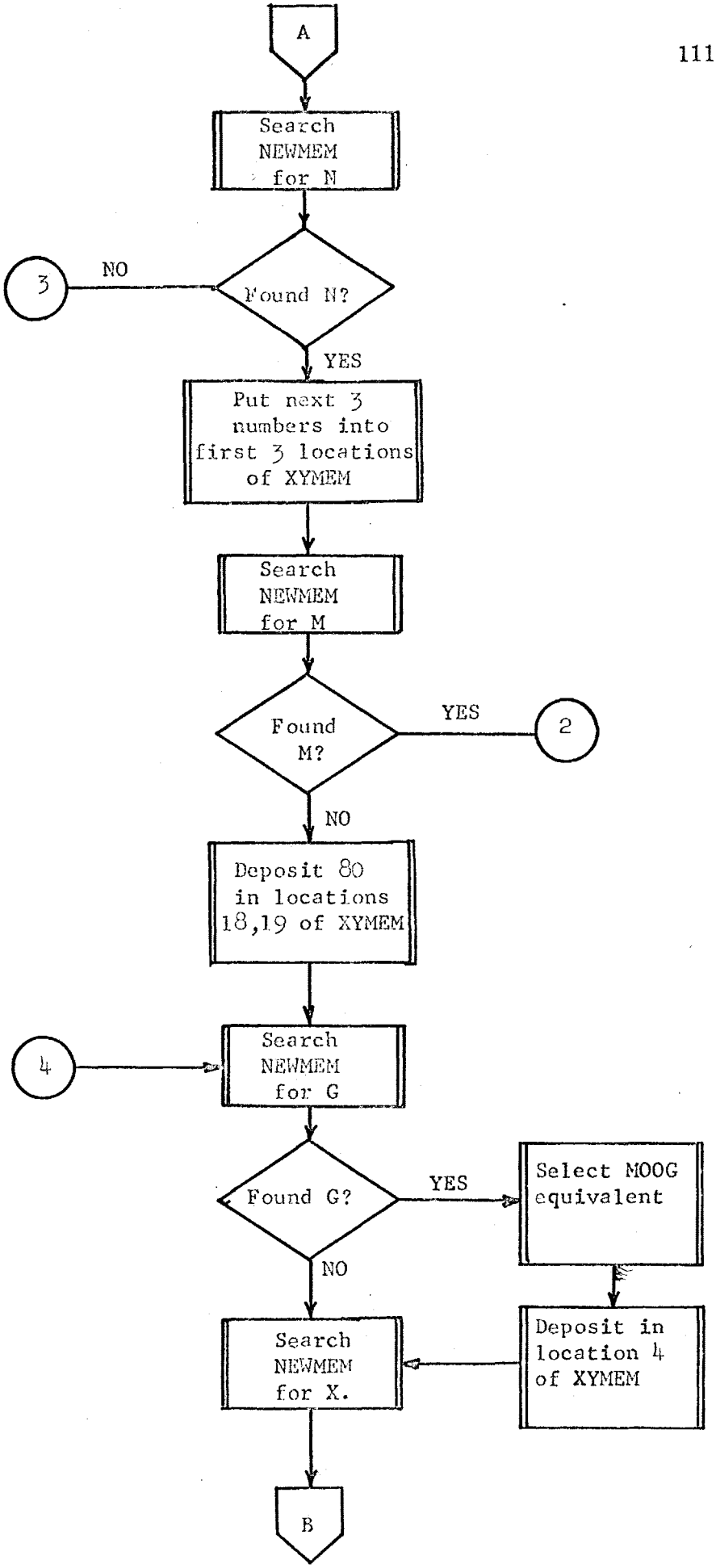


FIGURE 3.8

....continued....

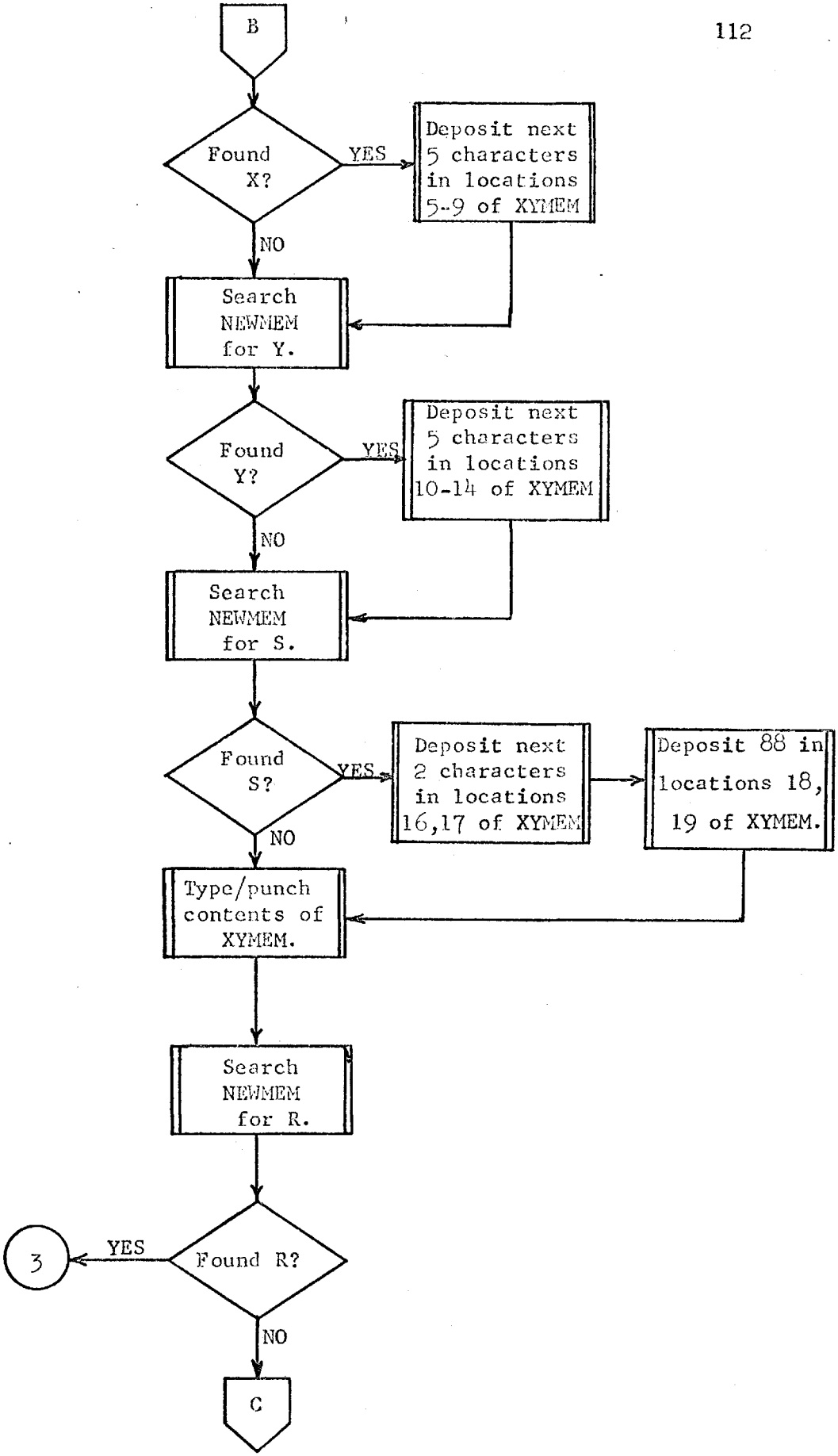


FIGURE 3.8

....continued....

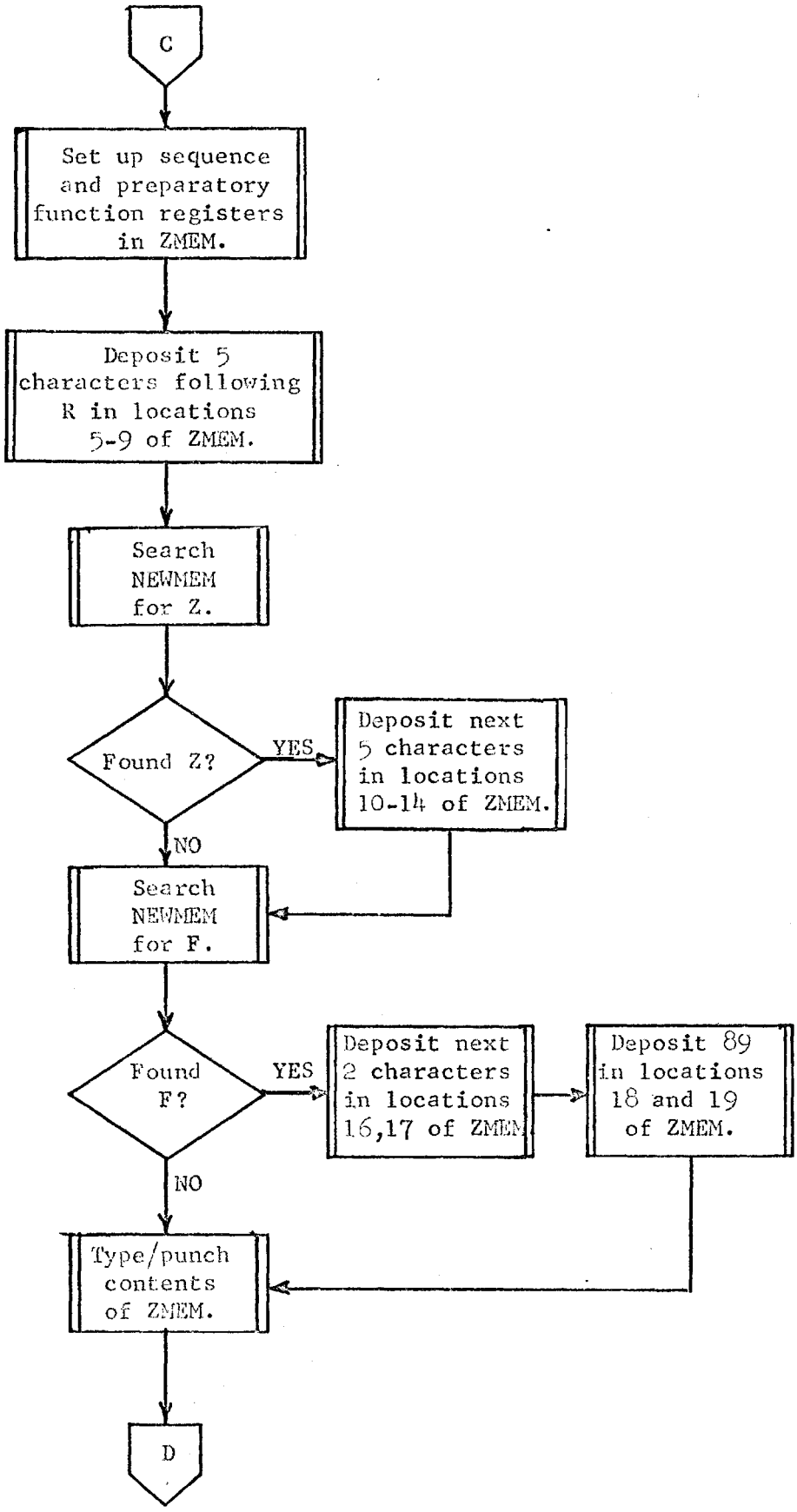


FIGURE 3.8

....continued....

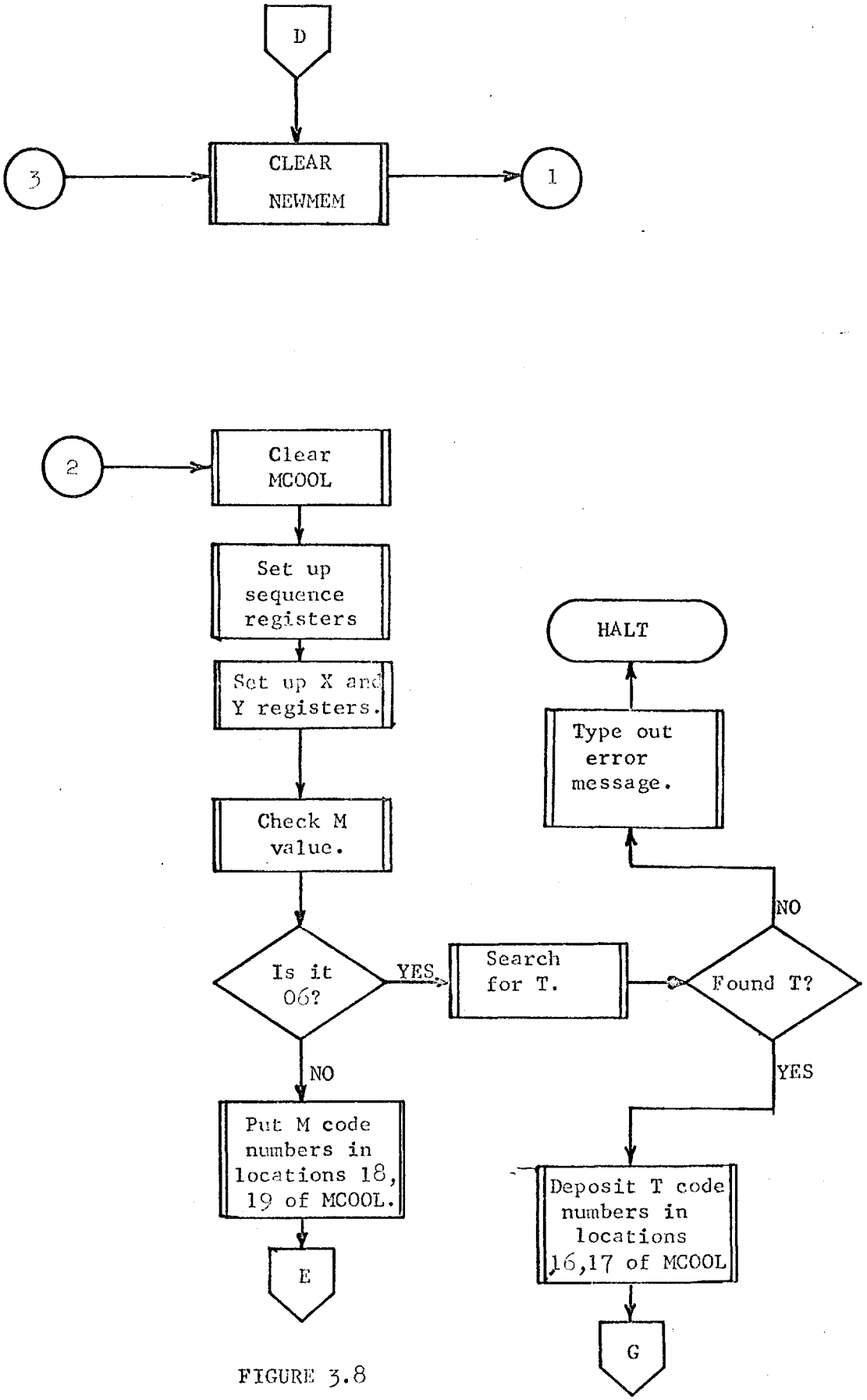


FIGURE 3.8

....continued....

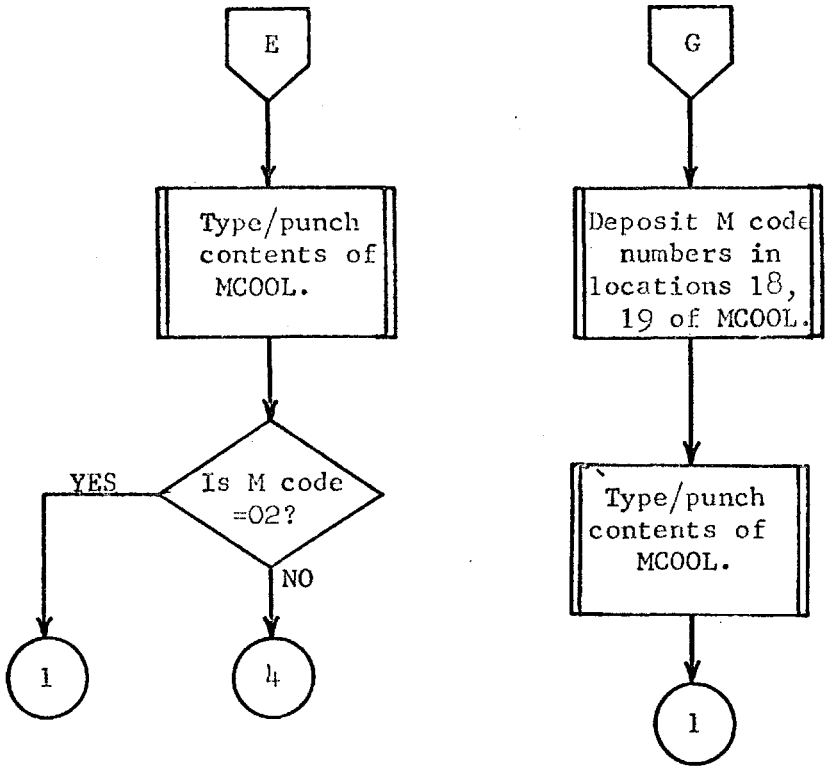

E
Type/punch contents of MCOOL.
Is M code =02?
YES
NO
1
4
G
Deposit M code numbers in locations 18, 19 of MCOOL.
Type/punch contents of MCOOL.
1

FIGURE 3.8

```
N001       X190000 Y000000                                    M02
N002       X000000                                      T01   M06
N003 G81   X027190 Y040560 R025030 Z026660 F12 S31       M08
N004       X049060
N005       X070940
N006               Y058690
N007               Y076810
N008       X049060
N009       X027190
N010 G80   X000000 Y000000                              T02   M06
N011 G81   X027190 Y040560 R016620 Z020370 F04 S31
N012       X049060
N013       X070940
N014               Y058690
N015               Y076810
N016       X049060
N017       X027190
N018 G80   X000000 Y000000                              T03   M06
N019 G81   X027190 Y040560 R021310 Z023360 F09 S29
N020       X049060
N021       X070940
N022               Y058690
N023               Y076810
N024       X049060
N025       X027190
N026       X190000 Y000000                                    M02
```

FIGURE 3.9(a) Input part program.

```
00101900000000000002
00200000000000000186
00300271904056000008
00350271904056031 88
00390250302666012 89
00450490604056000 80
00650490605869000 80
00750490607681000 80
00850490607681000 80
00950271907681000 80
01000000000000000286
01150271904056031 88
01190166202037004 89
01250490604056000 80
01350709404056000 80
01450709405869000 80
01550709407681000 80
01650490607681000 80
01750271907681000 80
01800000000000000386
01950271904056029 88
01990213102336009 89
02050490604056000 80
02150709404056000 80
02250709405869000 80
02350709407681000 80
02450490607681000 80
02550271907681000 80
02601900000000000002
00000000000000000000
```

FIGURE 3.9(b) Output MOOG part program.

4.0 TIME-SHARING SYSTEMS FOR TAPE GENERATION.

4.1 Introduction.

Another project for this thesis was loosely defined as writing a postprocessor for the MOOG Hydra-Point which could be used with the APT system available on the CDC 6400 at McMaster. Due to the rather considerable time spent on the translator and its inherent problems it was decided to search for a system which could be addapted to the operating requirements of the MOOG with a minimum of time and effort.

Subsequent investigations revealed that the General Electric Time Sharing system presently available on the terminal in the Computer-Aided Design room already contained a system for the part programming of point-to-point NC machines with $2\frac{1}{2}$ axis capability. This system is briefly dealt with in a prior section (2.3.3.3) of this thesis. It was decided to prepare a program in the G.E. system which would generate tapes for positioning and straight line milling work on the MOOG, which is presently programmed by special purpose computer programs, [60, 61] to cut mathematically defined curves.

This is not to call into question the work done by Mr. Husemeyer[60] and Mr. Wong[61], rather it is an effort to provide the sort of simple programming, which can be learned quickly by shop personnel, and applied directly to producing control tapes for the work going on in the Machine Shop at McMaster. The spe-

cial purpose programs are useful in that they enable the machine to do work for which it was not really designed.

4.2 Program Development

4.2.1 The Machine Tool Description File.

The Machine Tool Description File, or MTDF, has been discussed in terms of function, that is, to build up a machine-oriented vocabulary which can later be used to write a part program. The specific items of information required are as follows: [48]

i) Special vocabulary words.
ii) Standard vocabulary words.
iii) Standard and special characters.
iv) Number of registers available on the NC controller.
v) Word address characters for each register, if applicable.
vi) Output formats for each register specified.
vii) Designating whether the machine has absolute or incremental moves.
viii) Determination as to the necessity of repeating a register if the next value is identical to the one currently used in the register.
ix) Assignment of each register to its particular special or standard vocabulary word.
Answering specific questions relating to special registers and options.

Fortunately for the potential user, a great deal of the work has already been done in the preparation of the MTDF in the form of a generalized MTDF called GENMAC. This file of which a listing appears in figure 4.1, can be summoned from the remote terminal, modified and renamed to suit the requirements of the user, on the remote terminal.

With reference to the foregoing necessary items of information in the MTDF it should be noted that all but item (ii) may

GENMAC

```
100 COOLNT FEDRAT SPINDL TOOLNO FUTURE
110 FUTURE FUTURE FUTURE FUTURE FUTURE
120 FUTURE FUTURE FUTURE FUTURE FUTURE
130 REWIND DRILL  FACE   TAP    BORE
140 MILL   DEEP   DWELL  STOP   OPSTOP
150 CANCEL FUTURE FUTURE FUTURE FUTURE
160 FUTURE FUTURE FUTURE FUTURE FUTURE
170 FUTURE FUTURE FUTURE FUTURE FUTURE
180 ORIGIN TRANSL FROM   GOTO   GODLTA
190 GOCIRC GOLINE TMARK  SEQNO  RESET
200 FINI   BLOCK  RAPTO  FEEDTO XYSW
210 HBLOCK TRIGER MOVE   REPEAT FORMAT
220 END    CYCLE  DPT    PURGE  FNCALC
230 "="    ";"    "*"    " "    " "
240 " "    ";"    " "    " "    "_"
250 10
260 N   G   X   Y   R   Z   F   S   T   M
270 3.0 2.0 6.4 6.4 5.4 5.4 2.0 2.0 2.0 2.0
280 0   0   0   0   0   0   0   0   0   0
290 10  7   8   9 0 0 0 0 0 0 0 0 0 0 0 0
300 10 2 2 2 2 2 2 2 10 10 2 14*0
310 30 82 83 85 86 87 84 04 00 01 80 14*0
320 1   3   4   5   6   H   1   7   500.0
```

FIGURE 4.1 GENMAC listing of NCPPX program.

be user specified. These usually unaltered vocabulary words appear in lines 180 through 230 of figure 4.1.

4.2.2 The Preparation of the MOOG MTDF.

With the generous assistance of the numerical control specialist at C. G. E. in Toronto [63], a MTDF for the MOOG was quickly prepared. Most of the information used was found in the operators manual, a summary of which is found in Figure 3.6. Figure 4.2 shows the present status of the MOOG MTDF (MCMOOG by file name). A line by line description of the information and required modifications follows.

Lines 100 - 130

The first three lines refer to registers which hold multi-valued vocabulary words. There are fifteen such registers, all of which are available for the purposes of the user. Commonly used control words in this position of the MTDF could be, for example feedrate, spindle, speed, tool number, coolant, to mention a few. The MCMOOG MTDF does not require address letters and the multiple values occur primarily in the tool function code column, hence it was decided to set up a register "TFUNCT" to handle all codes for feedrate, spindle speeds and tool numbers, the numbers to be determined from the table.

The "COOLNT" control word was not basically modified, however the "RETRCT" function may be called up to either turn the spindle retract facility on or off. The other words listed in GENMAC (fig. 4.1) were deleted as their functions are now all

```
100 COOLNT RETRCT TFUNCT FUTURE FUTURE
110 FUTURE FUTURE FUTURE FUTURE FUTURE
120 FUTURE FUTURE FUTURE FUTURE FUTURE
130 POSITN TAP PECK DRILL MILL
140 BORE PITCH ZREAD PRSTOP ENDOFP
150 ATCHG MTCHG NOCHNG XFEED YFEED
160 SPINDL ZFEED RESFCT NOX   NOY
170 FUTURE FUTURE FUTURE FUTURE FUTURE
180 ORIGIN TRANSL FROM   GOTO   GODLTA
190 GOCIRC GOLINE TMARK  SEQNO  RESET
200 FINI   BLOCK  RAPTO  FEEDTO XYSW
210 HBLOCK TRIGER MOVE   REPEAT FORMAT
220 END    CYCLE  DPT    PURGE  FNCALC
230 "="    ";"    "*"    " "    " "
240 "0" ";" " " " " " " "
250 9
260 " " " " " " " " " " " " " " " " " " " " " "
270 3.0 1.0 5.3 5.3 5.3 5.3 1.0 2.0 2.0
280 1 1 1 1 0 0 1 1 1
290 9 9 8 12*0
300 8*2 9*9 7 3 4 5*0
310 0 1 4 5 6 7 8 9 00 02 86 06 80 84 85 88 89 0 1E10 1E10 5*0
320 1 3 4 5 6 N 1 0 0
```

FIGURE 4.2 Listing of MCMOOG MTDF.

taken over by the "TFUNCT" array.

Lines 130 - 170

The second group of control words is comprised of an array of as many as 25 single-valued vocabulary words. That is, calling up any one of these words by itself will cause a certain control code to be inserted into an assigned register. In specifying these words care must be taken to ensure that [49]:

1) They consist of from one to six characters delimited by a space or a comma

2) The control words start with an alpha character (upper case preferably). These may be followed by numeric symbols.

3) There is no duplication of these with the multi-valued vocabulary words or any other words in the MTDF vocabulary.

These constraints apply also to the first block of data, i.e. the multi-valued words.

The words assigned to execute the single valued codes were again drawn from the operator's manual and condensed so as to most appropriately describe their function. Of these, the first eight words refer to preparatory function codes, thus controlling the mode of machine operation by inserting the proper numeric code in the fourth column of the block (following the sequence number). Refer to figure 3.6 for the relationships.

The second series of nine words in this part of the MTDF, beginning with PRSTOP in line 140 and ending with ZFEED in line 160 generate codes which are entered in the Miscellaneous function

register. These may require slightly more explanation, which follows:

PRSTOP - Program Stop - 00
ENDOFP - End of Program - 02
ATCHG - Automatic tool Change - 86
MTCHG - Manual tool Change - 06
NOCHNG - No Change in Miscellaneous Function - 80
XFEED - X-Feedrate code for milling - 84
YFEED - Y feedrate coding - 85
SPINDL - Designates that the TFUNCT register refers to spindle speed - 88
ZFEED - TFUNCT register controls Z-axis feedrate - 89.

The RESFCT word refers to the Reserve function column in the MOOG block. This word must appear in the first executable series of statements of the part program. The NOX and NOY words (for no X and no Y) must be used when a non-motion block such as a "Read Z" block is called for. If these are not used, the strict format will be violated and the X and Y coordinates used previously will appear in that block.

Line 170, and all other FUTURE words are empty registers which may be used.

Lines 180 - 220

The contents of this block of words is generally left as is to form a basic vocabulary. Some of the motion commands such as GOLINE and GOCIRC may be used to generate temporary hole patterns, either linear or circular. All of the motion commands cause a block of information to be executed, hence only one of these should be specified, if it is required, per block instruction. An added condition is that the motion command be given after all other information in the block has been specified.

The ORIGIN statement permits the part programmer to use a location on the workpiece for his program origin and then specify where the machine origin will lie with respect to that location. Having done this the computer makes suitable additions and subtractions to the output data to validate it for machine usage.

These are only brief comments about the actual function and power of each statement. Reference 48 should be consulted for more precise and detailed information.

The vocabulary for the Part Programmer is now complete. He must now supply the system with various other bits of information concerning the actual formats, single valued word numeric values, and so on.

Line 230

Data Element 1 defines the TMARK register. The semicolon is traditionally the end of block character and the asterisk in the third location defines the "through" concept as in P1 through P10. The remaining locations are system spares.

Line 240

The zero in the first location signifies that all trailing zeros are to be included as for instance a 4.0 would always be read as 04000. Other possible modifications are suppression of leading or trailing zeros. The second data element again refers to the end of block character. Since the MOOG does not require a tab character or positive/negitive signs in its tape format the remaining elements are left blank.

Line 250

It has been established that there will be nine registers to serve the following data inputs!

1. Sequence number
2. Preparatory function
3. X coordinate register
4. Y coordinate register
5. Feedpoint or Rapto register
6. Final depth or FEEDTO register
7. Reserve function register
8. TFUNCT or Tool Function code register
9. Miscellaneous function register.

Line 260

Because no address letters are used in the MOOG format, all these locations are left blank.

Line 270

The formats of each register must be specified. The number preceding the decimal point refers to the size of the register and the number following the cedimal indicates how many decimal places the register contains. For example the number 15.603 has a format of 5.3.

Line 280

An item which changes from one machine to another is the repeatability or lack thereof, of the data in the block, regardless of whether or not this changes. As stated previously, the MOOG demands repeatability in everything except the Rapto and Feedto registers, hence a "1" is entered for every register except these two. This has the effect of necessitating only one call for a code such as the RESFCT in order to insert it in every block.

Line 290

The register storage positions for the Multi-valued words, are here specified. This now means that the COOLNT and RETRCT codes will only appear in register 9 and the TFUNCT values will be confined to register 8.

Line 300

A similar register storage designation must be applied to the single-valued words. This involves assigning the first eight words to register 2, the next nine words to register 9, the RESFCT to register 7, and the NOX and NOY functions to the third and fourth registers srepectively. The remaining words are "FUTURE" and are assigned zero.

Line 310

The single valued words, having been assigned to their locations also require numeric codes. These codes are specified in the order that the words appear in lines 130 to 160 above. Note especially that the RESFCT code is merely a zero and also that the NOX and NOY words execute their function by inserting a very high value into the register, consequently suppressing the output.

Line 320

The first five characters assign the sequence number, the first and second coordinates, the rapid spindle feed, and the controlled spindle feed to depth values to their proper registers. The letter N means nothing; some control systems have what is

called an HBLOCK wherein the sequence number is preceded by an "H" address letter. This facilitates identification of certain locations in the program for tape recycling. The insertion of anything but an H in this location suppresses this feature. Single or double spacing of the part-program listing is controlled by inserting either a 1 or a 2 in the location following the N. In this case single spacing was called for. Since the MOOG does not have a feedrate number calculation register, the next two locations are zeros.

This completes the MTDF for the MOOG. The system is now ready to generate control tapes/listings from a part program manuscript.

4.3 Part Programming in NCPPX with the MCMOOG MTDF.

To illustrate a few of the facilities available as well as methods involved in programming a short part program was written, shown in figure 4.4. Again, as in the foregoing section, a line-by-line commentary will be given to elucidate the program.

Before the actual program was written, a brief sketch was made of a sample part consisting of a line of holes and a bolt hole circle (fig. 4.3). The points were generated and loaded onto a file named KPTS by means of another G. E. program, called NCPTS. This program is exclusively interactive (on the Mark 1 system) andcalls for the typing of certain predefined mnemonic input codes to specify a given type of line or pattern. These are described in detail as to function and usage in reference 48.

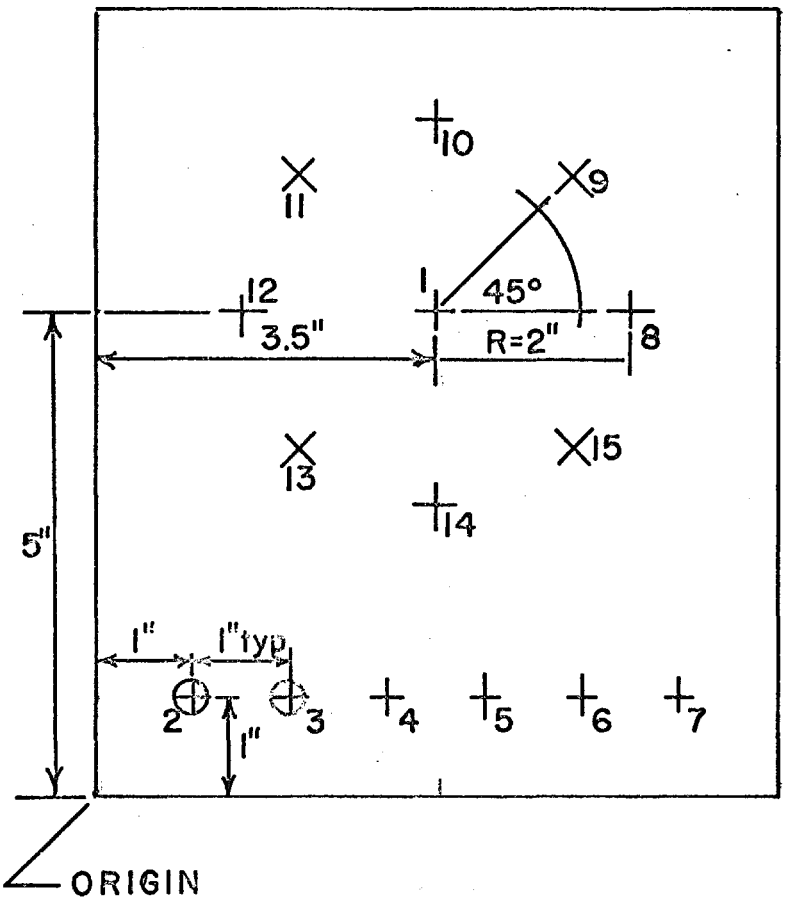


FIGURE 4.3 SKETCH OF PIECE PART AS DESCRIBED BY THE POINTS FILE.

```
10 DPT P16 ABS -3,8,0
20 DPT P17 ABS 16,8,0
30 ORIGIN -3,8,0
50 SEQNO 1,1 POSITN RESFCT TFUNCT 00 PRSTOP GOTO P17
60 TFUNCT 01 ATCHG GOTO P16
70 TFUNCT 00 COOLNT 08 BLOCK
80 DRILL TFUNCT 28 SPINDL GOTO P2
90 ZREAD NOX NOY RAPTO 2.4 FEEDTO 3.0 TFUNCT 12 ZFEED BLOCK
100 DRILL TFUNCT 00 NOCHNG GOTO P3 * P7
110 POSITN TFUNCT 02 ATCHG GOTO P16
120 DRILL TFUNCT 21 SPINDL GOTO P9
130 ZREAD NOX NOY RAPTO 1.3 FEEDTO 4.0 TFUNCT 08 ZFEED BLOCK
140 DRILL TFUNCT 00 NOCHNG GOTO P9 * P15
150 POSITN ENDOFP GOTO P17
160 20*FINI
```

FIGURE 4.4 Part-program for figure 4.3

For this workpiece the two codes used were PAL (for Points Along a Line), which called for the slope, y intercept, x starting, x stopping, and incrementaldistance values, and PAC (for Points Along a Circle), for which the center point, radius, starting angle, stopping angle and incremental angle had to be specified. This generated a total of 15 points which were entered in a file, the listing of which appears in figure 4.5. Each point is designated by a Pn where n is a number to differentiate one point from another.

Having established the points file, the attention is now returned to the part program (fig. 4.4)

Lines 10, 20.

Two more points are defined as P16 and P17. These points refer to the "toolchange" and "load" positions with respect to the part origin respectively.

Line 30

The orientation of the machine origin must be given with respect to the part origin.

Line 50

The SEQNO statement with its attendant control characters activates the program sequence counter to an initial value of one to be subsequently incremented by one for each block. The POSITN, TFUNCT and PRSTOP are then entered to fill registers so that with the motion instruction a full block of information is put out. The GOTO statement fills in the X and Y registers and executes

```
10010 P1        3.500000      5.000000
10020 P2        1.000000      1.000000
10030 P3        2.000000      1.000000
10040 P4        3.000000      1.000000
10050 P5        4.000000      1.000000
10060 P6        5.000000      1.000000
10070 P7        6.000000      1.000000
10080 P8        5.500000      5.000000
10090 P9        4.914214      6.414214
10100 P10       3.500000      7.000000
10110 P11       2.085786      6.414213
10120 P12       1.500000      5.000000
10130 P13       2.085787      3.585786
10140 P14       3.500000      3.000000
10150 P15       4.914214      3.585787
```

FIGURE 4.5 Points file listing for figure 4.3.

the punch or print routine to generate that block of information.

Line 60

An automatic tool change is called for with the TFUNCT designating the tool drum position and P17 the tool change position.

Line 70

A separate block is generated to turn the mist coolant on. There is no motion statement given here so a "BLOCK" statement forces out the information given.

Line 80

The first DRILL cycle is called for, the spindle code is given and the position of the first hole is designated. P2 locates the first hole position.

Line 90

A separate block must now be generated to control the Z axis movements. The NOX and NOY words suppress output of the X and Y registers so that the block length is not violated. ZREAD calls a "9" into the Preparatory function register and the TFUNCT 12 ZFEED specifies the Z axis feedrate. The RAPTO and FEEDTO registers occupy the space normally taken by the X and Y registers.

Line 100

The next sequence of operations has no changes in the machine functions, only the table positions, hence the DRILL, which must be restated, the zero TFUNCT register and the NOCHNG. GOTO P3 * P7 generates a series of five blocks in a linear pattern.

Line 110

A new spindle speed is programmed for the next tool and the operation begins on the Bolt hole circle.

Line 130

The Z axis information for the second series of drilling instruction is given.

Line 140

The bolt hole circle is completed.

Line 150

The table is moved to unload position and the program stop code is punched, turning all machine functions off.

Line 160

A 20 * FINI indicates to the computer that the end of the part program has been reached, and the execution can begin.

It should be stressed that this is merely a sample program which does not necessarily take full advantage of the available facilities of the rather limited vocabulary (as compared to some other languages). It should also be reiterated that this is only a point-to-point system and as such should not be applied to generalized contours, indeed the language is restricted to this type of machine. An output listing is shown in figure 4.6.

An alternative method of generating linear and circular hole patterns without using a points file is the use of the COCIRC and COLINE statements, as stated previously. Certain repeatable sequences may be written into cycle files and called up

```
ENTER Ø FOR LISTING OR 1 FOR TAPE? Ø

ØØ1Ø19ØØØØØØØØØØØØØØ;
ØØ2ØØØØØØØØØØØØØØ186;
ØØ3ØØØØØØØØØØØØØØØØ8;
ØØ45Ø4ØØØØ7ØØØØ2888;
ØØ59Ø24ØØØ3ØØØØ1289;
ØØ65Ø5ØØØØ7ØØØØØØ8Ø;
ØØ75Ø6ØØØØ7ØØØØØØ8Ø;
ØØ85Ø7ØØØØ7ØØØØØØ8Ø;
ØØ95Ø8ØØØØ7ØØØØØØ8Ø;
Ø1Ø5Ø9ØØØØ7ØØØØØØ8Ø;
Ø11ØØØØØØØØØØØØØ286;
Ø125Ø791Ø4Ø158602188;
Ø139Ø13ØØØ4ØØØØØ889;
Ø145Ø791Ø4Ø158600Ø8Ø;
Ø155Ø65ØØØ1ØØØØØØ8Ø;
Ø165Ø5Ø86Ø158600Ø8Ø;
Ø175Ø45ØØØ3ØØØØØØ8Ø;
Ø185Ø5Ø86Ø4414ØØØ8Ø;
Ø195Ø65ØØØ5ØØØØØ8Ø;
Ø2Ø5Ø791Ø4Ø4414ØØ8Ø;
Ø21Ø19ØØØØØØØØØØØ2;
Ø22Ø19ØØØØØØØØØØØ2;

ENTER Ø TO STOP OR 1 TO PUNCH TAPE? Ø
```

FIGURE 4.6 Output listing of part program.

as CYCLE1 for instance when that sequence is desired. These "built-in" facilities are all quite adequately documented in reference 48.

4.4 Summary

Much more could be said about the G. E. time sharing systems. The MARK II system for instance has a rather larger facility for the MTDF, although the principle remains the same. In this system it is possible to write a series of cyclic operations, unique to the machine, and compile them into a Cycle File. The combination of the Cycle File and the MTDF then becomes the post-processor. This postprocessor can handle $3\frac{1}{2}$ axis point-to-point work and $2\frac{1}{2}$ axis contouring when used with the NCPPL language. Reference 49 furnishes comprehensive documentation on the facilities available.

The most significant development here then is that McMaster is now able, through the use of the G. E. time-sharing system, 1 which has the added advantage of the educational after hours discount, to generate control tapes for its MOOG 83-1000 machining center with the aid of a computer. It is a highly recommended system in its present form and with the proper prerequisites both the MARK I and MARK II systems are available.

The generation of the tape takes place of course on the teletype terminal in the user's office. G. E. has anticipated that the requirements for more rapid tape generation will increase

and to that end they now produce and market their own terminal facility, the TERMINET which has the capability of producing NC control tapes at the rate of 110 characters per second. This would mean that the cutting of a 10 foot long control tape would take on the order of one minute. The advertised speed of 110 chararters per second has not yet been achieved but in this instance the delay lies in the telephone-terminal interface, and not in the tape-punching unit.

It should be reiterated that the tape generation facility at McMaster is now operable, according to the terms of reference outlined in this section and the appropriate user manuals. Manual programming the MOOG has in every case shown itself to be an extremely tedious and error-prone task, and does not befit an organization such as CIM whose aim it is to demonstrate up-to-date techniques to NC user firms.

With this in mind I highly recommend this part of the project to further use and development by the people in CIM, wherein some of their aims and goals may be better realized.

5.0 Trends in NC

Much has here been made of the control tape from which the machine control unit, and ultimately the NC machine, receives its data. Judging from some of the work going on in the area of Direct Numerical Control (DNC) it may be that future machines will be plagued less and less by the formatting and coding problems dealt with here.

The concept of DNC is not radically different from the presently accepted norm, the difference being primarily the elimination of the tape interface and its requirements. In practice, however, it provides for a much more flexible output control to the machine when a full-feedback system is used. Under such a system a constant monitoring of the power inputs is maintained such that when these change due to, for instance tool wear or composition changes in the metal, some predefined action will be taken by the machine. This may involve a toolchange, or a change in spindle speed, but it is still all inside one part program.

Some areas of the NC machine tool industry prefer to let the computer do all the work in producing a part. This may involve the control and direction of a number of NC machines and their associated material handling units in an optimized manner. One such installation was displayed at the Leipzig Spring fair held in March of 1970. [64]

Such systems are of course extremely complex and require

a great deal of time and effort in their execution and development. On a lesser scale are mini-computer controlled systems which with the use of a program interrupt facility are capable of handling as many as eight machines performing similar operations simultaneously. [63]

Industry has thus far been reluctant to fully accept and utilize DNC. It may have no choice in the future but to go the DNC route in the face of rising costs and overhead. It may however also arise out of need, as did the whole NC machining concept. Whatever the circumstances, DNC promises to further revolutionize the machine tool industry. What its attendant drawbacks and problems may be have not yet been clearly delineated, but it does open up a whole new area of computer controlled manufacture encompassing a complete manufacturing system.

6.0 Conclusions

A great deal has been said and written about numerical control in manufacture, its justification, its effect on industry, advantages of, disadvantages of, and so on. Considerably less has been said about some of the details, leaving potential users at somewhat of a loss when problems related specifically to the NC aspect of their operation arise.

In this thesis, three areas of problem have been dealt with, not in any complete or absolute manner, as this is infeasible to the point of impossibility. The first of these was the language survey. Given the criterion in section 2.3.5 the author wishes to emphasize that this was undertaken purely as a cataloging effort, i.e. an exercise in bringing together in a rather more condensed form than is presently available, information on the programming languages which are specifically designed for NC manufacture. If conclusions are to be drawn from this, they could be delineated as follows:

1) There appears to be a trend away from the large systems to smaller systems, as more and more point-to-point users accept the idea of computerized tape preparation,

2) There is still a considerable duplication of effort on the part of many in the development of a language in many cases for prestige reasons.

3) APT forms the basis for the majority of part-programming languages.

4) The advent of time-sharing has put the computer assist within the reach of virtually anyone who has a telephone.

5) The considerable programming effort involved in the development of a language is bound to preclude the further development of still more languages.

Their numbers are likely to diminish, as they already have, with the remaining languages adopting many of the desirable features of those that were discarded.

Other conclusions and issues will no doubt arise, but then the purpose here was information and not analytical discussion. The second part, the Tape format Translator, (Sections 3.0 - 3.3) forced out into the open some of the basic issues of conflict in this one area of NC machine controller design and construction. These were:

1) Four of five distinctly different tape formats, although the character coding was the same.

2) A confusing and sometimes conflicting disregard for any established standard in the coding of feedrates and spindle speeds (including preparatory functions).

3) Tape readers in use presently may be either pneumatic (no parity check), electromechanical (slow, noisy) or light-sensitive (very fast, reliable).

If any recommendations for machine control units are to be drawn from this they are:

1) The controller should be programmed to read either

Word address or Universal formats, as these are the most legible.

2) The controller should in its feedrate coding (and so on) use one of the established codes, either EIA Magic Three, EIA, or Bendix Inverse Time.

3) The controller should contain a separate register or memory buffer for each parameter which it is to control.

4) The controller should read character-by-character, stopping only when an end-of-block character is recognized. This setup demands a slightly more complex buffer-reader combination but it is much more efficient in operation and less prone to partiy error. The MOOG pneumatic reader is a case in point. It reads block by block, but if the tape position on the reader block is not within the tolerances, the spindle is bound to make unscheduled and highly undesirable movements in the cutting of a workpiece.

With these design parameters in mind, the difficulties in developing a usable translator may be more clearly seen. If such a project was again desired, it would be best related to two or more specific machines, as the required inputs to the whole range of machine tools are simply too diverse to handle in a project this size. If such a generalized tape translator could be developed, the need for postprocessors would be greatly diminished. It may also be desirable to work on a larger computer system, in a higher level language.

Many system breakdowns were experienced over the duration of the translator project development, calling into question the acclaimed reliability of the DEC PDP computer systems and their attendant peripheral devices, and delaying the completion of the project by weeks. Undoubtedly, many of these breakdowns resulted from an overuse in an academic setting with a corresponding deficiency in scheduled preventive maintenance. Unfortunately other systems such as NOVA and the small, but more expensive IBM and CDC units were not available, although it was felt by some that these slightly more sophisticated processors might be the answer.

The third major section of this thesis, the on-line tape generation facility was accomplished with considerable help from the G. E. Information Services in Toronto. It was done with one specific purpose in mind - to generate control tapes for the MOOG for ordinary programming as quickly and painlessly as is presently feasible. Whereas this section of the project may be academically the least significant, it is perhaps the most beneficial of the three here attempted.

These then have been the areas covered, a survey, a software development, and a direct application of an existing time-sharing computer-assisted NC tape generation facility at McMaster. They range from the debatable to the coldly practical but in every case the basic assumption or criterion has been to take data processing out of the office into the digital computer to

reduce part-program errors and their attendant consequences in the manufacture of workpieces for commercial use.

APPENDIX I

GLOSSARY OF NC TERMINOLOGY

GLOSSARY OF NC TERMINOLOGY

Absolute System - A numerical-control system in which all positional dimensions, both input and feedback, are given with respect to a common datum point. The alternative is the incremental system.

Adaptive Control - A technique, in the process of development, with the aim of automatically adjusting cutting conditions (feeds and speeds) to an optimum to satisfy some criterion, such as minimum machining cost. It involves the in-process mea urement of cutting parameters, together with an on-line computer fed with information about the costs of running the machine tool.

Address - In numerical-control programming, a symbol which indicates the significance of the information immediately following. For example, in the instruction X01575, X is the address signifying the digits 01575 as a co-ordinate on the X axis.

Alphanumeric Coding - A code system in which the characters used are the letters of the alphabet and the numerals 0 to 9.

Amplifier - A device for increasing the voltage, current, or power level of an electrical signal. In a numerical-control system, it is normally the part of the servo-system which amplifies the error signal and provides the power to drive the machine slides or the servo-valve controlling a hydraulic drive.

Analog - One physical variable is the analog of another physical variable if both variables have the same mathematical characteristics.

In numerical control, the term analog applies to a system which utilizes electrical voltage magnitudes or ratios to represent physical axis positions.

Auxiliary Function - A function of a machine other than the con-

trol of the coordinates of a workpiece or cutter. Usually on-off type operations such as starting and stopping a spindle, coolant pump, lubricating pump, feeds, speeds, etc.

Axis - A principal direction along which the relative movements of the tool and workpiece occur. There are usually three linear axes, mutually at right-angles, designated X, Y, and Z.

Binary Code - A code based on binary numbers.

Binary-Coded-Decimal System - A system of number representation in which each decimal digit is represented by a group of binary digits.

Block - A "word" or group of "words" considered as a unit separated from other such units by an "end of block" character (EB). On a punched tape, it consists of one or more characters or rows across the tape that collectively provide sufficient information for an operation.

Block Reader - A tape reader which will read a complete block of information at the same time.

Buffer Storage - A place for storing information in a control system or computer for anticipated utilization. Information from the buffer storage section of a control system can be transferred almost instantaneously to active storage which is that portion of the control system commanding the operation at the particular time. Buffer storage offers the ability of a control system to act immediately on stored information rather than wait for this information to be read into the machine via the tape reader which is relatively slow.

Cartesian-Co-ordinates - A means whereby the position of a point can be defined with reference to a set of axes at right-angles to each other.

Channel - Also known as level or track. A path parallel to the edge of the tape along which information may be stored by

the presence or absence of holes or magnetized areas. The EIA standard 1-inch-wide tape has eight channels.

Character - A number, letter or symbol read on one line across the tape.

Closed Loop System - A system in which the output, or some result of the output is measured and fed back for comparison with the input. In a numerical control system, the output would be the position of the table or head; the input would be the tape information which ordinarily differs from the output. This difference would be measured and result in a machine movement to reduce and eliminate this variance.

Code - A system describing the formation of characters on a tape for representing information in a language that can be understood and handled by the control system.

Command - A signal or group of signals or pulses initiating one step in the execution of a program.

Decode - To translate from coded language into an easily recognizable language. (Opposite of encode)

E. I. A. Standard Code - A standard code for positioning, straight-cut, and contouring control systems proposed by the U. S. Electronic Industries Association in their Standard RS-244. It uses eight-track paper tape, 1 in. wide, and is also known as the 8-B code. It has been recognized by the German V. D. I. and may possibly become a universal standard. It has recently been accepted by the American Standards Association as an American standard for numerical control.

Encode - To translate from an easily recognizable language into a coded language.

End-of-Block Character - A character indicating the end of a block, used to stop the tape reader after a block of information has been read.

Error Signal - The difference between the output and input signals in a servo-system.

Feedback - The transmission of a signal from a late stage to an earlier stage in a system. In a closed-loop numerically controlled system, a signal, depending on the actual position of the machine slide, is fed back and compared with the input signal which specifies the demanded position. These two signals are compared and generate an error signal.

Floating Zero (also described as Free Floating Zero) - A characteristic of a numerical machine tool control permitting the zero reference point on any axis to be established readily at any point on the travel. The control retains no information on the location of any previously established zeros.

Format - Physical arrangement of the input media as possible locations of holes on a punched tape or as magnetized areas on a magnetic tape. Also the general order in which information appears on the input media.

FORTRAN - FORmula TRANslation. A universal computing language or autocode developed by IBM and used for describing numerical processes in such a way that they can be understood both by human being and by computers. For a computer to accept programs in FORTRAN, it has to have a compiler to translate FORTRAN programs into machine (computer) code. In the APT system, mathematical or logical calculations not catered for in the APT program can be programmed in FORTRAN.

Hardware - The component parts used to build a computer or control system.

Incremental System - A control system in which each co-ordinate or positional dimension, both input and feedback, is taken from the last position, rather than from a common datum point, as in the absolute system.

Manuscript - A form used by the part programmer for listing the detailed instructions which can be transcribed directly by the tape preparation device or fed to a computer for further calculation.

Manual Data Input - A means of inserting data manually into the control system. This data is identical to information that could be inserted by means of a tape.

Open Loop System - A control system that has no means for comparing the output with the input for control purposes (i.e., there is no feed-back)

Parity Check - A hole punched in one of the channels whenever the total number of holes representing a character would be even. The net result would be an odd number of holes in every character. This is utilized as a check when reading the tape

Point-to-Point Control System - A system in which controlled motion is required only to reach a given end point with no path control during the transition from one end point to the next. A typical application is in a numerically controlled drill press.

Preparatory Function - A command on the tape changing the mode of operation of the control which is generally noted at the beginning of a block and consists of the letter character "g", plus a two-digit number.

Programmer (Part-Programmer) - A person who prepares the planned sequence of events for the operation of a numerically controlled machine tool. The programmer's principal tool is the manuscript on which the instructions are recorded.

Reproducibility - The ability to be precise or repeatable over a relatively long period of time.

Resolution - The least interval between two adjacent discrete details which can be distinguished one from the other.

Input resolution:

The smallest increment of dimension that can be programmed as input to the system. Helps to eleminate program "round off error"

Feedback resolution:

The samllest increment of dimension that the feedback device can distinguish and reproduce as an electrical output. Analog systems, such as ours, have what is termed infinite resolution. Any movement of the feedback device will result in an electrical output denoting that move.

Digital systems have a finite resolution in that a discrete motion must occur before a pulse is created to represent, electrically, that motion. A typical resolution for digital systems is .00002 in.

Software - Computer programs used as an assist in part programming. The APT program is an example of an automatic programming software package.

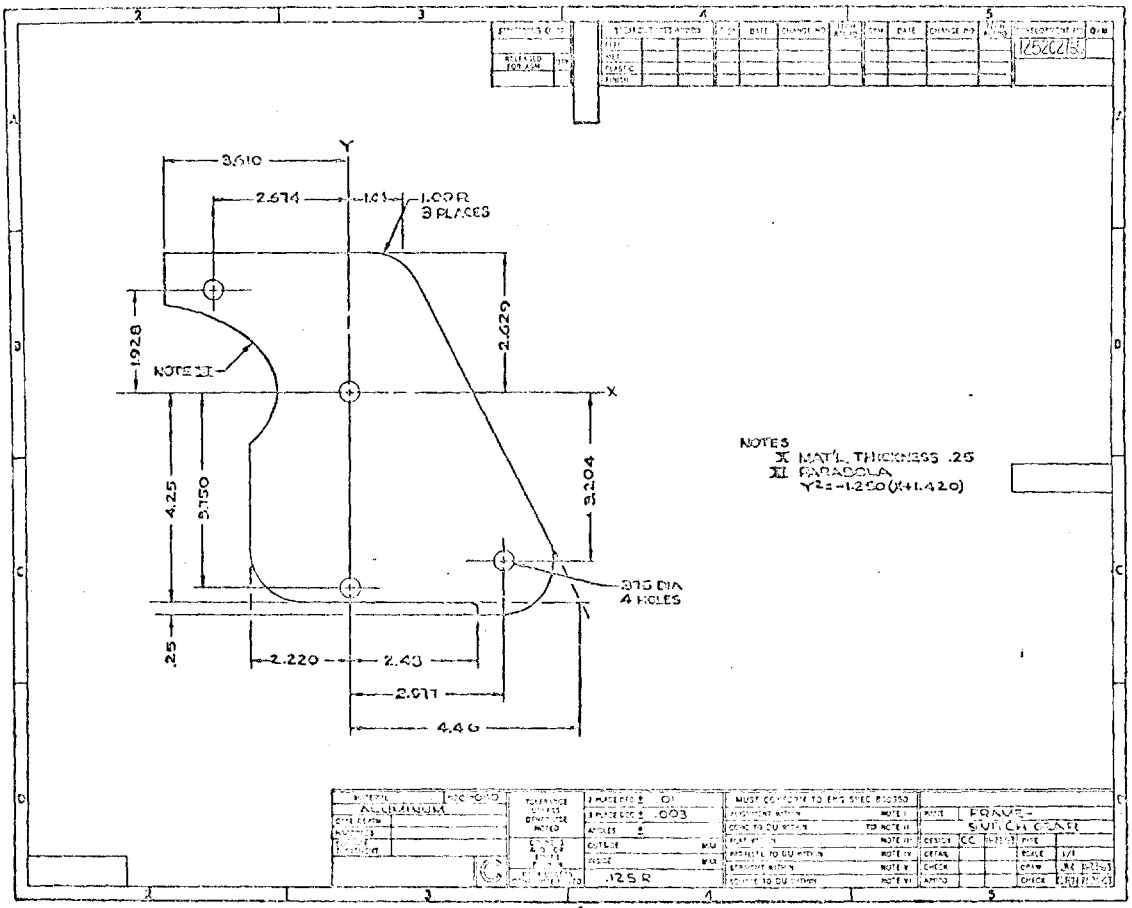
Storage - A device into which information can be introduced, held, and then extracted at a later time.

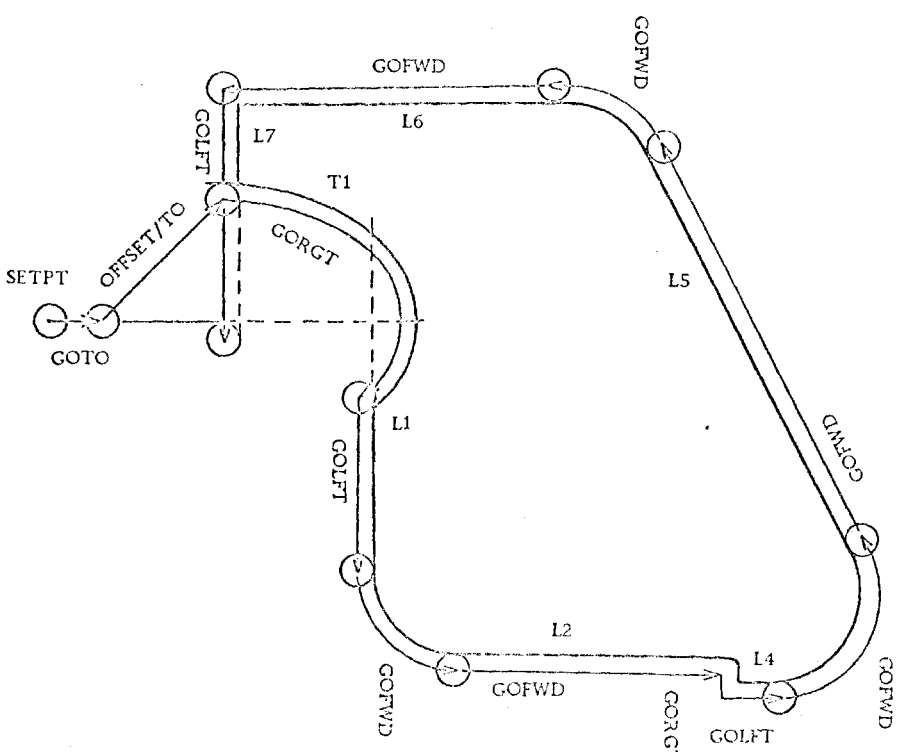
Word - An ordered set of characters which may be used to cause a specific action of a machine tool.

Word Address Format - Addressing each word in a block by one or more characters which identify the meaning of the word.

APPENDIX II

ADAPT TEST PART AND PART PROGRAM AS PREPARED FOR THE 6400/6600 APT SYSTEM, AND TAKEN FROM REFERENCE 36.


3.610
2.674
.161
1.00 R
3 PLACES
1.928
NOTE II
2.629
X
Y
4.25
3.750
3.204
.375 DIA
4 HOLES
.25
2.220
2.43
2.977
4.46
NOTES
I MAT'L THICKNESS .25
II PARABOLA
Y²=-1.250(X+1.420)
ALUMINUM
.003
.125 R
FRAME-
SWITCH COVER


GOFWD
GOFWD
GOLFT
L7
L6
T1
SETPT
OFFSET/TO
GORGT
L5
GOTO
L1
GOLFT
GOFWD
L2
L4
GOFWD
GOFWD
GORGT
GOLFT

```
PARTNO  ADAPT TEST PART                                                                                                  01
NOPOST                                                                                                                   02
CUTTER/.25                                                                                                               03
CLPRNT                                                                                                                   04
CLEARP/XYPLAN,17.0                                                                                                       05
      RESERV/PT,19                                                                                                       5.5
TOOLNO/1,9.7                                                                                                             06
SPINDL/200,CLW,HIGH                                                                                                      07
COOLNT/ON                                                                                                                08
FEDRAT/30.0                                                                                                              09
TOLER/.003                                                                                                               10
FROM/(SETPT=POINT/-5.5,0,1)                                                                                              11
GO TO/-5,0,1                                                                                                             12
GODLTA/0,0,1.3125                                                                                                        13
PARA=MACRO/NAME                                                                                                          14
N=1                                                                                                                      15
X=-4                                                                                                                     16
1) Y=SQRTF(-1.250*(X+1.420))                                                                                             17
PT(N)=POINT/X,Y,0                                                                                                        18
      PT(20-N)=POINT/X,(-Y),0                                                                                            19
       N=N+1                                                                                                             20
       X=X+.300                                                                                                          21
       IF(N-10) 1,2,2                                                                                                    22
2)     PT(10)=POINT/-1.420,0,0                                                                                           23
 NAME=TABCYL/NOZ,SPLINE,PT(1),PT(2),PT(3),PT(4),PT(5),PT(6),PT(7), $                                                     24
        PT(8),PT(9),PT(10),PT(11),PT(12),PT(13),PT(14)                                                                   25
 TERMAC                                                                                                                  26
 CALL/PARA,NAME=T1                                                                                                       27
       INDIRP/PT(2)                                                                                                      28
       PSIS/(PLANE/0,0,1,-0.3125)                                                                                        29
       OFFSET/TO,T1                                                                                                      30
 TLRGT,GORGT/T1,PAST,2,INTOF,(L1=LINE/-2.220,0,-2.220,-4.25)                                                             31
       GOLFT/L1                                                                                                          32
 L2=LINE/-2.22,-4.25,2.48,-4.25                                                                                          33
       GOFWD/(CIRCLE/XLARGE,L1,YLARGE,L2,RADIUS,1)                                                                       34
       GOFWD/L2                                                                                                          35
       GORGT/(LINE/2.48,-4.25,2.48,0)                                                                                    36
GOLFT/(L4=LINE/2.48,-4.50,0.-4.50)                                                                                       37
 L5=LINE/4.46,-4.25,1.04,2.629                                                                                            38
       GOFWD/(CIRCLE/YLARGE,L4,XSMALL,L5,RADIUS,1)                                                                       39
       GOFWD/L5                                                                                                          40
 L6=LINE/1.04,2.629,-3.610,2.629                                                                                         41
       GOFWD/(CIRCLE/XSMALL,L5,YSMALL,L6,RADIUS,1)                                                                       42
       GOFWD/L6                                                                                                          43
 L7=LINE/-3.610,2.629,-3.610,0                                                                                           44
       GOLFT/L7,PAST,(LINE/10,0,-10,0)                                                                                   45
         GODLTA/0,0,1.3125                                                                                               46
 COOLNT/OFF                                                                                                              47
SPINDL/OFF                                                                                                               48
            GOTO/SETPT                                                                                                   49
      FINI                                                                                                               50
```

DS IS/L2 CARD NO.35 TAPE NO. 37

X	Y	Z
2.3550000	-4.3750000	-.3125000

DS IS CARD NO.36 TAPE NO. 39

X	Y	Z
2.3550000	-4.6250000	-.3125000

DS IS/L4 CARD NO.37 TAPE NO. 41

X	Y	Z
2.9703569	-4.6250000	-.3125000

(0) = CIRCLE/ 2.9704 -3.5000 0.0000 1.0000 CARD NO. 39 TAPE NO. 43

DS IS CARD NO. 39 TAPE NO. 44

X	Y	Z
3.0525703	-4.6250000	-.3125000
3.2086909	-4.6025339	-.3125000
3.3601514	-4.5585104	-.3125000
3.5039905	-4.4937903	-.3125000
3.6373956	-4.4096391	-.3125000
3.7577584	-4.3077021	-.3125000
3.8627255	-4.1899725	-.3125000
3.9502445	-4.0587521	-.3125000
4.0186043	-3.9166067	-.3125000
4.0664680	-3.7663156	-.3125000
4.0929000	-3.6108174	-.3125000
4.0973834	-3.4531524	-.3125000
4.0798306	-3.2964034	-.3125000
4.0405847	-3.1436352	-.3125000
3.9777271	-2.9991705	-.3125000

DS IS/L5 CARD NO.40 TAPE NO. 46

X	Y	Z
1.4277664	2.1298295	-.3125000

(0) = CIRCLE/ .4204 1.6290 0.0000 1.0000 CARD NO.42 TAPE NO. 48

DS IS CARD NO.42 TAPE NO. 49

X	Y	Z
1.3911666	2.2034465	-.3125000
1.3088140	2.3240523	-.3125000
1.2115697	2.4330077	-.3125000
1.1010638	2.5284863	-.3125000
.9791485	2.6088876	-.3125000
.8478674	2.6728641	-.3125000
.7094210	2.7193434	-.3125000
.5661300	2.7475462	-.3125000
.4203961	2.7540000	-.3125000

ADAPT TEST PART — CARD NO.01 — TAPE NO. 2

CUTTER/ .250000

CARD NO.03 — TAPE NO. 4

CARD NO.05 — TAPE NO. 6

CLEARP / XYPLAN 17.0000

CARD NO.06 — TAPE NO. 8

TOOLNO / 1.0000 9.7000

CARD NO.07 — TAPE NO. 10

SPINDL / 200.0000 CLW HIGH

CARD NO.08 — TAPE NO. 12

COOLNT / ON

CARD NO.09 — TAPE NO. 14

FEDRAT / 30.0000

OUTTOL/ .003000 .003000 .003000

CARD NO.10 — TAPE NO. 16

INTOL/ 0.000000 0.000000 0.000000

CARD NO.10 — TAPE NO. 17

FROM /SETPT — CARD NO.11 — TAPE NO. 19

X	Y	Z
-5.5000000	0.0000000	1.0000000

DS IS — CARD NO.12 — TAPE NO. 21

X	Y	Z
-5.0000000	0.0000000	1.0000000

DS IS — CARD NO.13 — TAPE NO. 23

X	Y	Z
-5.0000000	0.0000000	2.3125000

CALL CARD NO.27 TAPE NO. 24

DS IS/TI CARD NO.30 TAPE NO. 28

X	Y	Z
-3.7684456	1.5773473	-.3125000

DS IS/TI CARD NO.31 TAPE NO. 30

X	Y	Z
-3.5310560	1.4898492	-.3125000
-3.1483342	1.3335268	-.3125000
-2.8181646	1.1831602	-.3125000
-2.5456051	1.0441265	-.3125000
-2.3893388	.9560980	-.3125000
-2.1758883	.8220245	-.3125000
-2.0950000	.7656319	-.3125000
-1.9618426	.6633956	-.3125000
-1.8423603	.5575502	-.3125000
-1.7380688	.4463429	-.3125000
-1.6601308	.3431125	-.3125000
-1.6073006	.2525309	-.3125000
-1.5690024	.1572564	-.3125000
-1.5485755	.0592078	-.3125000
-1.5478048	-.0518317	-.3125000
-1.5701101	-.1608197	-.3125000
-1.6136715	-.2650383	-.3125000
-1.6799376	-.3717202	-.3125000
-1.7746281	-.4880576	-.3125000
-1.9014762	-.6121121	-.3125000
-2.0601796	-.7400516	-.3125000
-2.2631016	-.8790585	-.3125000
-2.3450000	-.9299363	-.3125000

DS IS/LI CARD NO.32 TAPE NO. 32

X	Y	Z
-2.3450000	-3.2500000	-.3125000

(0) = CIRCLE/ -1.2200 -3.2500 0.0000 1.0000 CARD NO.34 TAPE NO. 34

DS IS CARD NO.34 TAPE NO. 35

X	Y	Z
-2.3450000	-3.3322134	-.3125000
-2.3234261	-3.4841684	-.3125000
-2.2814244	-3.6317882	-.3125000
-2.2197724	-3.7723399	-.3125000
-2.1396116	-3.9032215	-.3125000
-2.0424260	-4.0220099	-.3125000
-1.9300147	-4.1265061	-.3125000
-1.8044588	-4.2147754	-.3125000
-1.6680829	-4.2851838	-.3125000
-1.5234115	-4.3364277	-.3125000
-1.3731231	-4.3675585	-.3125000
-1.2200000	-4.3750000	-.3125000

DS IS/L6			CARD NO.43	TAPE NO. 51
	X -3.7350000	Y 2.7540000	Z -.3125000	
DS IS/L7			CARD NO.45	TAPE NO. 53
	X -3.7350000	Y -.1250000	Z -.3125000	
DS IS			CARD NO.46	TAPE NO. 55
	X -3.7350000	Y -.1250000	Z 1.0000000	
			CARD NO.47	TAPE NO. 57
COOLNT / OFF				
			CARD NO.48	TAPE NO. 59
SPINDL / OFF				
DS IS/SETPT			CARD NO.49	TAPE NO. 61
	X -5.5000000	Y 0.0000000	Z 1.0000000	
FINI END OF PART PROGRAM			CARD NO.50	TAPE NO. 63

APPENDIX III

IMPLEMENTATION OF 6400/6600 APT ON THE McMASTER COMPUTING AND DATA CENTER.

IMPLEMENTATION OF 6400/6600 APT ON THE MCMASTER UNIVERSITY CDC6400

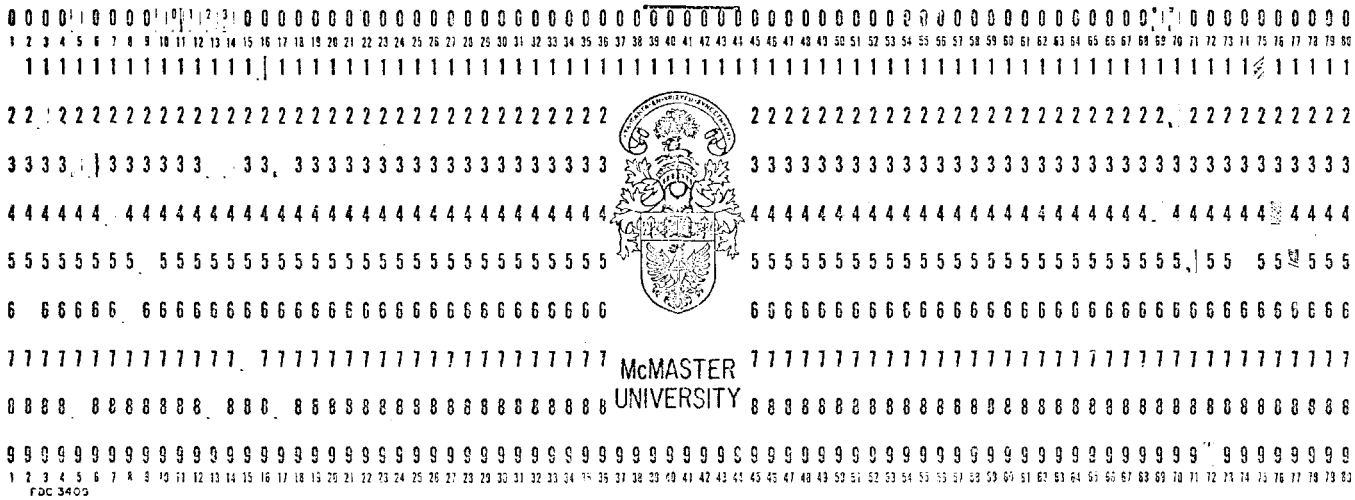
McMaster computing center now has its own in-file version of APT III. The original system tapes were found unworkable in the as-received condition for the following reasons:

1. The APT III version put out by CDC is structured so as to conserve memory space, that is, not all of the program is read into the core at any one time. Instead, a short subroutine at the beginning of the program, written in COMPASS calls up segments of the main program as they are required. These segments are called "Overlays" and they may be addressed either by program name or by file number and classification (See Publication #3.16, October 21, 1970, Data Processing and Computing Center for further details on overlays).
2. It was assumed by the software people of CDC that this system would be loaded into the peripheral processor library and would hence have all program names fully accessible to the central processor. Due to the core requirements of this type of installation and the relative infrequency of its use it was decided not to load APT onto the system. This alternative usage, however, kept the program names out of the reach of the central processor. The alteration to place APT on a permanent tape file consisted of calling up the overlay segments by file name and number rather than by program name.

Hence APT may now be simply called up with the following set of control cards:

```
8              END OF FILE
*          6-8-9 END OF RECORD

     PART PROGRAM DECK
PARTNO PARTNAME
*          6-8-9 END OF RECORD
 UNLOAD,LIBTAP.
 LIBTAP.
REQUEST,LIBTAP,MT. READ RACK 3403 (APT 2.2)
JOB ,CM65000,TP1.                                          USER NAME
```



A long program will require a slight modification of these control cards.

E. Klaassen
13 August 1971

APPENDIX IV

LISTING OF THE MOOG TRANSLATOR PROGRAM

```
                *10
0010    0000    INDEX,      0
                *100
0100    0000    TYPE,       0
0101    6041                TSF
0102    5101                JMP .-1
0103    6046                TLS
0104    7300                CLA CLL
0105    5500                JMP I TYPE
0106    0000    EOB,        0
0107    7300                CLA CLL
0110    1125                TAD END
0111    4100                JMS TYPE
0112    5506                JMP I EOB
0113    0000    COUNT,      0
0114    2100    XYMEM,      2100
0115    0000    XYCTR,      0
0116    2000    NEWMEM,     2000
0117    0000    MEMCTR,     0
0120    2200    MCOOL,      2200
0121    0000    COOLCT,     0
0122    2140    ZMEM,       2140
0123    0000    ZCTR,       0
0124    7600    CR,         7600
0125    0200    END,        200
0126    7777    M1,         7777
0127    7776    M2,         7776
0130    7775    M3,         7775
0131    7773    M5,         7773
0132    7755    M19,      7755
0133    0000    STOP1,      0
0134    0000    STOP2,      0
0135    0000    STOP3,      0
0136    0000    STOP50,     0
0137    0000    STOP5,      0
0140    0277    K277,       277
0141    0243    K243,       243
0142    0026    SIX,        26
0143    0010    EIGHT,      10
0144    0031    NINE,       31
0145    0177    MASK,       177
0146    1777    NEWMM,    1777
0147    1271    CHEX,     CHEKX
0150    1304    CHKY,     CHEKY
0151    0040    ZERO,     40
0152    0502    CLER,       CLEAR
0153    0533    ERROR,      ERR
0154    1200    RESET,      RESTT
0155    0600    MSUBR,      MSUB
```

```
0156  1000  SORTG,     SRTGEE
0157  1224  SORTX,     SRTEX
0160  1054  SORTY,     SRTWYE
0161  1071  SORTR,     SRTARE
0162  1126  SORTF,     SRTEFF
0163  1206  WRITZ,     WRTZED
0164  1241  PACK2,     PCK2
0165  0400  LINK1,     CHEKS
0166  2117  ESS,       2117
0167  0323  CHEKGEE,   CHEKG
0170  1321  LEDR,    LEADR
0171  1255  ERROR2,  ERR2
0172  0225  AGAIN,   AGEN
0173  7756  M18,     7756
0174  0474  RETURN,  RETRN
            *200
0200  7300  START,      CLA CLL
0201  6046              TLS
0202  1351              TAD CONST
0203  3113              DCA COUNT
0204  1146            TAD NEWMM
0205  3010            DCA INDEX
0206  4552              JMS I CLER
0207  4570            JMS I LEDR
0210  7300            CLA CLL
0211  1352              TAD LASTX
0212  3115              DCA XYCTR
0213  1125              TAD END
0214  3515              DCA I XYCTR
0215  1353              TAD LASTM
0216  3121              DCA COOLCT
0217  1125              TAD END
0220  3521              DCA I COOLCT
0221  1354              TAD LASTZ
0222  3123              DCA ZCTR
0223  1125              TAD END
0224  3523              DCA I ZCTR
0225  1116  AGEN,    TAD NEWMEM
0226  3117            DCA MEMCTR
0227  1114            TAD XYNEM
0230  3115            DCA XYCTR
0231  1120            TAD MCOOL
0232  3121            DCA COOLCT
0233  1122            TAD ZMEM
0234  3123            DCA ZCTR
0235  6031  LOOP1,     KSF
0236  5235              JMP .-1
0237  6036              KRB
0240  3517              DCA I MEMCTR
0241  1517              TAD I MEMCTR
0242  1124              TAD CR
0243  7450              SNA
```

```
0244  5253          JMP LPND
0245  7001         IAC
0246  7450           SNA
0247  4570         JMS I LEDR
0250  2117           ISZ MEMCTR
0251  7300           CLA CLL
0252  5235           JMP LOOP1
0253  4554  LPND,    JMS I RESET
0254  1517  MOREN,   TAD I MEMCTR
0255  1355           TAD NN
0256  7650           SNA CLA
0257  5264           JMP OUTN
0260  2117           ISZ MEMCTR
0261  2136           ISZ STOP50
0262  5254           JMP MOREN
0263  5574           JMP I RETURN
0264  2117  OUTN,    ISZ MEMCTR
0265  1130           TAD M3
0266  3135           DCA STOP3
0267  1517  SEQN,      TAD I MEMCTR
0270  3515             DCA I XYCTR
0271  2117             ISZ MEMCTR
0272  2115             ISZ XYCTR
0273  2135             ISZ STOP3
0274  5267             JMP SEQN
0275  4554           JMS I RESET
0276  1517  LOOP2,   TAD I MEMCTR
0277  1356             TAD MM
0300  7650             SNA CLA
0301  4555             JMS I MSUBR
0302  2117             ISZ MEMCTR
0303  2136             ISZ STOP50
0304  5276             JMP LOOP2
0305  1166             TAD ESS
0306  3115             DCA XYCTR
0307  1515  XS,         TAD I XYCTR
0310  1124             TAD CR
0311  7650             SNA CLA
0312  5317             JMP T1
0313  1151           TAD ZERO
0314  3515             DCA I XYCTR
0315  2115             ISZ XYCTR
0316  5307             JMP XS
0317  1360  T1,         TAD T80
0320  3115             DCA XYCTR
0321  1143             TAD EIGHT
0322  3515             DCA I XYCTR
0323  7300  CHEKG,      CLA CLL
0324  4554             JMS I RESET
0325  1517  LOOPG,     TAD I MEMCTR
0326  1357             TAD GG
0327  7650             SNA CLA
```

```
0330  5335            JMP  GOUT
0331  2117            ISZ  MEMCTR
0332  2136            ISZ  STOP50
0333  5325            JMP  LOOPG
0334  7410            SKP
0335  4556  GOUT,     JMS  I SORTG
0336  4547          JMS  I CHEX
0337  1136          TAD  STOP50
0340  7450          SNA
0341  5343          JMP  NEXT
0342  4557          JMS  I SORTX
0343  4550  NEXT,    JMS  I CHKY
0344  1136          TAD  STOP50
0345  7450          SNA
0346  5350  JMP LK
0347  4560          JMS  I SORTY
0350  5565  LK,      JMP  I LINK1
0351  7500  CONST,   7500
0352  2123  LASTX,     2123
0353  2223  LASTM,2223
0354  2163  LASTZ,     2163
0355  7673  NN,        7673
0356  7654  MM,        7654
0357  7631  GG,        7631
0360  2121  T80,       2121
            *400
0400  7300  CHEKS,     CLA CLL
0401  4554            JMS  I RESET
0402  1517  LOOPS,     TAD  I MEMCTR
0403  1356            TAD  SS
0404  7650            SNA  CLA
0405  5212            JMP  SOUT
0406  2117            ISZ  MEMCTR
0407  2136            ISZ  STOP50
0410  5202            JMP  LOOPS
0411  7410            SKP
0412  4310  SOUT,     JMS  SORTS
0413  7300  XYTP,     CLA  CLL
0414  6046            TLS
0415  1114            TAD  XYMEM
0416  3115            DCA  XYCTR
0417  1515  TYPEX,   TAD  I XYCTR
0420  1124            TAD  CR
0421  7450            SNA
0422  5227            JMP  ENDX
0423  0145            AND  MASK
0424  4100            JMS  TYPE
0425  2115            ISZ  XYCTR
0426  5217            JMP  TYPEX
0427  4106  ENDX,     JMS  EOB
0430  7300  CHEKR,     CLA  CLL
0431  4554            JMS  I RESET
```

```
0432  1517  LOOPR,    TAD I MEMCTR
0433  1357            TAD RR
0434  7650            SNA CLA
0435  5242            JMP RSM
0436  2117            ISZ MEMCTR
0437  2136            ISZ STOP50
0440  5232            JMP LOOPR
0441  5274            JMP RETRN
0442  1122  RSM,       TAD ZMEM
0443  3123            DCA ZCTR
0444  1114            TAD XYMEM
0445  3115            DCA XYCTR
0446  1130            TAD M3
0447  3135            DCA STOP3
0450  1515  Z1,        TAD I XYCTR
0451  3523            DCA I ZCTR
0452  2123            ISZ ZCTR
0453  2115            ISZ XYCTR
0454  2135            ISZ STOP3
0455  5250            JMP Z1
0456  1144            TAD NINE
0457  3523            DCA I ZCTR
0460  4561            JMS I SORTR
0461  7300  CHEKF,     CLA CLL
0462  4554            JMS I RESET
0463  1517  LOOPF,     TAD I MEMCTR
0464  1360            TAD FF
0465  7650            SNA CLA
0466  5273            JMP FOUT
0467  2117            ISZ MEMCTR
0470  2136            ISZ STOP50
0471  5263            JMP LOOPF
0472  7410            SKP
0473  4562  FOUT,      JMS I SORTF
0474  1361  RETRN,   TAD CONST1
0475  3113            DCA COUNT
0476  1146            TAD NEWMM
0477  3010            DCA INDEX
0500  4302            JMS CLEAR
0501  5572            JMP I AGAIN
                      PAUSE

0502  0000  CLEAR,     0
0503  1151            TAD ZERO
0504  3410            DCA I INDEX
0505  2113            ISZ COUNT
0506  5303            JMP .-3
0507  5702            JMP I CLEAR
0510  0000  SORTS,     0
0511  7300            CLA CLL
0512  2117            ISZ MEMCTR
```

```
0513  1127            TAD M2
0514  3134            DCA STOP2
0515  1166            TAD ESS
0516  3115            DCA XYCTR
0517  1517  S2,       TAD I MEMCTR
0520  3515            DCA I XYCTR
0521  2117            ISZ MEMCTR
0522  2115            ISZ XYCTR
0523  2134            ISZ STOP2
0524  5317            JMP S2
0525  1143            TAD EIGHT
0526  3515            DCA I XYCTR
0527  2115            ISZ XYCTR
0530  1143            TAD EIGHT
0531  3515            DCA I XYCTR
0532  5710            JMP I SORTS
0533  0000  ERR,       0
0534  7300            CLA CLL
0535  6046            TLS
0536  4347          JMS CRLF
0537  1140            TAD K277
0540  4100            JMS TYPE
0541  1141            TAD K243
0542  4100            JMS TYPE
0543  1362            TAD K1
0544  4100            JMS TYPE
0545  7402            HLT
0546  5733            JMP I ERR
0547  0000  CRLF,     0
0550  7300          CLA CLL
0551  1364          TAD K215
0552  4100          JMS TYPE
0553  1363          TAD K212
0554  4100          JMS TYPE
0555  5747          JMP I CRLF
0556  7716  SS,         7716
0557  7667  RR,         7667
0560  7612  FF,         7612
0561  7701  CONST1,     7701
0562  0261  K1,         261
0563  0212  K212,     212
0564  0215  K215,     215
            *600
0600  0000  MSUB,     0
0601  2117          ISZ MEMCTR
0602  1117          TAD MEMCTR
0603  3373          DCA TEMP
0604  1120          TAD MCOOL
0605  3010          DCA INDEX
0606  1173          TAD M18
0607  3113          DCA COUNT
0610  4552          JMS I CLER
```

```
0611  1120          TAD MCOOL
0612  3121          DCA COOLCT
0613  1114          TAD XYMEM
0614  3115          DCA XYCTR
0615  1130          TAD M3
0616  3135          DCA STOP3
0617  1515  MSEQ,   TAD I XYCTR
0620  3521          DCA I COOLCT
0621  2115          ISZ XYCTR
0622  2121          ISZ COOLCT
0623  2135          ISZ STOP3
0624  5217          JMP MSEQ
0625  4547          JMS I CHEX
0626  1136          TAD STOP50
0627  7650          SNA CLA
0630  5243          JMP MWYE
0631  1375          TAD MX
0632  3121          DCA COOLCT
0633  1131          TAD M5
0634  3137          DCA STOP5
0635  2117  REPTX,  ISZ MEMCTR
0636  1517          TAD I MEMCTR
0637  3521          DCA I COOLCT
0640  2121          ISZ COOLCT
0641  2137          ISZ STOP5
0642  5235          JMP REPTX
0643  4550  MWYE,   JMS I CHKY
0644  1136          TAD STOP50
0645  7650          SNA CLA
0646  5261          JMP MF
0647  1376          TAD MY
0650  3121          DCA COOLCT
0651  1131          TAD M5
0652  3137          DCA STOP5
0653  2117  REPTY,  ISZ MEMCTR
0654  1517          TAD I MEMCTR
0655  3521          DCA I COOLCT
0656  2121          ISZ COOLCT
0657  2137          ISZ STOP5
0660  5253          JMP REPTY
0661  1373  MF,     TAD TEMP
0662  3117          DCA MEMCTR
0663  2117          ISZ MEMCTR
0664  1517          TAD I MEMCTR
0665  1371          TAD M26
0666  7650          SNA CLA
0667  4315          JMS TLCH
0670  1374          TAD MFUN
0671  3121          DCA COOLCT
0672  1373          TAD TEMP
0673  3117          DCA MEMCTR
0674  1517          TAD I MEMCTR
```

```
0675  3521          DCA I COOLCT
0676  2117          ISZ MEMCTR
0677  2121          ISZ COOLCT
0700  1517          TAD I MEMCTR
0701  3521          DCA I COOLCT
0702  4353          JMS MTYPE
0703  4106          JMS EOB
0704  1374          TAD MFUN
0705  3121          DCA COOLCT
0706  2121          ISZ COOLCT
0707  1127          TAD M2
0710  1521          TAD I COOLCT
0711  7450          SNA
0712  5572          JMP I AGAIN
0713  5567          JMP I CHEKGEE
0714  5600          JMP I MSUB
0715  0000  TLCH,    0
0716  7300          CLA CLL
0717  4554          JMS I RESET
0720  1517  LOOP6,   TAD I MEMCTR
0721  1372          TAD TT
0722  7650          SNA CLA
0723  5330          JMP CONT1
0724  2117          ISZ MEMCTR
0725  2136          ISZ STOP50
0726  5320          JMP LOOP6
0727  4571          JMS I ERROR2
0730  1377  CONT1,   TAD TTM
0731  3121          DCA COOLCT
0732  2117          ISZ MEMCTR
0733  1517          TAD I MEMCTR
0734  3521          DCA I COOLCT
0735  2117          ISZ MEMCTR
0736  2121          ISZ COOLCT
0737  1517          TAD I MEMCTR
0740  3521          DCA I COOLCT
0741  2121          ISZ COOLCT
0742  1143          TAD EIGHT
0743  3521          DCA I COOLCT
0744  2121          ISZ COOLCT
0745  1142          TAD SIX
0746  3521          DCA I COOLCT
0747  4353          JMS MTYPE
0750  4106          JMS EOB
0751  5572          JMP I AGAIN
0752  5715          JMP I TLCH
0753  0000  MTYPE,     0
0754  7300            CLA CLL
0755  6046            TLS
0756  1120            TAD MCOOL
0757  3121            DCA COOLCT
0760  1521  TYPEM,    TAD I COOLCT
```

```
0761   1124              TAD CR
0762   7450              SNA
0763   5370              JMP ENDM
0764   0145              AND MASK
0765   4100              JMS TYPE
0766   2121              ISZ COOLCT
0767   5360              JMP TYPEM
0770   5753   ENDM,      JMP I MTYPE
0771   7752   M26,       7752
0772   7735   TT,        7735
0773   0000   TEMP,      0
0774   2221   MFUN,      2221
0775   2204   MX,        2204
0776   2211   MY,        2211
0777   2217   TTM,       2217
                         PAUSE

               *1000
1000   0000   SRTGEE,    0              / PREP FUNCTION CODING
1001   1352              TAD GEE
1002   3115              DCA XYCTR
1003   2117              ISZ MEMCTR
1004   4564              JMS I PACK2
1005   7450              SNA
1006   5253              JMP ENDG
1007   0351              AND MASK3
1010   1131              TAD M5
1011   7450              SNA
1012   5230              JMP BORE
1013   7001              IAC
1014   7450              SNA
1015   5234              JMP TAP1
1016   7001              IAC
1017   7450              SNA
1020   5240              JMP TAP2
1021   7001              IAC
1022   7450              SNA
1023   5244              JMP PECK
1024   7001              IAC
1025   7450              SNA
1026   5250              JMP DRILL
1027   5253              JMP ENDG
1030   7200   BORE,      CLA
1031   1353              TAD BORER
1032   3515              DCA I XYCTR
1033   5253              JMP ENDG
1034   7200   TAP1,      CLA
1035   1354              TAD TAP1R
1036   3515              DCA I XYCTR
1037   5253              JMP ENDG
1040   7200   TAP2,      CLA
1041   1355              TAD TAP2R
```

```
1042  3515            DCA I XYCTR
1043  5253            JMP ENDG
1044  7200  PECK,     CLA
1045  1356            TAD PECKR
1046  3515            DCA I XYCTR
1047  5253            JMP ENDG
1050  7200  DRILL,    CLA
1051  1357            TAD DRILLR
1052  3515            DCA I XYCTR
1053  5600  ENDG,     JMP I SRTGEE
1054  0000  SRTWYE,   0           / Y MOVEMENT INSTRUCTIONS
1055  7300          CLA CLL
1056  1360            TAD WYE
1057  3115            DCA XYCTR
1060  1131            TAD M5
1061  3137            DCA STOP5
1062  2117  RESTOY,   ISZ MEMCTR
1063  1517            TAD I MEMCTR
1064  3515            DCA I XYCTR
1065  2115            ISZ XYCTR
1066  2137            ISZ STOP5
1067  5262            JMP RESTOY
1070  5654            JMP I SRTWYE
1071  0000  SRTARE,   0           /FEEDPOINT MOVEMENT INSTRUCT
1072  7300            CLA CLL
1073  1127            TAD M2
1074  3134            DCA STOP2
1075  1131  OVER,     TAD M5
1076  3137            DCA STOP5
1077  2117  RESTOR,   ISZ MEMCTR
1100  2123            ISZ ZCTR
1101  1517            TAD I MEMCTR
1102  3523            DCA I ZCTR
1103  2137            ISZ STOP5
1104  5277            JMP RESTOR
1105  2134            ISZ STOP2
1106  7410            SKP
1107  5312            JMP SRT
1110  4313            JMS CHEKZ
1111  5275            JMP OVER
1112  5671  SRT,      JMP I SRTARE
1113  0000  CHEKZ,    0           /FINAL DEPTH INSTRUCTIONS
1114  7300            CLA CLL
1115  4554            JMS I RESET
1116  1517  LOOPZ,    TAD I MEMCTR
1117  1361            TAD ZZ
1120  7650            SNA CLA
1121  5325            JMP ZOUT
1122  2117            ISZ MEMCTR
1123  2136            ISZ STOP50
1124  5316            JMP LOOPZ
1125  5713  ZOUT,     JMP I CHEKZ
```

```
1126  0000  SRTEFF,   0            /Z FEEDRATE INSTRUCTIONS
1127  1362            TAD ZF       \
1130  3123            DCA ZCTR
1131  1127            TAD M2
1132  3134            DCA STOP2
1133  2117  RESTOZ,   ISZ MEMCTR
1134  1517            TAD I MEMCTR
1135  3523            DCA I ZCTR
1136  2123            ISZ ZCTR
1137  2134            ISZ STOP2
1140  5333            JMP RESTOZ
1141  1143            TAD EIGHT
1142  3523            DCA I ZCTR
1143  2123            ISZ ZCTR
1144  1144            TAD NINE
1145  3523            DCA I ZCTR
1146  4563            JMS I WRITZ
1147  4106            JMS EOB
1150  5726            JMP I SRTEFF
1151  0007  MASK3,    7
1152  2103  GEE,      2103
1153  0007  BORER,    7
1154  0001  TAP1R,    1
1155  0010  TAP2R,    10
1156  0004  PECKR,    4
1157  0025  DRILLR,   25
1160  2111  WYE,      2111
1161  7727  ZZ,       7727
1162  2157  ZF,       2157
            *1200
1200  0000  RESTT,    0
1201  1116            TAD NEWMEM
1202  3117            DCA MEMCTR
1203  1254            TAD M50
1204  3136            DCA STOP50
1205  5600            JMP I RESTT
1206  0000  WRTZED,   0
1207  7300            CLA CLL
1210  6046            TLS
1211  1122            TAD ZMEM
1212  3123            DCA ZCTR
1213  1523  TYPEZ,    TAD I ZCTR
1214  1124            TAD CR
1215  7450            SNA
1216  5223            JMP ENDZ
1217  0145            AND MASK
1220  4100            JMS TYPE
1221  2123            ISZ ZCTR
1222  5213            JMP TYPEZ
1223  5606  ENDZ,     JMP I WRTZED
1224  0000  SRTEX,  0        / X MOVEMENT INSTRUCTIONS
1225  7300          CLA CLL
```

```
1226  1253            TAD XSTRT
1227  3115            DCA XYCTR
1230  1131            TAD M5
1231  3137            DCA STOP5
1232  2117  RESTOX,   ISZ MEMCTR
1233  1517            TAD I MEMCTR
1234  3515            DCA I XYCTR
1235  2115            ISZ XYCTR
1236  2137            ISZ STOP5
1237  5232            JMP RESTOX
1240  5624            JMP I SRTEX
1241  0000  PCK2,     0
1242  1127            TAD M2
1243  3134            DCA STOP2
1244  7006  PACK,     RTL
1245  7004            RAL
1246  1517            TAD I MEMCTR
1247  2117            ISZ MEMCTR
1250  2134            ISZ STOP2
1251  5244            JMP PACK
1252  5641            JMP I PCK2
1253  2104  XSTRT,    2104
1254  7715  M50,      7715
1255  0000  ERR2,     0
1256  7300            CLA CLL
1257  6046            TLS
1260  1140            TAD K277
1261  4100            JMS TYPE
1262  1141            TAD K243
1263  4100            JMS TYPE
1264  1270            TAD K262
1265  4100            JMS TYPE
1266  7402            HLT
1267  5655            JMP I ERR2
1270  0262  K262,     262
1271  0000  CHEKX,    0
1272  7300            CLA CLL
1273  4200            JMS RESTT
1274  1517  LOOPX,    TAD I MEMCTR
1275  1317            TAD XX
1276  7650            SNA CLA
1277  5303            JMP XOUT
1300  2117            ISZ MEMCTR
1301  2136            ISZ STOP50
1302  5274            JMP LOOPX
1303  5671  XOUT,     JMP I CHEKX
1304  0000  CHEKY,    0
1305  7300            CLA CLL
1306  4200            JMS RESTT
1307  1517  LOOPY,    TAD I MEMCTR
1310  1320            TAD YY
1311  7650            SNA CLA
```

```
1312  5316          JMP YOUT
1313  2117          ISZ MEMCTR
1314  2136          ISZ STOP50
1315  5307          JMP LOOPY
1316  5704  YOUT,   JMP I CHEKY
1317  7711  XX,     7711
1320  7710  YY,     7710
1321  0000  LEADR,  0
1322  6046          TLS
1323  1131          TAD M5
1324  3137          DCA STOP5
1325  1132  LO,     TAD M19
1326  3113          DCA COUNT
1327  1151  LI,     TAD ZERO
1330  4100          JMS TYPE
1331  2113          ISZ COUNT
1332  5327          JMP LI
1333  4106          JMS EOB
1334  2137          ISZ STOP5
1335  5325          JMP LO
1336  1517          TAD I MEMCTR
1337  1124          TAD CR
1340  7001          IAC
1341  7450          SNA
1342  7402          HLT
1343  5721          JMP I LEADR
AGAIN   0172
AGEN    0225
BORE    1030
BORER   1153
CHEKF   0461
CHEKG   0323
CHEKGE  0167
CHEKR   0430
CHEKS   0400
CHEKX   1271
CHEKY   1304
CHEKZ   1113
CHEX    0147
CHKY    0150
CLEAR   0502
CLER    0152
CONST   0351
CONST1  0561
CONT1   0730
COOLCT  0121
COUNT   0113
CR      0124
CRLF    0547
DRILL   1050
DRILLR  1157
EIGHT   0143
```

```
END       0125
ENDG      1053
ENDM      0770
ENDX      0427
ENDZ      1223
EOB       0106
ERR       0533
ERROR     0153
ERROR2    0171
ERR2      1255
ESS       0166
FF        0560
FOUT      0473
GEE       1152
GG        0357
GOUT      0335
INDEX     0010
K1        0562
K212      0563
K215      0564
K243      0141
K262      1270
K277      0140
LASTM     0353
LASTX     0352
LASTZ     0354
LEADR     1321
LEDR      0170
LI        1327
LINK1     0165
LK        0350
LO        1325
LOOPF     0463
LOOPG     0325
LOOPR     0432
LOOPS     0402
LOOPX     1274
LOOPY     1307
LOOPZ     1116
LOOP1     0235
LOOP2     0276
LOOP6     0720
LPND      0253
MASK      0145
MASK3     1151
MCOOL     0120
MEMCTR    0117
MF        0661
MFUN      0774
MM        0356
MOREN     0254
MSEQ      0617
```

```
MSUB     0600
MSUBR    0155
MTYPE    0753
MWYE     0643
MX       0775
MY       0776
M1       0126
M18      0173
M19      0132
M2       0127
M26      0771
M3       0130
M5       0131
M50      1254
NEWMEM   0116
NEWMM    0146
NEXT     0343
NINE     0144
NN       0355
OUTN     0264
OVER     1075
PACK     1244
PACK2    0164
PCK2     1241
PECK     1044
PECKR    1156
REPTX    0635
REPTY    0653
RESET    0154
RESTOR   1077
RESTOX   1232
RESTOY   1062
RESTOZ   1133
RESTT    1200
RETRN    0474
RETURN   0174
RR       0557
RSM      0442
SEQN     0267
STX      0142
SORTF    0162
SORTG    0156
SORTR    0161
SORTS    0510
SORTX    0157
SORTY    0160
SOUT     0412
SRT      1112
SRTARE   1071
SRTEFF   1126
SRTEX    1224
SRTGEE   1000
```

```
SRTWYE   1054
SS       0556
START    0200
STOP1    0133
STOP2    0134
STOP3    0135
STOP5    0137
STOP50   0136
S2       0517
TAP1     1034
TAP1R    1154
TAP2     1040
TAP2R    1155
TEMP     0773
TLCH     0715
TT       0772
TTM      0777
TYPE     0100
TYPEM    0760
TYPEX    0417
TYPEZ    1213
T1       0317
T80      0360
WRITZ    0163
WRTZED   1206
WYE      1160
XOUT     1303
XS       0307
XSTRT    1253
XX       1317
XYCTR    0115
XYMEM    0114
XYTP     0413
YOUT     1316
YY       1320
ZCTR     0123
ZERO     0151
ZF       1162
ZMEM     0122
ZOUT     1125
ZZ       1161
Z1       0450
```

APPENDIX V

REFERENCES

REFERENCES

1. Childs, J.J., "Principles of Numerical Control" (2nd Ed.) Industrial Press Inc., New York, N.Y: 10016.

2. Roberts, A.D., Prentice, R.C., "Programming for N/C Machines", McGraw-Hill, New York, N.Y., 1968.

3. Thornhill, R.B., "Engineering Graphics and N/C", McGraw-Hill Book Co., New York, N.Y., 1967.

4. DeVries, M.A., NCSI (Numerical Control Society, Inc.) "The Practical Impact of N/C", Proceedings of the 3rd annual Technical Conference April 1966, Numerical Control Society.

5. DeVries, M.A., "N/C, Tomorrow's Technology Today", Proceedings of the 5th annual meeting and Technical Conference, 1968, NCS.

6. DeVries, M.A., "N/C, A Vehicle for Progress", Proceedings of the 4th annual meeting and Technical Conference, 1967, NCS.

7. DeVries, M.A., "From Tapt to Time Sharing", Proceedings of the 6th annual meeting and Technical Conference, 1969, NCS.

8. Anon. "ALRP-APT Part Programming Manual Number 20" IITRI, 10 W35 St, Chicago, Ill. 60616.

9. Anon. "APT Dictionary", Univac 1107, Thin film memory computer system. IITRI, 1963.

10. Anon. "3D Contouring", N/C Tape Preparation Uniapt Part Programming Manual. United Computing Corp., 22500 South Avalon Blvd., Carson, California 90744. (213) 830-7720.

11. Anon. "Uniapt Facts", UCC (see above).

12. Anon. "Uniapt Questions and Answers", UCC.

13. Solow, Herbert, "How to Talk to Machine Tools", Fortune Magazine, March 1962, PP 120 ff.

14. Anon. "AUTOMAP - Automatic Machining Program IBM 1962", IBM Technical Publications Dept., 112 East Post Rd., White Plains, N.Y. (IBM Application Program Bulletin).

15. Thomas, R.A., "The Languages of Tape", American Machinist, Special Report number 545, January 6, 1964.

16. Anon. "AD-APT for the 1620-1311 Data Processing System Application Description", (IBM Application Program Bulletin), Number 1620, CN-06X.

17. Anon. "AUTOSPOT-Automatic System for Positioning Tools". IBM Data Processing Application.

18, Anon. "Programming of N/C Machine Tools", NEL Report No. 187, May 1965. Ministry of Technology, U.K.

19. Anon. "Computer-Aided Design", NEL No. 242, Ministry of Technology, U.K. August 1966.

20. Anon. "A Survey on Part-programming for PTP, N/C Machine Tools", Ministry of Technology, U.K. May 1967.

21. Anon. "This is N/C", Monarch Machine Tool Co.=

22. Anon. "2CL Part-programming Manual" (1st Revision) NEL No. 424, Ministry of Technology, U.K., July 1969.

23. Heidemann, M.V., "A Study of Some Aspects of Numerical Control Machine Tools", Partial requirements for Master of Engineernng, November, 1968.

24. Anon. "N/C 360 APT", IBM Application Program, Form H20-0181-0.

25. Anon. "Conference on Numerical Control Machines", Institute of Mechanical Engineers, 1 Birdcage Walk, London SW1, Oct. 30, 31, 1969.

26. DeGroat, George, "NCS Takes a Hard Look", American Machinist, Volume 115, No. 9, Page 57, May 3, 1971.

27. Gaertner, J.F., "Selling NC to the Workers", American Machinist, Volume 115, No. 6, Page 68, March 22, 19 .

28. Greening, J.H., "Build a 'Master Plan' for NC", AM Special Report No. 648, American Machinist, Vol. 115, No. 6, March 22, 1971.

29. Weller, J.A., "Where Canada Stands on NC", Canadian Machinery and Metalworking, Vol. 81, No. 3, Page 69, March 1970.

30. House, G.J., "How McMaster wants to spread NC Knowhow", Canadian Machinery and Metalworking, Vol. 81, No. 3, Pg. 76, March 1970.

31. "Beginner's course in Numerical Control", Canadian Machinery and Metalworking, Vol. 79, No. 3, PP 89-115, March 1968.

32. Thomas, L.J., Editor, "N/C Handbook", Bendix Industrial Controls Division, 12843 Greenfield Rd., Detroit, Michigan 48227, U.S.A. January 1971.

33. Stute, G. and Eitel, H., "The Milling Technology in EXAPT 3", Numerical Control Programming Languages, Proceedings of the 1st International IFIP/IFAC Prolamat Conference, Rome, 1969, Published by North-Holland Publishing Company, Amsterdam, 1970.

34. "APT Reference Manual", CDC 6400/6600 Computer Systems, Control Data Corporation, Documentation Department, Palo Alto, California 94304.

35. Guru, B.P., Publication No. 3.16 (Overlays) McMaster Data Processing Center Oct. 21, 1971.

36. Anon. "SYSTEM/360 AD-APT/AUTOSPOT, Numerical Control Processor (360A-CN-09X) Application Description", Publication No. H20-0463-0, IBM Technical Publications Dept., 112 East Post Rd., White Plains, N.Y. 10601.

37. Anon. "SYSTEM/360 AD-APT/AUTOSPOT (OS), Numerical Control Processor (360A-CN-12X)", Part Programming Manual, Publication No. GH20-0549-1.

38. Anon. "SYSTEM/360 AD-APT/AUTOSPOT Numerical Control Processor", Version 2, Part Programming Manual, Publication No. GH20-0375-2.

40. Anon. "EXAPT Information", Verein Zur Förderung des EXAPT-Programmiersystems e.V. 51 Aachen, Postfach 587, Deutschland.

41. Arrowsmith, J.R., "The Chaos of NC Languages", Canadian Machinery and Metalworking, March 1971.

42. Anon. "EXAPT Technical and Economic Data", System, Science and Software, EXAPT U.S.A./Canada.

43. Nussey, I.D., "The Purpose and Future of Part Programming", Numerical Control Programming Languages (W.H.P. Leslie, Editor) 1970, Page 186.

44. Opitz, H. et al, "Calculation of Technical Data for the Computer Assisted Programming of NC Machine Tools." Numerical Control Programming Languages, 1970, Page 243.

45. Henderson, W.T.K., "The Facilities Available in the NEL/NC 2PL (Point-to-point) Subset", Ministry of Technology (U.K.) N.E.L. Memo X5/178, February 1969.

46. Shepherd, C. and Wilkinson, D.G., "Status of the 2C Processor", Ministry of Technology, NEL Memo X5/203, October, 1969.

47. Hughes, C.J., "The NEL NC Processor; 2CL, 2PL, 2C; Ministry of Technology, NEL Memo X5/213, November, 1969.

48. Anon. "Numerical Control Programming", Mark 1 Time-Sharing Service, General Electric Information Systems No. 71222A, February 1969.

49. Anon. "Numerical Control Part-Programming Languages", MARK II Users Guide, General Electric Information Systems No. 007429, November 1970.

50. McCarroll, John D., "Computer Aided Part-Programming for Numerical Control", Industrial Development Division, Institute of Scince and Technology, University of Michigan, 1969.

51. Vliestra, J. and Wielenga, D.K., "Philcon", Chapter 1.4, Numerical Control Programming Languages North-Holland Publishing Co. (1970).

52. Anon. "System/360 AUTOSPOT, Numerical Control Processor (360A-CN-08X), Version 2, Part Programming Manual", Third Edition, IBM Technical Publications Dept., 112 East Post Road, White Plains, N.Y. 10601, Publication No. H20-0373-2.

53. Anon. "Introduction to Data Processing for Numerical Control of Machine Tools", IBM Technical Publications Dept., 112 East Post Road, White Plains, N.Y. 10601, First Edition November, 1970.

54. Anon. "System 360 AUTOSPOT, NC Processor (360A-CN-08X) Application Description", Technical Publications Dept., White Plains, N.Y. 10601, First Edition, Publication No. 420-0462-0.

55. Anon. "Quickpoint 8", Digital Equipment Corporation, 146 Main Street, Maynard, Mass. 01754, U.S.A.

56. Ackley, D.A., Barber, J.D., and Struckel, M.R., "APT III FORTRAN IV Three-Dimensional Postprocessor", Bendix Industrial Controls Division, Detroit, Michigan 48228, Technical Paper 66-13.

57. Bobrowicz, V.F., "General APT III Computer Independent FORTRAN IV Modular Postprocessor Philosophy", Bendix Industrial Controls Division, Detroit, Michigan, Technical Paper 66-14.

58. Hoffman, G.W., Zager, V.A., "Multi-Axis Milwaukee-Matic Software", Bendix Industrial Controls Division, Detroit, Michigan, Technical Paper 66-11.

59. Anon. "Introduction of Programming", Software Group, Programming Department, Digital Equipment Corporation, Maynard, Massachusetts 01754 (1968).

60. Husemeyer, N.C., "Costing and Contouring with Point-to-Point Numerical Control", Department of Mechanical Engineering, McMaster University, Hamilton, 1971, Master's Thesis.

61. Wong, M., Master's Thesis Project, (to be published), Department of Mechanical Engineering, McMaster University, Hamilton.

62. Cuthbertson, I., NC specialist at Canadian General Electric, Toronto, technical and programming assistance.

63. Hartung, D.B., "Multi-Machine Computer Controlled Director", Tomorrow's Technology Today, Paper No. 2, Numerical Control Society, Princeton, N.J., 1968.

64. Ingersoll, R., "Remote DNC Bows at Leipzig", American Machinist, Vol 115, No. 10, May 17, 1971.

65. Anon., "System/360 APT numerical Control Processor (360A-CN-10X) Version 4 Part Programming Manual", IBM Application Program, Publication No. GH20-0309-4, November 1970.

66. Anon., "System/360 APT Numerical Control Processor - Versions 3 and 4 (360A-CN-10X) Application Description", IBM Application Program, Publication No. GH20-0181-2, April 1970.

67. McWaters, J.F., and Henderson T.K., "The NEL 2C,L Processor", Numerical Control Programming Languages (1970), Pages 153ff.