Mark Century

GECENT
POSTPROCESSOR

Computer Programmer Manual

GENERAL ELECTRIC

NUMERICAL EQUIPMENT CONTROL DEPARTMENT, WAYNESBORO, VIRGINIA

Mark Century

GECENT Postprocessor

Computer Programmer Manual

MANUAL NO. 247

ISSUED TO

MARTIN, EARNEST
President
Martin Tool Works, Inc.
3320 Tollview Drive
Rolling Meadows, Illinois 60008

GENERAL ELECTRIC

NUMERICAL EQUIPMENT CONTROL DEPARTMENT, WAYNESBORO, VIRGINIA

DISCLAIMER

Although the GECENT* Postprocessor has been extensively tested by the General Electric Company, no warranty, expressed or implied, is made by the Company as to the accuracy and functioning of the GECENT Postprocessor or related program material or services, and no responsibility is assumed by the General Electric Company in connection therewith.

© COPYRIGHT 1962, 1963, 1964, 1970

by

GENERAL ELECTRIC COMPANY, USA

Regulations of the U.S. Department of Commerce prohibit the reexportation of this technical data to Albania, Bulgaria, Communist China,the Communist controlled area of Vietnam, Cuba, Czechoslovakia, East Germany,Estonia, Hungary, Latvia, Lithuania, North Korea,Outer Mongolia, Poland, Rumania and the U.S.S.R.

GENERAL ELECTRIC
COMPANY

WAYNESBORO, VIRGINIA . . . . TELEPHONE 703—942-8161

COMMUNICATION AND
CONTROL DEVICES
DEPARTMENT

We are pleased to present you with your personal copy of the GECENT Postprocessor Computer Programmer's Manual. It has been registered in your name at the address given on the title page.

The documentation for the GECENT III program is in two volumes, one for the computer programmer and a separate volume for the part programmer. Some of you will receive both volumes; but in some cases, the Part Programmer's Manual will be mailed to another designated individual in your company.

Additional copies of either manual may be purchased at $42 each should others in your organization require them. Contact your local General Electric Industrial Salesman.

Please check your address. It is almost impossible to keep our mailing list up-to-date. If your address changes or if you transfer this manual to another person, please let me know immediately. I will be updating this manual periodically; therefore, it is to your advantage to keep me posted if your address changes.

This is your manual. We want it to serve you well. Should you find errors or omissions in this documentation, please send them to me. Your comments and criticisms will be appreciated.

Richard A Thomas

Richard A. Thomas, SOFTWARE COORDINATOR

RAT:mh

Enclosure

GENERAL ELECTRIC

COMPANY

WAYNESBORO, VIRGINIA 22980 . . . . TELEPHONE 703—942-8161

NUMERICAL EQUIPMENT

CONTROL DEPARTMENT

December 15, 1970

SUBJECT: GECENT Postprocessor Computer Programmer Manual

Attached is the first revision to your GECENT Postprocessor Computer Programmer's Manual. Not only is new copy enclosed, but you will notice a new format is being used. When you attempt to open your book, you will discern why.

It will be necessary for you to drive the closing stick open using a screw driver and mallet, as the contents are too tightly packed. Remove pages 6-1 through 6-60 in your manual and replace with the attached pages.

The new material, and more to follow, will reduce the number of pages and permit normal entry to the manual. This mailing is a part of our continuing effort to keep your manual as current as possible.

Richard A Thomas

R. A. Thomas, SOFTWARE COORDINATOR

RAT/pm

Attachment

FORWARD

This manual completely describes the GECENT III Postprocessor System as to its computer design, theory of organization, and details of operation.

It has been assumed that the reader is familiar with APT, FORTRAN IV, and with general computer practices and furthermore, has a knowledge and understanding of the postprocessor part programming vocabulary. The GECENT III Part Programmers Manual is a supplement to this manual and is referred to many times.

The various subsections of this document are complete in themselves. A relevant section may be identified so that further investigation into the subject may be made. Mandatory readings are Sections 1 and 2, since these sections delineate the fundamentals which must first be known before proceeding with further detail. Section 3 gives a detailed description of the elements of the entire postprocessor, and Section 4 itemizes some of the special functions of the postprocessor. Section 5 should be read by anyone who plans to work with the postprocessor, modifying it or simply maintaining it.

For further information or assistance on a part of this manual or the postprocessor, the reader is directed to:

The GECENT Postprocessor Program
General Electric Company
Building 305, Mail Drop H-8
Evendale, Ohio 45215

February 1970

TABLE OF CONTENTS

TABLE OF CONTENTS (con't)

TABLE OF CONTENTS (con't) Page

* Information not available for this distribution

TABLE OF CONTENTS (con't) Page

* Information not available for this distribution

TABLE OF CONTENTS (con't) Page

* Information not available for this distribution

TABLE OF CONTENTS (con't) Page

1.0 INTRODUCTION

The GECENT* III Postprocessor System is a highly generalized, modularized computer program for processing an APT CL tape for a General Electric Mark Century numerical control system. Except for a few output subroutines, the program is written in FORTRAN IV and uses many of the FORTRAN IV capabilities such as: DATA statements, logical IF statements, labelled COMMONs, and overlays.

The GECENT III postprocessor is designed for all third generation computers which have an APT system; it requires no more than the equivalent complement of core that APT may use; for, depending upon the NC machine type, the postprocessor may require as little as 20K words of storage or as much as APT uses in one core load. For multihead processing, the postprocessor requires two scratch devices (TAPES2 and TAPES3).

Approximately five output subroutines are written in machine language.** This was done to minimize computational time.

In order to overcome possible computer incompatibilities and also to reduce subroutine compilation sizes, the floating point numbers 0 through 5, 10, 100, and 360 are defined as FLZ, FL1 and so on. The integers 0 through 7 are defined as INTZ, INT1, INT2, and so on.

The structure and design of the postprocessor adheres to the recommended ALRP postprocessor guidelines, and consistancy with EIA, ASA, and NAS standards has been kept.

Processing through the postprocessor is in one pass for ~~all~~ ALL NC machines except multihead machines which inherently require two passes. Processing speeds are extremely fast since optimum programming methods have been utilized where possible.

* Trade mark of General Electric Co. ** Only two machine language subroutines are used with the IBM 360 System.

1.0 INTRODUCTION (cont'd)

The postprocessor can produce four types of printout and three types of punched output. There are a number of other special features which permit a greater ease in part programming and increase the scope of applications. These are all itemized and documented in detail.

Before proceeding, some important definitions must be established. With respect to the word "command" as used in this manual, a "command" refers to the programmed coded symbol fed to the NC machine control; the command initiates the NC machine action. Thus, a feedrate and a feedrate command, although related, are two entirely different things. The feedrate is the actual value rated in IPM, while the feedrate command is the coded F number fed to the NC machine control. The same interpretation applies to the spindle and spindle speed command.

In the same sense, a "command block" is a set of coded data in command form which is fed to the NC machine control for the execution of one or more functions.

The abbreviations NC for "Numerical Control", IPM for Inches Per Minute, and RPM for Revolutions Per Minute are used throughout this manual.

The convention of using lower case letters for the part coordinates and capital letters for machine coordinates is consistent throughout. Thus, xyzijk refer to the three linear axes xyz of the part coordinate system, and ijk refer to the direction cosines of the backward directed vector of the tool axis. The corresponding machine coordinates are then XYZABC, where XYZ are the linear axes and ABC are the rotary axes of the NC machine.

The notation $\langle \text{terms} \rangle$ is used to indicate that an integer result is obtained from the terms in the brackets. For example,

$$X = \frac{10}{3} = 3.3333,$$

but

$$X = \left\langle \frac{10}{3} \right\rangle = 3.$$

1.0 INTRODUCTION (cont'd)

REPLACE TEMP W/DTEMP BELOW

One final but very important restriction must be mentioned; in the use of the parameters TEMP and ITEMP which are located in COMMON, TEMP is dimensioned at 10. These parameters serve the general purpose of providing a temporary storage space for a subroutine. In order to avoid any potential error, the rule is never to use TEMP or ITEMP in any subroutine which calls another subroutine (except library subroutines). For if subroutine A uses TEMP and calls subroutine B which also uses TEMP, then subroutine B will have destroyed TEMP for subroutine A. These types of errors are extremely difficult to find, so for safety's sake, the above restriction must be observed.

The parameter names in the body of the manual are IBM 360 names.

APPENDIX SECTION 7.9 GIVES THE [redacted] PARAMETER CROSS REFERENCE. THE SYMBOL "n" SPECIFIES n NUMBER OF BLANKS.

2.0 GENERAL DESCRIPTION

In the sections which follow there is given a brief but detailed survey of the postprocessor, its theory of operation, characteristics of programming flow, and general structure. A discussion of each major NC machine type is made with particular emphasis concerning the affect upon the logical structure and flow of the postprocessor.

Sections 2.1, 2.2, and 2.3 should be read for the general overview of the postprocessor while the various subsections of Section 4 should be consulted for information regarding a particular type of NC machine.

2.1 THEORY OF OPERATION

The GECENT III postprocessor is a generalized, modularized system of subroutines which optimizes processing operation by loading and utilizing only those postprocessor segments required for a given machine tool. The structure of the postprocessor is based upon overlays which are selected at load time to form the requisite body of subroutines for postprocessing a part program.

The GECENT program is written completely in FORTRAN IV except for basic output subroutines such as CONBCD. (See Section 2.4.2) These subroutines have been deliberately kept in machine language so as to obtain maximum processing computer speed.

The GECENT III structure and theory of operation is based upon the commonality of features which exist in postprocessors for the various types of machine tools. In every postprocessor there are common functions which must be performed; whereas, certain other functions are required only for a specific type of NC machine. By grouping these functions and using them as needed, it is possible to put the postprocessor together at load time as a function of the specific NC machine being processed.

Some items which are common to all postprocessors are: CL tape reading, producing punched and printed output, processing of standard postprocessor statements, such as, PARTNO, PPRINT, MACHIN, and so on. These common functions are therefore grouped together into the basic overlay.

In a similar manner, other postprocessor functions can be grouped according to their common usage by positioning machines, lathes, mills, or whatever the machine class may be.

2.1 THEORY OF OPERATION (cont'd)

In brief, the technique of operation is as follows. Upon entry into APT Section IV, the postprocessor control element loads in a basic overlay which represents the minimum structure of the postprocessor. To this basic structure are added those overlay modules which are required by the selected machine. These may include the machine subroutine and possibly a spindle type subroutine, interpolation module, and one of the multiple printed output modules, as well as the main module for a lathe, mill, or whatever type NC machine is being processed. In any event, after primary initialization, there resides in core only those overlays pertinent to the machine tool for which the given part program is being processed.

An important point to note is that once the proper overlays are established in core there is no further processing of the overlays; that is, an overlay is not repeatedly pulled into core, overlayed later by another overlay, then the original overlay pulled is again, and so on. With the exception of the initialization overlay, no other segment overlay is replaced during the processing of any non-multihead machine tool part program.

Multihead processing inherently requires a two-pass system to merge the output data of each head. (See Section 2.4.5) In this case, an additional overlay replaces the basic structure overlays when head merging is performed. But once again there is no flip-flopping of overlays.

It bears repeating: when an overlay is pulled into core, it resides there until its function is completed at which time it may be overlayed by a new function overlay; but once used, it is never pulled back into core.

2.2 OVERLAY STRUCTURE

The general overlay structure is given in Diagram 2.2A

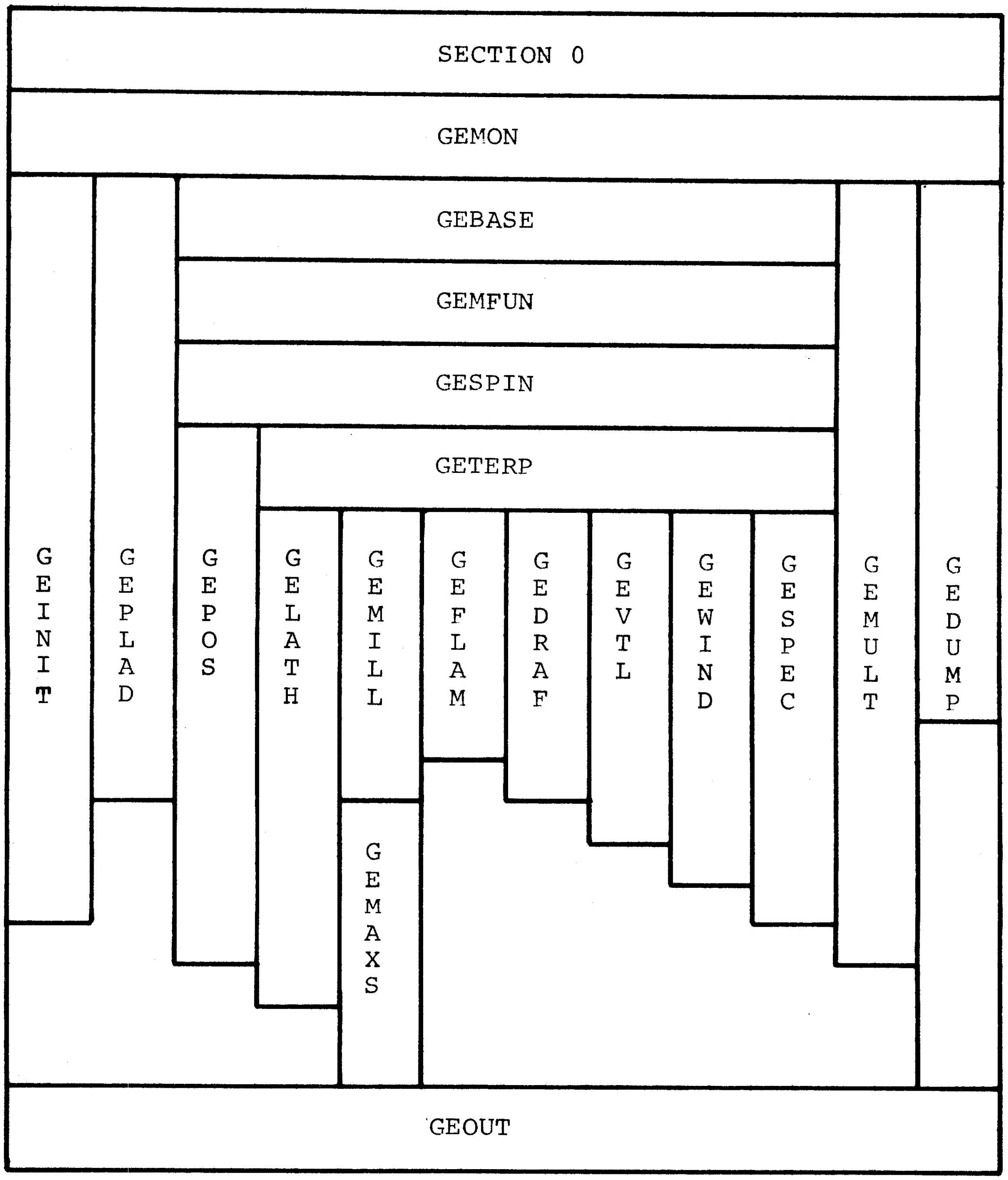


Diagram 2.2A

2.2.1 NAMING CONVENTIONS FOR THE GECENT III POSTPROCESSOR

Each major overlay is identified by the prefix "GE". ~~Only overlays have the prefix "GE"~~.

Overlays	Function
GEMON	The monitor overlay which directs the selection and processing of all other overlays.
GEINIT	The initializing overlay which establishes the starting conditions for postprocessing.
GEPLAD	The planning overlay which produces tool setups and machinability features for positioning machines
GEBASE	The basic overlay which contains the postprocessor subroutines common to all machine tools
GEPOS	The position machine overlay
GELATH	The lathe overlay
GEMILL	The 3-axis mill overlay
GEMAXS	The multiaxis overlay
GEMULT	The multihead sequence overlay
GEOUT	The output overlay
GETERP	The interpolation overlay containing linear and circular processing.
GEFLAM	Flame cutter overlay
GEWIND	Filament winder overlay
GEDRAF	Drafting machine overlay
GESPEC	Special purpose overlay
GEVTL	Vertical turret lathe (special cases).
GEWELD	Welding machine overlay
GEDUMP	The error dump overlay

2.2.2 OVERLAY LOADING TECHNIQUE

The method used in the GECENT III postprocessor to load the proper overlay modules is to interrogate the table data provided in the machine subroutine which has been selected by the input part program. (See Section 5.6 for a description and usage of the machine subroutine.)

When control is transferred to the postprocessor from APT Section IV, the control monitor overlay GEMON pulls in the initialization overlay GEINIT and transfers control to it. Within the GEINIT overlay are contained the machine subroutines and other basic initialization subroutines. The CL tape is read until the MACHIN statement is found. When found, the machine number is used to select and call the corresponding machine subroutine. For example, the statement MACHIN/GECENT, 4 causes machine subroutine MACH04 to be selected and called.

When the machine subroutine is called, the machine tool characteristic tables TABLEG, TABLEM, OPTAB, and SRTAB are set up. The postprocessor can now determine the required overlays by interrogating the pertinent option set in OPTAB. This is done after control is returned to overlay GEMON.

Upon return to GEMON, overlay GEBASE is loaded in. The other overlays are pulled in dependant upon the following option settings:

If option 1 = 0, contouring is designated and overlay GETERP is called.

If option 1 = 1, positioning is designated and overlay GEPOS is called. (See Section 5.6.2 for the special negative setting of this option.)

The technique of pulling in an overlay is dependant upon the computer used. Most computers [1] pull in an overlay when a subroutine in that overlay is called from a higher level overlay. Program control is then transferred to the called subroutine. Other computers [2] can pull in an overlay by name, as the overlay GELATH, without necessarily transferring control to that overlay.

[1] For example: IBM360 Models 40, 50, 65, 75; UNIVAC 1107, 1108; CDC 3600, 3800, 6400, 6600; RCA Spectra 70.

[2] For example: GE625, 635.

2.2.2 OVERLAY LOADING TECHNIQUE (cont'd)

In the descriptions which follow, when reference is made to an overlay being selected by an option, the actual loading of the overlay occurs by either of the above mentioned two techniques. For example, the reference to option 132 for a lathe implies that the actual loading of the overlay takes place by either of the following two methods:

(A) Pull in the overlay GELATH when a lathe subroutine is required, e.g., subroutine SFMO.

(B) Pull in the overlay GELATH by name when option 132 is zero.

(See Section 5.4.1.2 for the complete description of the overlay loading methods used by the different computers.)

Continuing with the selective loading of overlays:

When option 132 is 0, pull in GELATH.

When option 132 is 1, pull in GEMILL.

When option 132 is 2, positioning is designated, but GEPOS would already be pulled in under control of option 1.

When option 132 is 3, pull in GEDRAF. (drafting machine)*

When option 132 is 4, pull in GEFLAM. (flame cutter)*

When option 132 is 5, pull in GEWELD. (welder)*

When option 132 is 6, pull in GEWIND. (filament winder)*

When option 132 is 7, pull in GEVTL. (vertical turret lathe)*

Other main overlays can be defined and used as needed.

If option 116 ≠ 0, multiaxis processing is indicated and overlay GEMAXS is called.

The GEOUT overlay can be one of five print sequences; viz., GEOUT1, GEOUT2, GEOUT3, or GEOUT4. See Section 3.5.

* These settings for option 132 are not currently used.

2.2. OVERLAY LOADING TECHNIQUE (cont'd)

If option 164 = 1, use GEOUT1.

If option 164 = 2, use GEOUT2.

If option 164 = 3, use GEOUT3 for multihead machines.

If option 164 = 4, use GEOUT4 for non-multihead machines.

Other pertinent items for postprocessing, though not necessarily for loading overlays, are:

A. Option 19 for spindle type. Any one of several types may be used. (See Section 4.9.)

B. If option 133 is non-zero, a special function is to be performed for the given machine tool. The particular MACFUN will have to be used. (See Section 5.6.1.)

C. If the modifier PLAN is given in the MACHIN statement and option 1 is + 1, the GEPLAD overlay is called in to process and redevelop a new CL tape before continuing with the regular GECENT III sequence. (See Section 4.10 on GEPLAD.)

D. If multihead postprocessing is in operation (flag MULTHD is non-zero), the overlay GEMULT is called in when the first pass through the postprocessor is completed. (See Section 2.4.5 on multihead processing.)

E. Whenever a fatal error occurs, the overlay GEDUMP is automatically pulled in to produce a comprehensive print of all the pertinent postprocessor parameters. (See Section 5.7.)

Once all of the overlays are loaded into core, control is given to GEBASE which processes the entire CL tape, and upon completion, returns control back to GEMON which then calls DISPAT to return control back to APT Section IV.

2.2.3 SAMPLE POSTPROCESSOR STRUCTURES

<table>
<tr><td colspan="2">GEMON</td></tr>
<tr><td rowspan="4">GEINIT</td><td>GEBASE</td></tr>
<tr><td>GETERP</td></tr>
<tr><td>GELATH</td></tr>
<tr><td>GEOUT1</td></tr>
</table>

Lathe

OPTAB(1)	=	0.
OPTAB(132)	=	0.
OPTAB(164)	=	1.0

<table>
<tr><td colspan="3">GEMON</td></tr>
<tr><td rowspan="4">GEINIT</td><td rowspan="4">GEPLAD</td><td>GEBASE</td></tr>
<tr><td>GEMFUN</td></tr>
<tr><td>GEPOS</td></tr>
<tr><td>GEOUT1</td></tr>
</table>

Drill

OPTAB(1)	=	1.0
OPTAB(132)	=	2.0
OPTAB(133)	=	1.0
OPTAB(164)	=	1.0

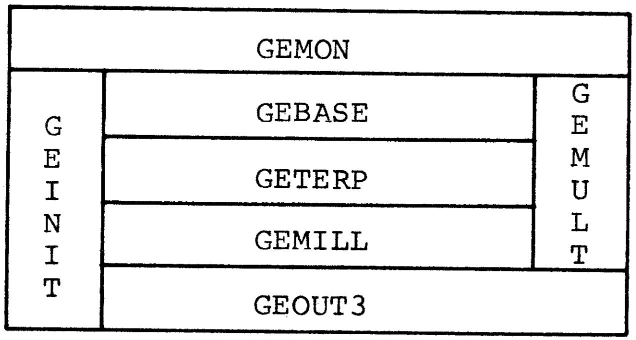
MACHIN/GECENT, n, PLAN

<table>
<tr><td colspan="2">GEMON</td></tr>
<tr><td rowspan="5">GEINIT</td><td>GEBASE</td></tr>
<tr><td>GETERP</td></tr>
<tr><td>GEMILL</td></tr>
<tr><td>GEMAXS</td></tr>
<tr><td>GEOUT4</td></tr>
</table>

Multiaxis Mill

OPTAB(1)	=	0.
OPTAB(116)	=	1.0
OPTAB(132)	=	1.0
OPTAB(164)	=	4.0

2.2 3 SAMPLE POSTPROCESSOR STRUCTURES (cont'd)



Multihead Mill

OPTAB(1)	=	0.
OPTAB(132)	=	1.0
OPTAB(164)	=	3.0

COMBIN is given designating multihead operation.

2.3 DESCRIPTION OF PROGRAM FLOW

The following is a brief description of the method and technique used for processing a postprocessor statement. The method described pertains to all machine types since there is no special flow except for the special function MACFUN. (See Section 5.6.1).

Postprocessing begins in overlay GEBASE with a call to the input subroutine INPUT which reads a CL ~~tape~~ record. The record is stored in the input buffers [redacted] DATASL and ICLDAT. Program flow proceeds as a function of the contents of the input arrays; i.e., the flow may be either for a motion record or for a non-motion record. In either event, the postprocessor, in processing the data of the input arrays, sets up the command block array DBFSEG which ultimately is converted to BCD and made output.

The DBFSEG array is dimensioned at 30 to provide storage for all possible letters of the alphabet and to allow room for other output parameters. The first fifteen cells of the array have specific assignments and are the cells most commonly used by nearly all machine tools. The other cells (up to cell 26) are used as required for machine tools which have multiple heads, slides, or additional registers for various and uncommon functions.

2.3.1 FIXED ORDER AND ASSIGNMENTS OF DBFSEG

The fixed assignments of DBFSEG are given below:

DBFSEG	Function	
1	Sequence number	(N)
2	Preparatory code	(G)
3	Primary coordinate axis, abscissa	(X)
4	Primary coordinate axis, ordinate	(Y)
5	Third primary coordinate axis	(Z)
6	Rotary axis (for a head)	(A)
7	Rotary axis (for a table)	(B)
8	Direction cosine or arc center offset for abscissa	(I)
9	Direction cosine or arc center offset for ordinate	(J)
10	Direction cosine or arc center offset for third primary axis	(K)
11	Feedrate command	(F)
12	Spindle command	(S)
13	Tool or turret code	(T)
14	Miscellaneous code	(M)
15	Command block identification code	(CODE)
16	Rapid traverse	(R)
18	Third rotary axis	(C)
19	Feedrate in IPM	
20	Spindle speed in RPM	

2.3.1 FIXED ORDER AND ASSIGNMENT OF DBFSEG (cont'd)

The letters in the right column represent the BCD letter address for these registers as set up in the standard REGSTR table. Actually, any letter or Hollerith character may be used, e.g., DBFSEG(2) can be assigned the letter H. The only requirement is that the DBFSEG cell be used for its assigned function; in this case H must be the preparatory code.

The DBFSEG array in being set up is assigned a CODE number (stored at DBFSEG(15)) which identifies the command block type, since at output time each command block type is processed differently. The command block CODE is used elswhere within the postprocessor, but its primary function is for output branching and subsequent processing.

The reason a command block is set up and identified with a CODE (DBFSEG(15)) instead of being made direct output, is because under certain circumstances the block must be saved for later possible modification as a function of other blocks which will either precede or follow its output. For example, a command block's feedrate may be altered because of A/D (acceleration/deceleration) restrictions; or the block may be merged with other blocks for multihead sequencing. Thus, each block must be uniquely identified so that the postprocessor knows which course of operation to follow in processing that block.

2.3.2 COMMAND BLOCK IDENTIFICATION CODES

CODE	Identification
0	Contouring linear move having increments ΔX, ΔY, ΔZ
+1	Non-motion block
-1	Non-motion block having auxiliary code by itself
+2	Rotary absolute move, absolute system
-2	Rotary incremental move, incremental system
+3	A FROM point
-3	Turret corrective move (generated by NOW modifier)
+4	A dwell block
-4	A preparatory code by itself, not a dwell
+5	An END block
-5	A RESET block
+6	An INSERT block
-6	A BREAK block
+7	A PPRINT block
-7	A PARTNO block
+8	A TMARK block
-8	A LEADER block
+9	A postprocessor warning or error comment block
-9	A postprocessor information block; not made output
+10	Circular move CLW in XY plane
-10	Circular move CCLW in XY plane

2.3.2 COMMAND BLOCK IDENTIFICATION CODES (cont'd)

+11 Circular move CLW in ZX plane

-11 Circular move CCLW in ZX plane

+12 Circular move CLW in YZ

-12 Circular move CCLW in YZ plane

+13 A thread block with a five digit lead

-13 A thread block with a six digit lead

+14 A turret correction on head 1 and a motion on head 2(multihead processing).

-14 A turret correction on head 1 and a motion on head 1(multihead processing).

+15 An auxiliary head motion having feed command in IPM.

+16 Position move in X and Y; generated by a GOTO statement.

-16 A positioning move in Z; generated by a CYCLE statement.

+17 An Op/n or PRFSEQ information block.
A combined multihead move using the head 1 feedrate.

-17 A combined multihead move using the head 2 feedrate.

+18 A FINI block.

Every command block must have an identifying CODE otherwise an error is assumed.

When the command block's basic elements have been set up and the CODE determined, subroutine OUTPUT is called. For the appropriate CODE, this subroutine adds the feedrate, spindle command, sequence number, preparatory code, and auxiliary code, and essentially completes the setup of the command block.

2.3.2 COMMAND BLOCK IDENTIFICATION CODES (cont'd)

After the command block DBFSEG has been completely set up, it is then sent to GEOUT for output processing. At this time the cells of DBFSEG contain the numeric value for each particular related register that is to be made output, except when the command block is in BCD, as for a PARTNO. These numeric values are in floating point format but must be converted to BCD for output. This is accomplished by subroutine CONBCD.

Values from DBFSEG are taken cell by cell, sent through subroutine CONBCD, and the converted BCD equivalent is stored in the output array BCDIMG. BCDIMG (dimensioned at 38) is originally all blanks.

Each converted BCD equivalent from DBFSEG is stored in BCDIMG at the location indicated by that particular register's order number and value in the REGFOR table. This is explained in greater detail in Section 2.4. At the top of each page is printed the register symbols (as given by the REGSTR table), and each item stored in BCDIMG is located at the cell which lines it up with its related letter address in the title. When the setup of BCDIMG is complete, it is printed.

The same line image is then prepared for punched output.* All that is needed for punched output is to precede each cell value with the appropriate letter address for that register. For example, the sequence number 240 and preparatory code 01 may be in BCDIMG (in BCD form) as:

* The print line image referred to here is for the Incremental Printout only since this printout is a reflection of the punched tape for the NC machine. The Absolute and Operator Printouts do not necessarily represent the punched tape data.

2.3.2 COMMAND BLOCK IDENTIFICATION CODES (cont'd)

BCDIMG

1	2	3	4	5	6
		240		01	

This line image, when printed places the values under their appropriate register heading, as:

N	G	X	Y	Z	
240	01	etc.	etc.	etc.	etc.

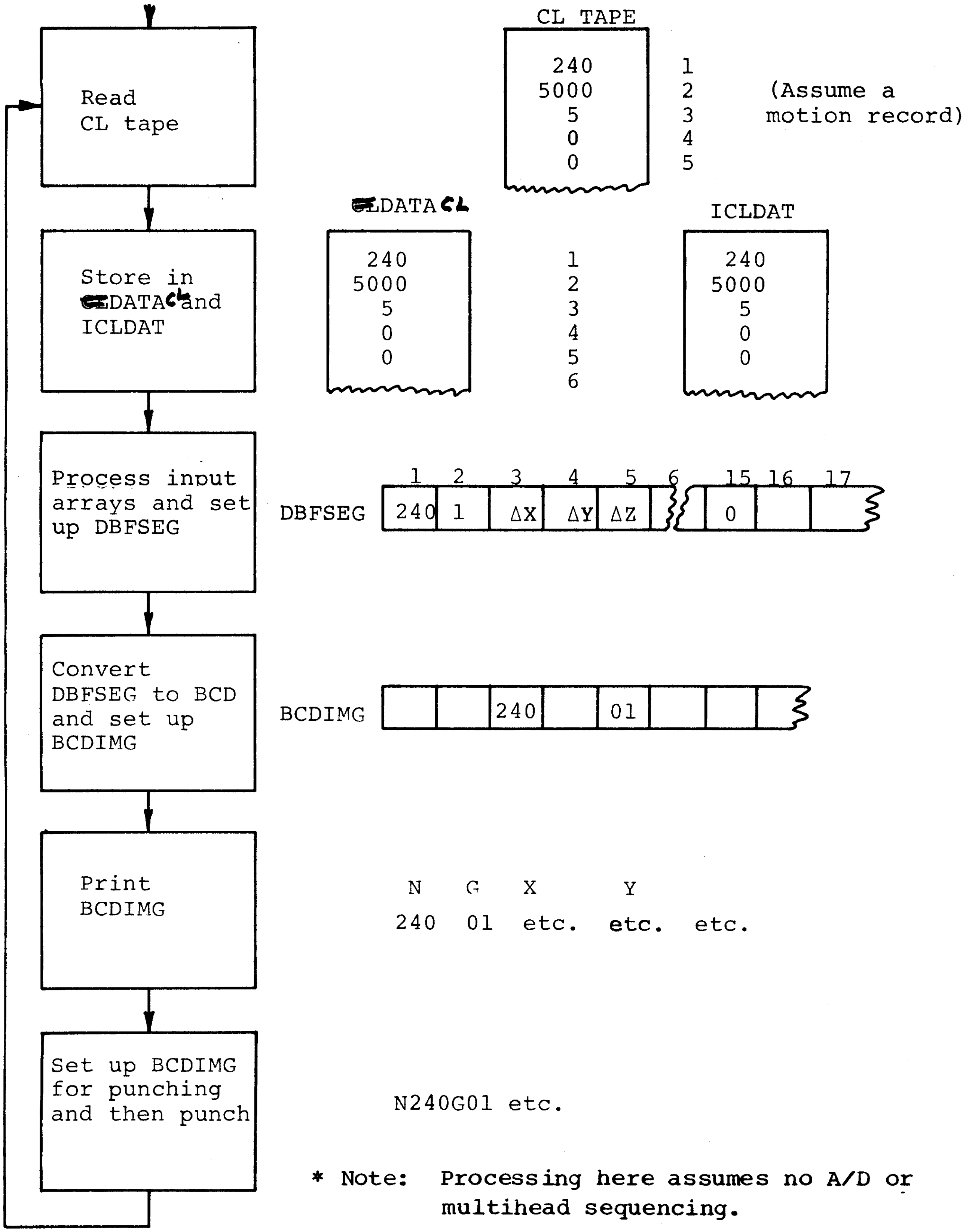
Hence, by adding in the BCD register letter address, BCDIMG is then ready for punching.

1	2	3	4	5	6
	N	240	G	01	

Thus the Incremental Printout of each register value gives the true representation of the punched output since the print image is also the punch image. The punch subroutines do not punch periods or blanks; however, for convenience the print image carries these symbols.

When output is complete, program flow returns to GEBASE which repeats the entire process.

2.3.3 SCHEMATIC OF PROGRAM FLOW*



* Note: Processing here assumes no A/D or multihead sequencing.

2.4 PROGRAM FLOW BY MACHINE TYPE

The GECENT III postprocessor handles a variety of machine tool types among which are positioning machines, lathes, 3-axis mills, multiaxis mills, multihead lathes and mills, welders, flame cutters, and other special purpose machines. The prime requirement for these and any other machine tools is that they be equipped with a Mark Century numerical control system.

The design of the GECENT III postprocessor is such that at load time the postprocessor is structured for the particular machine tool type being processed. The program flow through each type structure is basically the same, but there are certain variants which are unique to each type. The main type structures are for positioning machines (drills, grinders, boring machines, etc), 2 and 3-axis contouring, multiaxis milling, and multihead processing.

Generally with Mark Century numerical controls, contouring machine tools such as lathes and profile mills have an incremental system, whereas positioning machine tools, such as drills and boring mills, have an absolute system. (See Section 3.4 for definitions of these systems.) However, there are exceptions, such as a lathe having a positioning, absolute control, and a drill having a contouring, incremental control. These exceptions are special cases and are treated in a manner slightly different from the main types.

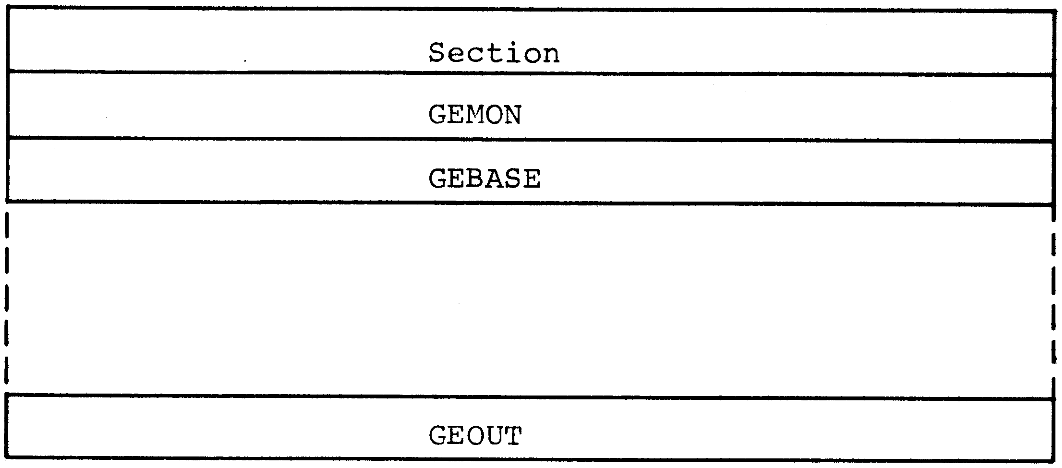
There are still other special cases, such as a filament winder or an electronic beam welder, which are treated separately by a special MACFUN sequence. (See Section 5.6.1.)

In the following description, contouring machines utilize an incremental departure system and positioning machines utilize an absolute coordinate system. Contouring machines which have a rotary table are assumed to have an incremental system on the rotary table also. Positioning machine tools which have a rotary table are assumed to have an absolute system rotary table.

Program flow through GEMON, GEINIT, GEBASE, and GEOUT is always the same for any machine tool type. It is generally only in the specific machine tool type overlays that program flow takes a different course or utilizes special functions. The description in Section 2.3 clearly defines this standard flow.

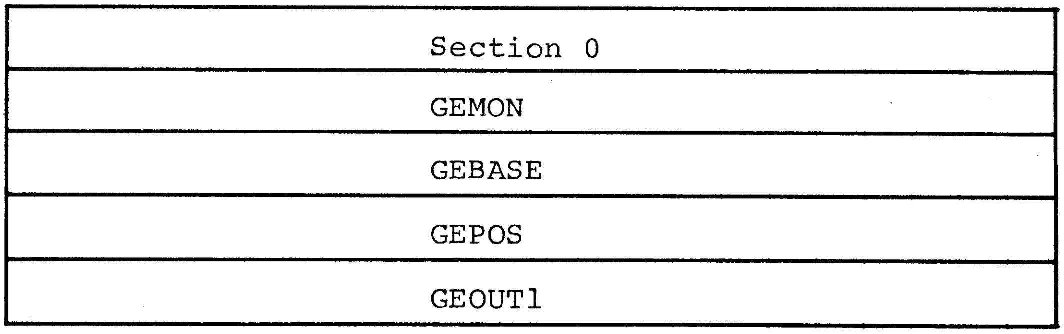
2.4 PROGRAM FLOW BY MACHINE TYPE (cont'd)

Section 2.2.2 described the manner in which the machine tool type is structured in the postprocessor by selective overlay modules. It is at this point in the program flow that the following descriptions continue.



2.4.1 POSITIONING MACHINES

The positioning overlay structure has the configuration shown in the diagram below. Although GEOUT1 is indicated, any of the GEOUT's could be used. GEOUT1, however, is normally sufficient for positioning machine printout verification.



2.4.1.1 POSITIONING MACHINE CHARACTERISTICS

The chief characteristic of the positioning sequence is that all output motion data is in absolute coordinates, and, unless a TRANS statement is given, the CL print coordinate points are identical with the postprocessor output points. The same is true for rotary table motions.

Positioning machines may have a rotary device other than a positioning table, e.g., a rotary indexer or a table with a few fixed positions. In all such cases, their operation either relies upon an auxiliary function M code or an absolute rotary register (A,B, or C). The important point to note here is that these and any other devices on a positioning machine utilize an absolute reference system which makes it possible to group the subroutine representing these features into a common overlay. This overlay is the GEPOS overlay.

Thus, the salient feature of GEPOS is that it is the main processing element in the GECENT III postprocessor which uses an absolute coordinate system.

Besides the absolute coordinate system, positioning machines also normally have discrete feedrate values; that is, only certain values are obtainable within a given minimum and maximum range. For example, a machine tool may accept only an integer value of feedrate in IPM in the range of 1 IPM to 20 IPM. A value of 3.7 would be unacceptable and would be converted to 4 IPM.

The feedrate command can be formulated from any one of several positioning feedrate types. (See option 78.) For example, in a Type 0 feed command, F_C may be such that:

$$F_C=2*F_{IPM} \cdot$$

Other types are defined and illustrated in Section 4.1.2.

Positioning machines which have a rotary table may also have a separate feedrate register for the table, and the rotary table feedrate command may also be formulated from any one of several types. (See Section 4.1.3).

2.4.1.2 POSITIONING MACHINE PROGRAM FLOW

After the CL tape is read and the input arrays CLDATA CL and ICLDAT are set up, subroutine GEBASE branches to subroutine MOTION for any motion record (FROM, GODLTA, GOTO). Subroutine MOTION tests option 1 which for a positive value branches to subroutine POSMOV to process the move as a positioning move in an absolute system.

Subroutine POSMOV sets up DBFSEG with the motion and feedrate values. Various positioning machines may have differing requirements for their motion registers. Some machines will accept the X, Y, and Z values all in the same block, while others require that Z be in a separate block following the XY block. Subroutine POSMOV tests option 130 for this requirement and produces the desired form of output.

This subroutine also suppresses redundant X, Y, and Z values since these values are modal in the control.

In setting up DBFSEG with the motion values, the subroutine assigns the command block value CODE to identify its condition types. CODE is set to +16 if X, Y, and Z are stored in one block. CODE also is +16 for X and Y in one block, and -16 for Z in a block by itself.

The current feedrate is next added to DBFSEG. If the condition is non-rapid, DBFSEG(11) is set to FEDIPM which is the closest programmable feedrate in IPM available on the machine. If the condition is for a rapid traverse, DBFSEG(11) is set to the rapid value FRAPID; the value stored in DBFSEG(11) is made negative to indicate a rapid traverse condition. (See Section 4.1.4.)

Finally, before outputting the command block, the absolute values of X, Y, and Z are tested by subroutine TSTLIM for possible transgression of the permissible slide limits. Warning comments are printed which identify the axes limit transgressed. When this is fulfilled, subroutine OUTPUT is called to complete the setup of DBFSEG and to prepare it for eventual output.

Subroutine OUTPUT does this by adding in the CL tape record number as the sequence number in DBFSEG(1), by adding in the CODE to DBFSEG(15), by adding in the pending spindle command SPNCOM (if any) to DBFSEG(12), and by adding in the pending auxiliary function M code VALUEM (if any) to DBFSEG(14). GEOUT is then called to produce the printed and punched output.

2.4.1.2 POSITIONING MACHINE PROGRAM FLOW (cont'd)

Regardless of which printed output sequence is used, the basic output for a positioning machine always derives from the flow described below.

Subroutine POSIT is called to suppress redundant X and Y values as a function of option 40, and then subroutine POSFED is called to convert the feedrate in IPM to the feedrate command code.

Subroutine POSFED tests option 78 for the required positioning feedrate type and branches accordingly. (See Section 4.1.2 for a description of each type positioning feedrate). It might occur that the programmed feedrate in IPM may be changed because of its unavailability in command form. For example, the programmed feedrate is 2.4 IPM. However, in this range, the table of discrete feeds permits only 2.0 or 3.0 IPM. Therefore, the postprocessor uses the feed command corresponding to 2.0 IPM and changes the feedrate in IPM (FEDIPM) to 2.0 IPM. Hence, the printed value of feedrate is the true value used and is not necessarily the programmed feedrate.

The final function that is performed in GEOUT for a positioning machine is to put the current spindle command in the command block of DBFSEG(12) if the block contains a T code. This is to ensure continuance of the proper speed after a tool change occurs.

At this point in the program the command block is fully prepared for output and is subsequently printed and punched. (See Section 3.5).

2.4.2 LATHES

Section 0
GEMON
GEBASE
GETERP
GELATH
GEOUT

A spindle type is also implied in the above structure.

2.4.2.1 LATHE CHARACTERISTICS

In nearly all cases, lathes employ an incremental contouring system. The programmed cutter path as presented on the CL tape is converted from its absolute coordinate form into one or more incremental segments whose summation (disregarding a TRANS) regenerates the original set of absolute data points within the step size tolerance of the machine tool. For example, the path from absolute X, Y coordinates (2, 6) to (6, 4) produces increments $\Delta X=4$, $\Delta Y=-2$. (See Section 3.4 for a complete description of the methods used for producing incremental moves.)

Any incremental motion can be segmented into yet smaller incremental motions. The path length may be segmented because the original increment may be greater than the maximum allowable departure (option 4). Any one of these segments could be further segmented by the SFM sequence (See Section 4.5), and these smaller segments still more segmented because of G code optimization (See Section 4.1.5.1). In any event, the summation of all these segments result in the original segment length.

The standard axes configuration for a lathe per EIA and NAS standards is +Z for the abscissa and -X for the ordinate. Part programming is normally done in the first quadrant of the standard rectangular Cartesian coordinate system, hence, the postprocessor must rearrange and modify the XY data into its required +Z-X output format. This rearranging is done per the setting of option 59 and option 60. It is, of course, possible to request any axes configuration desired; e.g., +X-Y, +X+Y, and so on.

2.4.2.1 LATHE CHARACTERISTICS (cont'd)

The feedrate command can be any one of the three contouring types. (See option 10 and Section 4.1.1 for a complete description.)

When a feedrate is programmed in an IPR mode, the postprocessor converts it to IPM by multiplying the IPR value with the spindle speed. The resultant feedrate in IPM is tested and made to be within the minimum (option 48) and maximum (option 25) feedrate value

Special functions normal for a lathe are threading, SFM, and turret operations, hence, the related subroutines are located in the GELATH overlay. Non-lathe type machine tools which have these functions are treated by special subroutines or most generally by a MACFUN. (See Section 5.6.1.)

2.4.2.2 LATHE PROGRAM FLOW

After the CL tape is read and the input arrays ~~C~~LDATA CL and ICLDAT are set up, subroutine GEBASE branches to subroutine MOTION which for a GOTO/ motion record calls subroutine TSTFLG. This subroutine tests a series of flags for special conditions such as a rapid traverse, reinstate, safety retract, and threading. (See Section 4.0 for a description of these special conditions.)

Upon return from subroutine TSTFLG, subroutine MOTION tests option 1 which for a zero value branches to subroutine GOLINE for a linear interpolation move or to subroutine GOCIRC for a circular interpolation move. (See Section 3.4.3 for a detailed description of the linear interpolation mode processing and Section 3.4.4 of the circular interpolation mode.)

Since program flow can proceed with either of these two modes, each path is separately described.

2.4.2.2.1 LINEAR INTERPOLATION FLOW

In subroutine MOTION, the CL data points had been stored in the part coordinate present point vector DPRESP. For a linear interpolation move, program flow continues in subroutine GOLINE where these data points are truncated and rounded to the machine tool step size and then stored into the machine coordinate present point vector DPRESM. This action occurs in subroutine GEOM. For example, say the CL data point for X is 24.678891.

2.4.2.2.1 LINEAR INTERPOLATION FLOW (cont'd)

This value is stored in DPRESP(1). When subroutine GEOM is called, this value is truncated and rounded by subroutine SRAREC to become 24.6789, assuming the step size (option 14) to be 0.0001 inches. The rounded value is then stored in DPRESM(1).

Before leaving subroutine GEOM the postprocessor calls subroutine TSTLIM to test the present machine point for possible violation of slide limits. Warning comments are printed for all slide violation.

The postprocessor always works within the machine coordinate system in generating additional segments, making A/D corrections, computing departures, or in any sequence which deals with the coordinate data.

Subroutine DEPART is called to produce the incremental departures which are computed as the difference between the present and previous machine points e.g.,

$$\Delta X = DPRESM(1) - DPREVM(1).$$

The departures are now checked to see if any one of them exceeds the maximum allowable departure (option 4). If any departure is too great, subroutine SEGMNT is called to segment the programmed path into sufficiently small segments, such that each axis departure of each segment is less than or equal to the maximum departure.

When the departures are acceptable and computed through subroutine DEPART, the incremental values of each axis departure are stored in DBFSEG, i.e., ΔX is stored in DBFSEG(3), ΔY in DBFSEG(4), and for non-lathes, ΔZ in DBFSEG(5).

Two key flags are now tested to see if the program flow should be rerouted for special items. The flag SFMFLG is checked; and if non-zero, program flow is diverted to subroutine SFMO which produces and outputs a series of segments based upon the required spindle speed variations to produce the desired SFM effect. (See Section 4.5.)

The other key flag is the threading flag THFLAG which, if non-zero, calls in the threading sequence THREDO. This subroutine generates its own special output command block. (See Section 4.6.)

2.4.2.2.1 LINEAR INTERPOLATION FLOW (cont'd)

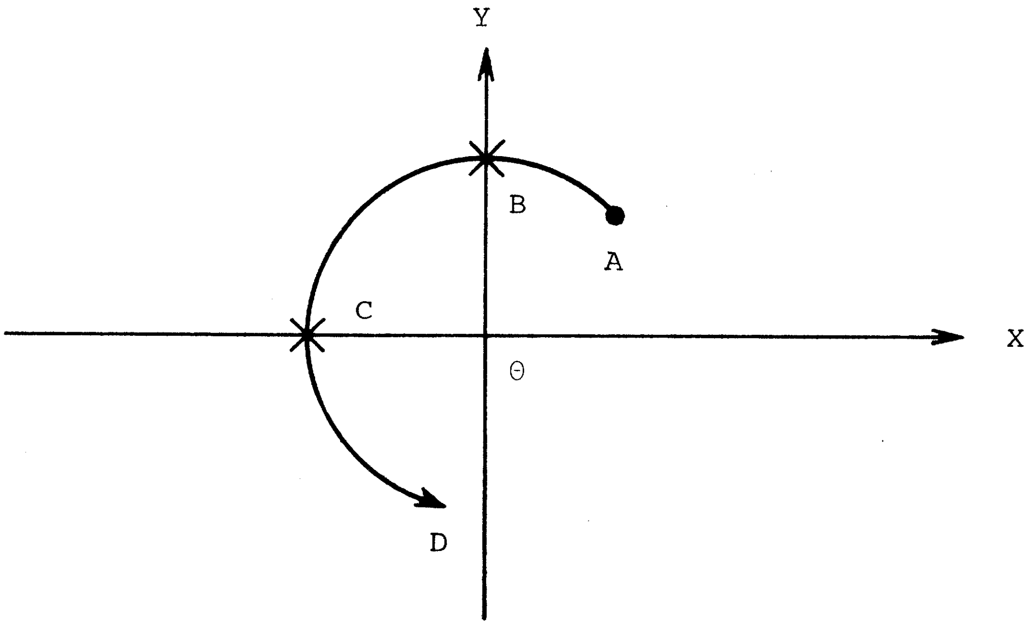
For linear moves the command block identifier CODE is set to zero.

At this point in the program flow for linear moves DBFSEG(3), (4), and (5) are set to their respective ΔX, ΔY, ΔZ values, and CODE=0. Subroutine OUTPUT is then called to complete the setup and eventual output of DBFSEG. This description continues in Section 2.4.2.2.3.

2.4.2.2.2 CIRCULAR INTERPOLATION FLOW

In subroutine MOTION, the CL data points had been stored in the part coordinate present point vector DPRESP. For a circular interpolation move (CIRFLG≠0), program flow continues in subroutine GOCIRC. The procedures for circular interpolation are discussed in detail in Section 3.4.4 and should be referred to for complete understanding. But, in brief, what takes place is that the circle data are reduced to their axes interception points which are stored in an array called DBUFER. Each point is in turn taken from DBUFER and individually processed to produce the incremental departures.

For example, assume the circle when plotted with its center at the origin looks like:



The circular interpolation sequence uses the CL data to determine the circle direction (CCLW), the plane of the circle (XY), the quadrants covered by the circle (quadrants I, II, III), and the axes interception points (B,C). The coordinate values of the points B, C, and D are stored into DBUFER.

2.4.2.2.2 CIRCULAR INTERPOLATION FLOW (cont'd)

Subroutine PROCQD selects each point from DBUFER and stores it into the part coordinate present point vector DPRESP. Processing continues exactly as described in Section 2.4.2.2.1 for a linear move except that in addition to computing the departures, the postprocessor also computes the arc center offsets through subroutine OFFARC.

The arc center offset for each axis is the absolute value of the incremental distance between the coordinate value of the circle's center and the coordinate value at the beginning point of the arc, i.e., (Arc Center Offset) = $|C_x$ - DPREVP(1)$|$.

The arc center offsets are stored in DBFSEG(8), (9), and (10) and correspond respectively to the X, Y, and Z registers for DBFSEG(3), (4), and (5).

For circular moves the command block identifier code is:

+10 for CLW in the XY plane

-10 for CCLW in the XY plane

+11 for CLW in the ZX plane

-11 for CCLW in the ZX plane

+12 for CLW in the YZ plane

-12 for CCLW in the YZ plane

At this point in the program flow for circular interpolation moves, DBFSEG(3), (4), and (5) are set to their respective ΔX, ΔY, ΔZ values; DBFSEG(8), (9), (10) are set to their arc center offset values, and CODE =±10, ±11, or ±12. Subroutine OUTPUT is then called to complete the setup and eventual output of DBFSEG.

2.4.2.2.3 OUTPUT OF AN INCREMENTAL MOVE

Subroutine OUTPUT completes the setup of the command block DBFSEG and prepares it for eventual output. The CL tape record number is added to DBFSEG(1) and CODE is stored in DBFSEG(15).

The feedrate in IPM is next added to DBFSEG(11). If a rapid traverse is in mode (FRAPID ≠ 0), the rapid feedrate is used, otherwise the current feedrate FEDIPM is used.

2.4.2.2.3 OUTPUT OF AN INCREMENTAL MOVE (cont'd)

All incremental moves require a dimension preparatory function G code which is selected according to the increment size (See Section 3.4.5.) Subroutine OUTPUT calls SELG to perform this function. For linear moves, this subroutine obtains a G code which is compatible with the path length of each of the linear slide motions. If the move is a circular interpolation move, then the subroutine obtains a G code which is compatible with the circle's radius and direction of arc (CLW or CCLW). After subroutine SELG obtains the proper G code, it stores it in DBFSEG(2).

A check is made to see if axis feedrate limitations vary on each axis; if so, subroutine FEDLIM is called to check and modify the feedrate accordingly. (See Section 4.1.5.2 for the complete description of this technique.) The feedrate, if modified, is again left in DBFSEG(11).

The command block DBFSEG is completed by adding the current spindle command, SPNCOM to DBFSEG(12), and the pending auxiliary function M code, VALUEM to DBFSEG(14). SPNCOM and VALUEM are DMBITS if there is no pending value. GEOUT is then called to produce the printed and punched output.

Regardless of which printed output sequence is used, the basic output for any incremental move always follows the flow sequence described below.

The first act performed for DBFSEG is to reorder it per the requested settings of options 59 and 60. Subroutine SHUFFL switches cell locations and makes the necessary modifications. For example, for a standard lathe option 59 is set so as to have the X value become the Z value, and the Y value to be a negative X value. Hence, if DBFSEG is set as

N	G	X	Y	Z		
100	1	8.2	4.8	DMBITS		

after subroutine SHUFFL it appears as

N	G	X	Y	Z		
100	1	-4.8	DMBITS	8.2		

2.4.2.2.3 OUTPUT OF AN INCREMENTAL MOVE (cont'd)

The shuffling effect is dependant upon the ISHVEC vector established in subroutine DECODE in GEINIT. (See Section 3.5).

A test is next made of option 143 to determine whether the sequence number should remain as the CL tape record number or to make it a unit increasing number. If the latter choice is indicated (option 143 ≠ 0), the unit number SEQNEW is stored in DBFSEG(1).

Subroutine CONTUR is now called to convert the feedrate in IPM to its feedrate command form. This conversion is done for either a linear or a circular interpolation move for any one of the three available contouring feedrate command formats (option 10). Furthermore, optimization of the feedrate by use of a multiplying constant in registers I, J, and K is done; or if the feedrate is for a rapid traverse, slide feedrates are used to maximize the rapid feedrate vector. (See Section 4.0 for a detailed description of each of these items.)

Subroutine FVARGO is called if the machine tool has a F command format which varies as a function of the preparatory G code. (See Section 4.1.1.2)

Further tests are made to ensure that the feedrate is not tape reader limited, and that the feed command is within the minimum and maximum allowable feed command range.

At the completion of all these tests, the feedrate in IPM is redetermined from the derived feedrate command number in the event that the feed command was not directly converted from the original value of feedrate in IPM. The derived feed command is stored into DBFSEG(11), and the corresponding feedrate in IPM is saved for eventual printing in the Absolute Printout.

Before exiting from subroutine CONTUR, the cut time for the move is computed and saved.

Continuing in GEOUT, redundancies of G, F, and S are suppressed as requested by option settings.

At this point in the program the command block is fully prepared to output, and is subsequently printed and punched per Section 3.5.

2.4.3 THREE-AXIS MILLING MACHINES

The following structure illustrates a typical milling overlay configuration. Note that any of the GEOUT's may be used for output purposes.

A spindle type is also implied though not specified in the configuration.

Section 0
GEMON
GEBASE
GETERP
GEMILL
GEOUT

2.4.3.1 MILLING MACHINE CHARACTERISTICS

To date, all milling machines employ incremental contouring systems.

The programmed cutter path as presented on the CL tape is converted from its absolute coordinate form into one or more incremental segments whose summation (disregarding a TRANS) regenerates the original set of absolute data points within the step size tolerance of the machine tool. For example, the path from absolute X, Y coordinates (2,6) to (6,4) produces increments $\Delta X=4$, $\Delta Y=-2$. (See Section 3.4.3 for a complete description of the methods used for producing incremental moves.)

Any incremental motion may be segmented into yet smaller incremental motions. The path length may be segmented because the original increment may be greater than the maximum allowable departure (option 4). These segments may be further segmented because of A/D consideration, and because of G code optimization (See Section 4.1.5.1). In any event, the summation of all these segments results in the original segment length.

The feedrate command can be any one of the three contouring types. (See option 10 and Section 4.1.1 for a complete description.)

2.4.3.1 MILLING MACHINE CHARACTERISTICS (cont'd)

A tool changer (if any) is considered to be a normal function of a mill, therefore, the related subroutines for effecting a tool change are all located in the GEMILL overlay. Special purpose tool changers, which require a return to a home position or any other special operation, are handled by special subroutines or most generally by a MACFUN. (See Section 5.6.1.)

2.4.3.2 MILLING MACHINE PROGRAM FLOW

The program flow for a non-multiaxis milling machine is identical with the sequence described in Section 2.4.2.2 for a lathe. The only exceptions are the references made to SFM and threading. In the case of a milling machine, a special SFM and thread sequence are used since their operations are different from a lathe.

2.4.4 MULTIAXIS MILLING MACHINES

The overlay structure of the multiaxis configuration is like that of a three-axis mill except that the additional overlay GEMAXS is added. In the diagram, the overlay GECLAS is indicated to emphasize the point that multiaxis processing requires the set of transformation relations for converting from part coordinates to machine coordinates.

The GEOUT referred to in the diagram must be GEOUT2, GEOUT3, or GEOUT4; GEOUT1 cannot be used because of the insufficient number of columns normally available on standard print sheets.

Section 0
GEMON
GEBASE
GETERP
GEMILL
GECLAS
GEMAXS
GEOUT

2.4.4.1 MULTIAXIS MILLING MACHINE CHARACTERISTICS

All multiaxis mills with Mark Century numerical controls utilize an incremental contouring system. The definitive characteristic of any multiaxis machine tool is the fact that the machine tool must have at least one rotary axis in addition to its translatory axes which can all move simultaneously. However, each rotary motion can be separately and independently moved, as can each translatory axis.

The rotary motions (table or head) when made output can be one of several forms of output units. (See option 118 for the available forms.) The translatory slides are always in inches or in millimeters in the metric system.

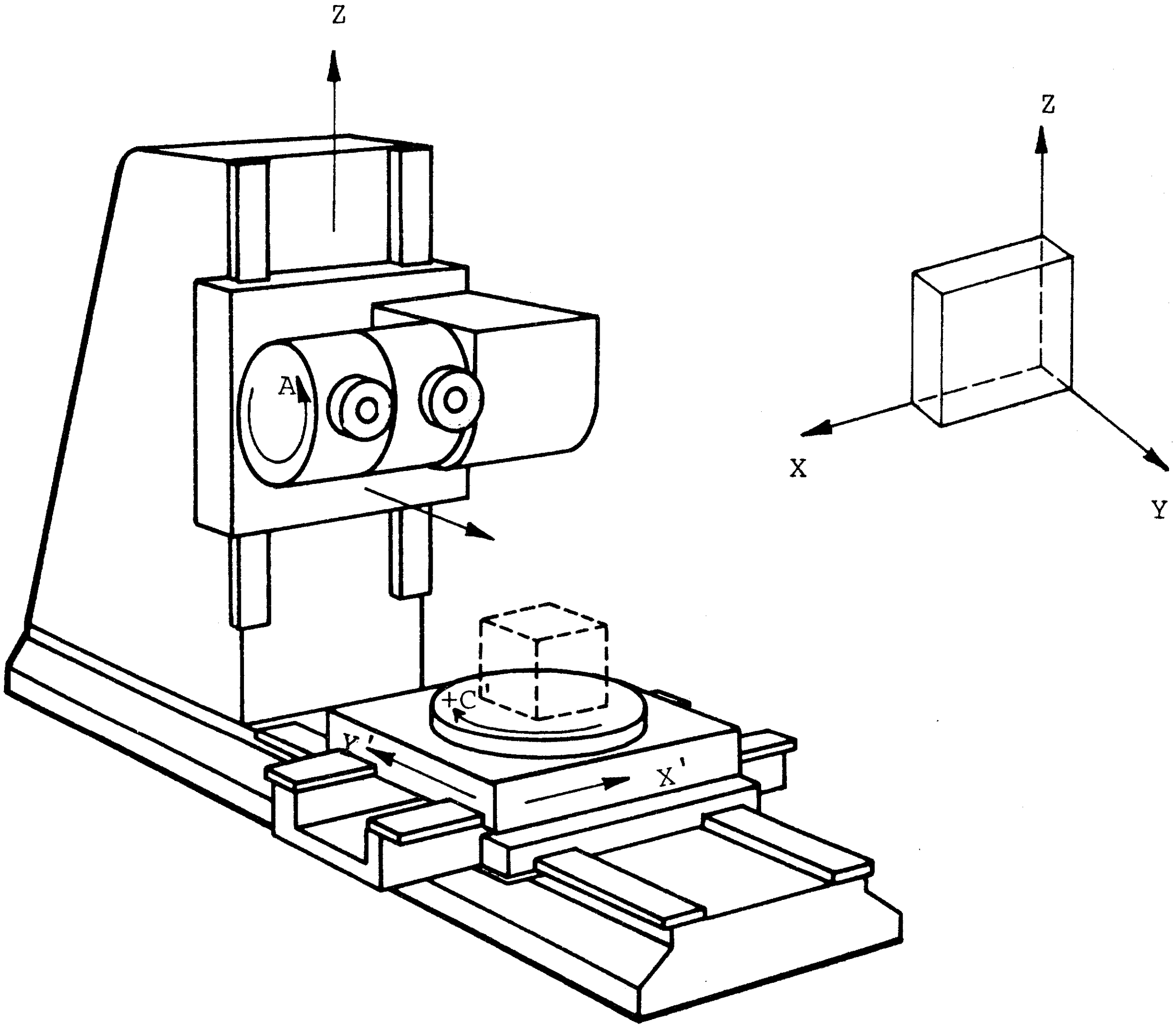


Diagram 2.4.4.1A

(Other classes are defined and further explained in Section 4.2)

2.4.4.1 MULTIAXIS MILLING MACHINE CHARACTERISTICS (cont'd)

The CL data for any motion are presented on the CL tape in absolute part coordinate form. Each point of the path is represented by the algebraic absolute (x,y,z) values accompanied by the direction cosines of the backward-directed vector of the tool. However, in order to function properly within the machine, the part coordinate data must be converted to the machine coordinate system.

The conversion of part coordinate data to machine coordinate data relies upon a set of transformation equations which are unique and dependant upon the axes configuration of each particular machine tool. In the GECENT III postprocessor these transforms are defined by the class in which they appear. For example, machine tools which have the axes configuration which conform to diagram 2.4.4.1A are said to have a Class 1 set of transformations.

A point taken from the CL tape is converted through the class transformation equations to become the corresponding value in machine coordinates. This conversion implicitly interprets the tool axis orientation in terms of rotory motions. Thus the part coordinates and direction cosines (x, y, z, i, j, k) are converted to the machine coordinates (X, Y, Z, A, B, C).

In addition to the segmentation sequences mentioned for 3-axis mills in Section 2.4.3.1, multiaxis processing can also include a segmentation due to the so-called "linearity" effect. Actually, the effect is produced because of the non-linear motion of the tool tip during a combined simultaneous motion of the linear and rotary axes. (See Section 3.4.7.3 for a detailed description of the linearity problem.)

The feedrate command can be any one of the three contouring types. (See option 10 and Section 4.1.1 for a complete description.)

The rotary motions of a multiaxis machine are always of an incremental type and do not position to any given absolute value. The postprocessor gives the part programmer the capability of programming the table as if it were an absolute reference system, but the actual rotary output is always an incremental move.

Rotary motions can be of several possible types, among which are rotary tables, rotary heads, swiveling heads, tilting tables, rotating columns, and so on.

2.4.4.1 MULTIAXIS MILLING MACHINE CHARACTERISTICS (cont'd)

There are some other special purpose sequences for a multiaxis mill, e.g., overcenter cutting, pallet changing, and these are all located in the GEMAXS overlay.

Special purpose tool changers which require a return to a home position or other special operation, are handled by special subroutines or most generally by a MACFUN. (See Section 5.6.1.)

2.4.4.2 MULTIAXIS MILLING MACHINE PROGRAM FLOW

The program flow for a multiaxis mill is essentially the same as for a NON-MULTIAXIS mill; the chief exceptions are that the multiaxis flow must convert the part coordinates to machine coordinates, and the rotary motions must be considered in the determination of incremental departures and feedrate command.

The description of program flow both for linear and circular interpolation is given in Section 2.4.2.2 for a lathe. The flow for a multiaxis machine follows that description except for some additional steps which are given here.

When subroutine GEOM is called, it in turn calls GEOM5 which calls subroutine CLASS, which employs the desired set of transforms for the machine tool class. This is specified in option 116. The part coordinates are transformed to machine coordinates and truncated and rounded to the machine tool step size. Rotary axes truncation and rounding is done by subroutine SROREC.

For example, the CL data are stored in the present point part coordinate vector as,

DPRESP(1) =	2.43682107	(x)
DPRESP(2) =	-13.24680110	(y)
DPRESP(3) =	0.01234567	(z)
DPRESP(4) =	0.00000000	(i)
DPRESP(5) =	0.00000000	(j)
DPRESP(6) =	1.00000000	(k)

2.4.4.2 MULTIAXIS MILL PROGRAM FLOW (cont'd)

Transformed, truncated, and rounded they are stored in the present machine point vector as,

DPRESM(1)=	22.6688	(X)
DPRESM(2)=	1.2468	(Y)
DPRESM(3)=	-5.0471	(Z)
DPRESM(4)=	15.6740	(A)
DPRESM(5)=	0.1234	(B)

Subroutine DEPART calls subroutine ROTMOV in order to compute the rotary departures. An important point to note is that the rotary moves are always kept in terms of their output units rather than in radians. This minimizes the processing time in that no conversion to and from output units is ever required.

Another function performed by subroutine ROTMOV, is that it always makes the absolute position of rotary moves positive and less than 360 degrees. For example absolute location of -400 degrees is made to be 320 degrees. Subroutine ROTMOV puts the rotary departures into DBFSEG(6) and (7). A convention of the postprocessor is that the head register is related to DBFSEG(6), while the table to DBFSEG(7). This is merely a convention and not a set rule.

After checking the linear departures versus the allowable maximum linear departure, similar tests are made with the rotary departures versus the rotary maximum departure. Subroutine SEGMNT is called if any maximum departure is exceeded.

When a segment is acceptable, several flags are tested to determine whether or not linearity testing should be performed. If so, subroutine LINRTY is called upon to produce the requisite number of segments to remove any "linearity" error. (See Section 3.4.7.3 for a detailed discussion of this subject.)

An important feature to be noted here is that when a departure exceeds the maximum departure and linearity testing is desired, subroutine SEGMNT is not immediately called upon to segment the path length to the necessary segments, but, rather, subroutine LINRTY is used since the expectation is that the path length will be sufficiently segmented in order to correct the "linearity" error.

2.4.4.2 MULTIAXIS MILL PROGRAM FLOW (cont'd)

A multiaxis move has motions both in the rotary and linear axes, but the posprocessor treats the move as if it were simply a linear motion; therefore a multiaxis motion command block is still identified by a CODE of zero.

An apparent contradiction can occur in command block identity. Rotary moves by themselves, when generated by a ROTATE statement, have their command blocks identified by CODE = -2. However, it is possible that in a multiaxis motion that ΔX, ΔY and ΔZ are zero, and only ΔA or ΔB are non-zero. Yet the command block CODE is still zero. This actually leads to no problem, and it is important that the command block generation source be known. The CODE uniquely identifies the source.

At this point in the program flow for linear multiaxis moves, DBFSEG(3), (4), (5), (6), and (7) are set to their respective ΔX,ΔY,ΔZ,ΔA, ΔB values, and CODE =0. Subroutine OUTPUT is then called to complete the setup and eventual output of DBFSEG is as described in Section 2.4.2.2.

Circular interpolation for multiaxis machines require an analogous determination of the equivalent of rotary axes "arc center offsets". These are normally the registers D(corresponding to the rotary A register) and E(corresponding to the rotary B register). These are not actually arc center offsets and are referred to as supplementary constants.

After subroutine GOCIRC has determined the axes interception points and stored them in the array BUFFER, subroutine PROCQD proceeds to process and output the points. For two- or three-axis machines, each interception point is merely the (x,y,z) coordinate value; but for multiaxis processing the tool axis vector direction cosines must be known. Therefore, subroutine PROCQD must determine the (i,j,k) values at each interception point before processing and outputting the point. (See Section 3.4.7.1 for the complete description of this technique.)

In addition to generating the tool axis direction cosines, subroutine PROCQD also outputs an information block. (See Section 5.5). This information block carries the angle of arc and circle radius which are information necessary in the determination of the feedrate command for a circular interpolation move.

2.4.4.2 MULTIAXIS MILL PROGRAM FLOW (cont'd)

Multiaxis circular interpolation moves are processed and made output with the same CODE value as for non-multiaxis moves. Hence, at this point in the program flow, DBFSEG(3), (4), (5), (6), and (7) are set to their respective ΔX, ΔY, ΔZ, ΔA, ΔB values; DBFSEG(8), (9), (10) to their arc center offset values, and DBFSEG(16) and (17) for the rotary supplementary constants D and E. The value of CODE is ±10, ±11, or ±12. Subroutine OUTPUT is then called to complete the setup and eventual output of DBFSEG.

When subroutine OUTPUT calls SELG to obtain the dimensional preparatory function G code, subroutine SELG first obtains the proper G code compatible with the linear (or circular) moves as described earlier. Then subroutine SELG calls subroutine SELGRO which accepts the already determined G code if it is compatible with the rotary moves; but if not, subroutine SELGRO obtains a G code compatible with both the linear (or circular) and rotary moves.

Output of a linear interpolation multiaxis move is essentially the same as for a non-multiaxis move, the main exception being that the effect of the rotary motions must be considered in the calculation of the feedrate command. In subroutine CONTUR, where this calculation is done, the postprocessor uses the part coordinate path length rather than attempting to find the machine coordinate space curve.

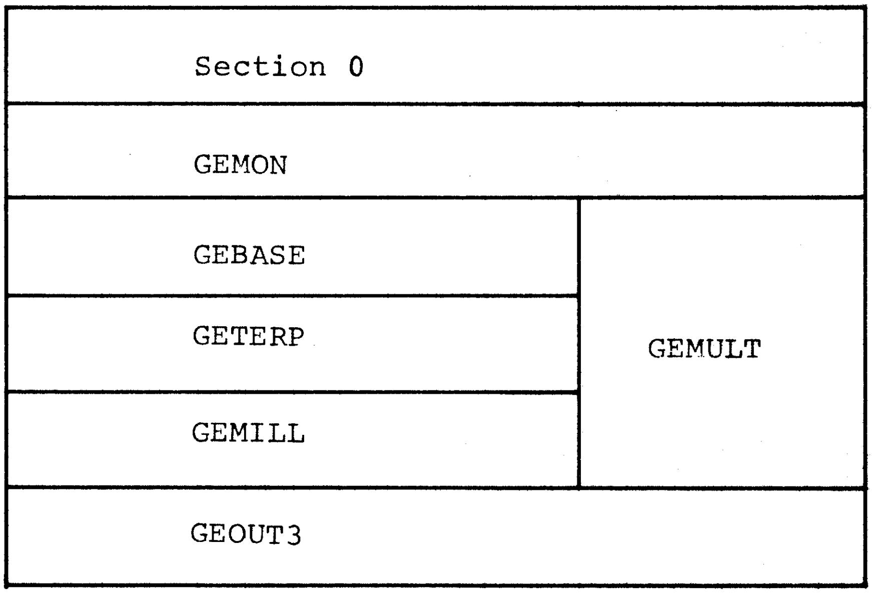
A circular interpolation multiaxis move computes a feedrate command from a different formula than does a non-multiaxis move, but otherwise all program flow is identical. (See Section 4.1.1.)

Processing of a multiaxis move requires no other special sequences in any of the permissible GEOUT's. In rather routine steps, the rotary motions are converted to an absolute location in degrees for printing in the Absolute Printout. The influence of the rotary motions is considered in other determinations such as the cut time, block read time, feedrate optimization, and so on, but these sequences, in effect, deal with all departures in a standard routine manner. There is no special branching for multiaxis processing.

At this point in the program, the command block is fully prepared for output and is subsequently printed and punched per Section 3.5.

2.4.5 MULTIHEAD MACHINES

The general overlay structure for a multihead configuration is illustrated below for a mill, but such a structure also applies to a positioning machine, lathe, multiaxis mill, or any other available machine type.



A spindle type is also implied in the above structure.

The key overlay for all multihead machines is GEMULT which is the main sequence for all multihead processing. Inherently, all multihead processing is a two-pass system; the first pass processes the CL tape for both heads individually, and the second pass merges the data for combined motion and output. In the overlay diagram, the second-pass GEMULT overlay replaces the first-pass overlays GEBASE, GETERP, and GEMILL.

Because of the large number of registers normally found on multiaxis machines, it is mandatory to use GEOUT3 as the output element. However, any one or combination of the three printouts can be obtained through option 17.

2.4.5.1 MULTIHEAD MACHINE CHARACTERISTICS

Generally speaking, most multihead machines are incremental contouring systems, but it is quite possible to have multihead absolute positioning systems as well. The chief characteristic of all multihead machines is that the NC machine must have more than one cutting head which can act simultaneously and independently of the other head(s). Machines with multiple heads which act in tandem, or are slave heads, or are mirror image operators, are not considered to be multihead machines. To be considered multihead, each head on the machine must be separately programmable so that single head operation or combined multihead simultaneous operation is obtainable.

There are other distinguishing features that a multihead machine may have, and these most commonly pertain to the heads. For example, depending upon the type of control system furnished, each head may or may not have its own feedrate register. This is a very important feature because a different type of merging process is used for each condition when simultaneous head operation is in effect.

An example will clarify the two methods. In diagram 2.4.5.1A is illustrated a simultaneous cut path for two heads.

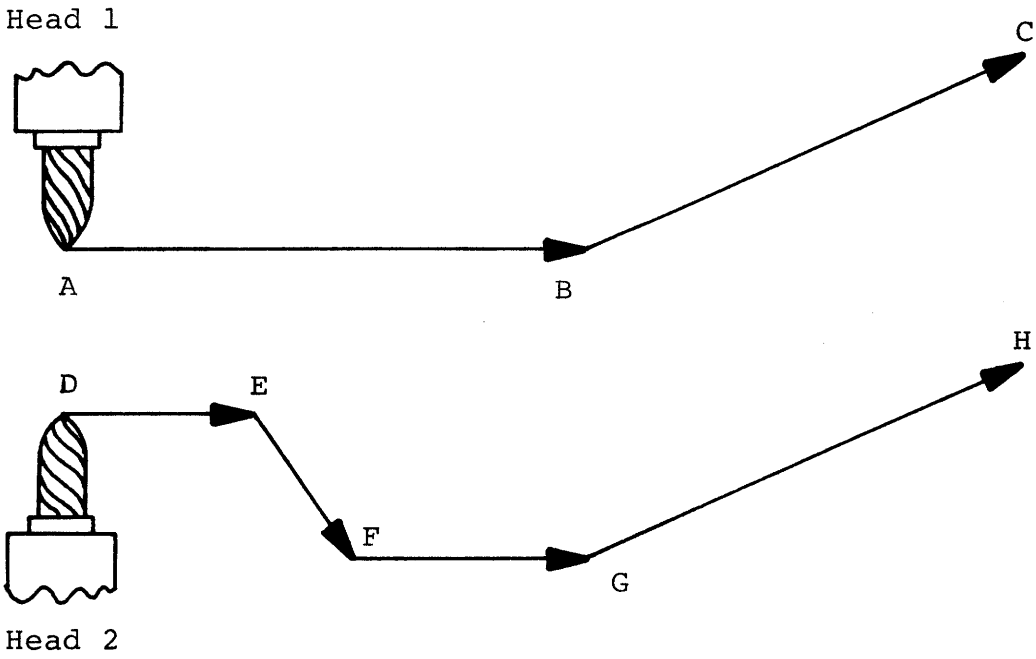


Diagram 2.4.5.1A

2.4.5.1 MULTIHEAD MACHINE CHARACTERISTICS (cont'd)

If the NC machine has only one feedrate register, the paths are segmented for equal times and may appear as shown in Diagram 2.4.5.1B, for when both heads share a common feedrate register, the merging process segments each head's cutter path so that identical times are produced for each head. (See Section 3.4.8 for a detailed description of the multihead merging technique.)

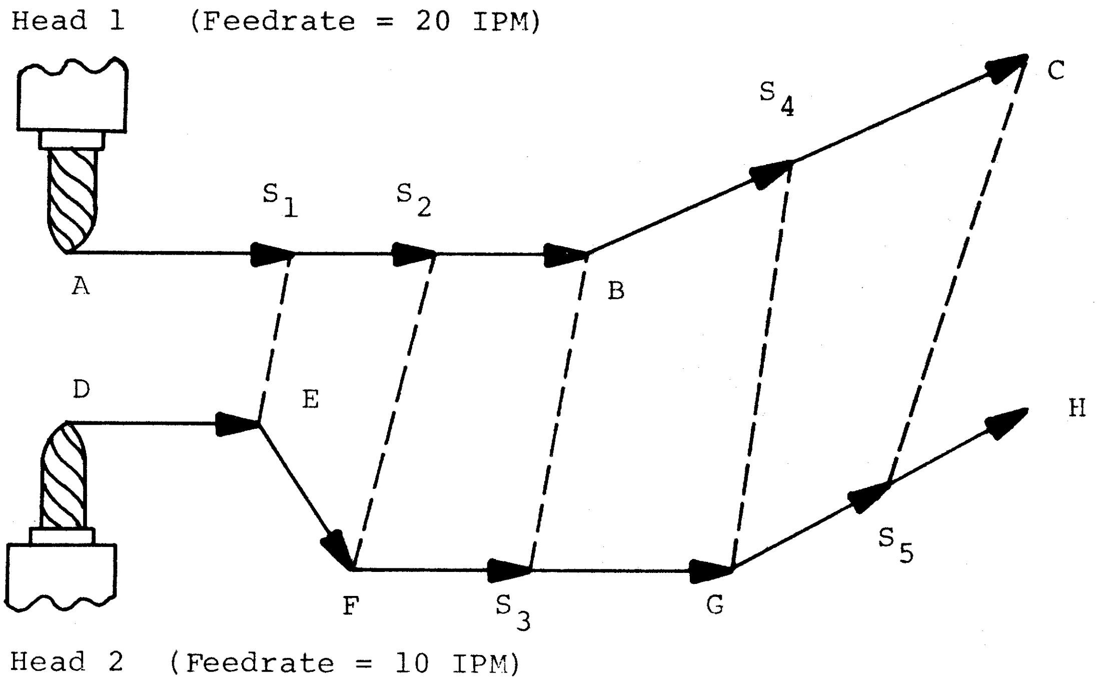


Diagram 2.4.5.1B

Path AB is segmented at S_1 so that the segments AS_1-DE are output as a merged block, as are also the segments S_1S_2-EF, S_2B-FS_3, BS_4 -S_3 G, and S_4C-GS_5. Head 1 will park at point C and wait until Head 2 completes the segment S_5H.

The location of the generated segment points S_1, S_2, and so on, are a function of the feedrates programmed for Head 1 and Head 2. If the feedrates are the same for both heads, then the points of segmentation are always laterally coincident with the other head's path end. This is illustrated in Diagram 2.4.5.1C.

2.4.5.1 MULTIHEAD MACHINE CHARACTERISTICS (cont'd)

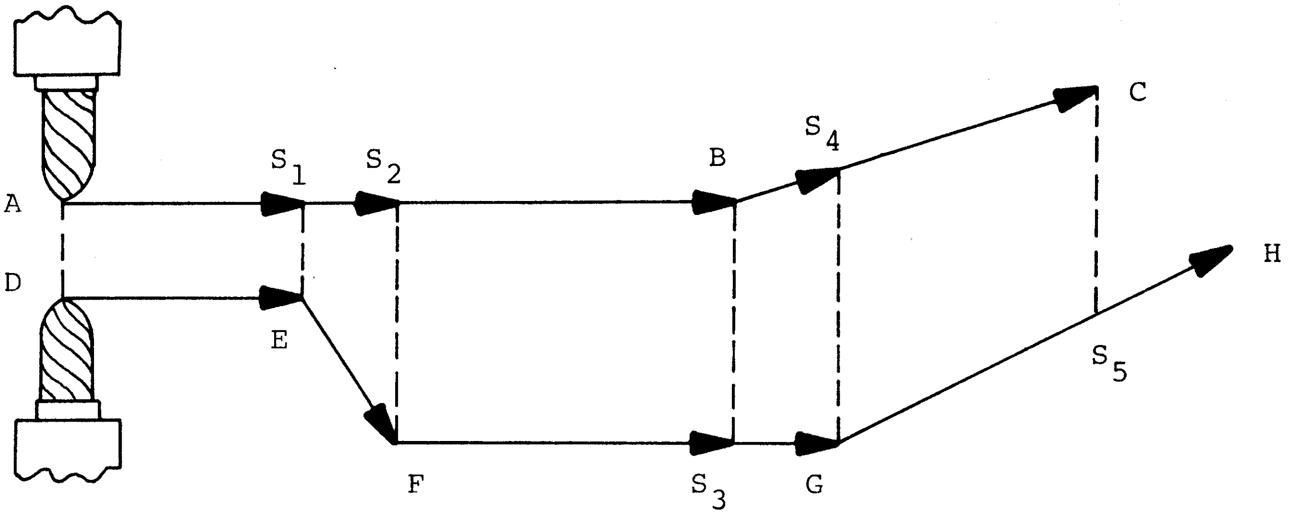
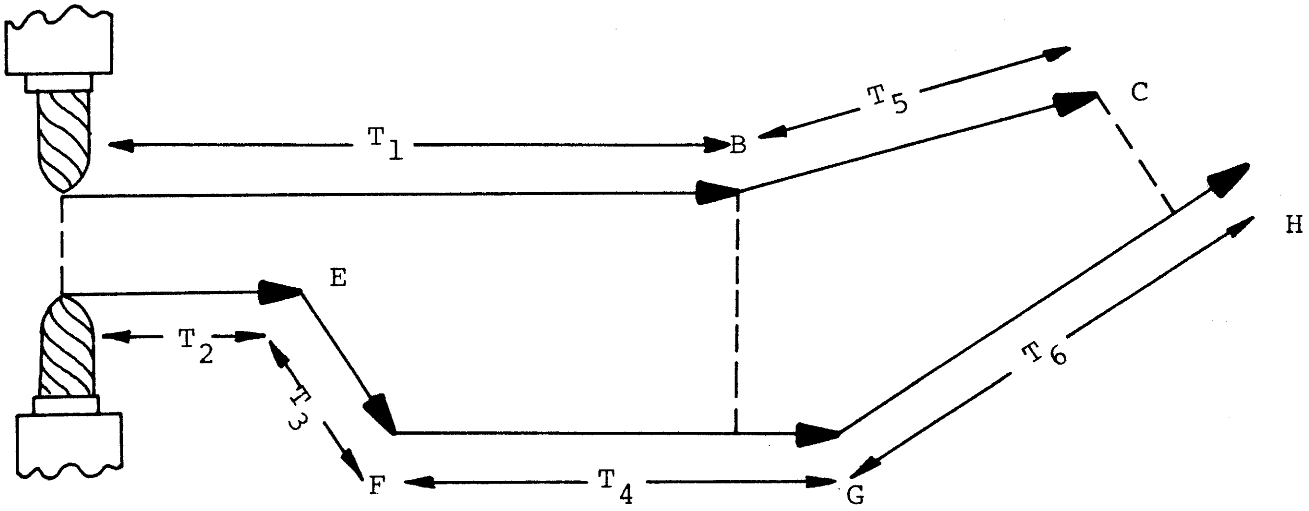


Diagram 2.4.5.1C

(See Section 3.4.8.2.1.1 for a complete description of the single-register technique.)

When each head has its own feedrate register, a different technique is used; now there is no need to segment the paths in order to obtain equal times. Instead, the cutter path for one head is made output separately as long as the other head has a path remnant to complete; thus, whichever head has the longest cut time, the other head will output the shorter (in time) paths until it becomes the head with the longer cut time. This technique will become comprehensible by considering the example as illustrated in Diagram 2.4.5.1D.

Head 1 (Feedrate = 10 IPM)



Head 2 (Feedrate = 10 IPM)

2.4.5.1 MULTIHEAD MACHINE CHARACTERISTICS (cont'd)

Initially, both heads must have their first paths combined: but thereafter, except for a rare circumstance, each head command block is output separately in an uncombined form.

Thus, initially the combined paths AB--DE are made output and the times T_1 and T_2 are computed. Since $T_1 > T_2$, when path DE is completed, Head 2 reads in path EF and computes T_3.

Since $T_1 > (T_2 + T_3)$, when path EF is completed, Head 2 reads in path FG and computes T_4.

Since $(T_4 + T_2 + T_3) > T_1$, when path AB is completed, Head 1 reads in path BC and computes T_5.

Since $(T_1 + T_5) > (T_2 + T_3 + T_4)$, when path FG is completed, Head 2 reads in path GH and computes T_6, and so on.

Note that except for the initial start-up, there has been no merging of command blocks; the only time when blocks must be merged is when the summated times for each head are equal, i.e.,

$$\underset{\text{Head 1}}{\left[\sum_{i=1}^{n} T_i\right]} = \underset{\text{Head 2}}{\left[\sum_{j=1}^{n} T_j\right]}$$

for when this condition occurs, both heads are at the same relative point as when initially starting.

Some NC machines may have circular interpolation in addition to the regular linear interpolation. Thus, it is possible to have mixed or the same interpolation modes on each head, e.g., linear on Head 1 and circular on Head 2, and so on. It must be understood, however, that these mixing capabilities are not always possible on every NC machine control combination: therefore, the fact that a multihead NC machine has both interpolation modes available does not necessarily mean that any combination can be applied to the heads.

Multihead machines often require a sharing of some item which is common to all heads. For example, the spindle speed on a multihead lathe or the feedrate register on single register controller. During the course of a part program, conditions may occur which, in effect, produce different values of the same item for each head. On a two headed lathe during a SFM mode, one head can easily generate a spindle speed which is completely

2.4.5.1 MULTIHEAD MACHINE CHARACTERISTICS (cont'd)

different from the other head. The question arises: which is the valid spindle speed? The answer is that both speeds are valid, but one speed is more important than the other; and only the part programmer knows which head has the highest priority. The postprocessor language available permits the part programmer to designate which head is the priority head, and the postprocessor accordingly selects from multiple-choice items.

Many conditions can arise during a multihead cutting sequence wherein one head stops cutting because it had completed its operation before the other head or for some other reason must stop cutting. When such a circumstance occurs, the non-operating head is usually parked, i.e., withdrawn from the workpiece and left to dwell until simultaneous or single-head operation can be continued. Parking frequently occurs when the NC machine has axis common to both heads.

After a head is parked and is to be brought back into operation, the postprocessor requires that the other head be first parked, and then both heads be brought into operation simultaneously. Synchronous motion is thereby achieved.

When a multihead machine has an axis common to all heads, it is necessary to ensure that motions along the common axis are identical for all heads. This requires a segmentation sequence which generates segments based upon equal value lengths of the common axis. (See Section 3.4.8.2.1.3 for a detailed description of this method.)

Diagram 2.4.5.1E illustrates the problem that exists when both heads share a common axis, the X axis in this example. For Head 1 and Head 2 to cut simultaneously along the X axis, the incremental motion in X must be identical for each head; futhermore, the X-axis component feedrate for each head must also either be equal or within some allowable tolerance.

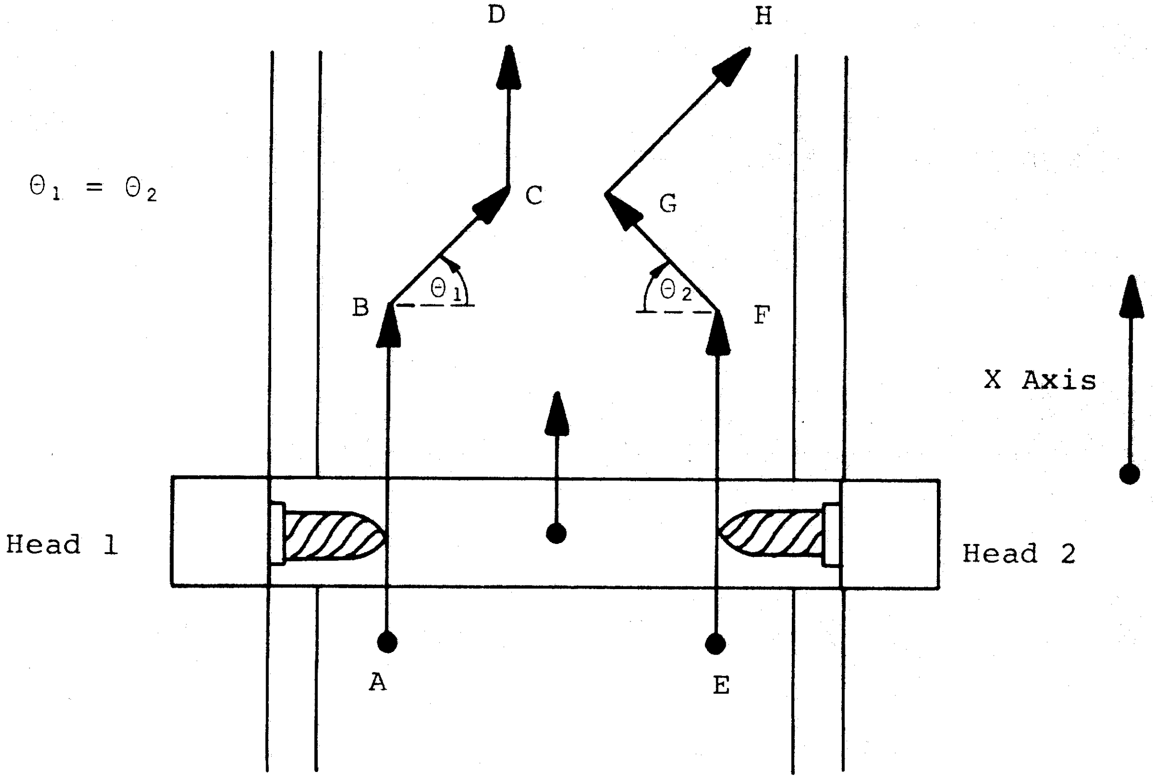
The conditions for merging paths under these requirements can normally be met when the head paths are bilaterally symmetrical in the common axis, or axially symmetrical in all axes, or when both paths are identical. Thus, in Diagram 2.4.5.1E, paths ABC and EFG could most likely be merged with ease, but greater difficulty would be realized with merging paths CD and GH.

Multihead processing through the first pass is completely standard except that special multihead considerations are sometimes used as for multiturrets or rapid traverse M-code output. Otherwise all first pass processing is normal. Motion

2.4.5.1 MULTIHEAD MACHINE CHARACTERISTICS (cont'd)

data, for example, are processed exactly as for single-head operation except that at output time, instead of printing and punching the data, the data is dumped on a scratch device for later processing by the second pass.

Since there are no radically special sequences for first pass processing, all information regarding positioning machines, lathes, mills, and multiaxis machines, can be obtained from the earlier sections, Section 2.4.1 through 2.4.4.2.



2.4.5.2 MULTIHEAD MACHINE PROGRAM FLOW

Postprocessor processing for a multihead machine is essentially no different than for a single-head machine except that a merging of the operations of both heads is done before outputting the command blocks. Regardless of the machine type being processed, CL data is dispatched through the first pass in the normal manner using all of the regular subroutine sequences. However, when subroutine OUTPUT is called to punch and print the command block, the postprocessor instead dumps the block onto an interim scratch tape. (The term "scratch tape" is used although in practice the scratch device may be a disc or a drum.) For head 1 the block is dumped onto TAPES2, and for head 2 it is dumped onto TAPES3.

The entire CL tape is thus processed until the FINI record is encountered, at which time TAPES2 and TAPES3 are rewound, and the second pass overlay GEMULT is pulled into core. Program control is then transferred to GEMULT.

After basic reinitialization is completed, subroutine GEMULT begins through subroutine CREAD to read data from scratch tape TAPES2, and with subroutine GMOUT, outputs each command block until a command block with a CODE = ±17 is found. This block contains the OP/n information, where n is the operation number. (See the Part Programmer's Manual for information on the OP/n statement.)

Once the CODE = ±17 block is found, the postprocessor then begins to read data from scratch tape TAPES3, and outputs each command block until once again a command block with a CODE = ±17 is found. The n value of the OP block from TAPES2 and TAPES3 are compared for equality, and if found equal, flags are set so that subsequent blocks read from TAPES2 and TAPES3 are merged. Merging of blocks continue until another command block with a CODE = ±17 is encountered, at which time merging halts until two identical OP values are again found.

2.4.5.2 MULTIAXIS MACHINE PROGRAM FLOW (cont'd)

An example will clarify the operational technique of output.

TAPES2	TAPES3
PARTNO	PARTNO
FROM	FROM
Motion A	OP/2
OP/1	Motion D
Motion B	OP/3
Motion C	Motion E2
OP/3	Motion F2
Motion E1	Motion G
Motion F1	OP/5
OP/4	Motion I
Motion H	OP/6
OP/6	Motion J2
Motion J1	OP/8
OP/7	END
END	

In the above example a simplified case is illustrated wherein TAPES2 and TAPES3 carry the first-pass dump command blocks in symbolic form. In actual fact each record is a DBFSEG command block of the form:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	30
N	G	X	Y	Z	A	B	I	J	K	F	S	T	M	CODE	

2.4.5.2 MULTIHEAD MACHINE PROGRAM FLOW (cont'd)

For the above example, GEMULT would process the tapes as follows:

A. Beginning with TAPES2, output the PARTNO, FROM point, and Motion A; when OP/1 is detected, transfer processing to TAPES3.

B. From TAPES3, output the PARTNO and the FROM point; when OP/2 is detected, compare the opcodes of TAPES2 and TAPES3, i.e., 1 versus 2.

C. Since the opcodes are unequal, and since the TAPES2 opcode is less than the TAPES3 opcode, transfer processing to TAPES2.

D. Output Motion B and Motion C; when OP/3 is detected, compare the opcodes of TAPES2 and TAPES3, i.e., 3 versus 2.

F. Since the opcodes are unequal, and since TAPES3 opcode is less than the TAPES2 opcode, transfer processing to TAPES3.

G. Output Motion D; compare OP/3 versus TAPES2 opcode.

H. The opcodes are equal, therefore, combine all following blocks on each tape. Combine Motion E1 and Motion E2, and output. Combine Motion F1 and Motion F2, and output.

I. When OP/4 of TAPES2 is detected, compare versus opcode of TAPES3, i.e., 4 versus 3.

J. Since the opcodes are unequal, and since the TAPES3 opcode is less than TAPES2 opcode, processing is transferred to TAPES3.

K. Output Motion G; compare OP/5.

L. Output Motion H; compare OP/6.

M. Output Motion I; compare OP/6.

N. Combine Motion J1 and Motion J2.

O. Process TAPES2 END

P. Process TAPES3 END.

2.4.5.2 MULTIHEAD MACHINE PROGRAM FLOW (cont'd)

When outputting a DBFSEG block, whether or not it is a mergeable block, the data of DBFSEG is stored into the array GMHBUF, which is dimensioned at 60 and is completely analogous to the order of DBFSEG.

Since DBFSEG is dimensioned at 20, GMHBUF is doubled to allow for a merger of two DBFSEG blocks. A DBFSEG row for Head 1 is stored into GMHBUF(1) through (20), while Head 2 stores in GMHBUF(21) through (40).

When a command block from TAPES2 or TAPES3 is not merged, it is made output through GEOUT3 as if it were a single-head operation. But when two blocks are merged, a variety of special sequences are first gone through which may generate additional command blocks or otherwise cause modifications to the original command blocks.

For example, the combined blocks could be a merger of two linear interpolation moves, or a combination of linear-circular, or even of two circular interpolation moves. Segmentation of the blocks can result as a function of the equal time merger needs of linear moves (subroutine GMLINE) and circular moves (subroutine GMCIRL).

Then, also, segmentation may result if the NC machine has only one feedrate register; or may result if the multiheads share a common axis (subroutine FXMULT). Additional blocks can also be generated by automatic parking sequences (subroutine FXPARK) or safety retracts. (The details of these sequences are discussed in Section 3.4.8.)

When a command block is sent to subroutine GMOUT for output, the command block first has the finishing touches made to it, e.g., the proper G code is selected, and the feed command determined, and other minor functions are performed. But at this point in the program, the command block is fully prepared for output, and is subsequently printed and punched through GEOUT3 as described in Section 3.5.6.

3.0 DETAILED DESCRIPTIONS

This section covers in detail the five major elements of the postprocessor. Each section is complete in itself and constitutes a full reference for that particular element.

The elements are discussed in the order in which they are used by the postprocessor, from input to output. Although most details are given, special sequences are deferred to Section 4.0 for their particular analysis.

Section 2.0 should be read prior to this section if an overall understanding of the postprocessor is sought.

3.1 CONTROL ELEMENT

When the APT System completes its processing of the input part program, it transfers program control to the control element of the postprocessor through APT Section IV DISPAT. The control element of the GECENT III postprocessor is the monitor overlay GEMON. The control element monitors the overall program flow of the postprocessor, directing the flow through the proper overlays and subroutines necessary for the given NC machine. The first function performed by GEMON is to initialize the COMMON areas and key parameters of the postprocessor. Initialization is done in a separate overlay GEINIT which is overlayed after it has completed its purpose; it is never called again.

After initialization, GEMON directs the construction of the overlay structure, putting together the requisite modules of the postprocessor as needed for the given NC machine. This structuring is accomplished by one of two methods which are dependant upon the computer used.

The details of initialization and structuring are given below in the following subsections.

GEMON remains in core along with APT Section 0 and Section IV and is never overlayed. When GEMON completes its primary functions of initialization and structuring, it transfers control until a FINI statement is encountered and processed. If multihead processing is in use, GEMON next gives control to GEMULT which retains control until a FINI statement. In either event GEMON terminates postprocessor operation by returning program control to APT Section IV DISPAT.

3.1.1 POSTPROCESSOR INITIALIZATION (GEINIT)

The COMMON parameters and arrays are all set to zero since this value is considered to be the initial condition for all flags, counters, and other parameters in COMMON.

Next, the postprocessor defines key parameters which are extensively used throughout the program. The integer values of 0 through 7 are defined as INTZ, INT1, INT2, etc. Similarly, floating point numbers are defined as FLZ, FL1, FL2, and so on up to FL5, and including FL10, FL100, FL360, and FLM1 for -1. These numbers are defined and located in COMMON in order to minimize the core size of a compiled subroutine which would otherwise define these numbers as a local variable and also to eliminate computer dependencies as much as possible. For example, on some computers an integer zero has a different bit configuration than does a floating point zero and so a separate representation for both is essential.

Very frequently it becomes necessary for the postprocessor to test for a null condition. The parameter DMBITS and DPBITS are used to designate a null condition since zero is very commonly a non-null value. The parameter DMBITS is defined as -40404040.0 and DPBITS = -DMBITS for the positive counterpart. Thus, when a flag or some parameter is tested and found to be DMBITS, the postprocessor knows that no condition has been established, and that a null condition exists. Basic reference tables, like TABLEG and TABLEM, are initially set to a null condition, i.e., set to DMBITS.

The subroutine STDMAC is called to establish the standard settings of all the option values in the OPTAB table, as well as for the REGSTR, REGFOR, and SRTAB tables.

GEINIT reads the CL tape in order to find the MACHIN statement so that the designated Machine Subroutine can have its NC machine data loaded into the proper tables. Once the MACHIN statement is found (Record Type 2000, Subtype 1015), it checks to see if the GECENT III postprocessor is called, and if so, to next check for the called Machine Subroutine. See Section 3.1.1.1 for the technique of obtaining the proper postprocessor and Machine Subroutine.

3.1.1 POSTPROCESSOR INITIALIZATION (GEINIT) (cont'd)

The called for machine number is saved in the current machine flag CURMAC which is then used to call the related Machine Subroutine. For example, the statement

MACHIN/GECENT, 40

sets up a call to subroutine MACH40. Once the Machine Subroutine sets up the NC machine characteristrics in the various tables, it's function is complete and it is overlayed as are all the subroutines in GEINIT. However, the MACFUN portion, if one exists, is not overlayed. (See Section 5.6.1.)

The final initializing function of GEINIT is done by subroutine ASSIGN which inspects the established values of the characteristics tables, and sets key flags accordingly. For example, the departure limit table RNGDEP is set up for either a metric or an inch system as designated by option 138.

3.1.1.1 SELECTING THE POSTPROCESSOR AND MACHINE SUBROUTINE

The postprocessor verifies it's own selection and obtains the proper Machine Subroutine by referring the APOSTP* table which exists in APT System COMMON and is set up in APT Section 0. If two MACHIN statements are given in a part program as:

MACHIN/BRACK, 10

MACHIN/GECENT, 10

then the processing sequence indicated is that the BRACK postprocessor is the first to be used to process the CL tape. Upon completion, the CL tape is rewound and the GECENT III postprocessor is next used with Machine Subroutine MACH10. The means by which the postprocessor is able to select the proper postprocessor is by use of the IPOSTP table.

* On the IBM System 360, this table is called APOSTP but is equivalent to IPOSTP in the GECENT III postprocessor.

3.1.1.1 SELECTING THE POSTPROCESSOR AND MACHINE SUBROUTINE (cont'd)

Basically, the IPOSTP table contains the BCD names of the postprocessor coupled with the given machine number. Thus in the above example the IPOSTP table would have in sequence:

BRACK	(BCD)
10	(Floating Point)
GECENT	(BCD)
10	(Floating Point)

An index counter, which is maintained by APT Section 0, points to the IPOSTP table location of the postprocessor to be selected. In the above example, the pointer at the beginning of the part program run, points to the location of the BRACK BCD reference; when processing is complete and the CL tape rewound, the pointer points to the location of the GECENT III BCD reference.

The method by which the pointer information is passed to the postprocessor varies by computer as does the stored data in the IPOSTP table. The most common method used by most computers* is to have the pointer stored as the first item in IPOSTP; the second item in the table gives the number of postprocessors yet to be processed, including the current one. Following these first two items are the pairs of postprocessor and machine number for each MACHIN statement. For the above example, the IPOSTP table initially appears as:

```
IPOSTP(1) = 3
IPOSTP(2) = 2
IPOSTP(3) = BRACK
IPOSTP(4) = 10
IPOSTP(5) = GECENT
IPOSTP(6) = 10
```

The IPOSTP table is dimensioned at 20, hence, under this method it is possible to multiple postprocess a given part program for as many as nine different postprocessors or Machine Subroutines.

* IBM 7090/94, GE635, UNIVAC 1107/8, CDC 3600/3800, 6400

3.1.1.1 SELECTING THE POSTPROCESSOR AND MACHINE SUBROUTINE (cont'd)

Another method used by some computers** has the pointer stored in the parameter NUMPTR which is in APT System COMMON. Otherwise, the IPOSTP table is set up similarly.

When the postprocessor encounters a MACHIN statement on the CL tape, it first compares the CL tape BCD name versus IPOSTP (IP) where IP is the pointer. If the comparison is not equal, the MACHIN statement is disregarded; if it is equal, the MACHIN machine number is compared versus IPOSTP (IP + 1), and so on. In this way the postprocessor can always recognize and accept a current MACHIN statement and process it accordingly.

3.1.1.1.1 PTONLY/2 RUN

A PTONLY/2 run is used when a CL tape is already available, and it is therefore necessary only to postprocess the tape. Thus, if a CL tape had been saved from a part program computer run which used the statement MACHIN/GECENT, 4, the saved CL tape can later be processed by the special APT input:

```
PTONLY/2

MACHIN/GECENT, 4

FINI
```

However, frequently the occasion arises when one desires to run the saved CL tape but with a different NC machine, and consequently, different MACHIN statement than that which exists on the CL tape. The postprocessor permits this because it obtains the current machine reference not from the CL tape but from the IPOSTP table. On a PTONLY/2 run the postprocessor does no checking of the postprocessor name or machine number versus the CL tape since it is meaningless to do so.

Therefore, if a CL tape were developed for multiple postprocessing or multiple machine processing, the postprocessor on a PTONLY/2 run will recognize <u>none</u> of the multiple MACHIN statements for <u>reprocessing</u> the CL tape. But, it will recognize <u>all</u> of the MACHIN statements for <u>changes</u> to the option table.

** IBM System 360, RCA Spectra 70

3.2 INPUT ELEMENT

The input to the GECENT III postprocessor is always the CL tape. The sole intent of the input sequence is to obtain the record data from the CL tape, and store all of it into the floating point array, CLDATA, and part of it into the integer or BCD array, ICLDAT. To expedite processing, certain key parameters which serve as counters and flags must also be determined from the input record. Basic among these is the flag NWPR which gives the number of words per record, i.e., the total number of items (either floating, integer, or BCD) in each logical record. The other flags and counters are discussed below.

The means by which a record is read from the CL tape varies by computer and the APT System used. Therefore, in the description which follows, the overall general scheme of input which is common to all computers is discussed first; then in the following sections, the information for each major computer system is given.

3.2.1 GENERAL INPUT FLOW

The CL tape is always read by calling subroutine INPUT, and this subroutine is called from only two overlays: GEINIT and GEBASE. Therefore, the subroutine INPUT resides in GEMON in order to be available to both of these overlays. Subroutine INPUT is a computer-oriented subroutine and, hence, differs for each computer. These differences are detailed below in the sections which follow.

The two input arrays are CLDATA (dimensioned at 246) and ICLDAT. The dimension of ICLDAT varies by computer, but it is always either dimensioned as a separate array of 20 or is equivalenced to CLDATA. The CLDATA array is used for obtaining floating point data while the ICLDAT array is used for accessing integer or BCD data.

Before subroutine INPUT is called, the CLDATA and ICLDAT arrays are always cleared to zero and the indicator flag INDPTS is set to 5. This indicator is used only when the CL record is a motion record, but it is always preset prior to reading a record.

The format of the CL tape may also vary by computer and APT System, and so each factor must be considered separately. Fundamental to them all are certain standard items, such as the identification and basic structure of each record type. (See Section 3.3.) Futhermore, the CL tape is always a buffered tape.

3.2.1 GENERAL INPUT FLOW (cont'd)

The data on the CL tape is in the form of physical records which are comprised of logical records whose maximum size is 245 words. Each logical record derives from an APT part program statement, hence, it can be a motion or non-motion type record. Non-motion records never contain more than twenty items, but motion records can easily reach the maximum size.

The calling sequence of the subroutine which actually reads a tape is usually slightly different for each computer, but they all carry at least the following items in their calling sequence:

CLTAPE - Identifies the CL tape in the computer's APT System.

IRETN - The return variable flag, which in the GECENT III postprocessor has the meanings:

IRETN<0, normal end-of-record read;

IRETN=0, end-of-file is read;

IRETN>0, an error made in reading.

NWPR - the number of words in the record.

CLDATA - the array into which the CL tape record is read.

The calling sequence may also require additional information on blocking factors and directions on how to proceed with reading of a record.

The output from the tape reading subroutine consists of the array CLDATA being filled and the word count NWPR and the condition flag IRETN being set.

Upon return from subroutine INPUT to the calling subroutine, the condition flag IRETN is tested to see if a "good read" occurred. A test for only a negative value of IRETN is made since an EOF is considered an error because an EOF should never occur in a normal sequence. A FINI record always concludes tape read operation.

There are only fourteen types of records carried on the CL tape, each identified as a Type 1000, Type 2000, and so on to Type 14000. Although the postprocessor reads each record, it only processes certain types and disregards the others. The recognized types are discussed in detail in Section 3.3 and Section 3.4. The types may be broadly classified as those for a motion record and those for a non-motion record. The motion record is processed in the Motion Element while the others are processed in the Auxiliary Element.

3.2.1 GENERAL INPUT FLOW (cont'd)

The general format of a CL tape record is as indicated below.

Location	ITEM	TYPE
1	Record Number	Integer
2	Record Type	Integer
3	Record Subtype	Integer
4	Data	Integer,Floating Point,BCD
.	.	.
.	.	.
.	.	.
245	Data	Floating Point

Regardless of the computer used, the first item of a record read from the CL tape is always the CL tape record number. In subroutine GEBASE after IRETN indicates that the record was read correctly, the parameter SEQCTR is set equal to ICLDAT(1) to pick up and save the record number for possible use as a sequence number.

The format is not the same for all record types and is given here in this form only for purposes of clarification. The point to be noted is that a CL tape record can be a mixture of integer numbers, floating point numbers and BCD values. For proper processing the postprocessor must refer to each element of the record with the proper FORTRAN statement; this is why there are two input arrays, CLDATA and ICLDAT. When a floating point number is to be accessed, the CLDATA array is used, as in

```
DPRESP(1)=CLDATA(6).
```

Or, for integer accessing,

```
ITYPE=ICLDAT(2).
```

When conversion from an integer to floating point (or vice versa) is desired, reference is made as in the example below.

```
SEQCTR=ICLDAT(1).
```

3.2.1 GENERAL INPUT FLOW (cont'd)

The important feature to note is that the arrays CLDATA and ICLDAT both contain the same information. Thus, it is possible but completely erroneous to program:

```
DPRESP(1)=CLDATA(2)
```

for now an integer number is stored in a floating point parameter. Hence, it is vital to know the record format so that the proper input array is referenced.

The two input arrays exist in the postprocessor in two different methods, each a function of the computer in use. One method is used for hexadecimal type computers and another method for non-hexadecimal computers.

3.2.1.1 HEXADECIMAL TAPE RECORDS

The COMMON storage has the double-precision array CLDATA dimensioned at 246, and the single-precision array ICLDAT dimensioned at 20. The CL tape record is all in double-precision and, therefore, is stored into CLDATA. From there the first twenty items are obtained and stored in single-precision form into ICLDAT. (See Section 3.2.2).

3.2.1.2 NON-HEXADECIMAL TAPE RECORDS

The COMMON storage has only the array CLDATA which is dimensioned at 246, and ICLDAT is simply equivalenced to it. No form of conversion or restorage is necessary.

3.2.2 INPUT SEQUENCES FOR IBM SYSTEM 360 AND RCA SPECTRA 70 COMPUTERS

The call to read a tape is through subroutine TAPERD which is in APT Section 0. The calling sequence as used by the GECENT III postprocessor is:

```
CALL  TAPERD  (CLTAPE, IRETN, NWPR, 4, ICLDAT(1),

1, ICLDAT(2), 1, ICLDAT(3),1, CLDATA(4), 0)
```

The first three items in the calling sequence have already been described; the fourth item states the number of arrays into which the CL tape record is to be read, i.e., into four arrays. Actually, what is desired is to read the first word into ICLDAT(1), the second word into ICLDAT(2), the third word into ICLDAT(3), and the remainder into CLDATA beginning at CLDATA(4). The couplet (ARRAY,n) specifies that n words are to be read into ARRAY; that is, the given sequence (ICLDAT(1),1), and so on. When n=0, as for (CLDATA(4), 0), the TAPERD subroutine reads into the given array until an end-of-record condition is reached.

Since the first three words are single-precision integers, they are stored in ICLDAT(1), (2), and (3), and the remaining double words are stored starting in CLDATA(4). These double words are then passed to subroutine STORGE where the integers in the array CLDATA are transferred to the single-precision integer array ICLDAT. This conversion and transference is done by taking advantage of the structure of the double and single-precision words.

Floating point double words look like:

```
 1        32  33       64
| XXXXXXXX | XXXXXXXX |
```

where the X's represent hexadecimal digits. A single-precision integer 4 looks like:

```
 1        32  33       64
| XXXXXXXX | XXXXXXX4 |
```

Subroutine STORGE checks the left half of the word, and if zero, knows it is an integer, and therefore, stores only the right half into ICLDAT.

The flags IRETN and NWPR are set according to the conventions described earlier.

3.2.3 INPUT SEQUENCE FOR UNIVAC COMPUTERS

The call to read a tape is through subroutine TAPERD which is in APT Section 0. The calling sequence as used by the GECENT III postprocessor is:

```
CALL TAPERD (CLTAPE, IRETN, NWPR, 1, CLDATA,0)
```

The first three items have already been described; the fourth item states the number of arrays into which the CL tape record is to be read, i.e., into one array, and that being the next item, CLDATA. The final item, zero, directs the TAPERD subroutine to read the record into CLDATA until an end-of-record condition is reached.

The flags IRETN and NWPR are set according to the conventions described earlier.

3.2.4 INPUT SEQUENCE FOR GE 600 SERIES COMPUTERS

The call to read a tape is through subroutine GETNXR which is in APT Section 0. The calling sequence as used by the GECENT III postprocessor is:

```
CALL GETNXR (CLTAPE, IRETN, NREC, NWPR, 1, CLDATA, 0)
```

The parameters CLTAPE, IRETN, and NWPR have already been described. The parameter NREC is the tape record number, and is unused by the GECENT III postprocessor. The next two items designate that one array, CLDATA, is to receive the input record. The final item, 0, directs the GETNXR subroutine to read the record into CLDATA until an end-of-record condition is reached.

The IRETN flag is set differently from regular usage, and so it is reconverted to conform to standard GECENT III postprocessing. Upon return from subroutine GETNXR, IRETN is 0 for a "good read" and is 5 for an end-of-file. IRETN is positive for an error in reading. Therefore, the flag is reset as follows:

IRETN is made -1, if it originally is zero;

IRETN is made 0, if it originally is +5;

IRETN is untouched if it is any positive value since this is already detectable as an error in subroutine GEBASE.

3.2.5 INPUT SEQUENCE FOR CDC COMPUTERS

The call to read a tape is through subroutine TAPERD which is in APT Section 0. The calling sequence as used by the GECENT III postprocessor.

```
CALL TAPERD (CLTAPE, IRETN, NWPR, 1, CLDATA,0,0,0,0,0)
```

The first three items have already been described; the fourth item states the number of arrays into which the CL tape record is to be read, i.e., into one array, and that being the next item, CLDATA. The zeroes following, in effect, directs the TAPERD subroutine to read the record into CLDATA until an end-of-record-condition is reached.

The flags IRETN and NWPR are set according to the conventions described earlier.

3.3 AUXILIARY ELEMENT

As was discussed in Section 3.2 the Auxiliary Element of the postprocessor is concerned with the disposition of all acceptable record types. By acceptable record types it is meant those CL tape records which are passed on from APT Sections I, II, or III, by the CL tape to APT Section IV and are, therefore, data of possible use to the postprocessor. The acceptable records are Types 1000, 2000, 3000, 5000, 6000, 9000, and 14000. All of these record types are processed in the Auxiliary Element except Type 5000 which is for motion records and is processed in the Motion Element. The other remaining record types are all handled in only two subroutines: Type 2000 records are processed in subroutine AUXLRY and the other types in subroutine GEBASE. Each of these record types is discussed in the following sections which give their purpose, format structure, and use by the GECENT III postprocessor.

3.3.1 RECORD TYPE 1000 - BCD PART PROGRAM STATEMENT

The original part program statement in BCD form is given in this record. It serves no useful function except that it can provide a means for identifying the statement currently being processed if such information is desirable. The format structure is:

(1) Record Number

(2) Record Type = 1000

(3) BCD identifier used only in APT Section II.

(4) First BCD word of statement.

(5) Second BCD word of statement.

and so on for (NWPR-3) words.

This record type is disregarded in the GECENT III postprocessor unless an error occurs. In this case the error dump sequence backspaces the CL tape one record to obtain the Type 1000 record and prints it to identify the source statement which induced the error.

3.3.2 RECORD TYPE 2000 - POSTPROCESSOR STATEMENTS

Part program statements, such as SPINDL, FEDRAT, COOLNT, are passed on the CL tape as Type 2000 records. With the exception of motion records this record type is the most common record processed.

The format structure is fixed only in the first three items of the record, but beyond this point the format structure can be of varying record length and word kind, i.e., either integer, floating point, or BCD words. Since it is virtually impossible to give all the possible format structures, a generalized description is given which illustrates the manner in which APT Section I passes on this type of record.

Each postprocessor statement consists of a major word and either several, one, or no minor words (or modifiers). Modifiers to the right of the slash are called minor words. Statements which have no minor words, do not require a slash (/) and are called basic statements. Those statements which have minor words must have a slash (/) immediately following the major word; such statements are called "common". In the proposed APT_IV language structure certain of these statements are categorized as generic or replacement, but for this discussion, all statements without a slash (/) are defined as common.

3.3.2 RECORD TYPE 2000 - POSTPROCESSOR STATEMENTS (cont'd)

There are two kinds of basic words: (1) those which stand alone as one major word, as END, RAPID; and (2) those which have BCD information strung out after the major word, as PARTNO, PPRINT. Basic words of the first kind have numeric subtype codes always less than 1000. The record subtype identifies the particular basic word being processed. The basic words (and their respective subtype numeric code) which are recognized by the GECENT III postprocessor are the following:

Basic Word	Numeric Code
END	1
STOP	2
OPSTOP	3
RAPID	5
SWITCH	6
RETRCT	7
DRESS	8
PICKUP	9
UNLOAD	10
GOHOME	14
RESET	15
BREAK	17
PPRINT	1044
PARTNO	1045
INSERT	1046

3.3.2 RECORD TYPE 2000 - POSTPROCESSOR STATEMENTS (cont'd)

The CL tape format structure of basic words of the first kind is always the same.

(1) Record Number

(2) Record Type = 2000

(3) Record Subtype (N<1000)

and, NWPR =3.

Basic words of the second kind also have identical CL tape format structures.

(1) Record Number

(2) Record Type = 2000

(3) Record Subtype (N>1000)

(4) First BCD Word

(5) Second BCD Word

(6) Third BCD Word

and so on for (NWPR-3) words.

The common words usually have a variable format structure which can be described only generally. In the APT III System, minor word modifiers are given numeric code equivalents which are fixed point integers. Any numbers which appear to the right of the (/) are passed on to the CL tape as floating point numbers. Hence, in general, a common word postprocessor statement will have a mixture of integer and floating point numbers in its CL tape record. Thus, a statement such as

```
SPINDL/10,RPM,RANGE,2,CLW
```

appears on the CL tape as:

3.3.2 RECORD TYPE 2000 - POSTPROCESSOR STATEMENTS (cont'd)

(1) Record Number

(2) Record Type = 2000

(3) Record Subtype = 1031

(4) 10 (floating point)

(5) 78 (integer code for RPM)

(6) 145 (integer code for RANGE)

(7) 2 (floating point)

(8) 60 (integer code for CLW)

and NWPR = 8.

Because of vocabulary variable formats and couplet usage the same statement can be written as

```
SPINDL/10,CLW,RANGE,2
```

in which case the CL tape record appears as:

(1) Record Number

(2) Record Type = 2000

(3) Record Subtype = 1031

(4) 10

(5) 60

(6) 145

(7) 2

and NWPR = 7.

Hence, it is readily apparent that common word CL tape records follow no set format structure.

3.3.2 RECORD TYPE 2000 - POSTPROCESSOR STATEMENTS (cont'd)

There is only one postprocessor common word statement which has a BCD word in its CL tape record and that is the MACHIN statement. The modifier following the slash (/) calls for the postprocessor; this modifier is always in BCD form.

MACHIN/GECENT,1,OPTAB,4,40

This is a special case statement since it is the only postprocessor statement recognized and dealt with in APT Section I where the postprocessor table is set up. The CL tape record for the above statement appears as:

(1) Record Number

(2) Record Type = 2000

(3) Record Subtype = 1015

(4) GECENT (BCD Word)

(5) 1 (floating point)

(6) 170 (integer code for OPTAB)

(7) 4 (floating point)

(8) 40 (floating point)

and NWPR = 8.

3.3.2.1 MAJOR WORD LIST

Common words have numeric subtype codes always greater than 1000. The common words (and their respective subtype numeric code) which are recognized by the GECENT III postprocessor are the following:

Common Word	Numeric Code
AIR	1011
AUXFUN	1022
CLAMP	1060
CLRSRF	1057
COMBIN	1071
COOLNT	1030
COUPLE	1049
CUTCOM	1007
CYCLE*	1054
DELAY	1010
DRAFT	1059
FEDRAT	1009
FLAME	1067
LEADER	1013
LINTOL	1068
LOAD*	1075
MACHIN	1015
MCHTOL	1016

* In the parlance of the proposed APT-IV vocabulary these words are generic.

3.3.2.1 MAJOR WORD LIST (cont'd)

Common Word	Numeric Code
MODE	1003
OP	1073
OPSKIP	1012
ORIGIN	1027
OVRCNT	1085
PITCH	1050
PIVOTZ	1017
POSITN	1072
PPUNCH	1082
PREFUN	1048
PRFSEQ	1069
REWIND	1006
ROTATE*	1066
SAFETY	1028
SELECT*	1074
SEQNO	1019
SET*	1087
SPINDL	1031
THREAD	1036
TMARK	1005

3.3.2.1 MAJOR WORD LIST (cont'd)

Common Word	Numeric Code
TOOLNO	1025
TRANS	1037
TURRET	1033
WELD	1076
XOFSET	1084

Other common words which are not in the above list are not recognized in the GECENT III postprocessor therefore, disregarded; no warning comment or error is issued.

The minor word list that the GECENT III postprocessor recognizes in common word statements is given in Section 3.3.2.2.

For a CL tape Record Type 2000, subroutine GEBASE calls subroutine AUXLRY, where a test is then made on the Record Subtype, which causes the postprocessor to branch to the proper subroutine. In all cases subroutine AUXLRY calls the subroutine which has the same name as the major word, e.g., the statement SPINDL/10, CLW is processed subroutine SPINDL; the statement SELECT/READER is processed in subroutine SELECT.

The reader is directed to the GECENT III Part Programmer's Manual for a complete description and usage of the above postprocessor statements.

3.3.2.2 MINOR WORD LIST

The following minor words (and their respective APT-III numeric code) are recognized in postprocessor common word statements by the GECENT III postprocessor.

Minor Word	Numeric Code
ABSPOS	322
ANGLE	252
ALL	51
BAR	207
BEAM	297
BORE	82
BOTH	83
CCLW	59
CLW	60
COARSE	195
DECR	62
DFLCTN	296
DEEP	153
DOWN	113
DRAG	299
DRILL	163
DWELL	197
ECODE	323
FACE	81
FEED	270
FINE	193
FLOOD	89
FRONT	148
HEAD	238
HED	238
HIGH	62
IN	48

3.3.2.2 MINOR WORD LIST (cont'd)

Minor Word	Numeric Code
INCR	66
INDEXR	242
INHIBT	279
INSPEC	173
IPM	73
IPR	74
LARGE	7
LEFT	8
LINCIR	95
LINEAR	76
LOCK	114
LOW	63
MAGZIN	178
MANUAL	158
MASTER	181
MAXIPM	96
MAXRPM	79
MEDIUM	61
MILL	151
MINUS	10
MIST	90
~~MMPR~~ MMPR	74
NEUTRL	166
MXMMPM	96

.3.2.2 MINOR WORD LIST (cont'd)

Minor Word	Numeric Code
NEXT	162
NOBACK	194
NOW	161
OFF	72
OFSETL	275
ON	71
OPER	231
OPTAB	170
ORIENT	246
OUT	49
OVRIDE	192
OXYGEN	169
PART	260
PALLET	239
PLUS	19
PREHET	171
RADIUS	23
RAIL	93
RANGE	145
REV	97
READER	241
REAR	149
RIGHT	24
ROCK	248
ROTREF	68

3.3.2.2 MINOR WORD LIST (cont'd)

Minor Word	Numeric Code
RPM	78
SADDLE	150
SCHEDL	250
SFM	115
SHIFT	249
SHORT	174
SIDE	94
SLAVE	180
SMALL	26
STEP	92
TABLE	177
TAP	168
TAPKUL	91
THRU	152
TILT	247
TLPOT	167
TOOL	87
TOOL	240
TORCH	172
TRAV	154
TUL	240
TURET	179
TURN	80

3.3.2.2 MINOR WORD LIST (cont'd)

Minor Word	Numeric Code
UP	112
XAXIS	84
XCOORD	116
XYPLAN	33
YAXIS	85
YCOORD	117
YZPLAN	37
ZAXIS	86
ZCOORD	118
ZXPLAN	41

3.3.3 RECORD TYPE 3000 - SURFACE DATA

This record type contains data descriptive of the circular drive surface which the cutter path is to follow. This data is essential in the GECENT III postprocessor when circular interpolation is available on the control (option 9 =1). When only linear interpolation is available, this record type is disregarded.

The format structure of this record type is given only for a circular drive surface since no other drive surface plays a special role in the GECENT III postprocessor. In most cases the APT System does not pass on drive surface data except for circles and cylinders.

(1) Record Number

(2) Record Type = 3000

(3) Surface Use Indicator (2 for a DS)

(4) Tool Position (1=TO, 2=PAST, 4=Tangent)

(5) Drive Surface Type=4 for a circle

(6) Number of words in canonical form

(7) Name of surface in BCD

(8) Surface name subscript

(9) X value for circle center

(10) Y value for circle center

(11) Z value for circle center

(12) X component of axis vector

(13) Y component of axis vector

(14) Z component of axis vector

(15) Radius of circle

The APT Section 10 Manual should be consulted for more details on this record type.

3.3.3 RECORD TYPE 3000 - SURFACE DATA (cont'd)

Only items 5, 9, 10, 11, 12, and 13 are used by the GECENT III postprocessor. If ICLDAT(5)≠4, the drive surface record is disregarded and the flag CIRFLG is set to zero indicating the path is not for circular interpolation. Otherwise, CIRFLG is set to +1 and the circle center is saved in the array CIRDAT. The motion record immediately following this Type 3000 record is processed using circular interpolation (See Section 3.4.4.).

3.3.4 RECORD TYPE 5000 - MOTION RECORDS

See Section 3.4 for the processing of this record type.

3.3.5 RECORD TYPE 6000 - ARELEM FLAGS

The data in this type record provides information regarding the cutter and cutting tolerances. Actually, the record provides other information which could be of use to a postprocessor, but currently the GECENT III postprocessor only makes use of the items mentioned below.

The complete format structure of this type record is as follows:

(1) Record Number

(2) Record Type = 6000

(3) Record Subtype

(4) ---NWPR) Other related data

The items contained in the fourth location and beyond are dependent upon the record subtype. The subtypes marked with an asterisk are the only items recognized by the GECENT III postprocessor.

Subtype = 1

Record item 4 is 0 for CUT and 1 for DNTCUT.

Subtype = 2

Record item 4 is 0 for 2DCALC; 1 for 3DCALC; and 2 for NDTEST.

Subtype = 4*

Record item 4 is the INTOL.

3.3.5 RECORD TYPE 6000 - ARELEM FLAGS (cont'd)

Subtype = 5*

Record item 4 is the OUTTOL.

Subtype = 6*

Record items 4 through 10 contain data defining the cutter, as follows:

(4) Diameter of cutter, D*

(5) Radius of cutter, r*

(6) Offset of corner radius center, E

(7) Height of corner radius center, F

(8) Cutter point angle, α

(9) Cutter side angle, β

(10) Length of cutter, h

This record type is processed in subroutine GEBASE where the inner tolerance INTOL is saved in TOLIN, the outer tolerance OUTTOL in TOLOUT, and the cutter radius CUTRAD is determined by D/2. The resultant CUTRAD is compared with r to determine whether or not a ball tool is in use, since D/2=r for a ball tool. Accordingly, parameter CUTTER=0 for a non-ball cutter, and =1 for a ball cutter. This information is pertinent to the postprocessor since circular interpolation in all planes is possible only with a ball cutter.

3.3.6 RECORD TYPE 9000 - ARELEM PARAMETERS

The only function served by this record type is to designate the existence of multiaxis processing. Although the format structure can vary, the only format recognized by the GECENT III postprocessor is:

(1) Record Number

(2) Record Type = 9000

(3) Record Subtype = 2

and NWPR =3.

All records for which the subtype is not 2 are disregarded.

The multiaxis flag AXMULT is set to 1 to indicate the existence of a multiaxis condition, and the parameters NCOM and NAXES are set as:

NAXES = 5

NCOM = 6

(See Section 3.4.1 on how these parameters are used.)

3.3.7 RECORD TYPE 14000 - FINI

This record identifies the FINI statement and initiates the termination sequence in the postprocessor. The format structure is:

(1) Record Number

(2) Record Type = 14000

and NWPR = 2.

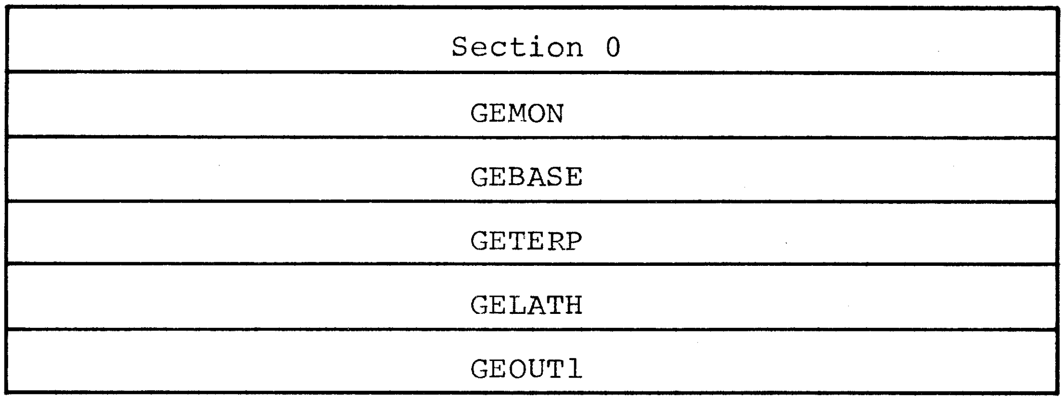
The postprocessor outputs a command block of CODE = 18 for a FINI statement.

3.4 MOTION ELEMENT

In the GECENT III postprocessor, a motion may be an absolute positioning move, a linear interpolation incremental move, a circular interpolation incremental move, a multiaxis move, or a rotary move. Each of these moves follows a separate processing path through the postprocessor, during which it becomes engaged in a variety of tests, path modifications, and optimizing sequences before it is finally made output. Processing of these motions is generally quite complex, but the major effects they undergo are detailed in the sections following. Special sequences which normally require greater coverage are briefly touched upon but explained in detail in a later section.

3.4.1 OBTAINING MOTION DATA FROM THE CL TAPE

A typical overlay structure is diagrammed below for a lathe which has an incremental contouring system.



The key motion overlay in the structure is GETERP which contains all of the incremental linear and circular interpolation sequences. GETERP is not needed for a positioning machine since any special positioning move adjustments are done in the GEPOS overlay. A multiaxis milling machine has the added overlay GEMAXS in core to supplement GETERP.

Processing within the Motion Element follows after the Input Element (Section 3.2) completes its reading of the CL tape. Subroutine GEBASE branches to subroutine MOTION when the CL tape record is a type 5000, and it is here that the program routing of the motion occurs.

3.4.1 OBTAINING MOTION DATA FROM THE CL TAPE (cont'd)

A CL tape motion record has the following format structure for a non-multiaxis move:

(1) Record Number

(2) Record Type = 5000

(3) Record Subtype (=3,4,5 or 6)

(4) Subscript of point, vector, or surface

(INDPTS) (5) BCD name or surface

$$\left.\begin{array}{ll}(6) & x_1\\ (7) & y_1\\ (8) & z_1\end{array}\right\}\quad \text{NCOM} = 3$$

$$\left.\begin{array}{ll}(9) & x_2\\ (10) & y_2\\ (11) & z_2\end{array}\right\}\quad \text{NCOM} = 3$$

$$\begin{array}{ll}\vdots & \vdots\\ \text{(NWPR)} & z_n\end{array}$$

3.4.1 OBTAINING MOTION DATA FROM THE CL TAPE (cont'd)

A multiaxis record appears as:

	(1)	Record Number	
	(2)	Record Type = 5000	
	(3)	Record Subtype (=3,4,5 or 6)	
	(4)	Subscript	
(INDPTS)	(5)	BCD name	
	(6)	x_1	NCOM = 6
	(7)	y_1	
	(8)	z_1	
	(9)	i_1	
	(10)	j_1	
	(11)	k_1	
	(12)	x_2	NCOM = 6
	(13)	y_2	
	(14)	z_2	
	(15)	i_2	
	(16)	j_2	
	(17)	k_2	
	.	.	
	.	.	
	.	.	
(NWPR)		k_n	

The record subtypes are discussed in Section 3.4.2.

The data x, y, z are the algebraic part coordinates derived from the part program and the data i, j, k are the backward directed direction cosines of the tool. All three values x, y, z are always given, even for two dimensional programs, in which case one of the values (usually z) is zero or some constant.

3.4.1 OBTAINING MOTION DATA FROM THE CL TAPE (cont'd)

The postprocessor uses two key indices for obtaining and saving motion data, namely, NCOM and INDPTS.

NCOM designates the "normal" number of values to be found in a CL tape record for a non-multiaxis and multiaxis move; NCOM is 3 for non-multiaxis, and is 6 for a multiaxis record; see the CL tape record formats above. NCOM is initialized to 3, but is reset to 6 when the MULTAX record (Record Type 9000 - See Section 3.3.6) is encountered.

INDPTS is an indicator pointing to the CL tape record location which is one less than the value to be selected next. Before subroutine INPUT is called, it is always preset to 5. The manner in which it is used will become clear in the following descriptions.

The postprocessor saves the CL part coordinate data in the present point vector DPRESP, which is dimensioned at six and ordered as:

$$\left.\begin{array}{l}\text{DPRESP(1)} = x\\ \text{DPRESP(2)} = y\\ \text{DPRESP(3)} = z\end{array}\right\}\quad \text{linear locations}$$

$$\left.\begin{array}{l}\text{DPRESP(4)} = i\\ \text{DPRESP(5)} = j\\ \text{DPRESP(6)} = k\end{array}\right\}\quad \text{tool direction cosines}$$

If a TRANS statement had been given, the TRANS values of x, y, and z (stored in the vector TRANSL) are added to the corresponding value of DPRESP. TRANSL is zero if no TRANS is given. Thus, at every instant of processing time, the postprocessor knows exactly where the tool control point is.

3.4.1 OBTAINING MOTION DATA FROM THE CL TAPE (cont'd)

For example, assume we have a two point record stored in ~~CL~~DATACL

CLDATA

(1) Record Number

(2) Record Type = 5000

(3) Record Subtype = 5

(4) Subscript

(5) BCD Name (INDPTS=5)

$$\left.\begin{array}{ll}(6) & x_1\\ (7) & y_1\\ (8) & z_1\end{array}\right\} \quad \text{First point}$$

$$\left.\begin{array}{ll}(9) & x_2\\ (10) & y_2\\ (11) & z_2\end{array}\right\} \quad \text{Second point}$$

NCOM = 3 NWPR = 11

The postprocessor selects the first point by

CL
DPRESP(I) = ~~CL~~DATA(INDPTS+I) + TRANSL(I)

for I = 1 to NCOM. After the point is processed and made output, INDPTS is increased by NCOM, and the new point is similarly selected and processed until INDPTS becomes greater than NWPR. It can be seen that regardless of the size of the record, or whether or not it is a multiaxis record, the sequence is a generalized process for all motion records.

Motion data as stored in DPRESP represents the cutter path in terms of the part coordinate system; but to actually machine the part, the data must be converted to the machine coordinate system.

3.4.1 OBTAINING MOTION DATA FROM THE CL TAPE (cont'd)

For non-multiaxis machines (with either absolute or incremental systems) conversion from part to machine coordinates is nothing more than a rounding of part coordinate data at the decimal location corresponding to the step size of the machine tool*. The step size or minimum programmable incremental is given in option 14 and stored in parameter STEP in subroutine ASSIGN.

For example, assume that STEP = 0.0001. This means that data beyond the fourth decimal location cannot be recognized by the numerical control system, therefore all data must be truncated at this point, and in order to avoid an accumulative error or a loss of path accuracy, the data is also rounded. As mathematically demonstrated in Section 7.2, this method of rounding guarantees that the maximum accumulated error on any axis can never become larger than one half the minimum step size of the NC machine. In fact, if the tool is programmed to the beginning point of the part program, the accumulated error is zero.

Some examples will illustrate the rounding method; assume STEP = 0.0001.

CL Tape Value	Rounded Value
22.24686231	22.2469
22.24685231	22.2469
22.24684231	22.2468
0.00005	0.0001
0.000005	0.0000

*The step size of a machine tool is the minimum distance that an axis moves for one servo pulse. Translation and rotation axes may have the same or different minimum step sizes.

3.4.1 OBTAINING MOTION DATA FROM THE CL TAPE (cont'd)

Subroutine SRAREC performs the rounding for linear data, and subroutine SROREC rounds for rotary data. The relation used is:

$$X = \left\langle \left(\frac{|x|}{STEP} + 0.5001 \right) \right\rangle * \left(\frac{|x| * STEP}{x} \right)$$

The part coordinate values are retained in DPRESP, and the rounded values are stored in the vector DPRESM which represents the present point in machine coordinates. DPRESM is dimensioned at 6 and ordered as:

$$\left.\begin{array}{l} \text{DPRESM(1)} = X \\ \text{DPRESM(2)} = Y \\ \text{DPRESM(3)} = Z \end{array}\right\} \quad \text{linear locations}$$

$$\left.\begin{array}{l} \text{DPRESM(4)} = A \\ \text{DPRESM(5)} = B \\ \text{DPRESM(6)} = C \end{array}\right\} \quad \text{rotary locations}$$

A convention of this manual uses the lower case letters x, y, z, i, j, k to represent the part coordinate data, and the capital letters X, Y, Z, A, B, C to represent the corresponding machine coordinate data. The rotary values A, B, and C derive from a multiaxis move and represent the rotary motions necessary to maintain the vector orientation of the tool as given by the direction cosines.

Converting from part to machine coordinates for multiaxis moves is considerably more involved than simply rounding the part data, for now it becomes necessary to consider the orientation of the tool relative to the part surface. This may involve a swivel of the tool, or a tilt of the table, or any number of possible rotary motions. Therefore, it is evident that a unique relationship exists between the part and machine coordinate systems, and this relationship usually varies for different multiaxis machine tool configurations. This relationship is mathematically expressible in terms of a set of transform equations which permit conversion between part and machine coordinate systems. The set of transforms is identified by the class associated with a particular multiaxis machine tool configuration, as class 1, class 2, and so on. (See Section 4.2.)

3.4.1 OBTAINING MOTION DATA FROM THE CL TAPE (cont'd)

When the postprocessor converts the data in DPRESP to DPRESM it utilizes the related set of transforms for the machine tool class, and then rounds the converted data to the step size of the NC machine. Subroutine GEOM is the subroutine responsible for converting DPRESP into DPRESM.

3.4.2 MOTION RECORD SUBTYPES

The technique described above for selecting motion data and saving them in DPRESP and DPRESM is used by the processing sequences for each of the motion subtypes. Any additional activities are detailed for that subtype.

Motion record subtypes range from subtype 1 to 6*, but only subtypes 3 through 6 are considered in the GECENT III postprocessor. Subtype 1 (for an INDIRP) and subtype 2 (for an INDIRV) are disregarded by the postprocessor.

3.4.2.1 SUBTYPE 3 FROM POINT

Subroutine MOTION branches to subroutine FROM to process this subtype. The FROM point is stored into DPRESP, sent through subroutine GEOM where it is transformed, rounded, and saved in DPRESM, then set up in the command block DBFSEG and sent to subroutine OUTPUT for printing. The block CODE is made + 3 for the FROM point.

There is never more than one point (x,y,z) or (x,y,z,i,j,k) in each FROM point record.

In the GECENT III postprocessor, multiaxis FROM points <u>must</u> be given in part coordinate form. This convention is in keeping with APT practices and maintains a consistency in that all data from the CL tape is always in the part coordinate system.

Some NC machines may utilize a fixed FROM point or home position from which all machining operations begin. In such cases, it is desirable to ensure that the given FROM point is in keeping with the fixed FROM point. The postprocessor provides a branch to the MACFUN (see Section 5.6.1) wherein a comparison test is made of the given versus the required FROM point, and when different, prints a warning comment, "THE FROM POINT IS NOT THE HOME POSITION". This is not a fatal error, for indeed, many a case arises where it is desirable to begin machining from a point which is not the home position.

---

*Special subtypes 7 and 8 for linearity testing are not currently recognized in the GECENT III postprocessor.

3.4.2.2 SUBTYPE 4 GODLTA POINT

In APT III for some computers, GODLTA points are not always passed on to the CL tape as an incremental record because the APT system algebraically adds the increments to the current path points, and the resultant record is passed on as a Subtype 5 for a GOTO record.

For example: GODLTA/2,-4,6

Present path point is: 22,10,8. APT converts the record to appear as:

GOTO/24,6,14

However, for a PTONLY/1 run, the GODLTA record is not converted to the GOTO type record.

In the GECENT III postprocessor a GODLTA record is processed by adding the DATA CL to the postprocessor previous point data (DPREVP) and by changing the record subtype index to 5 to make it appear as a GOTO record and be processed accordingly.

There is never more than one point (xyz) in each GODLTA record.

3.4.2.3 SUBTYPE 5 GOTO POINT

This motion record is by far the most common record passed on to the CL tape. It represents the algebraic location (xyz) or (xyzijk) of each cut vector that approximates the programmed cutter path; hence, for non-linear curves a CL tape motion record can consist of hundreds of points.

Each point is read one at a time from the buffer CL DATA, processed, and made output. This is the sequence used when linear interpolation is the processing mode, but a different course is followed for circular interpolation. (See Section 3.4.4.)

3.4.2.4 SUBTYPE 6 CONTINUATION RECORD

A part programmed non-linear path can very easily produce several hundred cut vectors, but each CL tape record is limited to a maximum of 80 (xyz) points or 40 (xyzijk) multiaxis points. Therefore, APT issues several records to represent the path; these records are the Subtype 6 continuation records. They are processed exactly as a Subtype 5 GOTO record for linear interpolation but are used slightly differently for circular interpolation. This process is described in Section 3.4.4.

3.4.3 PROCESSING A MOTION RECORD

After the postprocessor obtains a point from the CL tape and stores it into DPRESP and DPRESM, it then processes the motion to make it acceptable to the NC control system. A number of tests and modifications are detailed in the description below.

After the motion has been made output, the postprocessor retains the present point but redefines it as the previous point, because the next point read from the CL tape will become the new present point. Accordingly, after a motion is processed and made output, the elements of DPRESP and DPRESM are stored in the previous point vectors DPREVP and DPREVM, respectively.

Diagram 3.4.3A illustrates how these vectors are used for processing motions.

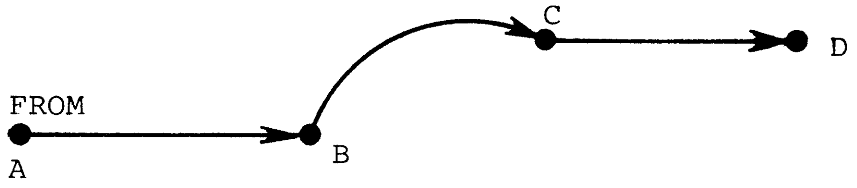


Diagram 3.4.3A

Initially, beginning with the FROM point, DPRESP contains point A and DPREVP is null. After the FROM point is made output, DPREVP becomes point A. After point B is output, DPREVP is B and DPRESP is C; and so on. Note that the same method is used for both the linear and circular interpolation sequences. The vectors DPREVM and DPRESM are reset at the same time that DPREVP and DPRESP are set.

3.4.3.1 PROCESSING A LINEAR INTERPOLATION MOTION

To produce a motion which utilizes the linear interpolation capability of the NC control system, the postprocessor need deal only with the data contained in the DPREVM and DPRESM vectors.

Positioning machines, and all machine tools which utilize an absolute system, output the values contained in DPRESM. For example, in Diagram 3.4.3.1A,

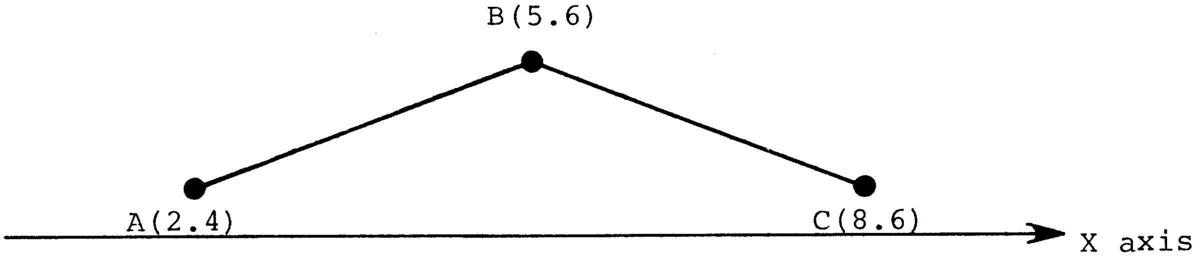


Diagram 3.4.3.1A

the X axis values of the path ABC are output as X2.4, X5.6, and X8.6. (The decimal point is not punched in the output tape but is used here for illustration.)

Contouring machines generally utilize an incremental system which requires that all motions be in the form of increments. In the above example, the X axis values for the path ABC are then X3.2 and X3.0. These incremental moves are referred to as departures. Thus, the departure from point A to point B is +3.2; the departure from point C to B is -3.0. A departure is defined to be the algebraic difference between the coordinate values of two adjacent points in a rectangular Cartesian coordinate reference frame.

3.4.3.1 PROCESSING A LINEAR INTERPOLATION MOTION (cont'd)

The postprocessor computes the departures in the machine coordinate system only, i.e., with DPREVM and DPRESM. Thus, for a multiaxis machine, the typical departures may result as follows:

Departure =	Present Point -	Previous Point
ΔX = 11.5730	22.68411	11.1111
ΔY = 3.3699	3.2466	-0.1233
ΔZ = 0.0000	-1.0000	-1.0000
ΔA = 25.0000	50.0000	25.0000
ΔB =-25.0000	-25.0	0
ΔC = 0.0000	0	0

When a CL tape motion record is processed for a line, subroutine MOTION branches to subroutine GOLINE which is the main subroutine for linear moves. After calling subroutine GEOM to convert from part to machine coordinates, it calls subroutine DEPART to compute the departures of the linear move.

Each computed departure is compared with the parameter HSTEP (which contains half the STEP size); and if the departure is less than HSTEP, it is set to zero. The reason for this is to make the axis move appear as a zero move, i.e., as no move at all, since the amount of motion specified by the departure is not physically possible on the NC machine. Because of the non-exact representation of floating point numbers, it is quite possible that the result of a subtraction can appear to be smaller than the STEP size but yet be a legitimate value. For example, the difference between the points 0.0005 and 0.0004 may appear as 0.00009999 rather than 0.00010000. The value is less than the STEP size of 0.0001 but is certainly a valid value. This is the reason why the above test uses HSTEP rather than STEP. This phenomena also illustrates the necessity for rounding up the departures; this rounding occurs later in the program.

When all departures are zero, there is, in effect, no move, and the postprocessor must disregard further processing and return to obtain a new point. This is accomplished by setting the return flag RETURN to -1 which then causes the postprocessor to reroute the program flow back to subroutine MOTION where the next point is selected.

3.4.3.1 PROCESSING A LINEAR INTERPOLATION MOTION (cont'd)

Before exiting from subroutine DEPART the departures are stored in their proper

DBFSEG(3) = ΔX

DBFSEG(4) = ΔY

DBFSEG(5) = ΔZ

The return flag RETURN is set to +1 to indicate that a motion has been accepted.

Unless segmentation (see Section 3.4.3.1) is required, the motion block is essentially ready for output. The command block CODE is set to zero to indicate a linear move, and unless SFM or threading are in mode, subroutine OUTPUT is called to produce ultimately the printed and punched output as described in Section 3.5.

3.4.3.2 SEGMENTATION OF A LINEAR MOVE

Because of the decimal format of each machine axis register, there is a limitation to the number size that a register can accept. If the format for a register is 24.0, that is, two digits to the left and four digits to the right of the decimal, then obviously the largest number that the register can accept is 99.9999. This limitation in the case of the motion registers is called the maximum departure; hence, each component value of a motion must be less than or equal to the maximum departure.

When departures result which exceed the maximum departure, the path must be segmented into sufficiently small segments. For example, if the maximum departure is 9.999 inches, a move in X of 40 inches must be segmented into five segments of 8 inches each. The postprocessor always obtains the largest segment possible.

Subroutine MOTION tests each of the computed departures versus the maximum departure, DEPMAX, which is specified in option 4. If any of the departures exceed DEPMAX, then subroutine SEGMNT is called to produce and output the proper size segments. One restriction should be noted here, namely, the multiaxis processing which calls for linearity testing must bypass the segmentation sequence in deference to the requirements of the linearity sequence. As explained in Section 3.4.7.3, linearity error to be measured accurately must consider the unsegmented total path.

3.4.3.2 SEGMENTATION OF A LINEAR MOVE (cont'd)

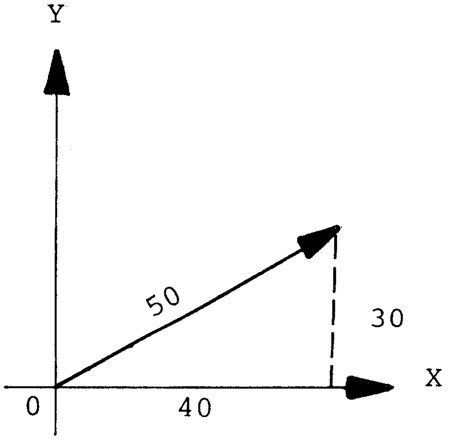
When the maximum departure is exceeded, subroutine SEGMNT segments the given path into sufficiently small motions and then outputs them. Path segmentation is fairly straightforward: linear ratioing is used to obtain the points of segmentation, and there is no limit to the number of segments produced. Basically, the postprocessor determines the number of required segments from the relation:

$$\text{Number of Segments} = < \frac{\text{Largest Departure}}{\text{Maximum Departure}} + 1 >$$

Next, the segment size is determined from:

$$\text{Segment Size} = \frac{\text{Departure}}{\text{Number of Segments}}$$

Actually, the segment size for each axis is determined so as to make linear interpolation to the segmented point unnecessary. Thus, the point of segmentation is found simply by adding the axis segment size to the current point value. Example: Assume a maximum departure of 9.9999 and move of 50 inches as shown in the diagram 3.4.3.2A



The number of segments is:

$$\text{NSEG} = < \frac{40}{9.9999} > + 1 = 5$$

The segment sizes are:

$$S_x = \frac{40}{5} = 8$$

$$S_y = \frac{30}{5} = 6.$$

DIAGRAM 3.4.3.2A

3.4.3.2 SEGMENTATION OF A LINEAR MOVE (cont'd)

Assuming the path starts from (0,0), the segments then are:

Segment 1:	(8, 6)
Segment 2:	(16, 12)
Segment 3:	(24, 18)
Segment 4:	(32, 24)
Segment 5:	(40, 30)

Each segment is individually produced and made output by calling subroutine OUTPUT.

The above example vividly illustrates the basic techniques used in segmentation, but unfortunately, actual cases are rarely that simple. Whenever the original path is segmented unevenly, there is the danger of losing accuracy in the last decimal digit. For example, by the above scheme a path of 10 inches, if segmented for a maximum departure of 3.9999 inches, produces three segments of 3.3333 inches such that their summation is 9.9999 inches, a loss of 0.0001 inches. The postprocessor must therefore provide a means of adding a pulse to the segment at the appropriate time, for it will not do to simply add in the necessary pulses on the last segment to make the result end up at the proper point. If this were done, the postprocessor would be deviating the tool from its directed path.

This is illustrated in exaggerated form in diagram 3.4.3.2B the heavy line represents the actual path; the small paths are the segments.

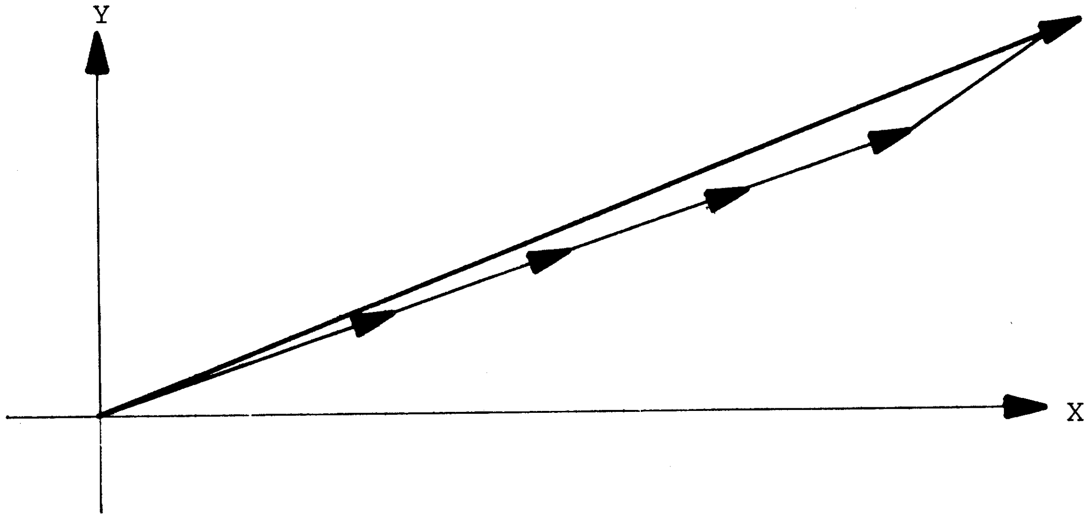


Diagram 3.4.3.2.B

3.4.3.2 SEGMENTATION OF A LINEAR MOVE (cont'd)

Hence, it is essential for the postprocessor to maintain optimum accuracy for each generated segment so that the produced segmented path adheres as closely as possible to the designated path. The method used by the GECENT III postprocessor is to recompute the segment length each time it is to be generated. The new segment is determined by taking the difference between the true present point and the machine previous point. In the description which follows, only the X axis is used, but the same technique applies to all axes.

First of all, the true, unrounded, untruncated segment length S_x is computed as:

$$S_x = \frac{\Delta X}{NSEG}$$

where Δx is the x-axis departure, and NSEG is the number of segments required. The output segment length is now determined by the following steps:

1. Add S_x to the previous machine point x_0 to obtain the present machine point x_1.

$$x_1 = x_0 + S_x$$

2. Subtract the previous machine point x_0 from the present machine point x_1 to get the increment δ_x .

$$\delta_x = x_1 - x_0 \qquad (\delta_x \geq S_x)$$

3. Round δ_x to the step size to obtain δ_x'

$$\delta_x' = \left\langle \frac{\delta_x}{STEP} + 0.5 \right\rangle * STEP$$

4. The increment δ_x' is the required output value. δ_x' is now added to the previous point x_0 to become the new x_0.

$$x_0 = x_0 + \delta_x'$$

5. The four steps are repeated for NSEG times.

3.4.3.2 SEGMENTATION OF A LINEAR MOVE (cont'd)

An example will make the above description completely clear. Assume a DEPMAX of 3.9999, STEP = 0.0001, and a ΔX move of 10 inches. The previous point was 8. S_x= 3.3333333. The columns below when read across show the results of steps 1 through 4.

	(1)	(2)	(3)	(4)
x_0	x_1	δ_x	δ'_x	New x_0
8.0000	11.3333333	3.3333333	3.3333	11.3333
11.3333	14.6666666	3.3333666	3.3334	14.6667
14.6667	17.9999999	3.3332999	3.3333	18.0000

From the above example it can be seen at value 14.6667 how the needed pulse is automatically added at the appropriate time to maintain the optimum path accuracy. Also note that the final result (18.0000) is the exact final point, but x_1 is not. In the postprocessor the vector DPREVM is used analogously as x_0 , and the vector DPRESM as x_1. When the path is completed, DPRESM is reset to DPREVM so that both points are identical, for upon return to subroutine GOLINE, the normal exiting sequence is always to reset the vectors as:

DPREVM = DPRESM

Section 3.4.3.2.1 gives a mathematical demonstration illustrating that the segmentation technique used by the GECENT III postprocessor does not cause any significant error in the segmented path.

3.4.3.2.1 SEGMENTATION PROOF

Let Δ be the exact segment length and $[x]$ be the rounded value of x such that

$$x = [x] + \varepsilon,$$

where ε is the error difference.

Then, by the above described method where DPRESM contains the exact location of the path, and DPREVM contains the actual location, we have for a few segments along the x-axis:

DPRESM	DPREVM
x	x
$x+\Delta$	$x + [(x+\Delta)-x] = x + [\Delta]$
$x + 2\Delta$	$x + [\Delta] + [2\Delta - [\Delta]]$
$x + 3\Delta$	$x + [\Delta] + [2\Delta - [\Delta]] +$ $[3\Delta - [2\Delta - [\Delta]] - [\Delta]]$

Defining $\delta =$ DPRESM - DPREVM, after three segments we have

$$\delta = 3\Delta - [\Delta] - [2\Delta - [\Delta]] - [3\Delta - [2\Delta - [\Delta]] - [\Delta]]$$

If $[\Delta] = \Delta$, then

$\delta = 3\Delta - \Delta - \Delta - \Delta = 0$, illustrating no error would occur.

However, $\Delta = [\Delta] + \varepsilon$.

In general,

$$\delta = n\Delta - \sum_{i=1}^{n} a_i ,$$

where $a_i = [\Delta]$,

and $a_{j+1} = [(j+i)\Delta - \sum_{i=1}^{j} a_i]$.

3.4.3.2.1 SEGMENTATION PROOF (cont'd)

Hence, we may define ε_i where $|\varepsilon_i|$ <half step size, by:

$$a_1 = [\ \Delta] = (\Delta) + \varepsilon_1$$

$$a_2 = [2\Delta - (\Delta + \varepsilon_1)] = (\Delta - \varepsilon_1) + \varepsilon_2$$

$$a_3 = [3\Delta - (\Delta - \varepsilon_1 + \varepsilon_2) - (\Delta + \varepsilon_1)] = (\Delta - \varepsilon_2) + \varepsilon_3$$

$$a_n = [n\Delta - (\qquad)] = (\Delta - \varepsilon_{n-1}) + \varepsilon_n$$

and

$$\sum_{i=1}^{n} a_i = n\Delta + \varepsilon_n$$

as the telescoping series collapses.

Therefore,

$$|\delta| = \left| n\Delta - \sum_{i=1}^{n} a_i \right| = \varepsilon_n$$

indicating that any error produced by the segmentation process is less than the step size.

3.4.4 PROCESSING A CIRCULAR INTERPOLATION MOTION

To produce a motion for circular interpolation the postprocessor must at first obtain the first and last points of the circle. There is no difficulty in obtaining the first point, but the last point is more difficult to obtain because circle data can easily consist of several continuation (subtype 6) records; and hence, the last point can be several CL records beyond the first point record. The postprocessor obtains the first and last points in the following manner.

Preceding every circle data (Type 5000, subtype 5) record is a Surface Data record (Type 3000-see Section 3.3.3) which indicates that the current cutter path is that of a circle. When processing this record type, the postprocessor sets the flag CIRFLG = 1 to indicate circular interpolation is called for. Therefore, after the motion record (Type 5000, subtype 5) is read and stored in ~~CL~~DATA,CL subroutine MOTION on the subtype 5 branch tests CIRFLG, and if non-zero, calls subroutine GOCIRC to begin the circular interpolation sequence.

The first thing that subroutine GOCIRC must decide is whether or not circular interpolation is possible with the current record. The subroutine makes a series of tests, and if any test indicates the impossibility of circular interpolation, the postprocessor immediately redirects the program flow to the linear interpolation sequence.

The postprocessor makes the following tests:

1. Is circular interpolation the specified mode? It is, if the LINCIR modifier was given, or if option 28 = 1.

2. Has more than one point of the circle been given? If so, the postprocessor has sufficient data to proceed.

3. Does the circle lie in a plane? Subroutine CHKAX is called to make this test. The non-planar axis values must be constant for <u>every</u> point in the record.

4. Is the circle radius greater than the maximum departure? If so, circular interpolation cannot be used.

3.4.4 PROCESSING A CIRCULAR INTERPOLATION MOTION (cont'd)

The circle radius used for this test is not the part circle radius r, but actually the distance from the part circle center to the cutter control point, R.

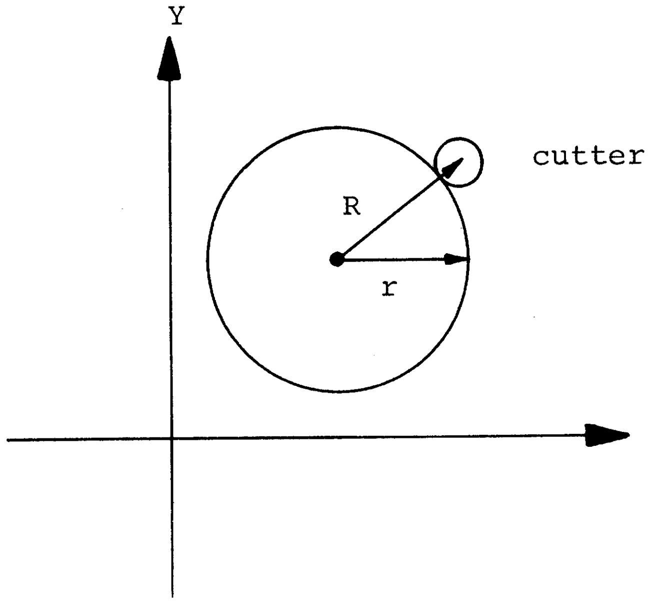


Diagram 3.4.4A

For large circles, R is not significantly larger than r, but when the circle is very small, the effect on the feedrate becomes a significant factor since the feedrate command is a function of R. See the Part Programmer Manual for a means of optimizing the feedrate by controlling the value of R.

5. The circle path must not be for a thread. Since the I, J, and K registers are necessary for both threading and circular interpolation, each event is mutually exclusive.

If each of the above tests is passed, the first point of the circle is saved in DCRPT1, which is dimensioned at three and ordered as x, y, z. The first point of the circle does not come from the CL data, since it is actually the previous point DPREVP; for as with all paths, the APT System does not repeat the starting point of a new path because that point is the same as the last point of the previous path.

3.4.4 PROCESSING A CIRCULAR INTERPOLATION MOTION (cont'd)

The last point of the CL record is saved in DCRPT2 as the potential last point of the circle. At this point in the program it is not known for a certainty that the last point of the CL record is truly the last point of the circle, for it is possible that the circle data may be continued on one or more subsequent subtype 6 continuation records. Thus, the postprocessor must read in the next record and check to see if it is a continuation record. If it is, the last point of the CL record is saved in DCRPT2, and the next record is read in, and so on, until the read-in record is <u>not</u> a continuation record.

The postprocessor searches for the last point by using the flag CIRSEQ. In subroutine GEBASE after a record is read, the flag CIRSEQ is checked; and if non-zero, program flow is routed immediately to subroutine GOCIRC. Hence, all the preliminary processing is disregarded so that the last point of the circle can be quickly found.

Subroutine GOCIRC checks to see if the CL record is a continuation record. If it is, a test is made to ensure that the circle still lies in a plane; however, at this point in the program it is too late to use linear interpolation should the circle not lie in a plane, therefore, only a warning comment is issued, and the circular interpolation sequence continues.

When a non-continuation record is found, the postprocessor knows that the last point saved in DCRPT2 is truly the last point of the circle. After the circle has been processed and made output, program flow is directed back to subroutine GEBASE (by setting the RETURN flag to +1) to the point where the ~~C~~LDATACL array is initially interrogated. There is no need to read the CL tape for a new record since the non-continuation record already exists in the ~~C~~LDATACL array.

Once the first and last points of the circle have been found, the postprocessor can proceed to output the circle path in the circular interpolation mode. To do this, the circle must first be segmented (if necessary) into its respective quadrant segments. The Mark Century numerical control does not process a circle path for more than ninety degrees, hence, a circle angle greater than ninety degrees must be reduced to two or more smaller angles, each of which must be no greater than ninety degrees. When a circle is greater than ninety degrees, it obviously lies in more than one quadrant if the circle center is at the reference frame origin. The postprocessor in segmenting the circle, segments it at the axis where the circle changes quadrants. Thus, in the diagram the circle AE is segmented at points B, C and D to produce the four segments AB, BC, CD, and DE.

3.4.4 PROCESSING A CIRCULAR INTERPOLATION MOTION (cont'd)

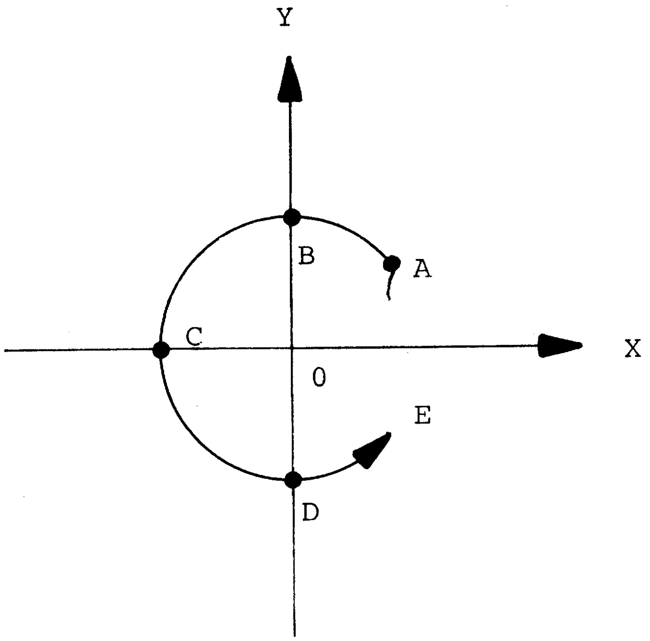


Diagram 3.4.4B

The segmentation sequence as described above is effected through the three subroutines CIRINT, QUADET, and QUADNT. Subroutine GOCIRC calls subroutine CIRINT which sets up a call to subroutine QUADET as a function of the circle plane. The parameter flag IPLANE is 0 for the XY plane, 1 for the ZX plane, and 2 for the YZ plane; this flag directs the postprocessor to the proper calling sequence of subroutine QUADET. The indices given in the calling sequence relate to the vectors CIRPT1 and CIRPT2, and specify the elements of the array which pertain to the circle plane. For example, if the circle lies in the XY plane, the indices 1,2,3 are given; these specify the values of the array as:

DCRPT1(1) for the X value,

DCRPT1(2) for the Y value, and

DCRPT1(3) for the non-planar value.

3.4.4 PROCESSING A CIRCULAR INTERPOLATION MOTION (cont'd)

For the ZX plane, the indices are 1, 3, 2, and specify:

DCRPT1(1) for the X value,

DCRPT1(3) for the Z value, and

DCRPT1(2) for the non-planar value.

With this information subroutine QUADET can determine the quadrant change points. The circle is translated to the origin by subtracting the circle center from the circle first and last points, i.e.,

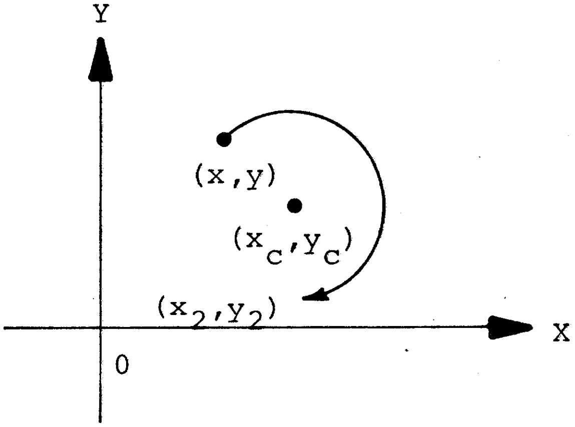
$$O_{x1} = x_1 - x_c$$

$$O_{y1} = y_1 - y_c$$

$$O_{x2} = x_2 - x_c$$

$$O_{y2} = y_2 - x_c$$

In the diagrams below, it is clear that the circle passes through the second, first, and fourth quadrants after the translation to the origin occurs.



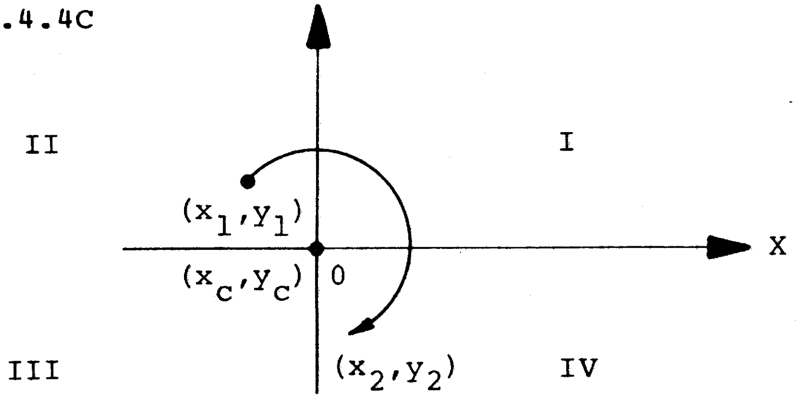


Diagram 3.4.4C

3.4.4 PROCESSING A CIRCULAR INTERPOLATION MOTION (cont'd)

The postprocessor determines the passed-through quadrants by subroutine QUADNT. This subroutine is called twice; first, with the beginning point of the circle, and then with the last point of the circle. For the first point of the circle, subroutine QUADNT sets the input parameter QA to the quadrant number in which the point falls, and, similarly, it sets the input parameter QB for the last point. In the example above, these values are found as:

QA = 2, QB = 4

Subroutine QUADNT finds these values in a most direct manner. The calling sequence to QUADNT gives, for example:

```
CALL QUADNT (1, PX1, PY1, QA),
```

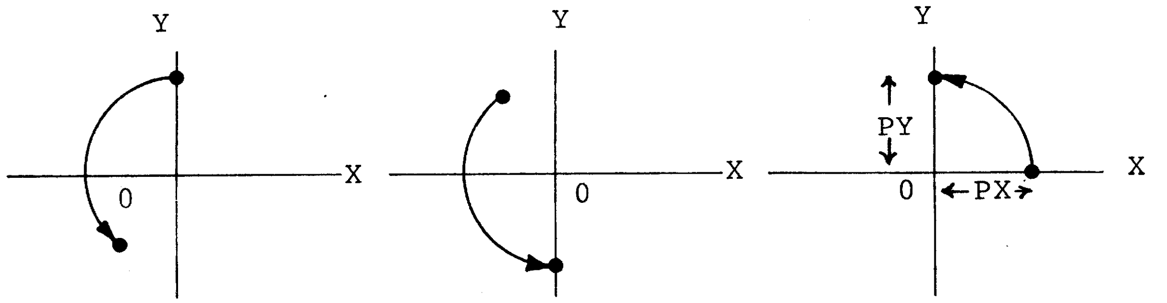
where 1 identifies the point (PX1,PY1) as the first point of the circle (with its center at the origin), and QA is the return output. The subroutine tests PX1 and PY1 for their sign condition. In the example above, PX1 is negative and PY1 is positive; from these conditions it is obvious that the point must lie in the second quadrant, therefore, QA = 2.

For the last point the call is:

```
CALL QUADNT (2,PX2,PY2,QB)
```

where 2 identifies the point (PX2, PY2) as the last point of the circle. Since PX2 is positive and PY2 is negative, the point can lie only in the fourth quadrant, and, accordingly, QB = 4.

An indeterminate condition arises when one or both of the points lie exactly on an axis as in the examples below.



First Point on Axis
(Diagram A)

Last Point on Axis
(Diagram B)

Both Points on Axes
(Diagram C)

Diagram 3.4.4D

3.4.4 PROCESSING A CIRCULAR INTERPOLATION MOTION (cont'd)

In diagram A, for example, does the first point lie in Quadrant I or II? Analytically, it could be either quadrant, but the postprocessor requires a unique decision. Hence, the following conventions are used:

1. When the first point lies exactly on an axis, the point is defined to lie in the quadrant through which the circle initially passes. In Diagram A, the first point then lies in Quadrant II.

2. When the last point lies exactly on an axis, the point is defined to lie in the quadrant through which the circle last passes. In Diagram B, the last point then lies in Quadrant III.

As indicated earlier, the postprocessor uses the signed values of PX and PY to find the proper quadrant value, but since under these conditions PX or PY is zero, the subroutine must now use the circle direction to determine the proper quadrant. In Diagram A, for the first point, PX is zero, and PY is positive, and the circle direction is CCLW. Hence, QA = 2. If the circle direction was CLW, QA = 1.

In Diagram C we have the condition where both the first and last points lie on the axes. Subroutine QUADNT processes the input conditions and yields the following results:

First Point: PX > 0, PY = 0, CCLW; QA = 1.

Last Point: PX =0, PY > 0, CCLW; QB = 1.

Hence, the path is defined to lie in one quadrant only.

The circle direction was determined earlier in the program when subroutine GOCIRC called subroutine DETDIR. This subroutine finds the circle direction by taking the first two points of a circle record and translating the circle center to the origin; the two vectors (from the origin to the circumference) are then crossed.

3.4.4 PROCESSING A CIRCULAR INTERPOLATION MOTION (cont'd)

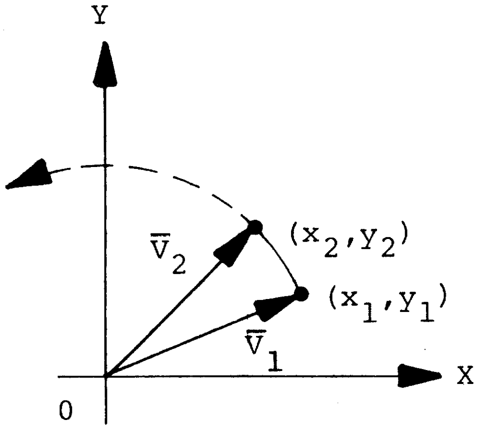


Diagram 3.4.4.E

For the two vectors $\bar{V}_1$ (x_1, y_1, z_1) and $\bar{V}_2$ (x_2, y_2, z_2), the cross product gives

$$\bar{V}_1 * \bar{V}_2 = \begin{vmatrix} \bar{i} & \bar{j} & \bar{k} \\ x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \end{vmatrix} = (x_1y_2 - x_2y_1)\bar{k} + (y_1z_2 - z_1y_2)\ \bar{i} + (x_1z_2 - x_2z_1)\bar{j}$$

Since one axis must always be constant, it is necessary to consider only two axes, viz., essentially, $(w_1v_2 - w_2v_1) = D$. For a constant z, w = x, v = y; for y constant, w = z, v = x; for x constant, w = y, v = z. If $D < 0$, direction is CLW; $D > 0$, direction is CCLW. If $D = 0$, the vectors are parallel and an error is assumed.

The subroutine sets the circle direction flag CIRDIR as:

CLW; CIRDIR = 0

CCLW: CIRDIR = 1

At this point in the program the postprocessor has all the requisite information to produce the circle segments, viz., the circle direction and the beginning and ending quadrants Subroutine QUADET now sets up the array DBUFER with the quadrant intersection points and the last point of the circle. DBUFER is dimensioned at (6,5) and ordered as x, y, z, i, j, k.

3.4.4 PROCESSING A CIRCULAR INTERPOLATION MOTION (cont'd)

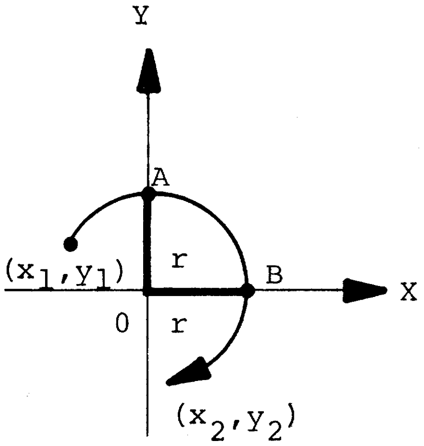


Diagram 3.4.4F

Thus, in our original example, the points A, B, and (x_2 ,y_2) are stored in that order into DBUFER.

When the circle path lies in only one quadrant, the last point (x_2,y_2) is stored into DBUFER.

There is no need to store the first point of the circle since it already exists in the DPRESP and DPRESM vectors. It will become clear shortly why only the last point of the circle is stored.

Subroutine QUADET determines the values of the points A and B by setting up the table XP (dimensioned at (2,4)) as follows:

XP:

	1	2	3	4
1	0	-r	0	r
2	-r	0	r	0

CLW

		1	2	3	4
(x)	1	0	-r	0	r
(y)	2	r	0	-r	0

CCLW

The XP table is initially set to zero and dependent upon the circle direction, the table is set at certain locations with the circle radius CIRRAD or its negative. The subroutine sequence automatically determines the values of (x_A , y_A) and (x_B , y_B) by selecting the values from the table and adding the corresponding circle center value to retranslate the point from the origin.

3.4.4 PROCESSING A CIRCULAR INTERPOLATION MOTION (cont'd)

Row 1 of the table gives the x value of the quadrant intersection point, while row 2 gives the corresponding y value. Thus, the table for CLW is stored as (reading columnwise):

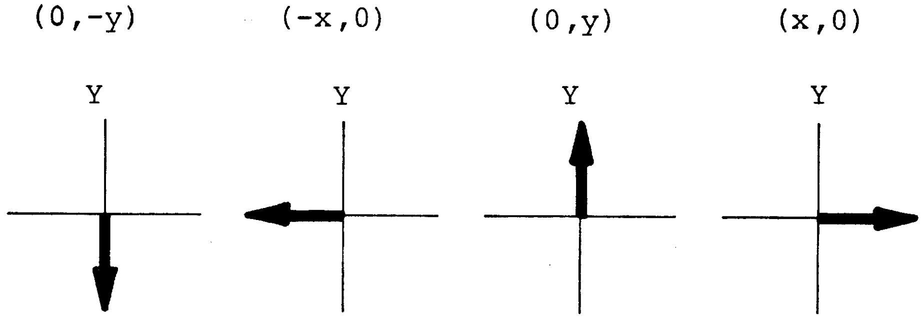


Diagram 3.4.4G

The subroutine automatically selects the proper column by computing the index JJM4 as:

$$JJM4 = (IA + II) \text{ modules } 4 + 1,$$

where IA is the starting quadrant number and II is the counter per quadrant, II = 1, 2, 3, or 4. Note that IA is the quadrant starting from the intersection point and not from the first point. Thus in the example above, the starting quadrant is measured from point A and not (x , y); therefore, IA is 1.

Another parameter used in the sequence is IDQ which gives the number of quadrant intersection points; in this example IDQ = 2 counting points A and B.

Referring to Diagram 3.4.4F of our example above, it will be seen how the points A and B are found in the following sequence.

For this example, IA = 1 and II initially is 1. Therefore,

$$JJM4 = (1 + 1) \text{ modulus } (4) + 1 = 3.$$

Hence, the third column of the CLW XP table is the quadrant intersection point A, i.e., $x_A = 0$, $y_A = r$. However, to obtain the true absolute values, we must ~~subtract~~ ADD the corresponding circle center value, since the circle data was originally translated to the origin; thus: $x_A = 0 + x_c$, $y_A = r + y_c$.

3.4.4 PROCESSING A CIRCULAR INTERPOLATION MOTION (cont'd)

This point is stored into DBUFER, and the subroutine looks for the next point. II now is 2, hence,

$$JJM4 = (1 + 2) \text{ modulus } (4) + 1 = 4,$$

and from the fourth column, we get $x_B = r + x_c$, $y_B = 0 + y_c$. This point is also stored into DBUFER, and since II = IDQ, the sequence ends by finally storing the circle last point (x_2, y_2) into DBUFER. The counter KTR is set to give the number of points stored into DBUFER; in the example above, KTR = 3.

The function of subroutine QUADET is now completed, and program flow returns to subroutine GOCIRC. At this point in the program the circle segments have been determined, and now all that has to be done is to output the segments. This is normally a simple process, but there are conditions which can arise to make the process more complex. In general, and for nearly all cases, the points can be made direct output from the array DBUFER. The complicating conditions which rarely arise are described in the special Section 3.4.4.1.

Before outputting the points in DBUFER, subroutine GOCIRC first computes the command block code CRCODE as:

CRCODE = 10, if the circle is in the XY plane.

CRCODE = 11, if the circle is in the ZX plane.

CRCODE = 12, if the circle is in the YZ plane.

CRCODE is a positive value if the circle direction is CLW and negative if CCLW.

If the circle plane had changed from the previous plane, the postprocessor outputs the new plane selection G code. For example, if the present circle were in the YZ plane when previously all circles were in the XY plane, then the plane selection G code (TABLEG(20)) would be made output. Subroutine PLNSEL performs this function.

When these preliminary preparations are completed, program flow goes to subroutine PROCQD to process and output the quadrant segments. In our example the array DBUFER contains the three points:

$$(x_A, y_A)$$
$$(x_B, y_B)$$
$$(x_2, y_2);\ KTR = 3$$

3.4.4 PROCESSING A CIRCULAR INTERPOLATION MOTION (cont'd)

The sequence in subroutine PROCQD selects the point (x_A, y_A) and stores it into DPRESP; next, subroutine GEOM is called which rounds and truncates the values, and stores it into DPRESM. Subroutine DEPART is called to compute the departures. It is evident that the processing thus far is simply that for a linear move as described in detail in Section 3.4.3.1. A special test is made on the departures (see Section 3.4.4.1), and if the test is passed, the command block is readied for output.

If an SFM mode exists, the postprocessor reroutes program flow to the SFM sequence (see Section 4.5) which ultimately outputs the segment. The flag SFMCIR is set to 1 to specify to the SFM sequence that a circle segment is being processed.

Before the command block DBFSEG can be made output, additional items must be added when the block is for a circular interpolation move. These items are the arc center offsets and are stored in DBFSEG(8), (9), (10).

The arc center offsets are the axial distances from the circle center to the beginning point of the circle. In the diagram the arc center offsets are the distances I and J. In general, I= $|x_c - x_1|$, J = $|y_c - y_1|$, K = $|z_c - z_1|$.

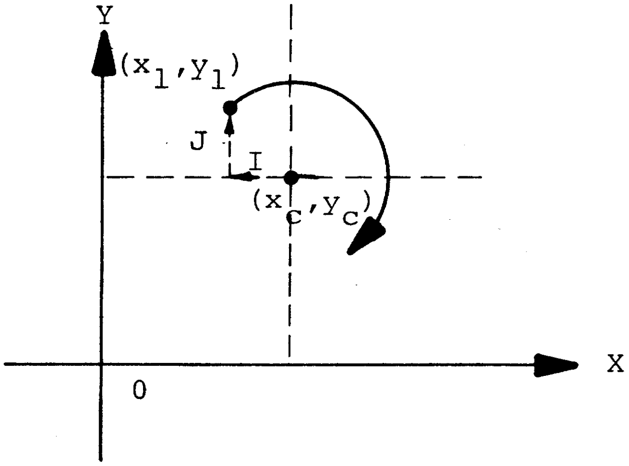


Diagram 3.4.4H

Subroutine OFFARC is the subroutine which computes the arc center offsets and stores them into DBFSEG. The offset values are rounded and truncated by subroutine SRAREC before being stored into DBFSEG.

3.4.4 PROCESSING A CIRCULAR INTERPOLATION MOTION (cont'd)

The command block is now ready for output. CODE is set to CRCODE, and subroutine OUTPUT is called to print and punch the blocks. DPREVP is set to DPRESP, DPREVM is set to DPRESM, and the process is repeated with DPRESP selecting the next point from DBUFER. The sequence repeats for KTR times.

When subroutine PROCQD completes its function, it returns to subroutine GOCIRC which sets the return flag RETURN to + 1, and returns to subroutine MOTION which returns to subroutine GEBASE. In GEBASE the return flag is tested, and since it is +1, program flow is directed to the internal sequence which begins interrogation of the ~~CL~~DATA cL array. There is no need to read in a new record at this time because the next record already exists in the ~~CL~~DATA cL array; this is the record which was the non-continuation (subtype 6) record which signaled an end to the circle data in subroutine GOCIRC.

3.4.4.1 SPECIAL CASE CONDITIONS

There are two special conditions which can result during a circular interpolation sequence that requires special testing and, if necessary, special treatment to alleviate the potential error. Both type errors can occur only on the beginning or end points of the circle. A description of each condition and the solution produced is detailed below.

The first kind of error that can occur is when a circle begins or ends just short of an axis, as illustrated in the diagrams below:

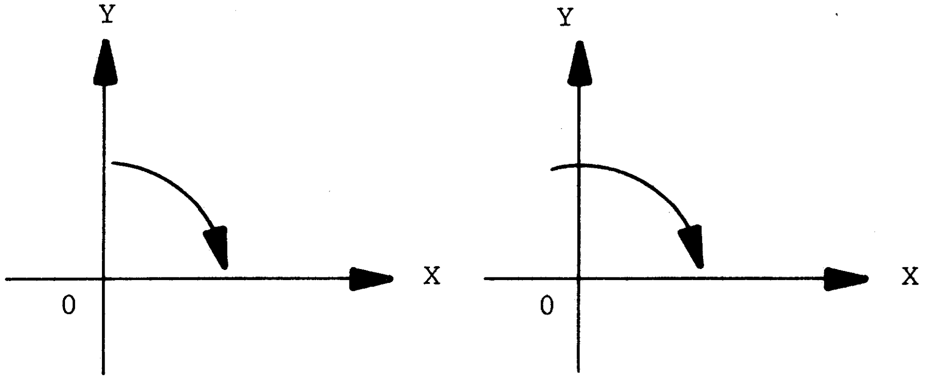


Diagram 3.4.4.1A

3.4.4.1 SPECIAL CASE CONDITIONS (cont'd)

These short distances can result as a function of the programmed tolerance, i.e., the tool, instead of landing exactly on the axis, goes beyond or falls short of the axis by a distance which is acceptable since it is within the programmed tolerance. If the distance should be very small, it can, in effect, produce a departure of zero as demonstrated by the example below where a blown-up view of the circle segment is illustrated.

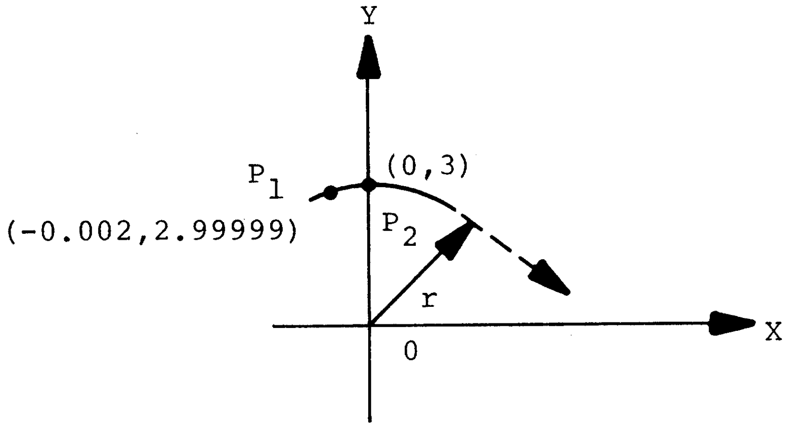


Diagram 3.4.4.1B

The beginning point P_1 of the circle is x = -0.002, y = 2.99999, and the quadrant intersection point P_2 is x = 0, y = 3. After subroutine SRAREC rounds and truncates to the step size, the values become:

$$P_1 = (-0.002, 3); \; P_2 = (0, 3).$$

The departures then are:

$$\Delta X = 0.002, \quad \Delta Y = 0.$$

A zero departure is unacceptable to the NC control system when the move is in the circular interpolation mode. If such an input is made to the control, an apparent machine stop occurs; actually, the halt is a dwell which can be of short duration or as long as several hours!

A zero departure never occurs while in circular interpolation, for at each quadrant intersection point there is always a non-zero X and Y departure. A zero departure can occur only as described above, i.e., at the beginning or end point of the circle, hence, that is the reason the test for a zero departure is made only on these points.

3.4.4.1 SPECIAL CASE CONDITIONS (Cont'd)

The solution to this problem is simple. After subroutine DEPART computes the departures, a test is made to see if either departure is zero. When either departure is zero, the postprocessor is directed to use linear interpolation for that segment only. The internal flag QDMODE is set to zero, thereby, specifying linear interpolation. CODE is set to zero, and subroutine OFFARC is bypassed; this, in effect, causes the segment to be output as a linear rather than as a circular interpolation move.

The other error condition which can occur in a circular interpolation mode is one resulting from APT Section II linear data for a circle. Diagram 3.4.4.1C illustrates how linear cut vectors can approximate a circle path. The only requirement is that each cut vector be within the tolerance band. Depending on the part geometry and the tolerances specified, several possible cut vector sequences can occur as illustrated in the diagram by cut sequence A and B. The fact that the end point of the approximating linear cut vectors can be anywhere within the tolerance band can cause an erroneous effect with the last point of the circle.

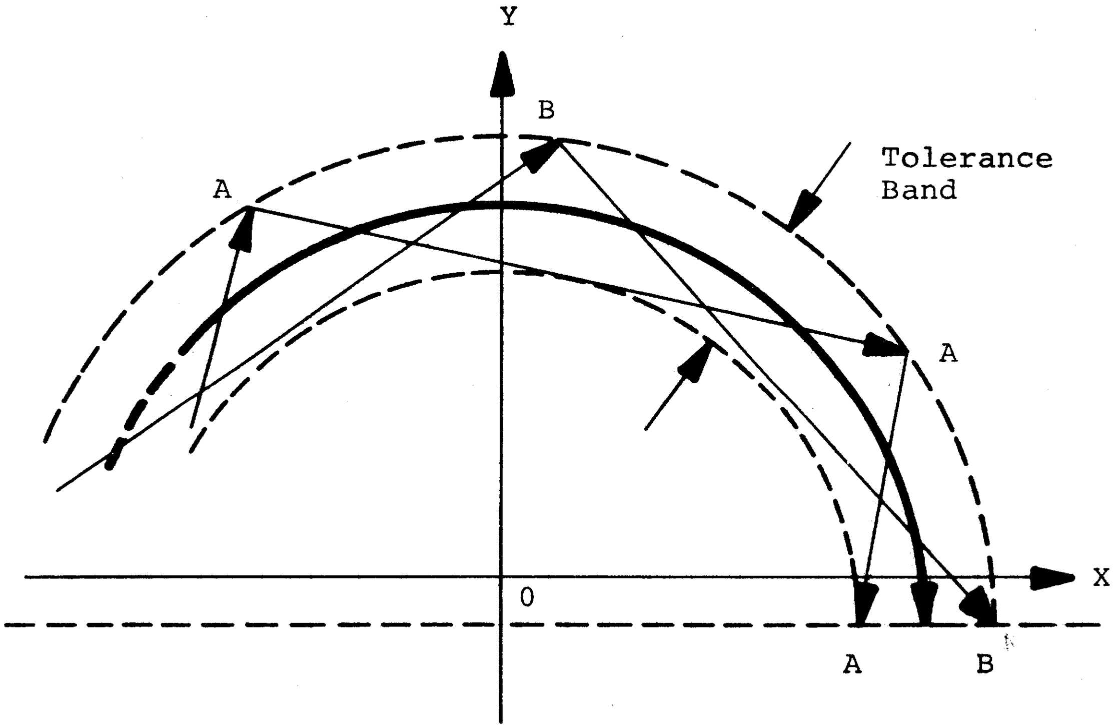


Diagram 3.4.4.1C

3.4.4.1 SPECIAL CASE CONDITIONS (cont'd)

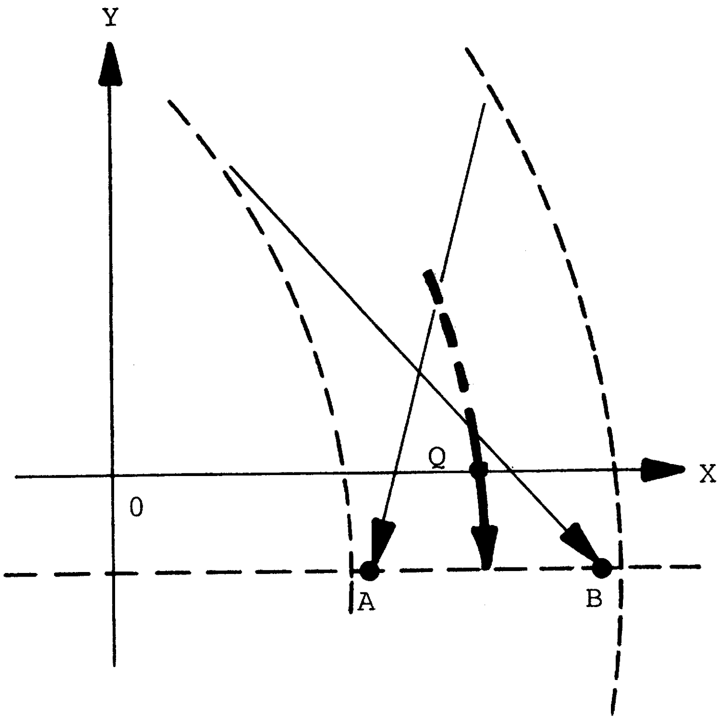


Diagram 3.4.4.1D

It is evident that the circle is of CLW direction. But when the circle is broken into its quadrant segments, it becomes clear that a problem can occur with the fourth quadrant segment. For example, if point B should be the circle end point, there is an apparent change of circle direction when moving from point Q to B, (see Diagram 3.4.4.1D.. The departures and arc center offsets for the fourth quadrant move will be inconsistent with the established circle direction and will, therefore, cause the control system to lose synchronization.

As with the first problem, the solution to this problem is simply to make the move in the linear interpolation mode. This is done in the same manner as described above.

This error condition is detected by comparing the non-zero axis component value at the quadrant intersection point versus the same axis component value at the end point. In the example of Diagram 3.4.4.1D at point Q, x is the non-zero value while y is zero. Therefore, Q_x is compared with B_x; and if B_x is not less than Q_x, the error is known to exist, for it is easily seen that B_x infers that the circle direction is CCLW and not CLW.

3.4.4.1 SPECIAL CASE CONDITIONS (cont'd)

The above mentioned test is done only after the circle has been translated to the origin; consequently, for all circles, regardless of direction, quadrant, or plane, the absolute non-zero axis component value must be larger than the corresponding axis absolute value at the circle end point; otherwise the error exists.

3.4.5 PROCESSING A ROTARY MOTION

As with a linear move, it is possible to have an absolute rotary move or an incremental rotary move. Positioning machines with a rotary table* will generally utilize an absolute system for the table which means that the value loaded into the table register is the actual location for the table to be moved. Contouring machines with a rotary device (table, head, column, and so on) utilize an incremental system which means that the value loaded into the rotary device register is the amount of motion to be made. An example of each move is illustrated below.

* The reference here is to a table which is dimensionally programmed, executable through an A, B, or C register. Indexers and devices which obtain their motion through miscellaneous function of other codes do not pertain.

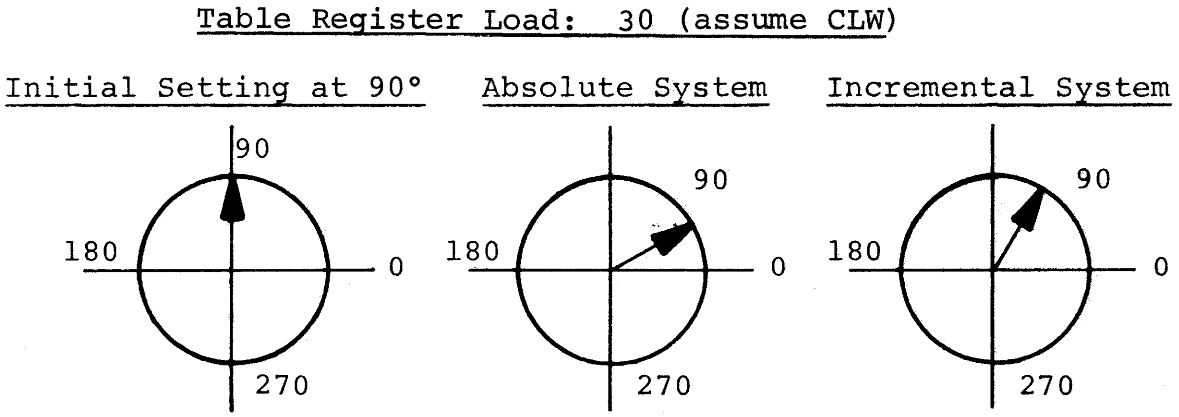


Table Register Load: 30 (assume CLW)

Initial Setting at 90°

Absolute System

Incremental System

Diagram 3.4.5A

3.4.5 PROCESSING A ROTARY MOTION (cont'd)

Subroutine ROTABL handles the rotary sequences for all such rotary systems regardless of the type of rotary device (table, head, and so on). From subroutine ROTABL the two major rotary processing subroutines are called, viz., subroutine ROTABA for an absolute system, or subroutine ROTABI for an incremental system. Each of these sequences is discussed in the next two following sections.

Fundamental to both of these subroutines, however, is the technique of obtaining both the incremental amount of move and the resultant absolute location after the move. The basic postprocessor statements (assumed a table)

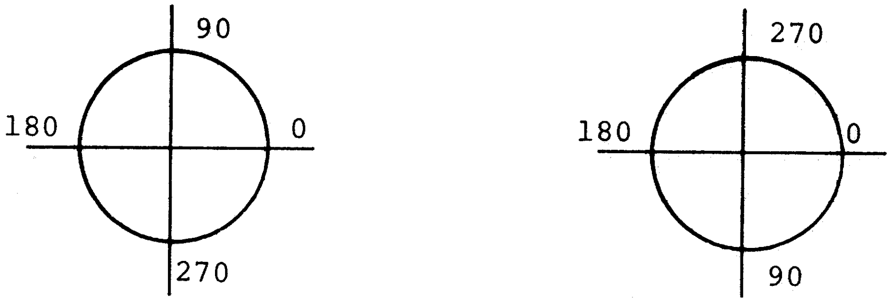
ROTATE/TABLE, INCR, α or ROTATE/TABLE, ATANGL, β

can be used for either an absolute or an incremental system. For either statement, the incremental amount ROTRAD and the absolute location ROTPOS are found. The absolute location must always be known in order to permit the use of the ATANGL or ROTREF modifiers. (See Section 3.4.7.2 for the use of option 29 which precludes the determination of ROTPOS.) The previous absolute location PRVPOS is also set before exiting from the subroutine.

The first item done in subroutines ROTABA and ROTABI is to establish the absolute location. The flag INCABS, which was set in subroutine ROTABL, specifies the nature of the given ROTATE statement, i.e., as to whether it gave an incremental move (α) or an absolute location (ATANGL, β). If INCABS is 0, an incremental move was given therefore, ROTRAD = CLDATA(5) and ROTPOS = PRVPOS + ROTRAD. If INCABS is +1, an absolute move was given, therefore, ROTPOS = CLDATA(6) and ROTRAD = ROTPOS - PRVPOS.

These determinations are, in effect, that which occur in the subroutines, but the precise manner is slightly different. For example, the table rotation direction can cause ROTRAD to be subtracted instead of added to PRVPOS in order to get ROTPOS. Also, consideration must be given as to how the table is absolutely scaled, that is, CLW or CCLW and also as to what direction of rotary move will cause an increase in the reading on the rotary scale. See the following diagrams.

3.4.5 PROCESSING A ROTARY MOTION (cont'd)



Scaled CCLW and off of the table

Scaled CLW and on the table

Diagram 3.4.5B

The postprocessor keeps the absolute location ROTPOS (or PRVPOS) always less than 360 degrees, i.e.,

ROTPOS = ROTPOS modulus 360.

And when ROTPOS becomes 360, the postprocessor resets it to zero. For consistency the postprocessor also keeps ROTPOS a positive value, and always treats negative angles as their positive complement, e.g., -40° is made 320°.

Although the rotation value (α or β) given in the ROTATE statement is in degrees, the output value may be in another form. Option 118 specifies the form of output.

For example, the output rotation value may be in terms of 100 parts per revolution; therefore, 90 degrees would be output as 25; 180 degrees as 50, and so on. Subroutine CONROT takes care of the conversion and anticonversion. The calling sequence for CONROT is:

CALL CONROT (VALUE, N),

where VALUE is the item of conversion, and N is plus or minus one. If N = + 1, VALUE is in degrees and is to be converted to output units; if N = - 1, VALUE is in output units and is to be converted to degrees.

The output value of rotation R is always rounded and truncated as a function of the rotation step size which is given by option 119.

3.4.5 PROCESSING A ROTARY MOTION (cont'd)

The rounding and truncation is determined in subroutine SROREC by the relation:

$$R = \left\langle \frac{R}{\text{OPTION}(119)} + 0.5001 * \frac{|R|}{R} * \text{OPTION}(119) \right\rangle$$

For example: OPTION(119) = 0.001, R = 10.246890. Then after subroutine SROREC, R = 10.247.

As with linear motions, there is a maximum allowable rotary move which requires that rotations greater than the rotary maximum must be segmented into sufficiently small rotations. The parameter ROTMAX carries the maximum and is obtained from option 111. Option 111 is given in degrees, but in subroutine ASSIGN the parameter ROTMAX is converted to output units.

The segmentation sequence for rotary moves must ensure that there is no loss of accuracy due to truncation or rounding. For example, a rotation of 90 degrees can be segmented in three 30 degree moves; in output units assume 360° = (revolution). These 30 degree values are equal to 0.08333 when rounded and truncated to a step size of 0.0001. Thus, we have

$$\begin{array}{ccl} 30 & = & 0.08333 \\ 30 & = & 0.08333 \\ \underline{30} & = & \underline{0.08333} \\ 90^\circ & & 0.24999 \neq 90^\circ. \end{array}$$

Therefore, it is apparent that segmentation must be done with the values in output units, and that a recovery sequence is required which adds in the potentially lost pulse at the proper time.

The postprocessor utilizes such a technique; and for the above example would produce values such that:

$$\begin{array}{ccl} 30 & = & 0.08333 \\ 30 & = & 0.08334 \\ \underline{30} & = & \underline{0.08333} \\ 90^\circ & = & 0.25000 \; = \; 90^\circ \end{array}$$

The technique is described in detail in Section 3.4.5.3.

3.4.5.1 ROTARY ABSOLUTE SYSTEM PROCESSING

The main processing subroutine for all rotary absolute systems is subroutine ROTABA which is in GEPOS. This subroutine processes rotary moves which are incremental or absolute, i.e., which evolve from the statements:

ROTATE/TABLE, INCR, α or ROTATE/TABLE, ATANGL, β .

The output of the motion for an absolute system is always the value of the table location itself. For example, the statement:

ROTATE/TABLE, ATANGL, 23, CLW

causes the output of the value 23 for the rotary table register. If another statement follows the above, such as

ROTATE/TABLE, INCR, 10, CLW ,

the output value is 33, the new absolute location.

Absolute system NC machines often have a rotary table which moves to its programmed point in the shortest direction. For example, if the table is sitting at 90 and is directed to go to 180, the table moves CCLW to 180 since this is the shortest route.

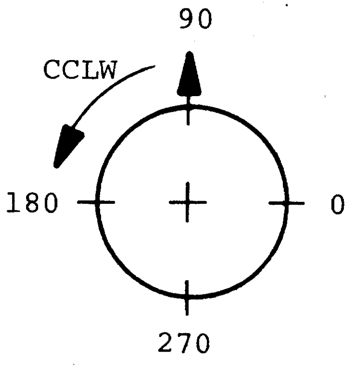


Diagram 3.4.5.1A

The postprocessor permits the part programmer to specify the direction of rotation, and should he choose the direction which is the longest route, the postprocessor effects this by producing two moves in the direction specified. In the example above, suppose the statement given was:

ROTATE/TABLE, ATANGL, 180, CLW

The postprocessor produces the two moves: (1) rotate to absolute angle 271 ° ; (2) rotate to 180° . This in effect causes a CLW rotation to the 180° position.

3.4.5.1 ROTARY ABSOLUTE SYSTEM PROCESSING (cont'd)

Note that option 117 plays an important part in this sequence since it is essential to know the direction of the scale.

A rotary table on a positioning machine can have its own feedrate register which is separate and distinct from the linear feedrate register. Option 139 indicates the existence and location of such a rotary feedrate register. For example, option 139 = 16 specifies that the sixteenth cell of DBFSEG is the rotary feedrate register location, and this is where the rotary feedrate is stored.

If there is a rotary feedrate, option 141 tells what type of feedrate it is; and the appropriate subroutine is called to convert the current feedrate to the proper form required for the rotary feedrate output.

The rotary position value is stored in DBFSEG(6) if the rotary device is a head, or in DBFSEG(7) if the rotary device is a table. Subroutine SET12 is called to obtain the current positioning mode G code, if any.

The command block CODE for an absolute system rotary move is +2, and this CODE is set prior to calling subroutine OUTPUT which ultimately prints and punches the block.

The table may also have its own rapid requirements such as an M code which establishes the rapid and feed condition. If so, and a RAPID is given, the postprocessor outputs the M code which puts the table in the rapid mode (TABLEM(42)). Then, before exiting from subroutine ROTABA, it outputs the M code which puts the table back into its feed mode (TABLEM(43)). The RAPFLG is set to zero since rapid for a table is considered to be one-shot only.

3.4.5.2 ROTARY INCREMENTAL SYSTEM PROCESSING

The main processing subroutine for all rotary incremental systems is subroutine ROTABI which is in GEMILL. This subroutine processes rotary moves which are incremental or absolute, i.e., which evolve from the statements:

ROTATE/TABLE, INCR, α or ROTATE/TABLE, ATANGL, β.

The output of the motion for an incremental system is always the departure from the previous to the new table location. For example, if the table is positioned at location 10, the statement:

ROTATE/TABLE, ATANGL, 23, CLW

causes the output of the value 13 for the rotary table register. If another statement follows the above, such as:

ROTATE/TABLE, INCR, 10, CLW

the output value is 10, the incremental distance from the previous to the new table location which is now 33.

The incremental amount of move ROTRAD is determined in subroutine ROTABI. If the modifier was INCR, then ROTRAD = α; if the modifier was ATANGL, then ROTRAD = β - PRVPOS. The sense of rotary direction ROTDIR is important to the value of ROTRAD and ROTPOS. The direction flag ROTDIR has the following meanings:

ROTDIR = - 1, CLW

ROTDIR = + 1, CCLW

If the rotation is an incremental move (INCR) and no rotary direction is specified, the postprocessor assumes CLW. If the rotation is an absolute move (ATANGL) and no rotary direction is specified, the postprocessor uses the minimum direction.

After the amount of move ROTRAD has been determined, it is tested versus the maximum rotary departure ROTMAX, and if greater, the rotary amount is segmented into sufficiently small rotations; see Section 3.4.5.3.

Subroutine ROTOUT is called to output the command block. It is in this subroutine that the incremental move is corrected per Section 3.4.5.3 and stored into DBFSEG(N), where N = 6 if the rotation is for a head, or N = 7 if the rotation is for a table.

3.4.5.2 ROTARY INCREMENTAL SYSTEM PROCESSING (cont'd.)

The command block CODE is set to - 2 for an incremental rotary move. Subroutine OUTPUT, before printing and punching, obtains the required preparatory function G code (stored in DBFSEG(2)), and the current feedrate (stored in DBFSEG(11)). See Section 3.4.6 for the description of G code selection.

The postprocessor retains the rotary absolute position if option 29 so specifies, hence:

DPREVM(N) = DPRESM(N)

in order to keep the previous machine point vector accurate.

3.4.5.3 SEGMENTATION OF A ROTARY MOVE

Rotary moves utilize an accurate rounding technique similar to that used by linear moves. In addition to rounding each rotary departure to the step size, the difference between the true move and the rounded move is saved and accumulated on each move. When the absolute value of the difference is equal to or greater than one half the step size, the difference is rounded up to the step size (maintaining the sign of the difference) and added to the rotary move. Then the rotary difference is reset to the difference between the true and the rounded differences.

The following example will clarify the problem.

Six rotary moves of 30° or .08333333 decimal parts of a revolution are to be output.

Let R be the true rotary departure;

R´ be the rounded rotary departure;

D be the true difference between the true rotary departure and the rounded rotary departure;

D´ be the rounded value of D.

Step size = .0001 inches.

3.4.5.3 SEGMENTATION OF A ROTARY MOVE (cont'd.)

(1) $R_1 = .08333333$

Rounding R_1 gives

$R_1' = .0833$

The difference between R_1 and R_1' is

$D_1 = .00003333$

Rounding D_1 gives

$D_1' = 0.0$

The departure of .0833 is output.

(2) $R_2 = .08333333$

$R_2' = .0833$

The difference between R_2 and R_2' plus the previous D_1 gives

$D_2 = .00006666$

Rounding D_2 gives

$D_2' = .0001$

D_2' is added to R_2' making the rotary move .0834.

Now D_2 becomes - .00003334 which is the difference between the old D_2 and D_2'.

(3) $R_3 = .08333333$

$R_3' = .0833$

$D_3 =-.00000001$

$D_3' = 0.0$

A rotary departure of .0833 is output.

3.4.5.3 SEGMENTATION OF A ROTARY MOVE (cont'd.)

(4) R_4 = .08333333

R_4' = .0833

D_4 = .00003332

D_4' = 0.0

.0833 is output.

(5) R_5 = .08333333

R_5' = .0833

D_5 = .00006665

D_5' = .0001

The rotary move becomes

$D_5' + R_5'$ = .0834

D_5 is reset to - .00003335

(6) R_6 = .08333333

R_6' = .0833

D_6 =-.00000002

D_6' = 0.0

.0833 is output.

The example shows that the absolute rotary position will always be within half the step size. Greater accuracy than this is not possible.

3.4.6 SELECTING THE PREPARATORY FUNCTION G CODE

An NC contouring machine which has an incremental system moves the tool according to the axis departures given in each command block. Thus, the move given by

$$\Delta X = -2, \quad \Delta Y = 3$$

moves the tool to a point reached by moving the X axis 2 inches in the negative direction and the Y axis 3 inches in the positive direction. Each motion block must have a preparatory function G code which not only tells the control system what type of mode exists (linear or circular), but also the dimensional magnitude of the move. In most cases, the numerical control system does not require that the G code be given in each block after the mode is once established. The postprocessor, however, sets up each command block with the current G code, and depending upon Option 38, may suppress redundancies.

Subroutine SELG selects the necessary G code for linear interpolation moves, whereas subroutine SELGCR selects the G code for circular interpolation and subroutine SELGRO selects the G code for rotary moves; subroutines SELGCR and SELGRO are each called from subroutine SELG.

In addition to selecting the necessary G code, each subroutine also determines the proper dimension multiplier GDIMUL which is used to determine the feed command for that block value; see Section 4.1.1. The range of magnitudes covered by the G codes available on the Mark Century numerical control can extend from 0.00001 to 9999.9999 inches. Most control systems do not have such a wide range.

Since a particular value G code can mean a different magnitude from one numerical control system to another, the postprocessor assigns a table location to a specific range of magnitude. For example, depending on the NC machine, a G12 can be either the range from 0.0001 to 0.0999 or the range from 100 to 999.999. There is no confusion, however, if we refer to a TABLEG location to identify a particular range since any value can be stored there.

3.4.6 SELECTING THE PREPARATORY FUNCTION G CODE (cont'd)

Subroutines SELG and SELGCR make use of the departure limit array RNGDEP ~~DEPLIM~~ which is set up in subroutine ASSIGN or GEINIT. The array is set up for either an English or metric system as designated by option 138.

DEPLIM

Inch		Metric*
0.1	(1)	1
1	(2)	10
10	(3)	100
100	(4)	1000
1000	(5)	Option (4)

The above table is used by both the linear and circular interpolation sequences in selecting the proper G code since the table defines the range magnitude assignable to each of the available G codes. In all cases the smallest magnitude G code that encompasses all the comparison values is used, i.e., the magnitude must be less than or equal to the largest dimension of the compared values.

For example, assume a G01 and G10 exist as defined below. Then a linear move of $\Delta = 8$ inches causes the selection G01 for the range of moves 0.1 to 9.9999 inches, whereas a $\Delta = 80$ inches causes the selection of G10 for the range 10.0 to 99.9999 inches; however, the G10 is compatible with both the $\Delta = 8$ and $\Delta = 80$. Thus, if

$$\Delta X = 8 \text{ and } \Delta Y = 80,$$

then, the G10 must be used since the magnitude covers both ΔX and ΔY. But this is not true for the G01 since $\Delta Y = 80$ is beyond the range of the G01.

* The array for the metric system can be different than shown since the values are a function of the available G codes and the type of control selected by option 165. The listing of subroutine ASSIGN should be referenced for this information.

3. .6. SELECTING THE PREPARATORY FUNCTION G CODE (cont'd.)

An important feature to note here is that if a G01 as defined was the only G code dimension available, all moves greater than 9.9999 inches would have to be segmented. Thus, the maximum departure (option 4) and the maximum magnitude must be identical.

Since a larger magnitude G code can be compatible with smaller dimensions, the question may be raised; why not always use the largest dimension G code and do away with the other dimensions? The reason is because higher feedrates are possible using the lower dimension G codes. This is made evident in Section 4.1.1.1 wherein a discussion of the feed command illustrates the affect of the G code.

The converse attempt to obtain higher feedrates by using small dimension G codes is precluded because of the loss of significant digits in the motion value by the numerical control system, e.g., a move of 23.2468 with a G01 results in the loss of the leading digit. Hence, instead of obtaining the given move, the actual move is 3.2468.

Related to each G code is a so-called "dimensional multiplier" which is a dimensional constant used in determining the feed command; see Section 4.1.1.1. This constant, as used in the feed command formula, is multiplied by 10. But the postprocessor, in order to economize on time and space, interprets the resultant product as if it were the actual value of the dimension multiplier.

For example, for a magnitude range of 0.1 to 9.9999 inches, the dimension multiplier is 1; dimension multiplier value is always multiplied by 10, the postprocessor interprets the term as the value 10 instead of 1. The parameter GDIMUL carries this value.

Although the postprocessor scans the tables in search of an available G code, it uses only those G codes which are actually available. An unavailable G code is indicated by DMBITS being stored at the related TABLEG location.

In the examples given in the following sections, the inch system is used exclusively, but the metric system is similar. The only difference between the two systems is the table of RNGDEP values, otherwise, all processing methods are identical except that a modification to the dimension multiplier is sometimes necessary.

3.4.6.1 SELECTING THE G CODE FOR A LINEAR MOVE

The selection of the linear G code resides completely in subroutine SELG where the departures stored in DBFSEG(3), (4) and (5) are compared versus the RNGDEP values, and a decision is made as to the proper G code to select; after selection, the G code is stored in DBFSEG(2).

The linear interpolation range of magnitudes and their table locations are as follows:

Linear Interpolation

Range	TABLEG
0.01 - 0.09999 inches	(13)
0.1 - 0.9999	(12)
1 - 9.9999	(2)
10 - 99.9999	(11)
100 - 999.9999	(14)
1000 - 9999.9999	(15)

Related to the above table is the departure limit table (see above) which indicates the selection of the G code for a given move.

If the Departure is,	use TABLEG	DIMULT
< RNGDEP(1)	(13)	0.1
< RNGDEP(2)	(12)	1
< RNGDEP(3)	(2)	10
< RNGDEP(4)	(11)	100
< RNGDEP(5)	(14)	1000
> RNGDEP(5)	(15)	10000

3.4.6.1 SELECTING THE G CODE FOR A LINEAR MOVE (cont'd.)

The examples below illustrate the use of the tables. Refer to the RNGDEP table given in Section 3.4.6. The examples also assume that all the dimensional G codes are available, but this is not normally true in actual practice.

Example 1: $\Delta X = 0.2$, $\Delta Y = 2$, $\Delta Z = 20$. ΔZ is the largest departure, and since

$$\mathrm{RNGDEP}(3) < \Delta Z < \mathrm{RNGDEP}(4),$$

the postprocessor uses the G code stored at TABLEG(11) and a GDIMUL = 100.

Example 2: $\Delta X = -0.001$, $\Delta Y = 0$, $\Delta Z = 0$.

$|\Delta x|$ is the largest departure, and since

$$|\Delta X| < \mathrm{RNGDEP}(1),$$

the postprocessor uses the G code stored at TABLEG(13) and a GDIMUL = 0.1.

Once the postprocessor G code has been determined, it is stored into DBFSEG(2), and the command block is essentially ready for output. If the metric system is in use, some final modifications may have to be made to GDIMUL; these modifications can also be a function of the control type (option 165) as well as of the metric system. The modification is simply to increase the size of GDIMUL to the value required for the metric control system.

The linear interpolation sequence for selecting the preparatory function G code also calls a special feedrate optimizing sequence (option 170) which can produce additional command blocks: see Section 4.1.5.1 for the description of this sequence.

And, finally, if the linear move is a multiaxis move (involving both linear and rotary motions), subroutine SELG calls subroutine SELGRO to select the proper G code for the rotary motion. The selected code for the rotary motion can override the selected code for the linear motion if the linear G code magnitude is not sufficiently large for the largest rotary motion. For example, assume that the following conditions exist:

$$\Delta X = 0.2, \quad \Delta Y = 0.1, \quad \Delta Z = 0.09, \quad \Delta A = 40°$$

3.4.6.1 SELECTING THE G CODE FOR A LINEAR MOVE (cont'd)

Subroutine SELG, in considering the linear departures, selects a G code whose dimensional magnitude is from 0.1 to 0.9999, say, a G01. However, this magnitude is not large enough for ΔA, since the corresponding rotary magnitude of G01 extends only to 35.9999°. Therefore, subroutine SELGRO selects the next larger dimensional code, say G10, which, in fact, extends to 359.9999°. This G code also embraces the linear moves, and though the execution time is now longer, both the linear and rotary moves can be simultaneously processed. See Section 3.4.6.3 for the table of magnitude ranges for rotary moves.

3.4.6.2 SELECTING THE G CODE FOR A CIRCULAR MOVE

Subroutine SELGCR, which is called from subroutine SELG, selects the circular interpolation G code. The criterion for selection is the radius of the circle; i.e., the magnitude of the radius CIRRAD is compared with the RNGDEP values, and a selection is made by selecting that G code whose dimensional magnitude is less than or equal to the circle radius. Actually, there is a double comparison, for once the correct magnitude is found, the subroutine next finds the related G code as a function of the circle direction.

The circular interpolation range of magnitudes by circle directions and their table locations are as follows:

Circular Interpolation

Range	TABLEG(CLW)	TABLEG(CCLW)
0.001-0.09999 inches	(23)	(33)
0.1 -0.9999	(22)	(32)
1 -9.9999	(3)	(4)
10 -99.9999	(21)	(31)
100 -999.9999	(44)	(45)
1000 -9999.9999	(49)	(50)

3.4.6.2 SELECTING THE G CODE FOR A CIRCULAR MOVE (cont'd.)

Related to the above table is the RNGDEP table (see Section 3.4.6) which dictates the selection of the G code for a given radius. The tables given below are for the CLW circles only, but a set for CCLW circles is analogous.

If The Radius is,	use TABLEG(CLW)	GDIMUL
< RNGDEP(1)	(23)	0.1
< RNGDEP(2)	(22)	1
< RNGDEP(3)	(3)	10
< RMGDEP(4)	(21)	100
< RNGDEP(5)	(44)	1000
≥ RNGDEP(5)	(49)	10,000

The examples below illustrate the use of the tables. Refer to the RNGDEP table given in Section 3.4.6. The examples also assume that all the dimensional G codes are available, but this is not normally the case.

Example 1: CIRRAD = 0.08 and CIRDIR is 0.

Since the circle direction is CLW, and

$$\text{CIRRAD} < \text{RNGDEP}(1),$$

the postprocessor uses the G code stored at TABLEG(23) and a GDIMUL of 0.1.

Example 2: CIRRAD = 40 and CIRDIR is 1.

Since the circle direction is CCLW, and

$$\text{RNGDEP}(3) < \text{CIRRAD} < \text{DEPLIM}(4),$$

the postprocessor uses the G code stored at TABLEG(31) and a GDIMUL of 100.

3.4.6.2 SELECTING THE G CODE FOR A CIRCULAR MOVE (cont'd)

Once the proper G code has been determined, it is stored into DBFSEG(2), and the command block is essentially ready for output. The program flow returns to subroutine SELG where a final check is made to see if the circular interpolation move is also a multiaxis move (involving both linear and rotary motions). If so, subroutine SELG calls subroutine SELGRO to select the proper G code for the rotary motion. The selected code for the rotary motion can override the selected code for the circular motion if the circular G code magnitude is not sufficiently large for the largest rotary motion. For example, assume that the following conditions exist:

CIRRAD = 8.67 and A = 40 .

Subroutine SELGCR selects a G code whose dimensional magnitude is from 1 to 9.9999, say, a G01. However, this magnitude is not large enough for ΔA, since the corresponding rotary magnitude of G01 extends only to 35.9999° . This G code also embraces the circular radius, and though the execution time is now longer, both the circular and rotary moves can be simultaneously processed. See Section 3.4.6.3 for the table of magnitude ranges for rotary moves.

3.4.6.3 SELECTING THE G CODE FOR A ROTARY MOVE

The selection of the rotary G code resides completely in subroutine SELGRO which is called from subroutine SELG and from subroutine OUTPUT. A comparison of the rotary departures stored in DBFSEG(6, 7, 18) is made relative to the maximum rotary departure (option 111).

The rotation G code dimensions are somewhat analogous to those for a linear move, and, in fact, use the same TABLEG values and concomitant GDIMUL as do the linear moves. The major difference is that the values are in rotary measure rather than linear measure.

The rotary move range of magnitudes and their table locations can be as follows:

Rotary Motion Example

Range	TABLEG
0-0.3599 degrees	(13)
0.36-3.5999	(12)
3.6-35.9999	(2)
36-359.9999	(11)

In actual testing during postprocessing, the above values are converted to output units since all rotary values are processed in their output form. Note that there is no dimension related to TABLEG(14) as for linear moves. Nor are rotary G codes selected on the basis of the RNGDEP table but rather on the basis of the rotary maximum departure (option 111). This is why the above table is not a fixed set of magnitudes and also why TABLEG(14) is not used. The above table is correct only if the rotary maximum departure is 359.9999.

The general case for the selection of G codes for rotary motions is given here in degrees, but it should be remembered that, in practice, the values are in output units, either degrees, or decimal parts of a revolution.

3.4.6.3 SELECTING THE G CODE FOR A ROTARY MOVE (cont'd.)

If the Rotary Departure is,	use TABLEG	GDIMUL
Maximum Rotation/1000	(13)	0.1
Maximum Rotation/100	(12)	1
Maximum Rotation/10	(2)	10
Maximum Rotation/1	(11)	100

The examples below illustrate the use of the tables, and assume that all the dimensional G codes are available though this is not normally the case. Degrees are assumed to be the output units.

Example 1: $\Delta A = 40°$, $\Delta B = 1°$, $\Delta C = 0.1°$.

Rotary maximum departure: option 111 is 360 .

ΔA is the largest departure and since

$$\frac{\text{option } 111}{10} < \Delta A < \frac{\text{option } 111}{1}$$

the postprocessor uses the G code stored at TABLEG(11) and a GDIMUL = 100.

Example 2: $\Delta A = 0.0001$, $\Delta B = 0.1$, and option 111 = 36°.

$|\Delta B|$ is the largest departure, and since

$$\frac{\text{option } 111}{1000} < |\Delta B| < \frac{\text{option } 111}{100},$$

the postprocessor uses the G code stored at TABLEG(12) and a GDIMUL = 1.

Once the proper G code has been determined, it is stored into DBFSEG(2), and the command block is essentially ready for output. However, if the machine motion is a multiaxis move (involving both linear and rotary motions), the selected code for the rotary move may override the previously selected linear G code if the magnitude of the linear G code is not sufficiently large for the rotary motion. For example, assume that the following conditions exist:

$$\Delta X = 0.2, \quad \Delta Y = 0.1, \quad \Delta Z = 0.09, \quad \Delta A = 40°.$$

3.4.6.3 SELECTING THE G CODE FOR A ROTARY MOVE (cont'd.)

Subroutine SELG, in considering the linear departures, selects a G code whose dimensional magnitude is from 0.1 to 0.9999, say, a G11. However, this magnitude is not large enough for ΔA, since the corresponding rotary magnitude of G11 extends only to 3.5999° Therefore, subroutine SELGRO selects the second larger dimensional code, G10, which extends to 359.9999°. This G code also embraces the linear moves, and though the execution time is now longer, both the linear and rotary moves can be simultaneously processed.

3.4.7 PROCESSING A MULTIAXIS MOTION

Sections 3.4.1 through 3.4.3 detail how a motion record is obtained from the CL tape and stored into the part coordinate present point vector DPRESP; The following description proceeds from that point.

Because of the rotary motions of a multiaxis machine, there is no one-to-one linear correspondence between the part and machine coordinate points as there exists with a linear three-axis machine. But there is a mathematical relationship between the part and machine points, such that the location of a point on the part plus the tool axis orientation at that point can be expressed in terms of the machine's linear and rotary motions. This relationship is the so-called Geometry Package, and the conversion is accomplished through transformation (or class) equations.

Hence, when a part coordinate point (x, y, z, i, j, k) is obtained from the CL tape, it must be converted to its machine coordinate form (X, Y, Z, A, B,C); the converted and rounded valued are stored in the present machine point vector, PRESMP.

The program sequence is as follows:

1. Store new part coordinate point in DPRESP.

2. Subroutine GEOM is called which for multiaxis processing in turn calls subroutine GEOM5.

3. Subroutine GEOM5 calls subroutine CLASS which then branches to the multiaxis geometry package, i.e., to subroutine CLASSn, where n ranges from 1 to 9.

3.4.7 PROCESSING A MULTIAXIS MOTION (cont'd.)

4. Subroutine CLASSn takes the values of DPRESP, and using the equations of transformation, computes the corresponding machine coordinate point.

5. The rounded and truncated point is stored in DPRESM.

From this point on, the postprocessor program flow is basically the same as for three-axis processing except for some special sequences, such as feedrate number determination and linearity testing; these special sequences are discussed in the later sections of this manual.

The flag MAFORK must always be preset before calling the class subroutines. When MAFORK = 1, the inverse transforms are computed, that is, the machine coordinate point is converted to the corresponding part coordinate point. In this case the data in the DPRESM vector is used as input, and the resulting point is stored in the DPRESP vector.

When MAFORK = 2, the direct transforms are computed, that is, the part coordinate point is converted to the corresponding machine coordinate point. In this case the data in the DPRESP vector is used as input, and the resulting point is stored in the DPRESM vector.

DPRESP(1) = x	MAFORK = 2	DPRESM(1) = X
DPRESP(2) = y	$\longrightarrow$	DPRESM(2) = Y
DPRESP(3) = z	MAFORK = 1	DPRESM(3) = Z
DPRESP(4) = i	$\longleftarrow$	DPRESM(4) = A
DPRESP(5) = j		DPRESM(5) = B
DPRESP(6) = k		DPRESM(6) = C

The MAFORK in some class subroutines is used also for other meanings, as, for example, in subroutine CLASS1 when MAFORK = 0, the subroutine selects the loaded tool gripper constants for use in the transform relations.

A special test is made in subroutine FROM5 to ensure that the CL point's direction cosines are valid; that is,

$$\sqrt{i^2 + j^2 + k^2} = 1 \pm \varepsilon,$$

or else a warning comment to this effect is issued.

3.4.7 PROCESSING A MULTIAXIS MOTION (cont'd.)

After the part coordinate data are transformed into the machine coordinate data, subroutine DEPART is called to compute the linear departures ΔX, ΔY, and ΔZ.

Subroutine DEPART calls subroutine ROTMOV in order to compute the rotary departures. An important point to note is that the rotary moves are always kept in terms of their output units. This minimizes the processing time in that no conversion to and from output units is ever required.

Another function performed by subroutine ROTMOV, is to make the rotary moves positive and less than 360 degrees. For example, a value of -400 degrees is made to be 320 degrees. Subroutine ROTMOV puts the rotary departures into DBFSEG(6), (7), and (18). A convention of the postprocessor is that the head register is related to DBFSEG(6), while the table to DBFSEG(7). This is merely a convention and not a set rule.

After checking the linear departures versus the allowable maximum linear departure, similar tests are made with the rotary departures versus the rotary maximum departure. Subroutine SEGMNT is called if any maximum departure is exceeded.

When a segment is acceptable, several flags are tested to determine whether or not linearity testing should be performed. If so, subroutine LINTRY is called upon to produce the requisite number of segments to remove any "linearity" error. See Section 3.4.7.3 for a detailed discussion of this subject.

An important feature to be noted here is that when a departure exceeds the maximum departure and linearity testing is desired, subroutine SEGMNT is not immediately called upon to segment the path length to the necessary segments, but, rather, subroutine LINRTY is used since the expectation is that the path length will be sufficiently segmented in order to correct the "linearity" error.

A multiaxis move has motions both in the rotary and linear axes, but the postprocessor treats the move as if it were simply a linear motion. Therefore, a multiaxis motion command block is still identified by a CODE of zero.

3.4.7 PROCESSING A MULTIAXIS MOTION (cont'd.)

An apparent contradiction can occur in command block identity. Rotary moves by themselves, when generated by a ROTATE statement, have their command blocks identified by CODE = -2. However, it is possible that in a multiaxis motion that Δ X, Δ Y, and Δ Z are zero, and only Δ A or Δ B are nonzero. Yet the command block CODE is still zero. This actually leads to no problem, and it is important that the command block generation source be known; the CODE uniquely identifies the source.

At this point in the program flow for linear multiaxis moves, DBFSEG (3), (4), (5), (6), and (7) are set to their respective ΔX, ΔY, Δ Z, ΔA, Δ B values, and CODE = 0. Subroutine OUTPUT is then called to complete the setup and eventual output of DBFSEG as described in Section 2.4.2.2.

If a third rotary axis exists on the NC machine, the departure ΔC is stored into DRFSEG(18). The third rotary axis is treated exactly the same as the other rotary axes.

Processing of a multiaxis move requires no other special sequences in any of the permissible GEOUT's. In rather routine steps, the rotary motions are converted to an absolute location in degrees for printing in the Absolute Printout. The influence of the rotary motions is considered in other determinations such as the cut time, block read time, feedrate optimization, and so on; but these sequences, in effect, deal with all departures in a standard manner. There is no special branching for multiaxis processing.

3.4.7.1 MULTIAXIS CIRCULAR INTERPOLATION

Circular interpolation for multiaxis moves will involve at least one of the rotary motions as well as at least two linear motions. This requires a determination of the rotary equivalent of "arc center offsets". These are not actually arc center offsets and are referred to as supplementary constants.

After subroutine GOCIRC has determined the axes interception points and stored them in the array DBFSEG, subroutine PROCQD processes and outputs the points. For two or three axes machines each interception point is merely the (x,y,z) coordonate value; but for multiaxis processing, the tool axis vector direction cosines must also be known. Therefore, subroutine PROCQD must determine the (i, j, k) values at each interception point before processing and outputting the point.

3.4.7.1 MULTIAXIS CIRCULAR INTERPOLATION (cont'd.)

In order to find the tool axis vector direction cosines, the postprocessor first translates the circle center to the origin. The two radii V_1 and V_2 (see Diagram 3.4.7.1A) include the angle swept through in the first quadrant from point 1 to point 2.

$$V_1 \cdot V_2 = |V_1| \; |V_2| \; \cos \theta ,$$

$$\text{or } \theta = \cos^{-1} \left(\frac{V_1 \cdot V_2}{R^2} \right).$$

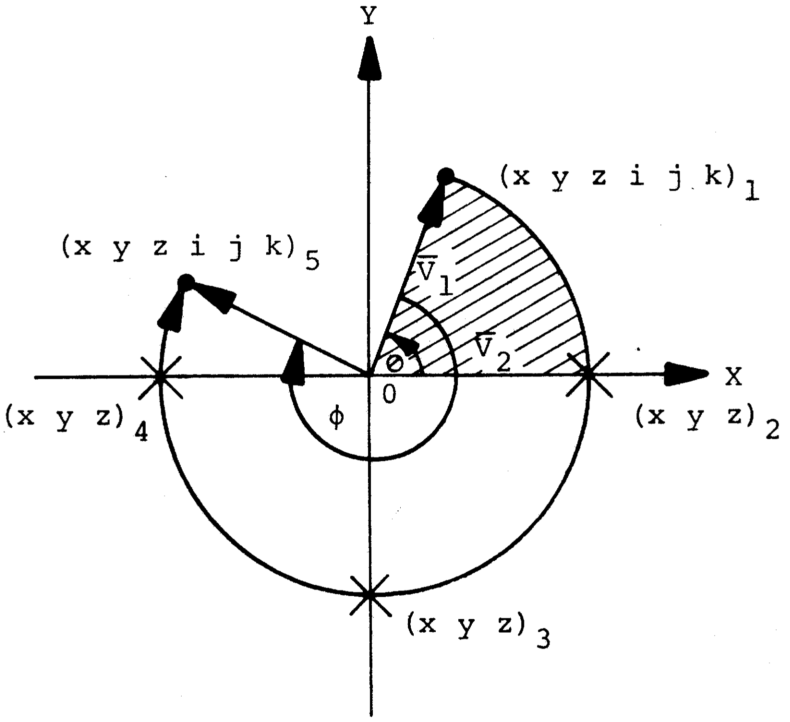


Diagram 3.4.7.1A

Each quadrant sector is treated separately to find the tool axis direction cosines since no circle move can be greater than ninety degrees; see Section 3.4.4. Angle α is the total angle swept through and is found by summating the individual angles from each quadrant. The following terms are computed:

$$\alpha = \frac{\sin (\theta - \gamma)}{\sin \theta}, \quad \gamma = \sum_{i=1}^{4} \theta_i / \phi, \quad \beta = \frac{\sin \gamma}{\sin \theta}.$$

3.4.7.1 MULTIAXIS CIRCULAR INTERPOLATION (cont'd.)

The direction cosines I, J, K are found from:

$$I = \alpha_{i_1} + \beta_{i_5}$$
$$J = \alpha_{j_1} + \beta_{j_5}$$
$$K = \alpha_{k_1} + \beta_{k_5}$$

The computed direction cosines are then normalized.

In addition to generating the tool axis direction cosines, subroutine PROCQD also outputs an information block. (See Section 5.5.) This information is necessary for the determination of the feedrate command for a circular interpolation move.

Multiaxis circular interpolation moves are processed and made output with the same CODE value as for non-multiaxis moves. Hence, at this point in the program flow, DBFSEG(8), (9), (10) to their arc center offset values, and DBFSEG(16) and (17) for the rotary supplementary constants D and E. The value of CODE is ±10, ±11, ±12. Subroutine OUTPUT is then called to complete the setup and eventual output of DBFSEG.

When subroutine OUTPUT calls SELG to obtain the dimensional preparatory function G code, subroutine SELG first obtains the proper G code compatiable with the linear (or circular) moves as described earlier. Then, subroutine SELG calls subroutine SELGRO which accepts the already determined G code if it is compatible with the rotary moves, but if not, subroutine SELGRO obtains a G code compatible with both the linear (or circular) and rotary moves.

3.4.7.2 ROTARY MOTION WITH ROTREF

The ROTREF modifier to a ROTATE statement calls for a rotation of the reference frame, but which frame and how the frame is to be rotated has not been clearly defined. The result has been that several interpretations, some even contradictory, have evolved.

The GECENT III postprocessor considers only two major interpretations, each of which is opposite to the other. The interpretation used is selected by option 198.

3.4.7.2 ROTARY MOTION WITH ROTREF (cont'd.)

In the GECENT III postprocessor ROTREF is one-shot only, and, in meaning, always calls for a rotation of the part reference frame such that after the rotation, the part programmer is permitted to continue operating in his original part reference system. This is the effect, but the problem arises in the interpretation placed upon the modifier ROTREF at the time of application.

The two interpretations used in the GECENT III postprocessor are illustrated below. A cube is to be machined such that each face has an identical cut sequence.

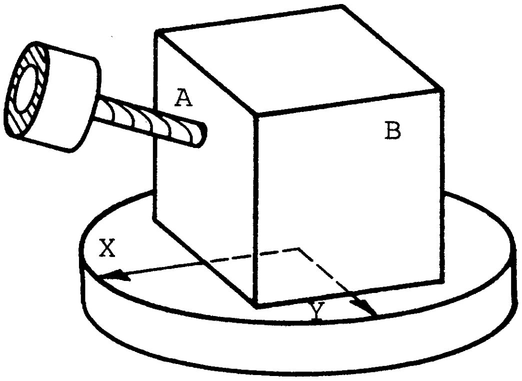


Diagram 3.4.7.2A

After programming face A, the programmer would like to turn the part and program face B with the same geometry, i.e., use identical part program statements as before. However, if the programmer says

ROTATE/TABLE, INCR, 90, CLW

the resulting rotation of the part and table appear as:

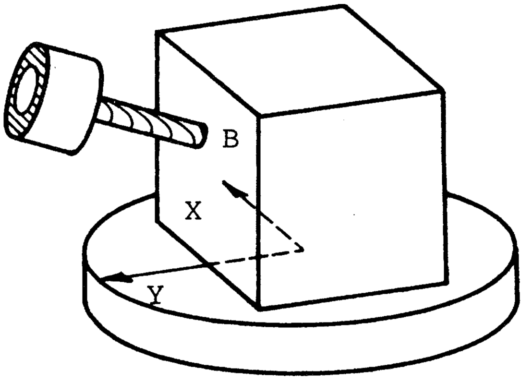


Diagram 3.4.7.2B

3.4.7.2 ROTARY MOTION WITH ROTREF (cont'd.)

This results because the part coordinate system (arrows) are fixed in the part, and, therefore, must move with the rotation.

At this point the tool tip has different direction cosines than did the sequence for face A.

In order to rotate the part coordinate system and the part geometry back to its prior position, the programmer can use the modifier ROTREF which here means:

> Rotate the part coordinate reference frame back to the prior position.

Hence, if instead of the above ROTATE/statement, the programmer had given

```
ROTATE/TABLE, INCR, 90, CLW, ROTREF
```

the result would look like:

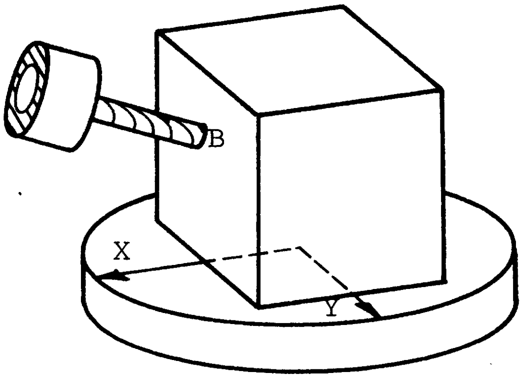


Diagram 3.4.7.2C

In effect no rotation of the part geometry occurred, and the same set of statements used on face A can now be used on face B.

The above explanation is one of two interpretations used by the GECENT III postprocessor which results when option 198 is zero. The direct opposite meaning results when option 198 is non-zero.

3.4.7.2 ROTARY MOTION WITH ROTREF (cont'd.)

This opposite meaning is useful for the case when the part programmer desires to think of the part system as fixed in space immediately above the table, so that it does not move with the part under a rotation.

In such a case, it is convenient for him to interpret ROTREF as meaning "rotate the part system with the part". In such instances, it might be necessary to program ROTREF on almost every rotation statement. Thus, beginning with Diagram 3.4.7.7A and option 198 = 1, Diagram 3.4.7.2B results when the ROTATE statement is given as:

```
ROTATE/TABLE, INCR, 90, CLW, ROTREF.
```

Diagram 3.4.7.2C results from

```
ROTATE/TABLE, INCR, 90, CLW, ROTREF
```

Subroutine ROTABI contains the programming sequence which produces the rotation of the part reference frame for both settings of option 198. In the following example we assume option 198 = 0, but the same method and ideas apply when option 198 = 1 and after allowing for the branching difference.

Let the tool be at the following position:

DPRESP(1) = x_1,	DPRESM(1) = X_1
DPRESP(2) = y_1,	DPRESM(2) = Y_1
DPRESP(3) = z_1,	DPRESM(3) = Z_1
DPRESP(4) = i_1,	DPRESM(4) = A_1
DPRESP(5) = j_1,	DPRESM(5) = B_1
DPRESP(6) = k_1,	DPRESM(6) = ________

Also at this point in the program the vector DPREVP = DPRESP and DPREVM = DPRESM.

The following statement is given

```
ROTATE/TABLE, INCR, 40, CLW, ROTREF.
```

3.4.7.2 ROTARY MOTION WITH ROTREF (cont'd.)

The initial effect upon the position vectors is

DPRESP(1) = x_1,	DPRESM(1) = X_1
DPRESP(2) = y_1,	DPRESM(2) = Y_1
DPRESP(3) = z_1,	DPRESM(3) = Z_1
DPRESP(4) = i_1,	DPRESM(4) = A_1 + 40 = A_2
DPRESP(5) = j_1,	DPRESM(5) = B_1
DPRESP(6) = k_1,	

Since the AXMULT and ROTREF flags = 1, subroutine ROTABI branches to the geometry transforms for the proper class. The flag MAFORK is set to 1 which calls for the inverse transforms, i.e., to convert the machine point to the corresponding part point. Hence,

DPRESP(1) = x_2,	DPRESM(1) = X_1
DPRESP(2) = y_2,	DPRESM(2) = Y_1
DPRESP(3) = z_2,	DPRESM(3) = Z_1
DPRESP(4) = i_2,	DPRESM(4) = A_2
DPRESP(5) = j_2,	DPRESM(5) = B_1
DPRESP(6) = k_2,	

However, we do not want the part coordinate system to rotate as a function of the rotary move. It was essential to obtain the influence of the rotation upon the tool axis setting (direction cosines), but the xyz location was not to change. This is accomplished by resetting the part point to the previous xyz point taken from DPREVP

3.4.7.2 ROTARY MOTION WITH ROTREF (cont'd.)

Therefore,

DPRESP(1) = x_1,	DPRESM(1) = X_1
DPRESP(2) = y_1,	DPRESM(2) = Y_1
DPRESP(3) = z_1,	DPRESM(3) = Z_1
DPRESP(4) = i_2,	DPRESM(4) = A_2
DPRESP(5) = j_2,	DPRESM(5) = B_1
DPRESP(6) = k_2,	

Since there is a change in the part coordinate data, the postprocessor calls subroutine GOLINE so that the new point is updated in both the DPRESP and DPRESM vectors. The requisite move is thereby produced.

The flag ROTREF is set in subroutine ROTABL. The flag AXMULT is set when a MULTAX part program statement is given.

Another option which affects the use of a ROTREF modifier is option 29 which tells the postprocessor whether or not to remember the absolute location of the table. For example, assuming that we start from 0, the statements

```
ROTATE/TABLE, INCR, 10, CLW

ROTATE/TABLE, INCR, 20, CLW

ROTATE/TABLE, INCR, 10, CLW
```

places the table at the 40 degree position. If option 29 is set to 1, the postprocessor remembers the position such that if another statement is given as

```
ROTATE/TABLE, ATANGL, 0, CLW
```

the postprocessor outputs a move of 320 degrees to position the table at 0. It is evident, therefore, that to be able to use the ATANGL or ROTREF modifiers, the postprocessor must remember the table location since the correct increment of rotation derives from the difference between the previous and present points.

3.4.7.2 ROTARY MOTION WITH ROTREF (cont'd)

A part program which wishes to use the ROTATE/TABLE statement merely to index the table to a new position so that a repeated cut sequence can be made, would have option 29 set to zero so that the table location is not remembered and, therefore, would not affect a later multiaxis move. For example, in Diagram 3.4.7.2A suppose we wished to drill a hole at the same spot in each of the four faces of the cube. In this case, the ROTATE statement merely indexes the table ninety degrees. When the drilling operation is completed, the part program can make a multiaxis move which is unaffected (and properly so) by the previous rotations.

It must be remembered that if option 29 is zero, use of the ATANGL and ROTREF modifiers is precluded. If option 29 is 0 and a ROTREF or ATANGL modifier is nevertheless given, the postprocessor prints a warning comment to this effect, and continues as if option 29 were equal to 1.

3.4.7.3 LINEARITY ERROR AND CORRECTION

The so-called linearity error is a direct result of the non-linear motion of the tool tip when there is a simultaneous motion of the linear and rotary axes. There is no linearity error when there are only three linear axes since an error is produced only by a change in the tool axis orientation relative to the part surface. Another type error called the transition error also can result; but this type error is completely resolvable only in APT Section II. See the IITRI report, The Transition Problem, December 27, 1965.

3.4.7.3.1 DESCRIPTION OF PROBLEM

The APT system generates linear cut vectors fitted within given tolerances along the given cutter path, Diagram 3.4.7.3A.

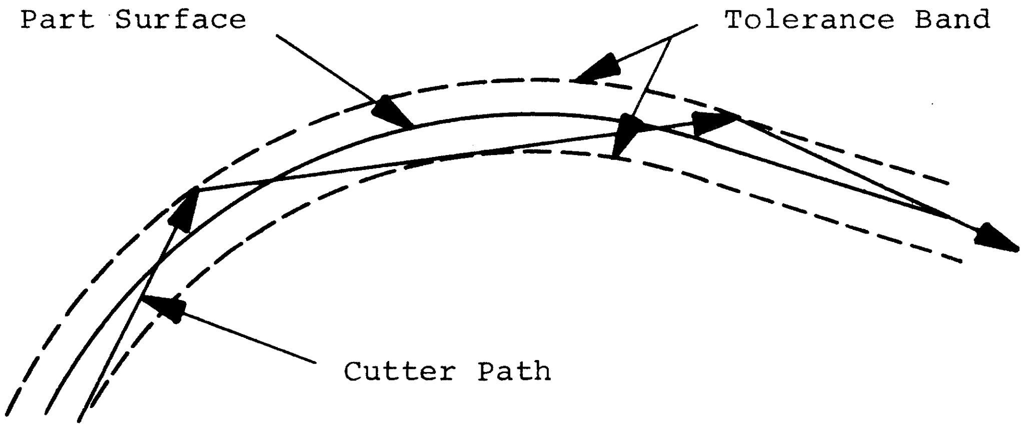


Diagram 3.4.7.3A

]

3.4.7.3.1 DESCRIPTION OF PROBLEM (cont'd)

In order to follow faithfully the required cutter path, the machine tool must follow the generated linear cut vectors within the allowable tolerance. But motions produced by a multiaxis machine with rotary axes result in non-linear motions of the cutter which can place the cutter outside the tolerance limits.

In the Diagram 3.4.7.3B, the tool is to move from point A to point B, and the tool axis is reoriented to a new angle θ. The actual cutting path does not follow the designated linear move from A to B because the tool orientation motions cause the tool end to deviate. However, the deviation may be acceptable if tolerance has not been exceeded.

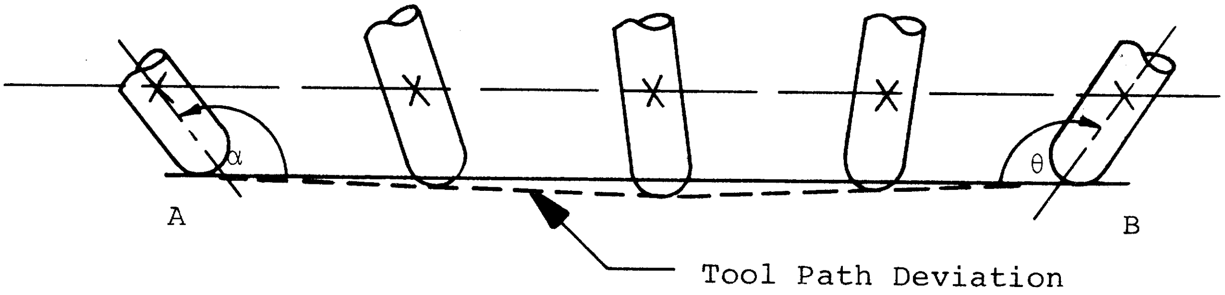


Diagram 3.4.7.3B

1. To determine when nonlinear motions cause the tool to exceed tolerance limits; and

2. To correct the tool's motion so that it stays within tolerance.

The approach taken by the GECENT III postprocessor to resolve the linearity problem is based upon a solution which keeps the actual machine tool path within some given tolerance of a linear interpolation of the tool path. This linearity tolerance is specified in the part program and may be changed as warranted by the cutter path.

3.4.7.3.2 Method of Solution

The specification of a "good" tolerance will dictate the accuracy of the cutter to adhere to the required path. The tolerance in discussion refers to the tolerance limits for determining when the cutter has deviated from the required path.

The given cutter tolerance (INTOL, OUTOL, TOLER), Diagram 3.4.7.3C, cannot be used as a linearity tolerance because it is not sufficient to restrict the tool to the required path; in most cases a finer tolerance is needed. This finer tolerance is called the linearity tolerance and derives from the following conditions.

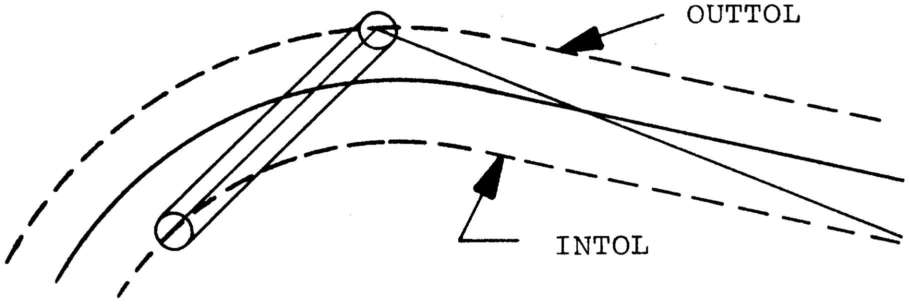


Diagram 3.4.7.3C

A cylinder of radius r (Diagram 3.4.7.3C) is constructed about each cut vector; and any time the cutter path goes outside of the cylinder, a linearity error is assumed. Steps are then taken to correct for the error.

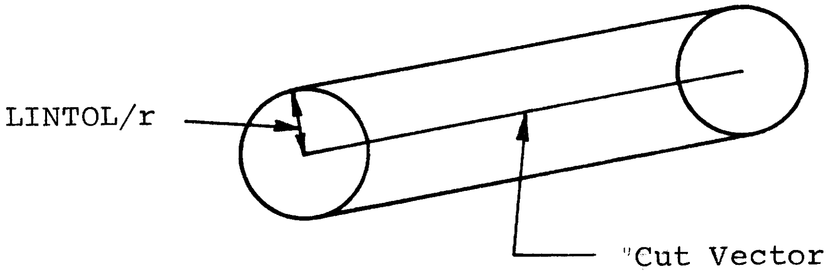


Diagram 3.4.7.3D

3.4.7.3.2 METHOD OF SOLUTION (cont'd.)

This consists of inserting a new point on the cut vector, ie., breaking the cut vector into two smaller segments. The most logical place to make the break is at the point where the linearity test was made, but this may not be the best place. If the error occurs very close to a cut vector end and the break made there, then more problems may arise. A/D limitations on the very short segments or linearity errors may occur on the large segment if the angle change in the tool axis is large. A better place to make the break is at the middle of the cut vector although the same problems may still arise. There is an advantage, however, in making the cut at the middle; viz., the cut segments are of optimum length thereby minimizing the above mentioned problems.

A study by IITRI* indicated two areas where linearity problem errors may occur. One error is due to the failure of the tool axis to orient itself correctly at its final (or inserted) position. For example, in Diagram 3.4.7.3E, when the tool moves from A to B the solid lines indicate the actual tool setting whereas the correct tool setting is the dotted figure. This tool axis variation results from a geometric error derived from the transform relations of the machine tool. The method to prevent this error relies upon a tolerance cone in which the tool axis is allowed to vary. If the tool axis falls outside the cone, a midpoint on the cutter path is inserted.

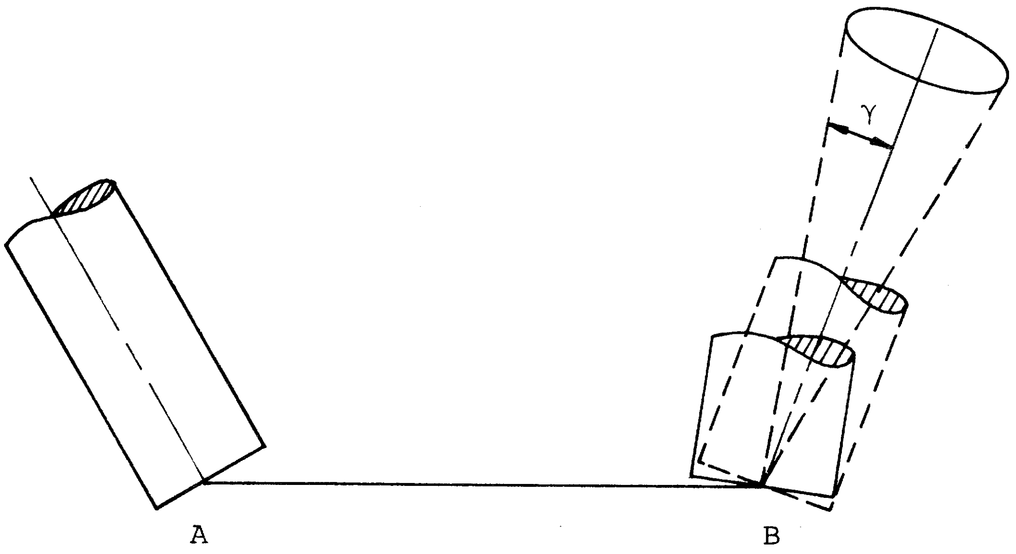


Diagram 3.4.7.3E

* Five-Axis Linearization Study, February 1964.

3.4.7.3.2 METHOD OF SOLUTION (cont'd.)

The second linearity error is path deviation as illustrated in Diagram 3.4.7.3F. Two types are shown: a symmetrical path (curve A) and a nonsymmetrical path (curve B). The symmetrical case can be corrected by inserting a point on the middle of the segment, but the non-symmetrical linearity deviation makes it difficult to apply the midpoint correction with any great degree of accuracy. In such cases the part programmer must tighten the tolerance to ensure linearity correction.

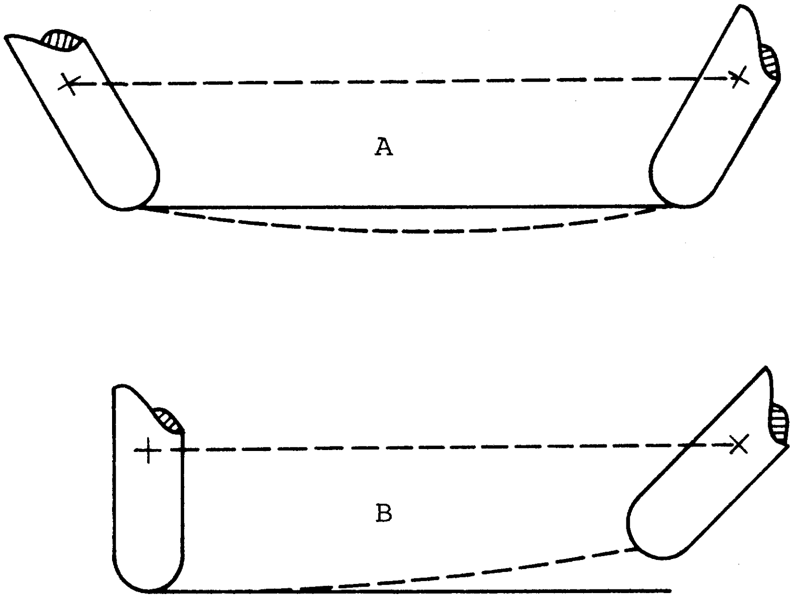


Diagram 3.4.7.3F

3 4.7.3.2 METHOD OF SOLUTION (cont'd)

In curve A, Diagram 3.4.7.3G, the non-symmetry results from the variation of the tool axis at points 1 and 2. In B the tool axis is at a mirror angle at points 1 and 2, and gives a symmetrical linearity deviation. These effects may not be the same (or have the same magnitude) on all types of machine tools.

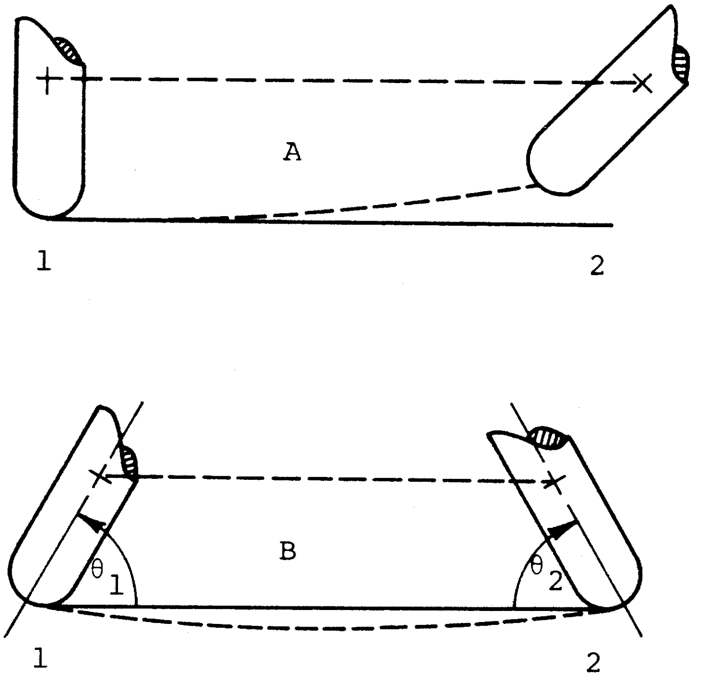


Diagram 3.4.7.3G

The part programmer can call for and control linearity testing by use of the following postprocessor statements.

```
                                          ON
LINTOL/r, γ          and/or         LINTOL/OFF
                                           O
```

3.4.7.3.2 METHOD OF SOLUTION (cont'd)

r is the radius of the linearity tolerance sphere (or cylinder); γ is the half-angle of the tool orientation cone. The part programmer must give the LINTOL statement prior to any motions which are to be linearity tested. Once given, the linearity testing sequence is modal, however, the part programmer may change the values of r and γ at any point in the program. The postprocessor will not perform linearity testing unless and until a LINTOL statement established a value for r and γ. The part programmer need not specify γ if he does not wish tool axis orientation testing (cone testing), but r must always be given. Tool axis orientation testing is important only for flat end mills whereby a gouge can result if the tool axis is not oriented properly. The part programmer can cancel both cone and sphere linearity testing at any point in the part program by giving LINTOL/O. Similarly, cone testing alone can be cancelled by LINTOL/r,O.

There are certain paths over which the postprocessor will automatically disregard linearity testing. Since linearity testing is important only when cutting, the postprocessor therefore excludes all non-cutting paths such as rapid traverse paths, retracts and advances of the tool during a tool changing sequence, and table rotations without a ROTREF.

In addition to these non-cutting paths there is also one type of cutting path over which the part programmer may not wish to have linearity testing, namely, cut paths which are produced as one-point CL tape. Single point records are most likely to give a straight line path with no rotary motions involved.

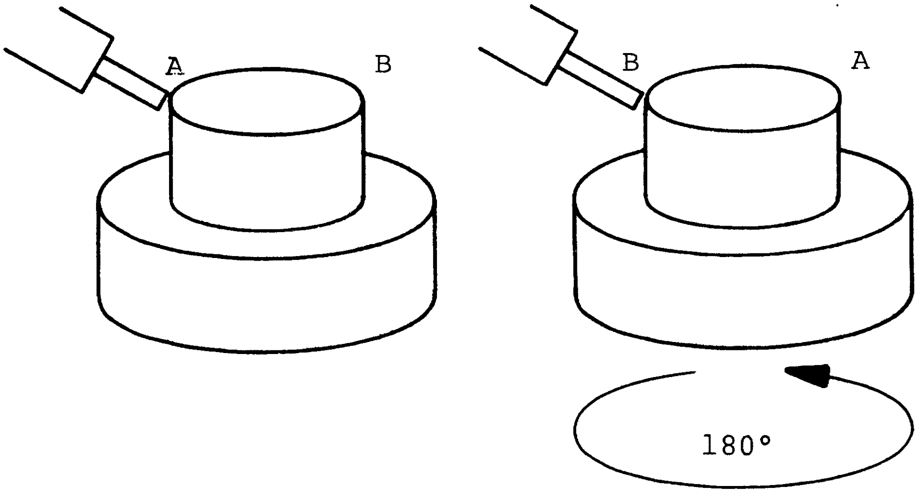


Diagram 3.4.7.3H

3.4.7.3.2 METHOD OF SOLUTION (cont'd)

The following example is a case wherein a rotary motion is produced from a one-point CL record. If linearity testing were to be performed for this motion, an improper cut would result.

In Diagram 3.4.7.3H, the tool is to cut from point A to point B; the part programmer can produce this by giving a GOTO/x,y,z,i,j,k statement which causes the table to rotate 180 degrees. If the

postprocessor were to do linearity testing on this one-point CL record, it would produce a large series of small corrective moves which causes the tool to cut straight through the part from point A to point B. This is because each midpoint along the path AB would appear to be far out of the tolerance sphere, hence, the postprocessor would insert a "correction" path bringing the tool back into tolerance but producing an improper cut.

The postprocessor can be made to disregard a one-point CL record for linearity testing by the statement LINTOL/OFF. Once given, this statement is modal until LINTOL/ON or LINTOL/r is given. Unless the LINTOL/OFF statement is given, the postprocessor continues to accept all one-point CL records for linearity testing.

Linearity testing is not done over circular paths when circular interpolation is used over the path.

A ROTATE/HED or ROTATE/TABLE statement with a ROTREF modifier is also tested for linearity when specified.

In order to detect linearity errors it is essential that the part coordinate data points correspond exactly with the machine coordinate points. For example, in Diagram 3.4.7.3I

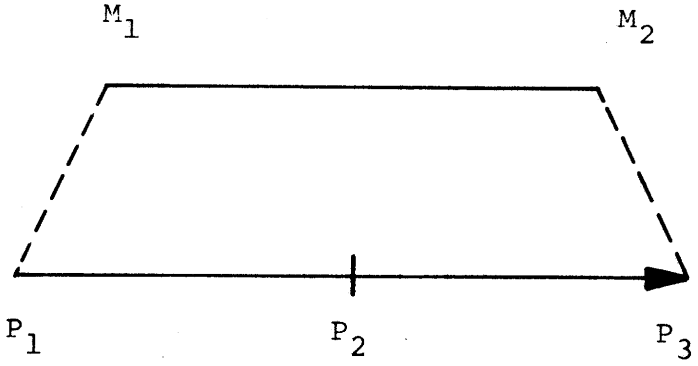


Diagram 3.4.7.3I

3.4.7.3.2 METHOD OF SOLUTION (cont'd)

M_1 corresponds to P_1, but M_2 corresponds to P_3, an invalid situation. Such an event could occur, for example, on a ROTATE/TABLE statement without a ROTREF modifier: that is, the part coordinate points have not changed but the machine coordinate points have. Since the postprocessor can detect linearity errors only when there is a one-to-one correspondence between the part and machine coordinates, it is essential that the part programmer keep the two coordinate systems compatible as long as linearity testing is to be used. Thus, caution must be used when programming ROTATE/TABLE, α and ROTATE/HED, α with no ROTREF modifier.

The value of r in the PPTOL/r, γ statement will normally be a function of the part and the particular machine tool axes configuration, and therefore, will vary considerably from part program to part program. Experience will undoubtedly provide the best value. However, a rule-of-thumb working value may be (INTOL + OUTTOL)/2.

The postprocessor does not perform linearity testing under the following conditions:

(1) No LINTOL given part program or a LINTOL/O was given. Parameter RADLIN is zero; branch to RETURN when RADLIN = 0.

(2) The path is a rapid traverse.

Parameter FRAPID is non-zero; branch to RETURN when FRAPID ≠ 0.

FRAPID is always zero for non-rapid paths.

(3) The path occurs during a tool change.

Parameter TOLCON is non-zero; branch to RETURN when TOLCON ≠ 0.

(4) The path is a one-point CL record.

Parameter RADLIN is set to zero which indicates LINTOL/OFF had been given; branch to RETURN if parameter NWPR < 11.

(5) The statement ROTATE/HED or ROTATE/TABLE is given without a ROTREF modifier.

3.4.7.3.2 METHOD OF SOLUTION (cont'd)

(6) Circular interpolation is used over a circular path.

The processing subroutine for circular interpolation paths (subroutine PRODQD) does not call subroutine LINRTY.

(7) Overcenter cutting occurs.

This refers to those NC machines which exceed a slide limit when cutting over center and the postprocessor makes an adjustment to allow for possible continuation.

3.4.7.3.3 PROCESSING METHOD OF SUBROUTINE LINRTY

In Diagram 3.4.7.3J let the path $P_1 P_2$ be the cutter path in part coordinates. Then $M_1 M_2$ is the corresponding resultant path in machine coordinates; M_1 and M_2 result when P_1 and P_2 are processed through the transform equations. The midpoint M_A of path M_1 M_2 is found by linear interpolation; similarly midpoint P_A is found on path P_1P_2. P_A represents the true, ideal midpoint of the cutter path if there was no linearity error due to the rotary motion of the slides.

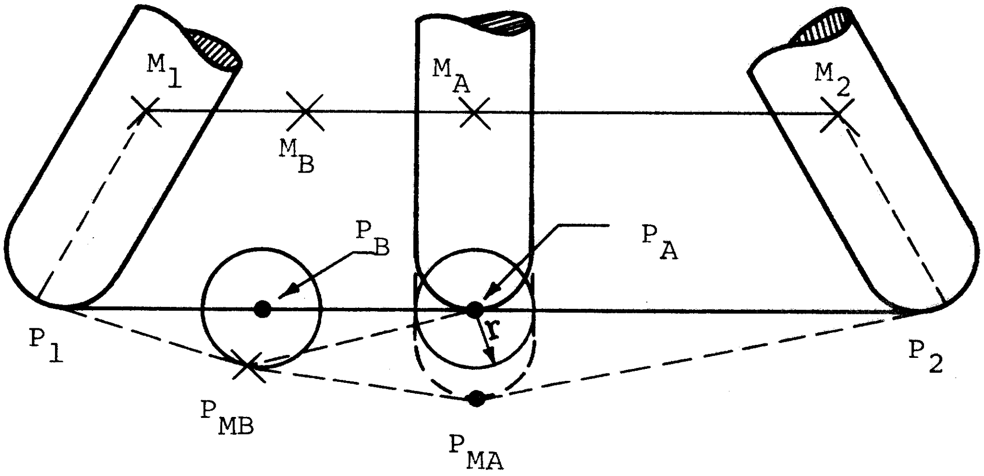


Diagram 3.4.7.3J

3.4.7.3.3 PROCESSING METHOD OF SUBROUTINE LINRTY (cont'd)

Midpoint M_A is converted to part coordinate midpoint P_{MA} by processing it through the inverse transform equations. As illustrated, P_{MA} falls outside the tolerance sphere of radius r, and is therefore detected as being a linearity error. Point M_A and P_A are saved as potential output points which represent a segment to correct the detected linearity error.Midpoints M_B and P_B are next determined, and the transformed P_{MB} is now found to fall within the tolerance sphere, therefore, no linearity error occurs here. Hence, it is sufficient to output the new point P_A. The corrected path now appears as in Diagram 3.4.7.3K.

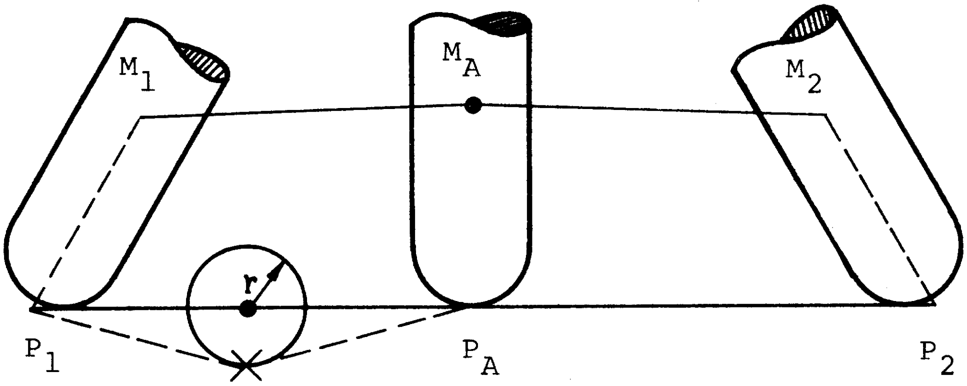


Diagram 3.4.7.3K

P_A is made P_1 and M_A is made M_1; testing then continues with the new paths P_1P_2 and M_1M_2.

In subroutine LINRTY points P_1 and P_2 are represented by the part coordinate system vectors DPREVP and DPRESP, respectively, while M_1 and M_2 are the machine coordinate system vectors DPREVM and DPRESM. The part coordinate vectors have the order:

x, y, z, i, j, k

where x,y,z are the CL data values plus any given TRANS values; i,j,k are the backward directed tool axis direction cosines. The machine coordinate vectors have the order:

X, Y, Z, A, B, C

3.4.7.3.3 PROCESSING METHOD OF SUBROUTINE LINTRY (cont'd)

where X,Y,Z are the transformed part coordinates for the slides, while A and B and C are the machine tool rotary motions in degrees. A or B or C may be zero for four-axis machines. When going from part to machine coordinates, the vector DPRESM is always the converted point related to DPRESP. Conversely, when going from machine to part coordinates, the vector DPRESP is always the converted point related to DPRESM. Hence, when point M_A is to be converted to point P_{MA}, the original vectors DPRESP and DPRESM are first saved, and then DPRESM is made equal to the midpoint vector M_A ($\equiv$HALFMP) to produce the part midpoint vector P_{MA} ($\equiv$ DPRESP).

Whenever a linearity error is detected, the flag LINFLG is set non zero to indicate this condition. Whenever a linearity error is found on the given path, the postprocessor will output segments (as needed) until the whole path P_1 P_2 is processed. Under such conditions the flag LINSIG is set non zero to indicate that linearity correction segments have been made output. Therefore, the subroutine return flag RETURN is set to non zero so that when regular processing continues after subroutine LINRTY, there will be no redundant output of the path P_1P_2 (converted to M_1 M_2). When subroutine LINRTY detects no error, RETURN is set to zero, and the cutter path is processed in the normal manner.

Note that option 29 must be set non zero if linearity testing is to be used. The table position must be known for the correct determination of the points of segmentation.

3.4.8 PROCESSING IN A MULTIHEAD ENVIRONMENT

All multihead processing inherently requires a two-pass system: the first pass processes the CL tape for both heads, and the second pass merges the data for combined motion and output.

3.4.8.1 FIRST-PASS CONSIDERATIONS

Multihead processing must be considered at the very beginning of the program when GEINIT is in core, for it is at this time that the multihead environment is established.

One of the first complexities to be resolved is how to establish the register (REGSTR) and format (REGFOR) conditions for each head when there is only one table available for each condition. This is resolved in the Machine Subroutine where the tables REGSTR and REGFOR are first set up for head 2, and then written onto TAPES1 where they are saved until GEMULT is in core.

3.4.8.1 FIRST-PASS CONSIDERATIONS (cont'd)

After TAPES1 is written, the REGSTR and REGFOR tables are then set up for head 1. Since these tables are in GECOM COMMON, they are available during all phases of the program.

In GEMULT during the first call to subroutine GMOUT, the subroutine GMSTOR is eventually called; and upon initial entry into this subroutine, the saved data on TAPES1 are reselected and stored into the tables GMWORD and GMFORM which are analogous to the tables REGSTR and REGFOR, respectively. GMWORD and GMFORM are in GECOT3 COMMON which makes them available for GEOUT3 processing.

Also, in the initial entry sequence of subroutine GMSTOR, TAPES1 and TAPES4 are rewound and opened for writing. TAPES1 and TAPES4 are used by GEPRO3 for saving the data for the Absolute and Operation Printouts, respectively. Later entries to subroutine GMSTOR are simply rerouted to GEOUT.

The postprocessor statement COMBIN/n designates multihead operation; and when the CL tape record (subtype 1071) for this statement is encountered, subroutine COMBIN is called wherein the flag MULTHD is set to n. This flag establishes the multihead environment for the postprocessor.

The currently operating head is selected by the statement SELECT/HEAD, n, and the CL tape record is processed in subroutine SELHED where the head flag IHEAD is set to n.

The postprocessor statement OP/n specifies the combining or processing sequence of operation for both heads. Subroutine OPCODE processes the CL data information for this statement (subtype 1073) and sets up the special CODE = 17 command block. The operation number n is saved in flag NOP, but the other data of the CL record are stored in a fixed manner into DBFSEG.

3.4.8.1 FIRST-PASS CONSIDERATIONS (cont'd)

DBFSEG(2) = n (from OP/n)

DBFSEG(7) = 0 if SFM; = 1 if RPM

DBFSEG(8) = head number (IHEAD)

DBFSEG(9) = t (See Part Programmer's Manual)

DBFSEG(10) = t (See Part Programmer's Manual)

DBFSEG(11) = 0 if there are restrictions to consider while merging, otherwise = 2 (NONE modifier)

DBFSEG(15) = 17 (CODE) to designate an OP block

This special DBFSEG block is in many respects similar to the Information Block (CODE = -9) (See Section 5.5), and in fact, serves the same purpose but in a more unique manner and exclusively for multihead operation.

The setup DBFSEG block is made output where it is stored on TAPES2 or TAPES3 depending on IHEAD.

A DBFSEG record of CODE = 17 also results from the statement PRFSEQ/ON and PRFSEQ/OFF. In this case DBFSEG is set up as:

DBFSEG(2) = NOP (opcode)

DBFSEG(8) = IHEAD if the PRFSEQ modifier is ON; otherwise, = 2 if OFF and IHEAD = 1, or = 1 if OFF and IHEAD = 2.

DBFSEG(11) = 3

DBFSEG(15) = 17 (CODE)

The setup DBFSEG block is made output where it is stored on TAPES2 or TAPES3 depending on IHEAD.

3.4.8.1 FIRST-CLASS CONSIDERATIONS (cont'd)

There are special multihead sequences in the subroutines for processing RAPID moves and TURRET statements, but these are of a nature whereby a particular head M code or T code is involved. In other words the output from these subroutines are no different than for single-head operation, but merely reflect the requirements of the particular head then in mode. Sections 5.2 and 6.0 (Subroutines Descriptions) and program listings should be consulted for further information on these items.

When a command block DBFSEG is ready for output, subroutine OUTPUT is called to ultimately print and punch the block. But for a CODE = 17, subroutine OUTPUT only adds the plus-minus value of SEQCTR to DBFSEG(1), and bypasses the other sequences since they are not yet needed. Instead, the command block is dumped onto TAPES2 for head 1 (IHEAD=1), or onto TAPES3 for head 2 (IHEAD=2).

When the FINI record (type 14000) is encountered on the CL tape, subroutine GEBASE outputs two FINI command blocks (CODE = 18), one for each head, i.e, for TAPES2 and TAPES3. An end-of-file is then written on TAPES2, TAPES3, and TAPES4 which are then all rewound.

When program control is returned to the monitor GEMON, it pulls in the overlay GEMULT which processes the dumped data for the second pass.

3.4.8.2 SECOND-PASS CONSIDERATIONS

Before processing can begin in GEMULT, the postprocessor must initialize key parameters and flags, clear arrays, and open TAPES2 and TAPES3 for reading; this is done in subroutine GMINIT.

Subroutine CREAD is then called to read a record from TAPES2 if IHEAD = 1, or from TAPES3 if IHEAD = 2. The result of the read is that the array AS2 (for TAPES2) or AS3 (for TAPES3) are stored with the dumped row of DBFSEG. The arrays AS2* and AS3* are dimensioned and ordered the same as DBFSEG, hence, when AS2 or AS3 are filled from tape with the dumped DBFSEG, the postprocessor thereafter treats them in the same manner as if it were considering a DBFSEG row.

Beginning with head 1, the postprocessor reads TAPES2 and checks the command block code, i.e., the fifteenth element of the row, to see if an OP/n block was read; a code of 17 indicates such a block. If no such block is detected, the postprocessor knows there is no merging necessary, and it outputs the command block as it is. Subroutine GMOUT is called to output the block. See Section 3.4.8.2.1 for details on outputting a single or combined multihead command blocks.

When a CODE of 17 is found, the postprocessor stops reading TAPES2 and begins reading TAPES3. Each read-in command block code is tested to see if it is an OP/n block (CODE = 17); and if not, the postprocessor again knows no merging is necessary and it outputs the block as it is.

When a CODE of 17 is detected, the postprocessor immediately makes a comparison of the TAPES2 opcode with the opcode from TAPES3.

The opcodes are first saved as:

IS23 = AS2(2) head 1 opcode,

IS33 = AS3(2) head 2 opcode.

(See Section 3.4.8.1 for method of storing DBFSEG for a OP/n statement.)

SEE FOOTNOTE
ON 3.4-84

3.4.8.1 SECOND-CLASS CONSIDERATIONS (cont'd)

If the opcodes are equal, then a merge of the blocks is indicated (See Section 3.4.8.2.2). If the opcodes are unequal, no merging is to take place, and the postprocessor reads the scratch tape of the head which has the lowest value opcode.

The above sequence is repeated wherein blocks are made output as long as the opcodes are equal, and the sequence continues until two opcodes are found which are equal.

When the opcodes are unequal (no merging) and a CODE = 17 block is found, the postprocessor makes an additional check to see if the block is a multihead information block for a SAFETY (DBFSEG(11) = 1) or for an SFM (DBFSEG(11) = 4). See Section 3.4.8.1 for a discussion of these items.

When the block is for a SAFETY, the retract values of X,Y, and Z are saved in SAFHD1 or SAFHD2 as the case may be.

Similarly, for an SFM block, the SFM value is saved in SFMHD1 or SFMHD2 as the case may be.

These retained values are used at some later point in the program.

*The arrays AS2 and AS3 are actually doubly dimensioned arrays of (30, 2), but for convenience and simplicity all references to AS2 and AS3 are made as if they were singly dimensioned arrays.

3.4.8.2.1 MERGING OF BLOCKS

When the opcodes are equal, merging of blocks from TAPES2 and TAPES3 commences and continues until the opcodes once again become unequal, at which time the processing sequence described at the beginning of Section 3.4.8.2 begins again.

The merging of command blocks is a highly complex affair which is dependent upon a variety of factors, all of which directly affect the methods of merging. Among the key factors that must be considered are:

(1) One feedrate register or two feedrate registers;

(2) Same interpolation modes or mixed interpolation modes on each head;

(3) Shared or common axes for both heads;

(4) The influences of the PRFHED, SAFETY, SFM, TURRET, and other postprocessor statements;

(5) Automatic parking and returning.

Since this subject is so complex, the best that can be accomplished here is to describe the theory involved and make a brief survey of some of the programming methods used. For more details the reader must refer to the multihead listings and individual subroutine write-ups in Section 5.2.

3.4.8.2.1.1 SINGLE FEEDRATE REGISTER MERGING

The theory of operation is first discussed with some examples to illustrate the methods used. Following the theory of operation is a brief description of how the method is programmed in subroutine GEMULT.

3.4.8.2.1.1.1 THEORY OF OPERATION

The fundamental operating requirement of single feedrate register merging is that the two heads must have identical cut times, but they do not necessarily have to have the same feedrates. This means that the part programmer has complete freedom to specify different feedrates for each head. Equal records are produced as output by segmenting the head with the longer cut time into two records; the cut time of the first record is equal to that of the other head. Details of the segmentation procedure are discussed below.

Though this method of combining cut sequences gives more freedom to the part programmer, the number of records of paper tape output increases. An option is provided in the program which reduces the number of records of output by giving the computer program more freedom to vary the feedrates, and thus to reduce the required number of segmentations.

Note that the procedures described usually produce unequal block times between the two heads. This can be remedied by changing the feedrate of one head so that the block times become equal. This approach is not wholly acceptable, because the part programmer usually wants block records to have equal cut or dwell times and, at the same time, keep the desired feedrates.

The postprocessor attempts to meet the above two requirements within well-defined limits. The concept used in merging two heads for simultaneous cutting is illustrated by the following examples.

Example 1: (See Diagram 3.4.8.2.1A)

Assume that there are two heads, A and B, each having two cutter motions, and we wish to combine the motion statements of these two heads. The intent is to produce simultaneous cutting without allowing dwells to occur on either head and to retain the specified feedrates, if possible.

3.4.8.2.1.1.1 THEORY OF OPERATION (cont'd)

The following symbols are used in the description:

L_{al} and L_{a2}	cutter motions for head A (where L is the vector length of a move)
F_{al} and F_{a2}	feedrates for head A motions
T_{al} and T_{a2}	cutting times for head A motions
L_{bl} and L_{b2}	cutter motions for head B
F_{bl} and F_{b2}	feedrates for head B motions
T_{bl} and T_{b2}	cutting times for head B motions

The first step is to calculate T_{al} and T_{bl}.

Now assume $T_{al} > T_{bl}$.

The ratio $T_{bl}/T_{al} < 1$ is computed.

The vector length L_{al} is then segmented into two records, which are computed as follows:

$$L_{all} = (T_{bl} / T_{al})*L_{al}$$

$$L_{al2} = L_{al} - L_{all}$$

The ratio (T_{bl}/T_{al}) is the basic factor in seqmenting a circle. The example implies linear motion. However, with ratio (T_{bl}/T_{al}) known, a circle can be segmented and, thus, the requirements spelled out in the example can be met.

The corresponding times for L_{all} and L_{al2} ~~and~~ ARE T_{all} and T_{al2}.

But note:

$$T_{all} = \frac{L_{all}}{F_{al}} = \frac{\left(\frac{T_{bl}}{T_{al}}\right) * L_{al}}{F_{al}} = \frac{T_{bl}}{T_{al}} * T_{al}$$

$$T_{all} = T_{bl}$$

The two requirements, namely, production of equal cut times and maintenance of desired feedrates, have been met. L_{all} and L_{bl} are now set up, and a block of output is generated.

3.4.8.2.1.1.1 THEORY OF OPERATION (cont'd)

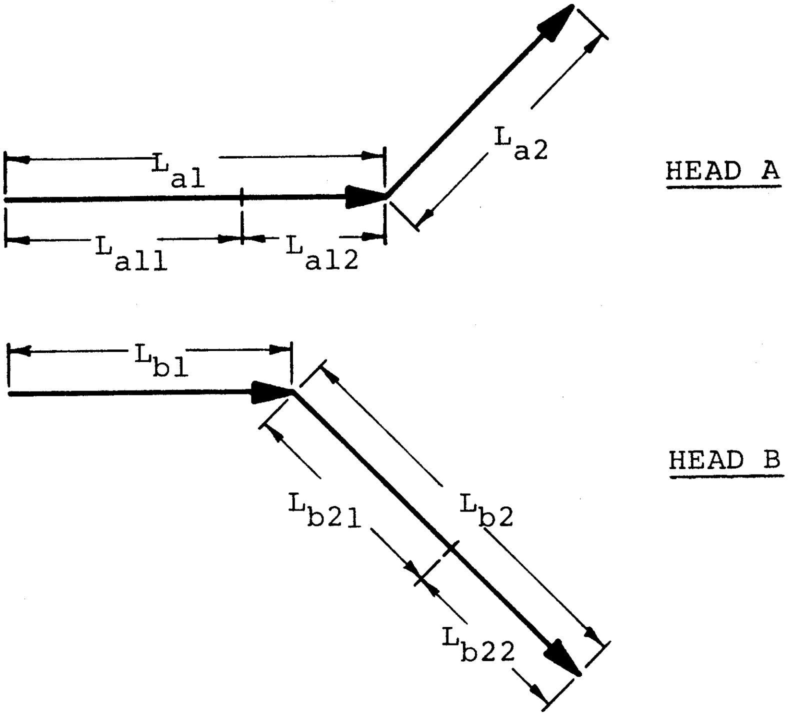


Diagram 3.4.8.2.1A

The next record for head B is read, and the following motions are compared:

L_{a12} with L_{b2}

Assume $T_{a12} < T_{b2}$

The new ratio is (T_{a12}/T_{b2})

and L_{b2} is segmented $L_{b21} = (T_{a12}/T_{b2}) * L_{b2}$

$L_{b22} = L_{b2} - L_{b21}$

3.4.8.2.2.2.2 THEORY OF OPERATION (cont'd)

Again, the requirements have been met, and the second record of output (L_{a12} and L_{b21}) is generated.

Next a record is read in from head A, and the following comparison is made:

L_{a2} with L_{b22}

The above procedure continues, until all moves in the specified combined cut have generated output blocks.

Obviously, in this example, one head will finish its cutter motions before the other head finishes. Thus, the following rule has been established:

> RULE 1: When two heads are designated for combined cutting, the last motion statement of each head should move the cutter away from the cutting surface.

The head that finishes first sits in a dwell condition until the computer program finishes the generation of the output blocks to complete the combined motion cuts of the second head.

Within the segmentation, the part which is to be used as the output block does not have tape reader limitation. This is true because the other head has not been segmented, and consequently, has a cut time greater than or equal to the tape reader limitation. This is illustrated in Example 2.

Example 2:

(Refer to Example 1 and Diagram 3.4.8.2.1A)

T_{a1} has been segmented so that $T_{a1} = T_{a11} + T_{a12}$; however, $T_{a11} = T_{b1}$. Since T_{b1} already meets the tape reader limitation, T_{a11} also meets this requirement.

Now, note that $T_{a12} = T_{a1} - T_{b1} = \Delta T$.

$T_{a;2}$ is the difference in time between T_{a1} and T_{b1}. Its value affects the solution of the problem as follows:

In GECENT III the constant TMAX is the maximum time which restricts feedrate. It is the maximum value of tape reading time in seconds (option 13) and servo setting time in seconds (option 69).

3.4.8.2.1.1.1 THEORY OF OPERATION (cont'd)

If T_{a12} > TMAX, processing may continue, with no further testing required.

If T_{a12} < TMAX, rule 2 is enforced:

> Rule 2: When a record for either head is considered for segmentation into two parts but the time of the second segment (T_{a12}) is less than TMAX, segmentation will not occur. The program then reduces the feedrate of the other head such that $T_{a1} = T_{b1}$.

For example, assume $T_{a1} > T_{b1}$.

T_{b1} is then adjusted so that $T_{a1} = T_{b1}$.

T_{b1} is adjusted by reducing the feedrate F_{b1}. The new feedrate value is calculated as follows:

$$F_{b1}\text{ (new)} = \frac{T_{b1}}{T_{a1}} * F_{b1}\text{ (actual)}$$

Thus,

$$T_{b1} = \frac{L_{b1}}{\frac{T_{b1}}{T_{a1}} * F_{b1}} = \frac{L_{b1} * T_{a1}}{T_{b1} * F_{b1}} = \frac{T_{b1} * T_{a1}}{T_{b1}} = T_{a1}$$

In example 1, two objectives were set: (1) The cut or dwell times for both heads in a given output record should always be equal. (2) Feedrates should be kept at the rates specified by the part programmer, whenever possible.

The computer program will never alter the first provision, and it changes the feedrate only under rule 2. However, the part programmer has the additional option of allowing the computer program to reduce the feedrate value within a given tolerance band, in addition to providing for feedrate reduction under rule 2.

Suppose the part programmer designates different feedrates for each head but is willing to allow a reduction in either feedrate of up to 10 percent tolerance band; see option 151. (Note: 10 percent is used here for sample purposes only; the tolerance band could be any value from 0 to 100 percent).

3.4.8.2.1.1.1 THEORY OF OPERATION (cont'd)

When the segmented record is tested against TMAX under rule 2, a second test is also provided. The second test compares the delta time against the time of the head that will not be segmented. If the ratio $\frac{\Delta t}{t}$ is less than 10 percent (or any specified tolerance band), segmentation is not performed; the feedrate value of one head is reduced so that the cutter times are equal for the given output record. This test reduces the number of output records in the combined mode without violating the feedrates by more than the tolerance band.

One final feature of this system should be noted. Programmed dwells are allowed in a combined cut and basically follow the same procedures already discussed.

Three restrictions governing the merging of two heads into combined cuts have been placed into the program, namely, RPM, SFM and IPM limitations. The part programmer may specify a desired RPM or SFM; however, if either value exceeds the designated tolerance band specified in the combined cut statement, the simultaneous cutting will not be allowed. A similar application results for an IPM limitation between heads. If these restrictions are not met at any given time while in a combined cut, one head will be withdrawn while the other head continues cutting. When the first head has finished its cutting sequence, the second head will return to the part and finish its cutting sequence.

3.4.8.2.1.1.2 PROGRAMMED PROCEDURE

In subroutine GEMULT after the opcodes have been ascertained as equal, the postprocessor seeks records from TAPES2 and TAPES3 which it can successfully merge together. Mergeable records are linear (CODE = 0), dwells (CODE = 4), and circular interpolation (CODE = ± 10, 11, 12) records. All other valid type records except CODE = 17 are made output without merging. The CODE = 17 records are recognized for SAFETY, SFM, and PRFSEQ statements, and the condition flags are set accordingly.

The indices ICODE and JCODE represent RESPECTIVELY the head 1 and head 2 VALUES OF (CODE + 1.0) ~~value, respectively~~. These indices are used to determine the condition of the two blocks to be merged and the condition flag ICIRLN is set accordingly as:

3.4.8.2.1.1.2 PROGRAMMED PROCEDURE (cont'd)

ICRLIN	Condition: Head1 - Head2
-1	Linear - Circular
0	Linear - Linear
+1	Circular - Linear
+2	Circular - Circular

Dwells and turret corrective moves are treated as lines; a dwell is treated as a linear move with zero feedrate.

For a Line - Line condition, subroutine GMLINE is called to combine the two command blocks; and for a Line - Circle, Circle - Line, or Circle - Circle condition, subroutine GMCIRL is called to combine the two command blocks. The subroutine does this in three possible ways in accordance with Section 3.4.8.2.1.1.1:

(1) Combine head 1 and head 2 with no changes;

(2) Use head 1 as it is, but segment head 2 into two parts such that the cut times of both heads are equal;

(3) Use head 2 as it is, but segment head 1 into two parts such that the cut times of both heads are equal.

The segmentation based upon equal cut times was discussed in the theory of operation in Section 3.4.8.2.1.1.1 above. The subroutine SPLIT is called to perform the actual dividing of the path using a ratioed linear proportion for straight line segmentation, and a vector ratioed sequence for circles; see the write-up on subroutine CIRSEG for the mathematical description of circle segmentation.

The technique for merging can best be explained by considering the example in Diagram 3.4.8.2.1B. The following assumptions are made:

(1) a preparatory function exists for each head;

(2) the feedrates on L_{al} and C_{al} are, respectively, 20 ipm and 10 ipm:

(3) the feedrate for head 2 is set to 10 ipm.

3.4.8.2.1.1.2 PROGRAMMED PROCEDURE (cont'd)

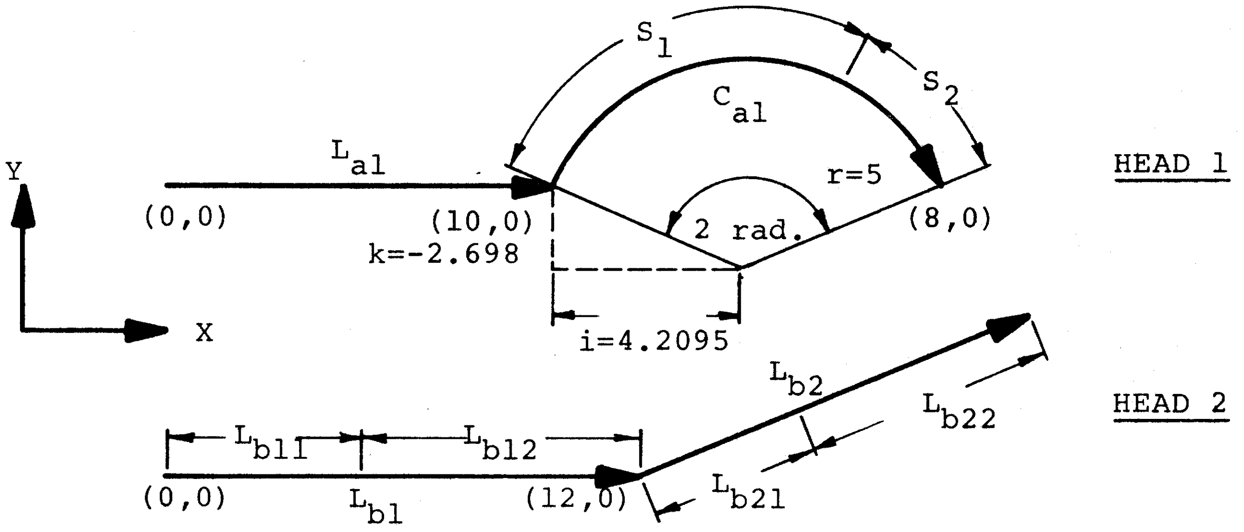


Diagram 3.4.8.2.1B

Since the first two paths to be merged are linear, subroutine GMLINE is called. The cutting times for L_{a1} and L_{b1} are calculated as:

$$T_{a1} = \frac{10}{20} = 0.5 \text{ min.}; \quad T_{b1} = \frac{12}{10} = 1.2 \text{ min.}$$

Since T_{b1} is greater than T_{a1}, subroutine SPLIT is called to segment L_{b1} into segments L_{b11} and L_{b12}:

$$L_{b11}(X) = \frac{0.5}{1.2} * 12 = 5 \text{ in.}$$

$$L_{b12}(X) = 12 - 5 = 7 \text{ in.}$$

3.4.8.2.1.1.2 PROGRAMMED PROCEDURE (cont'd)

L_{a1} and L_{b11} are merged together for a block of output. The circle record C_{a1} is read in, and C_{a1} is compared with L_{b12}. Subroutine GMCIRL is called with ICRLIN = 1. The arc length for C_{a1} is:

$$S_o = 2 * 5 = 10 \text{ in.}$$

The times for the two segments are:

$$T_{b12} = \frac{7}{10} = 0.7 \text{ min.}$$

$$T_{CAL} = \frac{10}{10} = 1 \text{ min.}$$

$$\Delta T = 1.0 - 0.7 = 0.3 \text{ min}$$

The ratio for segmenting is:

RADIO = 0.7

Since ΔT is positive, subroutine CIRSEG is called to segment the circle C_{a1}. See Diagram 3.4.8.2.1C for the definition of terms.

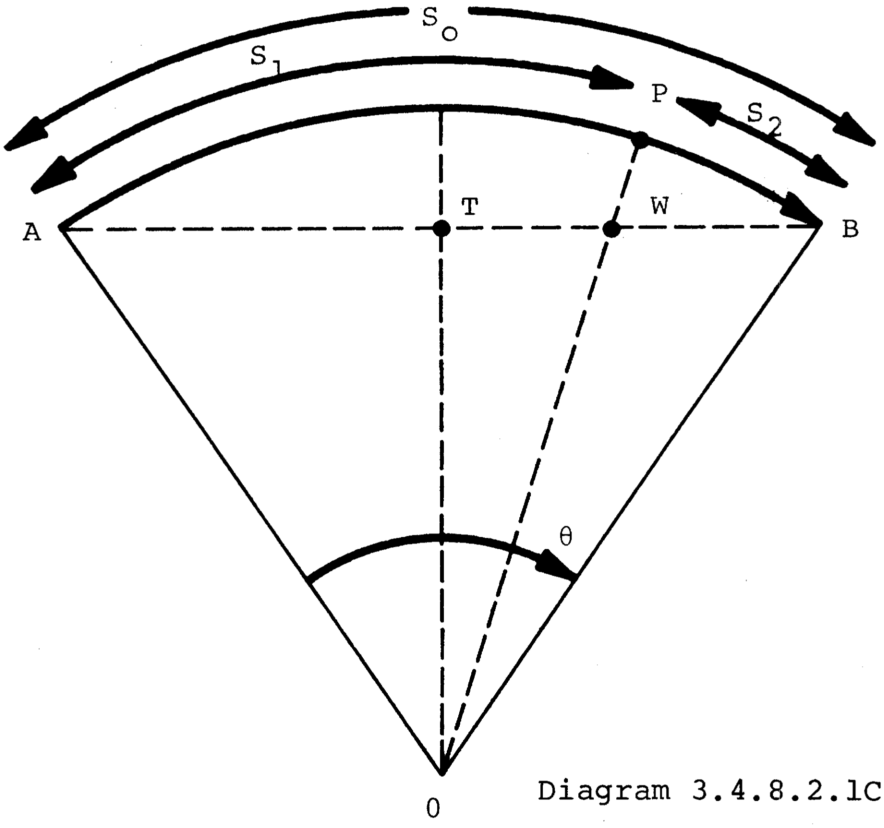


Diagram 3.4.8.2.1C

3.4.8.2.1 1.2 PROGRAMMED PROCEDURE (cont'd)

We need to find the vectors $\overline{AP}$, $\overline{PB}$, and $\overline{PO}$. The chord length for our example is:

$$|AB| = \sqrt{(18 - 10)^2} = 8.$$

$$|AT| = 4.$$

$$|OT| = AO* \cos \frac{\theta}{2} = 5* \cos 1 \text{ rad} = 2.7015$$

$$|WT| = |OT| \tan (\theta\text{-RADIO } \theta) = 2.7015 \tan (0.4)$$

$$|WT| = 1.1422$$

$$|AW| = 5.1422$$

$$AW(X) = \frac{|AW|}{|AB|} AB(X) = 5.1422$$

$$AW(Y) = 0.0$$

$$OW(X) = AW(X) - AO(X) = 5.1422 - 4.2095 = 0.9327$$

$$OW(Y) = 0.0 + 2.698 = 2.698$$

The length of OW is:

$$OW = \sqrt{0.9327^2 + 2.698^2} = 2.85$$

$$\overline{OP} = \frac{RADIUS}{OW} * \overline{OW}$$

$$OP(X) = \frac{5}{2.85} * 0.9327 = 1.636$$

$$OP(Y) = \frac{5}{2.85} * 2.698 = 4.732$$

$$\overline{AP} = \overline{AO} + \overline{OP}$$

3.4.8.2.1.1.2 PROGRAMMED PROCEDURE (cont'd)

The departures for S_1 are:

$AP(X) = 4.2095 + 1.636 = 5.8455;$

$AP(Y) = 2.698 + 4.732 = 7.430;$

$\overline{PB} = \overline{AB} - \overline{AP}$

The departures for S_2 are:

$PO(X) = 8 - 5.8455 = 2.1545;$

$PB(Y) = 0 - 7.430 = -7.430.$

The offsets for S_2 are:

$PO(X) = -1.636;$

$PO(Y) = -4.732.$

The departures and the offsets for S are now merged into a block with L_{b12}.

The remaining segment S_2 is merged with L_{b2}. The cutting times for L_{b2} and S_2 are computed to be:

$$T_{b2} = \frac{8.544}{10} = 0.854$$

$$T_{s2} = \frac{3}{10} = 0.3$$

The time for the segment S_2 is less than the time for L_{b2} so the line L_{b2} must be segmented.

$$L_{b21}(X) = \frac{0.3}{0.854} * 8 = 2.81;$$

$$L_{b21}(Y) = \frac{0.3}{0.854} * 3 = 1.05$$

These departures are merged in a block with departures PB and offsets PO which have been computed for the circle segment S_2.

Since head 1 has finished, the remaining line segment, L_{b22}, is output in a block by itself.

3.4.8.2.1.3 COMMON AXIS SEGMENTATION

In addition to the segmentation required for obtaining equal times or equal path lengths, a multihead combined path may also require a segmentation if each head shares a common axis, thereby requiring that the incremental motion along the common axis be identical for both heads. Gantry type machines with multiheads very commonly have this feature.

The segmentation resulting as a function of a common axis is possible with a single or with double feedrate registers. The main requirement in segmenting so as to obtain an equal increment along the common axis is that the component feedrate along the common axis be either equal or be within an acceptable limit of each other.

3.4.8.2.1.3.1 THEORY OF OPERATION

Diagram 3.4.8.2.1.3A illustrates a typical gantry-type multihead machine which is to cut simultaneously path aAB with Head 1 and path dD with Head 2.

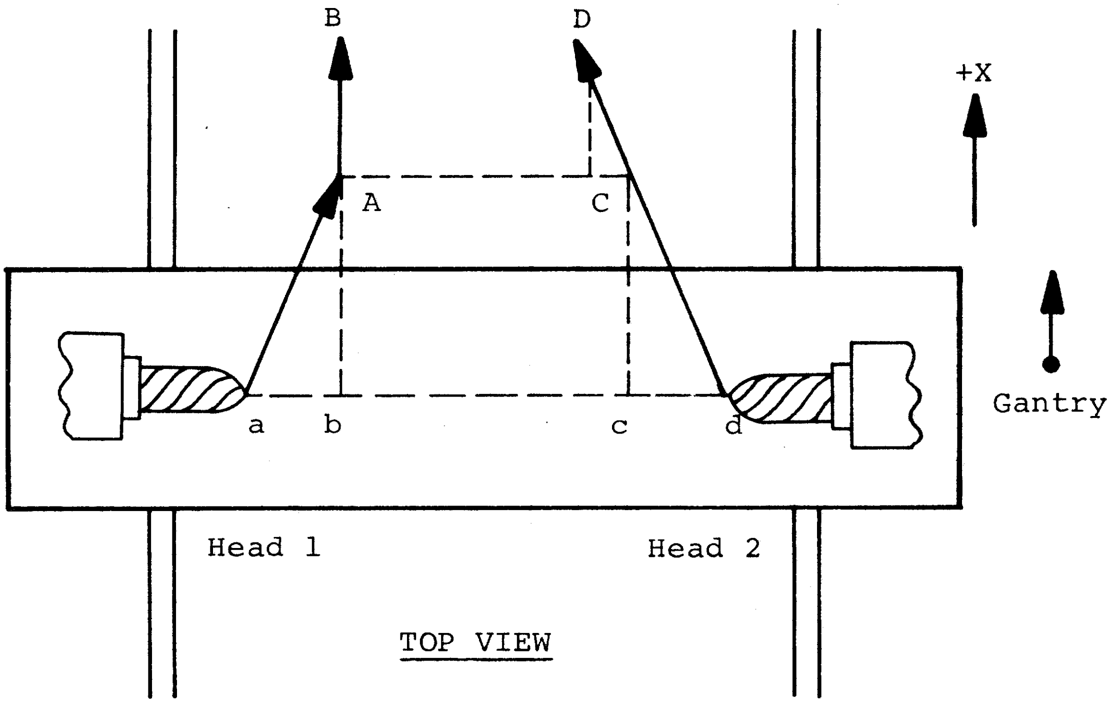


Diagram 3.4.8.2.1.3A

3.4.8.2.1.3.1 THEORY OF OPERATION (cont'd)

The requirement that ΔX for both Head 1 and Head 2 be identical is a necessary but not sufficient condition, because the component feedrate of both heads along ΔX must be approximately equal within some specified tolerance (option 157). In this example, the component feedrates along ΔX are equal since bA = cf; therefore, the path dD is segmented at point f, and consequently, Head 1 moves distance aA simultaneously with the Head 2 move from point d to f.

The simultaneous move along paths AB and fD may not be possible, however, if

$$\left| F_{x(AB)} - F_{x(fD)} \right| > \varepsilon$$

where

$F_{x(AB)}$ is the X axis feedrate along path AB,

$F_{x(fD)}$ is the X axis feedrate along path fD,

and, ε is option 157.

When circumstances arise which prohibit simultaneous cutting, the postprocessor completes the paths by separate sequences for each head. In the above example, Head 2 would be parked when path aA-df is completed, Head 1 would complete path AB, park, the gantry would return to point f, and Head 2 then would complete path fD.

It should be noted that when a head is parked, the other head is temporarily withdrawn also. This is done to prevent marring or scoring the workpiece.

Similarly, when both heads are to begin a simultaneous cut sequence (as when starting paths aA-df), both heads are first withdrawn and then simultaneously returned to the workpiece. This is done to ensure perfect synchronization of both heads.

A head is parked by moving it to the ΔX, ΔY, ΔZ distance given in the SAFETY statement. The X, Y, and Z values at that point are saved to allow a return to that point when cutting is to be resumed. Once parked, only the common axis value changes for the heads.

Only linear interpolation can be used in combined head moves, although circular interpolation can be used on single, unmerged head operations.

The common axis is designated by option 155. The postprocessor permits only one common axis and assumes that option 155 specifies which axis it is.

3.4.8.2.1.3.1 THEORY OF OPERATION (cont'd)

The general scheme of operation for combining command blocks for a common axis are as follows:

(1) Obtain a path for both Head 1 and Head 2.

(2) Combine that portion of the paths which share the same common axis range and whose component feedrates are within the allowable limit.

(3) Upon completion of the two paths, obtain two new paths.

(4) When the end of either path is encountered or when merging cannot otherwise be continued, complete the path using single head operation.

Some examples of common axis cut sequences are given below. With each example is given a brief description of the method of processing that particular example. The examples assume that the X axis is the common axis for both heads.

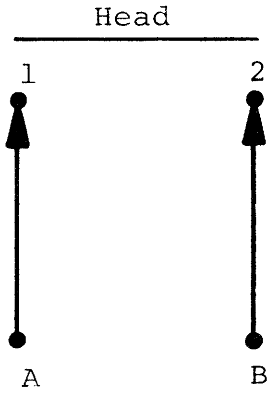


Diagram 3.4.8.2.1.3B

Paths are same length. Compare the X component feedrate on each head to see if they are within the option value tolerance. If they are, output the paths. If not, park Head 2 and cut A, then park 1 and cut B. Continue to next paths. Print a comment each time a head is parked.

3.4.8.2.1.3.1 THEORY OF OPERATION (cont'd)

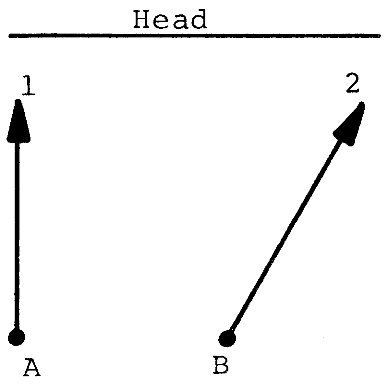


Diagram 3.4.8.2.1.3C

Compute the X component feedrates. If they are within the option tolerance, output the paths. If not, park 2, cut A. Then park 1, cut B. Print a comment each time a head is parked.

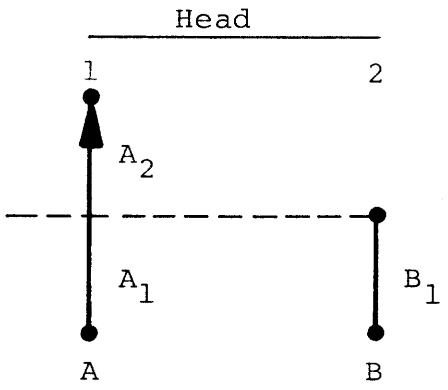


Diagram 3.4.8.2.1.3D

Path A will be segmented as follows:

A_1 and B_1 will be merged, if possible;

A_2 will be cut separately (Head 2 parked).

3.4.8.2.1.3.1 THEORY OF OPERATION (cont'd)

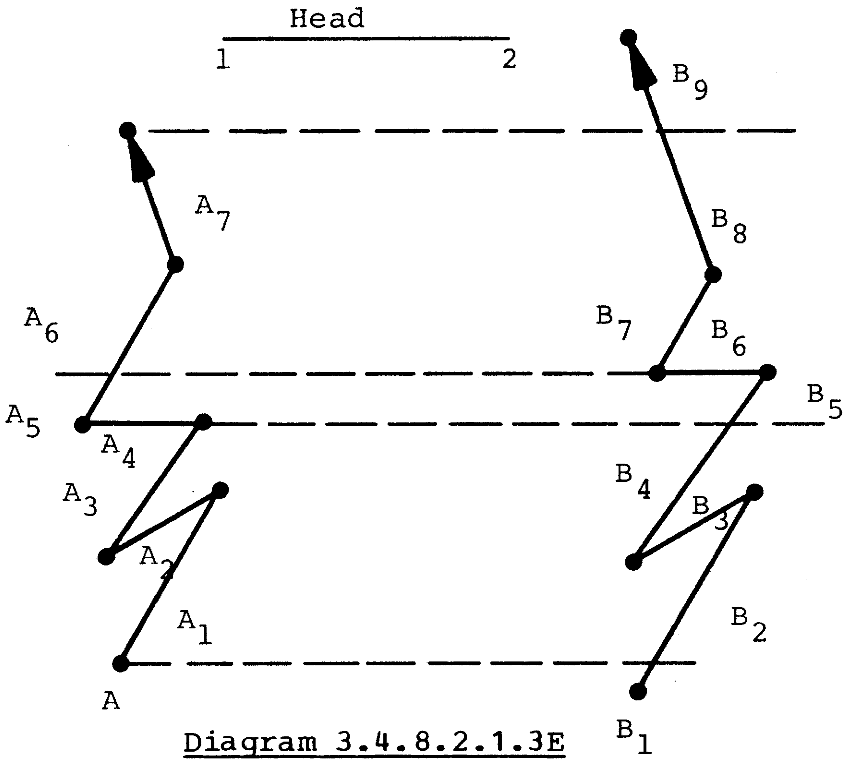


Diagram 3.4.8.2.1.3E

Park Head 1, cut B_1, cut A_1 and B_2, cut A_2 and B_3, cut $A_3$and B_4, park Head 1, cut B_5, cut B_6, park Head 2, cut A_4, cut A_5, cut A_6 and B_7, cut A_7 and B_8, park head 1, cut B_9.

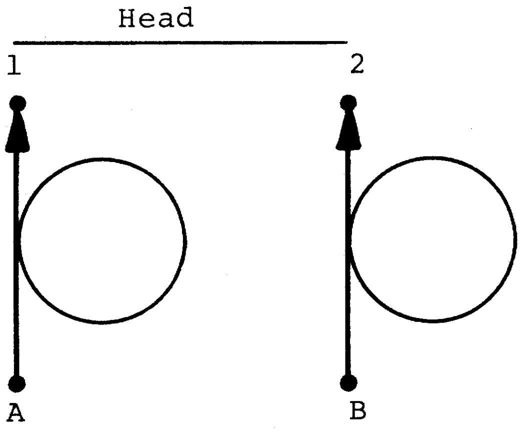


Diagram 3.4.8.2.1.3F

Cutting each circle can be a separate operation. If so, circular interpolation may be used.

3.4.8.2.1.3.1 THEORY OF OPERATION (cont'd)

The sequences could also be cut simultaneously; if so, linear interpolation is required. Care should be taken to insure that the X values on each circle are identical; otherwise, many small cut sequences may result which may be tape reader-limited.

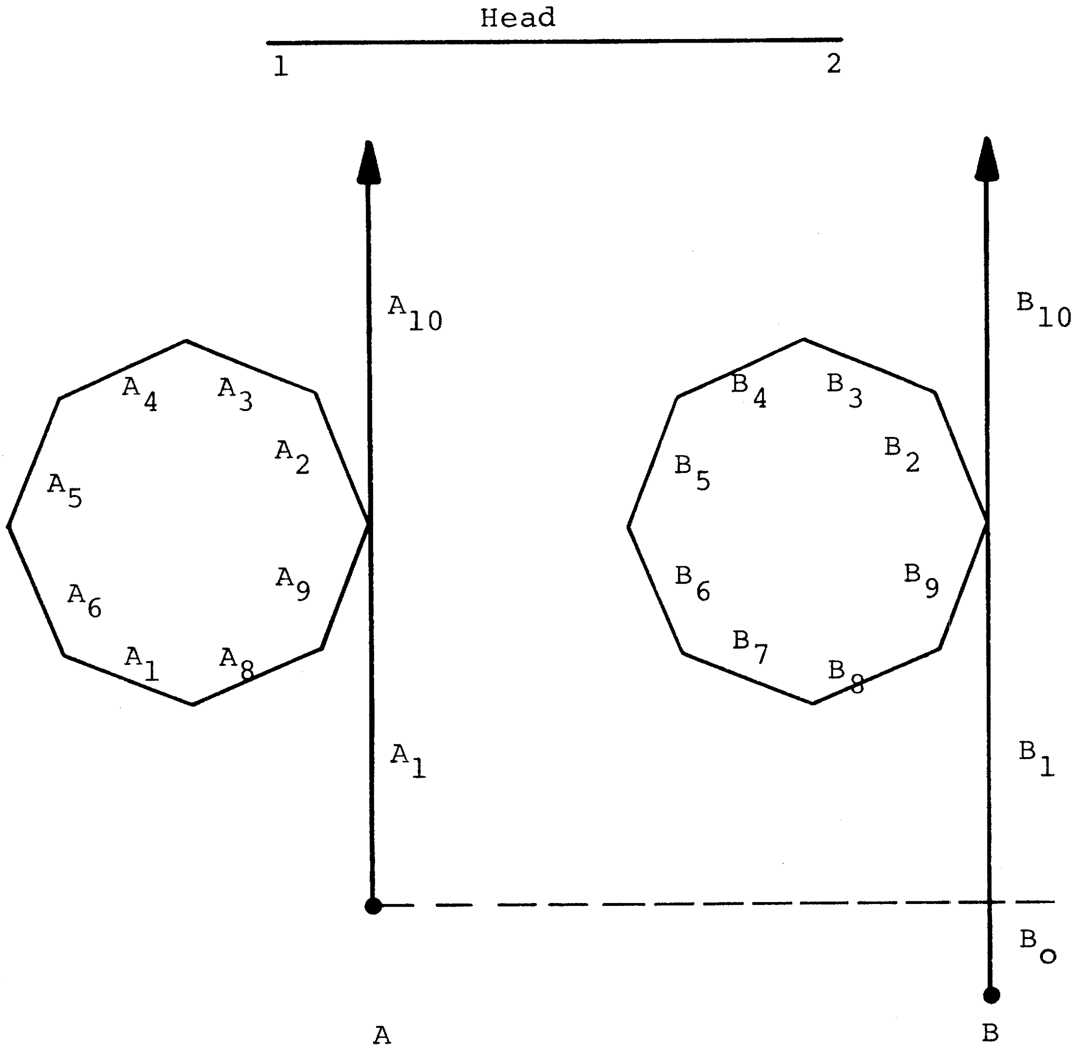


Diagram 3.4.8.2.1.3G

This sequence would be cut as follows:

cut B_0, cut A_1 and B_1, A_2 and B_2, A_3 and B_3, A_4 and B_4 , etc.

Care must be taken to insure that the X value of each segment is cut simultaneously.

3.4 8.2.1.3.1 THEORY OF OPERATION (cont'd)

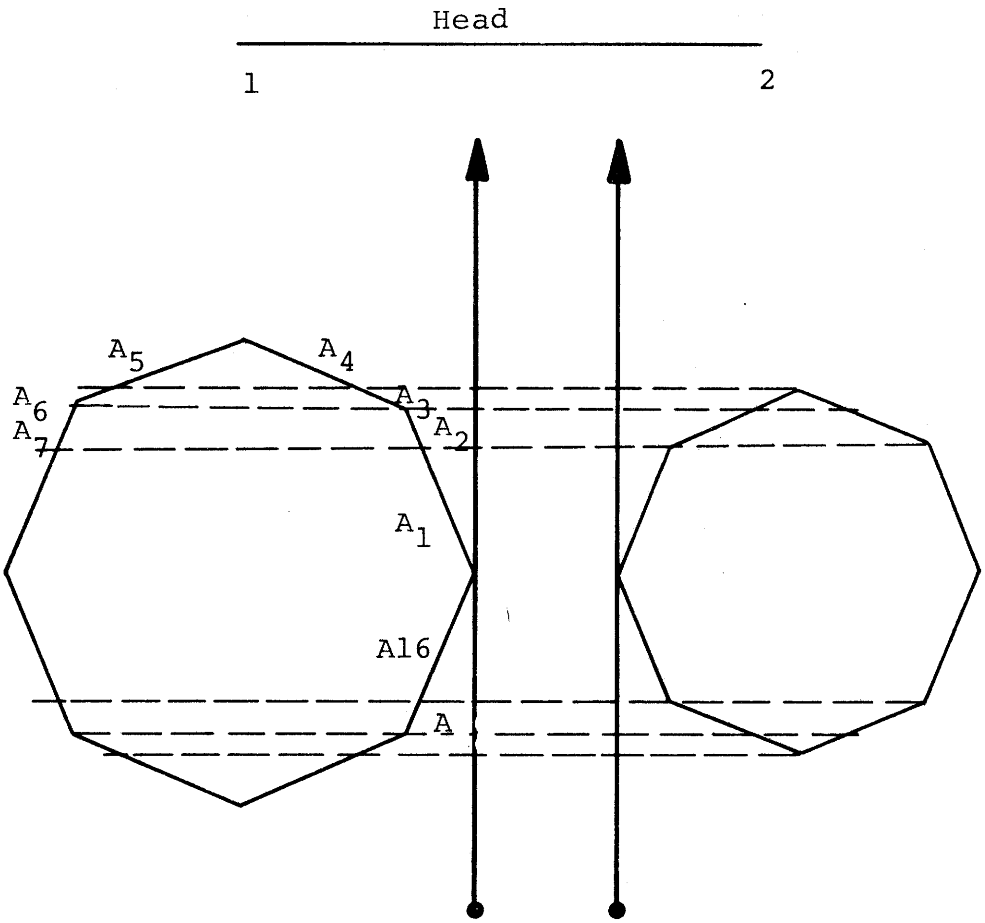


Diagram 3.4.8.2.1.3H

Cut A_1 and B_1, cut A_2, cut B_2, cut A_3 and B_3, cut A_4, cut A_5, cut A_6 and B_4, cut B_5, cut A_7, cut A_8 and B_6, cut A_9 and B_7, cut A_{10} cut B_8, cut $A_{11}$and B_9, cut A_{12},cut A_{13},cut $A_{14}$and B_{10},cut B_{11},cut A_{15},cut $A_{16}$and B_{12}.

(Park alternate head when a single move of a head is being) made.

3.4.8.2.1.3.2 PROGRAMMED PROCEDURE

In subroutine GEMULT before the call to the subroutine which carries out the merging of linear blocks (subroutine GMLINE), the postprocessor tests for the existence of a common axis option 155 and calls subroutine FXMULT if one exists. No such call precedes subroutine GMCIRL since the common axis segmentation sequence requires linear interpolation.

The generalized sequence of subroutine FXMULT is highly involved because of the many combinations of cut paths and affecting conditions; hence, the best means of explaining the methods of common axis segmentation is to illustrate the techniques by using simple examples. Once these special examples are followed, the overall general scheme will become clear.

We will use the simultaneous cut paths as illustrated in Diagram 3.4.8.2.1.3.2A.

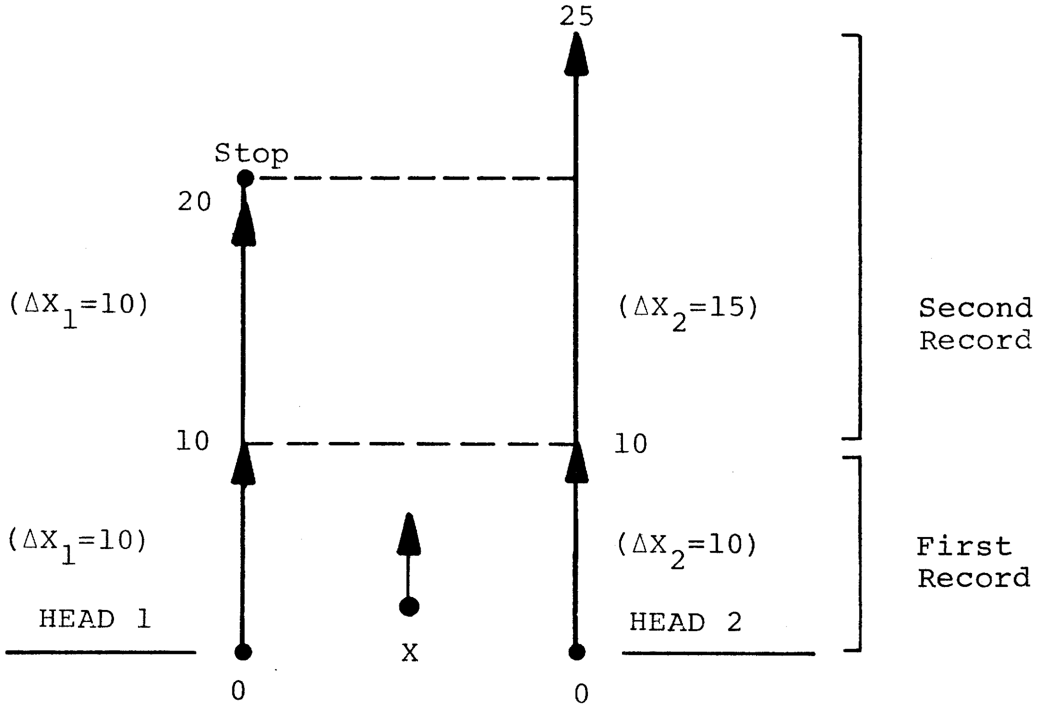


Diagram 3.4.8.2.1.3.2A

In this example we will consider only the X axis which is the common axis to both heads. We further assume that only one feedrate register exists.

3.4.8.2.1.3.2 PROGRAMMED PROCEDURE (cont'd)

Upon entry into subroutine FXMULT, the postprocessor computes the final absolute coordinate points of both heads. It must be recalled that during the second pass the information concerning the location of the tool (DPRESM and DPRESP vectors) is not available; all that is available are the incremental moves as dumped on the scratch tapes in the first pass.

However, in subrouting GMOUT the incremental moves are accumulated to obtain the current absolute XYZ values for each head; these values are stored in the vectors ABS2 and ABS3 for Head 1 and Head 2, respectively.

Therefore, upon entry into subroutine FXMULT, the vectors have the values (using X axis only):

ABS2	ABS3
0	0

This represents the beginning point of both heads; see Diagram 3.4.8.2.1.3.2A.

The subroutine requires the final point of the path; therefore, it computes the head vectors:

HIVEC(1) = ABS2(1) + AS2(3,1).

and

H2VEC(1) = ABS3(1) + AS3(3,1).

It will be remembered that AS2 is the command block (similar to DBFSEG) for the currently read Head 1 record read from TAPES2, while AS3 is for Head 2 from TAPES3. The XYZ values in AS2 and AS3 are increments.

(Note: For convenience and simplicity all vectors and other arrays will henceforth be treated as a single parameter. It must be remembered though that while the reference is to the X axis only, any axis could as well apply.)

In our example then,

H1VEC = 0 + 10 = 10,

H2VEC = 0 + 10 = 10.

3.4.8.2.1.3.2 PROGRAMMED PROCEDURE (cont'd)

The subroutine requires the knowledge of the beginning and end path values; these are the values H1X1 and H1X2 for Head 1, and H2X1 and H2X2 for Head 2.

$$H1X1 = 0,\ H1X2 = 10,$$

$$H2X1 = 0,\ H2X2 = 10.$$

The direction of both heads must now be determined; these are the flags H1DIR and H2DIR.

$$H1DIR = H1X1-H1X2 = 0 - 10 = -10,$$

$$H2DIR = H2X1-H2X2 = 0 - 10 = -10.$$

The subroutine has detected that there is truly a motion in X for both heads, so now it must determine whether or not both heads are moving in the same direction. This is done by the following ratio test:

$$\left[\frac{H1DIR}{H2DIR} = \frac{-10}{-10} = +.\right]$$

The postprocessor now knows that both heads are moving in the same direction, and a further test indicates the motion is in the positive direction, therefore, flag DIR = 1 for this condition.

The next question to be resolved is: do both paths have the same origin, i.e., both begin at the same point? This is resolved by the test:

$$DIR(H1X1 - H2X1) = 1(0-0) = 0$$

The test indicates that the origins are the same. Now, which path is longer?

$$DIR(H1X2-H2X2) = 1(10-10) = 0.$$

The test indicates that the path lengths are equal; hence, no segmentation is necessary and the two head paths can be output together if the X-axis component feedrates of each head are within the acceptable tolerance difference.

3.4.8.2.1.3.2 PROGRAMMED PROCEDURE (cont'd)

This test is done in subroutine FXTOL where the component feedrate is determined by:

$$F_X = \frac{\Delta X. \vec{F}}{S},$$

where F_X is the component feedrate, ΔX is the X-axis increment, F is the head feedrate, and $S = \sqrt{\Delta x^2 + \Delta y^2 + \Delta z^2}$.

If the two component feedrates are not within the acceptable tolerance difference, the return flag IND is set non-zero, and the postprocessor outputs each head move separately.

In this example it is clear that the component feedrates would be the same since ΔX is the same for both heads. However, this is true only if one feedrate register exists; with two feedrate registers the component feedrates can very easily be significantly different even though the ΔX's are equal.

The two paths are made output as a combined move; but before actually outputting the block, subroutine FXPARK is called to make sure that the tools are in the workpiece. It will be recalled that the heads are both retracted and then brought together into the workpiece when a new combined sequence begins.

Subroutine GMOUT outputs the combined command block as described in Section 3.4.8.2.2.

Referring to Diagram 3.4.8.2.1.3.2A, it is seen that we have combined and output the motion from X = 0 to 10. The flags RFLAG2 and RFLAG3 are set to zero and subroutine FXMULT returns to subroutine GEMULT where a new command block from TAPES2 and TAPES3 is read. The zero settings of RFLAG2 and RFLAG3 indicate that both tapes should be read.

3.4.8.2.1.3.2 PROGRAMMED PROCEDURE (cont'd)

After obtaining the next command blocks, subroutine FXMULT again determines the parameters as described above. Continuing with our example, the same sequence and results are summarized.

ABS2	ABS3	
10	10	
H1VEC	H1X1	H1X2
20	10	20
H2VEC	H2X1	H2X2
25	10	25
H1DIR	H2DIR	
-10	-15	

$$\left[\frac{\text{H1DIR}}{\text{H2DIR}}\right] = \frac{-10}{-15} = + . \quad \text{DIR} = +1.$$

Origin test: 1(10-10) = 0 - same origin.

Test to see which path is longer:

$$\text{DIR}(\text{H1X2}-\text{H2X2}) = 1(20 - 25) = -5.$$

It is found that Head 2 has the longer path, hence, the Head 2 path will be segmented at H1X2, i.e., at X = 20. Subroutine SEG is called to perform this function; the calling sequence to subroutine SEG specifies the point of segmentation. (See Section 6.0 for complete description of subroutine.)

After segmentation the two paths are combined and made output if subroutine FXTOL so designates. Flag H2FLAG is temporarily set to 1 indicating that the Head 1 path is completed but Head 2 is not. Under these conditions, H2X1 is reset as

$$\text{H2X1} = \text{ABS3} = 20$$

since at output, ABS3 has had ΔX added to it to obtain the current absolute point.

The flag RFLAG2 is set to 0 and RFLAG3 to 1 indicating that since the Head 1 path is completed, TAPES2 must be read for a new command block whereas TAPES3 must be bypassed since the Head 2 path still has a portion to be made output.

3.4.8.2.1.3.2 PROGRAMMED PROCEDURE (cont'd)

Thus, upon return to subroutine GEMULT, only TAPES2 is read. Since Head 1 calls for a STOP, it is parked, and the remainder of the Head 2 path is made output, then parked. The two heads are resynchronized with new data from TAPES2 and TAPES3 which begins a new sequence.

The individual subroutine write-ups for the common axis segmentation sequence must be consulted for greater details; see Section 5.2.

3.4.8.2.2 MULTIHEAD OUTPUT

The calling sequence to subroutine GMOUT has an integer flag which is set according to the condition that exists in DBFSEG. For example, if an unmerged block for head 1 is to be output, then the integer flag is 1.

The possible settings are:

(IH)	1	=	head 1 only
	2	=	head 2 only
	3	=	both heads are merged, but head 1 is the primary head
	4	=	both heads are merged, but head 2 is the primary head

The first function performed by subroutine GMOUT is to determine the absolute motion values of the point; these values are used by subroutine FXMULT and GEOUT3 in printing the Absolute and Operator Printouts. This function is done by subroutine GMABS which takes the incremental values of AS2 or AS3 and algebraically summates and stores them in the array ABS2 or ABS3 as the case may be. ABS2 and ABS3 are ordered as X, Y, Z, A, B for the absolute coordinate machine point.

The command block is now readied for output; this requires computing a feedrate command and obtaining the requisite G, M, S, and T codes.

3.4.8.2.2 MULTIHEAD OUTPUT (cont'd)

A motion command block at this time has the feedrate in IPM, hence, a conversion to the feed command is required. This conversion is done by subroutine GMOTIN, and the value is stored into the eleventh cell of AS2(or AS3). This subroutine is almost identical with subroutine CONTUR except that multihead preparatory function G codes are involved in the determination of the feedrate command.

If the resultant feedrate command is greater than the feedrate command maximum (FCOMAX), the postprocessor calls subroutine GFDLIM to optimize the feedrate command by using a ratio multiplier on the feedrate command and computing an I, J, K (as appropriate) value; see Section 4.1.5 for a complete description of this method.

The arrays AS2 or AS3 are now essentially complete, but they must be restored into GMHBUF which is the prime array used for all multihead output. Subroutine GEMISC obtains from AS2 or AS3 the spindle command and speed, any pending M code or T code, and stores them into GMHBUF. The storing sequence is a function of the head as follows:

	GMHBUF	AS2
Head 1	(12) = spindle command	(12)
	(13) = tool code (T)	(13)
	(14) = miscellaneous code (M)	(14)
	(20) = spindle speed in RPM	(20)
Head 2	(32) = spindle command	(12)
	(33) = tool code (T)	(13)
	(34) = miscellaneous code (M)	(14)
	(40) = spindle speed in RPM	(20)

When both heads share a common register, the convention used in the postprocessor is that the head 2 value for that register is stored into the corresponding head 1 location of GMHBUF.

3.4.8.2.2 MULTIHEAD OUTPUT (cont'd)

Subroutine GMOUT completes the restoring of GMHBUF by adding in the remaining cells of AS2 or AS3. The order of storage is as illustrated below.

GMHBUF	AS2		GMHBUF	AS3
(1)	(1)	N	(21)	(1)
(2)	(2)	G	(22)	(2)
(3)	(3)	X	(23)	(3)
(4)	(4)	Y	(24)	(4)
(5)	(5)	Z	(25)	(5)
(6)	(6)	A	(26)	(6)
(7)	(7)	B	(27)	(7)
(8)	(8)	I	(28)	(8)
(9)	(9)	J	(29)	(9)
(10)	(10)	K	(30)	(10)
(11)	(11)	F	(31)	(11)
(12)	(12)	S	(32)	(12)
(13)	(13)	T	(33)	(13)
(14)	(14)	M	(34)	(14)
(15)	(15)	CODE	(35)	(15)
(16)	(16)		(36)	(16)
(17)	(17)		(37)	(17)
(18)	(18)	C	(38)	(18)
(19) *	(19) *	F-IPM	(39) *	(19) *
(20)	(20)	S-RPM	(40)	(20)

*This value of feedrate is stored in GMHBUF before subroutine GMOTIN is called to convert the feedrate into its command form.

When GMHBUF is all set up, it is made output through subroutine GMSTOR which directs the program flow to GEOUT3 for printing and punching.

3.4.8.2.2 MULTIHEAD OUTPUT (cont'd)

If the command block being processed is a circular interpolation move (CODE = ± 10, 11, 12,), subroutine GMOUT calls subroutine PREPHD which selects the proper preparatory function G code that permits circular interpolation on that particular head. The G code is made output in a block by itself.

The above described output sequence pertains to single-head output, i.e., head 1 or head 2 only, and also to combined multihead output. When only head 1 is to be output, GMHBUF(1) through (20) are used; when only head 2 is to be output, GMHBUF(21) through (40) are used. When multihead combined moves are output, GMHBUF(1) through (40) are used.

When a combined multihead block is processed for output, subroutine GMOUT, in addition to the above described chores, must also do some special testing and modifying.

The input head flag (IH) is 3 or 4 for a combined block. Thus, an early branch in subroutine GMOUT directs the program flow to the test which determines the nature of the combined move, that is, the flag ICRLIN is tested for the following interpolation conditions:

ICRLIN	Condition
-1	head 1 is linear, head 2 is circular
0	both heads are linear
+1	head 1 is circular, head 2 is linear
+2	both heads are circular

Common to all these conditional combined moves is the determination of the feedrate command. For the combined condition which contains a linear and a circular interpolation move, a set of "arc center offsets" may also have to be determined for the linear head.

3.4.8.2.2.1 LINEAR-LINEAR

The first item considered for the combined linear-linear multihead move is the proper determination of feedrate. Subroutine GMOTIN is called to determine the feedrate command of head 1 and head 2; the parameters FRN1 and FRN2 contain the feedrate commands. If one of the head moves is a delay, i.e., a xero move, the feedrate command for that head is set to zero.

Next, the selection of the proper preparatory function G code is made. If a preparatory function register for each head is available (option 152 ≠ 0), each head G code is used, and GMHBUF(2) and (12) select from AS2(2) and AS3(2) to obtain the proper head G code accordingly.

If only one G register is available, a dimensional G code which is compatible with the size of motions for both heads is selected and stored in GMHBUF(2).

A similar determination is made for the F register, that is, the use of one F register common to both heads or the availability of an F register for each head; option 139 is non-zero for multiple F registers. GMHBUF (11) and (31) are set to AS2(11) and AS3(11) when multiple F registers are available. Otherwise, GMHBUF(11) above is stored with the feedrate command after it has been decided which head feedrate to use. The input head flag IH specifies this, for when IH = 3, then head 1 is the primary head, and so GMHBUF(11) = AS2(11), and CODE = +17 and is so stored into GMHBUF(15). But when IH = 4, then head 2 is the primary head, and GMHBUF(11) = AS3(11), and CODE = -17 and is so stored into GMHBUF(15).

The remainder of the GMHBUF block is next set up with the other cells of AS2 and AS3 and is made output through subroutine GMSTOR.

3.4.8.2.2.2 LINEAR-CIRCULAR OR CIRCULAR-LINEAR

As with the linear-linear condition described above, a similar determination must be made concerning the use of single or double preparatory function and feedrate command registers.

If a preparatory function register exists for each head, the value is set up in GMHBUF(2) and (32) from AS2(2) and AS3(2), and no further consideration is needed since each head can function separately according to its interpolation mode.

However, if only one preparatory function register is used for both heads, the linear head "arc center offset" must be determined in the following fashion.

The length S of the circular move and the radius R of the circle are determined so that the circle angle θ can be found. The "arc center offsets" for the linear head are next computed using θ and the deltas of the linear head.

1) $S = \sqrt{\Delta X_C^2 + \Delta Y_C^2 + \Delta Z_C^2}$ (delta motions)

2) $R = \sqrt{I_C^2 + J_C^2 + K_C^2}$

3) $B1 = S/2R$

4) $B2 = \sqrt{1 - B1^2}$

5) $\theta = a\ \tan^{-1} \left(\frac{B1}{\sqrt{1-B1^2}} \right)$

6) $I_L = \Delta X_L / \theta$

7) $J_L = \Delta Y_L / \theta$ Linear head "arc center offsets"

8) $K_L = \Delta Z_L / \theta$

Note the following setting of CODE for these mixed interpolation modes:

GMHBUF(15) = CODE = +17 for circular-linear;

GMHBUF(15) = CODE = -17 for linear-circular.

The block is made output after the remainder of GMHBUF is setup.

3.4.8.2.2.3 CIRCULAR-CIRCULAR

There must be two preparatory function registers for this condition to exist. The setup of GMHBUF is direct and with no further modifications needed. GMHBUF(2) is set from AS2(2) and GMHBUF(32) from AS3(2).

The feedrate commands are likewise setup, and CODE = +17 if IH = 3, otherwise, CODE = -17 for IH = 4.

The block is made output after the remainder of GMHBUF is setup.

3.5 OUTPUT ELEMENT

The GECENT III postprocessor can produce punched output in either tape image (PUNCHA) or Hollerith BCD (PUNCHB). Either type is selected by the designation of option 20:*

OPTION 20 = 0, use PUNCHB

OPTION 20 = 1, use PUNCHA

*See Section 5.6.2 for option 20 = -1 for magnetic tape output.

Four forms of printed output are available by option selection. See the GECENT III Part Programmers Manual for a complete description of the printed output.

The four major overlays of GEOUT are: GEOUT 1, which produces the Summary Print; GEOUT 2, which produces the Combined Print; GEOUT 3, the Multiple Print, which produces any one or combination of the Incremental, Absolute, or Operator Manuscript Printouts for multihead machines; and GEOUT 4, which is similar to GEOUT 3, except used for non multihead machines. However, the GEOUT overlays are mutually exclusive and only one overlay can be used for any given run.

The incremental data produced on any of the printouts are an exact copy of the data which are punched into the control tape. The tape, of course, does not include such things as postprocessor comments, blanks, FROM point, and so on. The same printed incremental line image is also punched into the tape. This control tape is accomplished in the following manner.

3.5.1 CONVERSION TO TAPE IMAGE

The print sequences convert the elements of DBFSEG to BCD, (unless, of course, they are already in BCD form), and sets these elements in the array BCDIMG.

The elements of DBFSEG are converted to BCD through subroutine CONBCD; the conversion format of each cell is specified in the related REGFOR table. For example, DBFSEG (2) contains the floating point value -3.2468. REGFOR (2) = -24.0 specifying a signed number with two places to the left and four places to the right of the decimal. The converted value produced in BCD becomes -03.2468. This value is printed under the X column on the Incremental Printout.

3.5.1 CONVERSION TO TAPE IMAGE (cont'd)

To punch this data the BCD letter address of the related register is first obtained from the REGSTR table and is inserted in BCDIMG ahead of the BCD value. In this example the letter address is selected from REGSTR (2) to produce the punched value X-032468. The punch routines do not punch blanks or periods. Trailing zeroes are not punched unless required, as for a positioning machine. (See option 1 and 51). Leading zeroes can be suppressed under certain conditions for certain numerical control systems, the Mark Century 100M control and others. See Section 4.9 for leading zero suppression information.

3.5.2 PRINTOUT VARIABLE FORMAT

A variable format is used with all four forms of printout; that is, each print format is structured according to the needs of each machine tool. The Machine Subroutine has the tables REGSTR and REGFOR which describe the input requirements for that particular machine. These tables are also used to set up the print format for the particular machine. The REGSTR table tells the postprocessor which registers are available, and the REGFOR table specifies the decimal structure of each register. Hence the postprocessor has all the necessary information to lay out the print format.

To print any given BCD value the postprocessor must know which print columns it is to use; for example, the X values may have to be printed in columns 11 through 19, Y columns 20 through 28, and so on. These print column values are determined in overlay GEINIT in subroutine CALCPn where n = 1 for GEOUT 1, 2 for GEOUT 2, 3 for GEOUT 3, and 4 for GEOUT 4. The print column values are determined for each available register (as given by REGSTR and REGFOR tables) and stored in a print vector.

There are four vectors determined in CALCPn, namely, the vectors NIP (initial print position), NFP (final print position), NPR (number of places to right of decimal), and NPT (total number of digits in each register). Each element of the vector has a one-to-one correspondence with the register tables. For example:

3.5.2 PRINTOUT VARIABLE FORMAT (cont'd)

(SUBSCRIPT)	NIP	NFP	NPR	NPT	REGSTR	REGFOR
(1)	3	5	0	3	N	30.
(2)	8	9	0	2	G	20.
(3)	12	19	-104	-6	X	-24.
(4)	22	29	-104	-6	Y	-24.

Zero is stored if the standard register assignment does not exist. For example, if the machine tool has no T code, REGSTR (13) = DBLNKS, REGFOR = 0, and thus NIP (13) = NFP (13) = NPR (13) = NPT (13) = 0.

The postprocessor determines the optimum spacing between columns before calculating the initial and final print positions. Hence, when GEOUT 1 is used to produce the Summary Print, the postprocessor first determines the number of columns it must set aside for the incremental data, and then checks to see how much room is left for the absolute data. The print vectors NIP and NFP are altered to produce the optimum spacing format. For example NIP (I) and NFP (I) for I=2,3,4,---14 may be increased by 3 to provide three columns of space between each printed value.

The print vectors also give additional information which is used to make format decisions in printing and punching. A negative value of the vector NPT tells the postprocessor that the algebraic sign of the register value must be made output. This is important since the XYZ registers may carry signs, whereas the G register does not. Also, if a value n of the NPR vector is over 100, i.e., (n + 100), this signals the postprocessor that the trailing zeroes are to be dropped for this particular element's output value. Positioning machines, which utilize an absolute coordinate system, must output the trailing zeroes, therefore, the elements of the print vectors for the registers XYZ would each be a value less than 100. A negative value of an element in the print vector NPR specifies that the related register value must also have a decimal point in its printed value.

The print vectors are stored in COMMON and are therefore available to all output subroutines. The output initialization subroutine GEPRE calls the proper subroutine CALCPn. Subroutine GEPRE and associated subroutines are overlayed when all the basic initialization is completed.

3.5.2 PRINTOUT VARIABLE FORMAT (cont'd)

The CALCPn subroutine in addition to calculating the print vector elements for the Incremental Printout (and subsequent punching), also analogously determine the print positions and other information for the Absolute and Operator Printouts. In general, the subroutines will expand the print width of the motion register values or incremental systems so as to encompass the probable programmed algebraic dimensions. For example, REGFOR(3) = -24.0 specifies a six digit incremental move for X. The Absolute format for X is made to have an eight digit spread to ensure that any programmed point will be properly printed.
The following sample illustrates the method.

REGFOR(3) = -24.0

	INCREMENTAL	ABSOLUTE
	X	X
FROM	892.3456	0892.3456
	95.6789	0988.0245
	10.0010	0998.0255
	90.1111	1088.1366
	(6 digit spread)	(8 digit spread)

A similar expansion is also made if the units system of the machine tool is metric. Note also that the FROM point format is also expanded.

The register title printed at the top of each page is set up in accordance with the values of the print vectors. The title structure is determined in CALCPn and then saved in the vector BCDREG which is in COMMON. When the title for each new page is required, subroutine TITLEn is called. This subroutine prints the postprocessor identification title, the machine tool identification and page number, the PARTNO, and then the register title as stored in BCDREG. A possible title printout for the Summary Printout GEOUT1 is given below.

3.5.2 PRINTOUT VARIABLE FORMAT (cont'd)

```
           ***GENERAL ELECTRIC GECENT III POSTPROCESSOR***

   MACHINE 14     BRACK LATHE NC40

   TEST CASE 2468A1001

   N  G  X  Z  I  K  F  S  T  M  ABSX  ABSZ  FIPM  SRPM
```

The sequence through subroutine CALCP1 is described in detail in Section 3.5.4.1, and should be read as a continuation of the above description.

3.5.3 GENERAL OUTPUT FLOW

The logical flow in the output section is basically the same for all the GEOUT's. GEOUT1 is the simplest output sequence whereas GEOUT3 is the most complex. It must also be remembered that GEOUT3 and GEOUT4 require multiple passes to produce the several printouts, hence, will obviously have a more complex and longer flow path. Each of the separate GEOUT's are discussed below in detail.

The structure of each GEOUT is functionally the same, i.e., the key subroutines of each GEOUT perform analagous functions. Therefore, the description which follows pertains to GEOUTn where n = 1,2,3, or 4.

For a complete description of the variable printout method, see Section 3.5.4.1 wherein subroutine CALCP1 is analyzed in detail. This description defines the basic techniques used by all the GEOUT's in the setting up and use of the column vectors.

3.5.3.1 INITIALIZATION

Initialization occurs in subroutine GEPRE in overlay GEINIT which performs all the "one-shot" chores required for output functions:

(1) Subroutine DECODE interprets options 59 and 60 and sets up the shuffle vector ISHVEC which directs the postprocessor in its reassignment of registers for output. This reassignment is done by subroutine SHUFFL which is called in subroutine GEPROn. The shuffle vector ISHVEC (dimensioned at six) has a fixed order, viz.,

			Standard value
ISHVEC(1)	=	X	3
ISHVEC(2)	=	Y	4
ISHVEC(3)	=	Z	5
ISHVEC(4)	=	I	8
ISHVEC(5)	=	J	9
ISHVEC(6)	=	K	10

If the machine tool is a standard milling machine, the above order of ISHVEC is unchanged and is not used by subroutine SHUFFL. In this case, flag ISHUFL = 0.

However, a standard lathe operates in Quadrant IV and uses the axes +Z-X and related registers +K+I. Therefore, ISHVEC is set as follows (determined by options 59 and 60:

ISHVEC(1)	=	5
ISHVEC(2)	=	-3
ISHVEC(3)	=	0
ISHVEC(4)	=	10
ISHVEC(5)	=	8
ISHVEC(6)	=	0

3.5.3.1 INITIALIZATION (cont'd)

Subroutine SHUFFL uses this vector to "shuffle" the normal command block data into the output format required for the particular machine tool. The command block DBFSEG for a lathe originally is set up as:

1	2	3	4	5	6	7	8	9	10	11	-	-
N	G	X	Y				I	J	-	F	-	-

After shuffling, DBFSEG is reordered as:

1	2	3	4	5	6	7	8	9	10	11	-	-
N	G	-Y		X			J	-	I	F	-	-

For example, the value of ISHVEC(1) being a 5 indicates that the normal value stored for X (at DBFSEG(3)) should be relocated at DBFSEG(5). This value is made output with the letter address Z thereby fulfilling the requirement of the abscissa lathe output.

A zero value in ISHVEC means to disregard the referenced register. The flag ISHUFL is set to 1 when shuffling is required.

(2) Output parameters and flags are set to their zero or null starting values. Storage arrays are cleared; BCD arrays are set to DBLNKS; conditional storage arrays are set to DMBITS.

(3) Subroutine CALCPn is called to set up the print column vectors and to determine the register title BCDREG.

(4) The first PARTNO is converted to its readable form and punched out. The readable PARTNO is produced either for Tape Image (PUNCHA) or BCD Hollerith (PUNCHB).

For Tape Image (option 20 = 1) subroutine PARNEM is used for even parity check readers, and subroutine PARNOM is used for odd parity check readers.

For BCD Hollerith (option 20 = 0) subroutine IDPART is used.

(5) The first PARTNO is saved in vector DPRTNO (dimensioned at 11) for later printing as a title on each page.

3.5.3.2 OUTPUT PROCESSING

Normal output processing for a command block relies principally on subroutine GEPROn which is the main processing unit of GEOUT. In addition to processing each command block, subroutine GEPROn also performs several other functions.

(1) The unit increasing sequence number (option 143=1) is generated in subroutine GEPROn. When called for, the generated number is stored in DBFSEG(1) and the CL tape record number is then printed at the far right side of the page under the title heading, CLREC.

The unit increasing sequence number is limited to the maximum size number permitted by the N register format. For example, if REGFOR(1) = 30.0, the largest number that N can become is 999. The maximum number is determined in subroutine ASSIGN in GEINIT and stored in the parameter SEQLIM. A number N greater than SEQLIM is made.

N = N modulus SEQLIM

(2) Incremental motions are accumulated to produce the absolute coordinate values. The absolute values are remodified by subtracting the respective ORIGIN value(stored in the array ORGIN) to produce the Operator's Print when requested. This, in effect, is simply done by subtracting the ORGIN(I) value from the relative FROM(I) value, where I = 1,2,3,4,5.

The accumulated incremental moves for the absolute or operator values are stored in the array DABVAL(I) for I = 3 to 7. DABVAL(11) is for the feedrate in IPM, and DABVAL(12) is for the spindle speed in RPM.

(3) After printing the BCDIMG for the command block, subroutine PPUNCH is called to count the command block characters, and then punch the command block. The block is punched according to the setting of option 20, that is, either for PUNCHA or PUNCHB.

3.5.3.2 OUTPUT PROCESSING (cont'd)

(4) A final chore of subroutine GEPROn is to print the total cut and dwell times, and the total tape footage generated by the part program.

There are several other special sequences performed in GEPROn which are described in detail in the sections pertaining. Some of these items are: processing of a dwell block (Section 4.10); turret corrective moves (Part Programmer Manual Section 4.8.13) processing of information blocks (Section 5.5); OPSKIP processing (Section 4.11); and determination of cut and dwell times (Section 4.12).

The prime purpose of subroutine GEPROn is to set up the command block for printing and punching. As described in Section 2.3.2, each non-BCD block has its floating point values converted to BCD through subroutine CONBCD. Subroutine SETLIM is then used to prepare the array BCDIMG which is printed by subroutine GEPRNn. Next, again using subroutine SETLIN, each letter address from the REGSTR table is inserted before each register value existing in BCDIMG. The BCDIMG is then punched by subroutine PPUNCH.

Each command block is processed differently according to its CODE value. However, there are common processing routes for some different CODE settings. For example, when the command block is a motion of dwell block, subroutine SHUFFL is called to rearrange and modify the XYZ IJK registers per option 59 and 60. Dwell blocks are remodified, as needed, to produce the proper dwell code. (See Section 4.10). If the command block is for an incremental motion, subroutine CONTUR is called to compute the feedrate command. If the command block is for an absolute motion (as for a positioning machine), the subroutine POSIT is called to suppress redundant coordinate points; this subroutine in turn calls subroutine POSFED to produce the positioning feedrate command.

According to the related option setting, (options 38 and 107) a command block which has a redundant G or F command has its redundancy suppressed. Redundant S commands are always suppressed except for a spindle neutral or when immediately following a STOP, OPSTOP or SPINDL/OFF.

3.5.3.2 OUTPUT PROCESSING (cont'd)

Other than the above items, special processing of each command block is done according to its CODE value. The description below briefly details the output processing for each type of command block. At this point in the program, DBFSEG is completely set up, and the CODE value is also set in ICODE as

$$ICODE = |CODE| + 1.$$

CODE=0

The command block is for a linear, incremental move. Processing is described above. The converted block is printed and punched.

CODE=+1

The command block is a non-motion block. The block may require an F command to produce a momentary dwell; see option 98. Or possibly, the block may require a dwell produced by a G04; see option 148. The converted block is printed and punched.

CODE=+2

The command block is for a rotary motion. The rotary value is converted to output units; but other than this, processing is as described above for a motion block. The converted block is printed and punched.

CODE=+3

The command block is for a FROM part. The block is shuffled per subroutine SHUFFL, and the accumulation of the Absolute and/or Operator Data is begun. The title FROM is printed as are the values of the FROM point. The FROM point is not punched.

CODE=-3

The command block is for a turret corrective move as generated by the NOW modifier. This block is processed exactly as if it were for a CODE=0 except that the corrective incremental moves are not summated nor printed on the Absolute Printout, though they are for the Operator Printout. (See Part Programmer Manual Section 4.8.13 for the explanation regarding corrective moves.) Otherwise, the converted block is printed and punched.

3.5.3.2 OUTPUT PROCESSING (cont'd)

CODE=+4

The command block is for a dwell block and is processed as described above. The block is printed and punched.

CODE=-4

The command block is for a preparatory G code by itself, not a dwell. The block is tested at DBFSEG(11) for the existence of an F command in the event that the block may be for a CUTCOM. The F command is left untouched if one exists, otherwise, the processing considers the possibility of adding in an F command per option 98. G codes in CODE = -4 blocks are never suppressed regardless of redundancy. The block is printed and punched.

CODE=+5

The command block is for an END. This code signals the postprocessor to perform special actions before output, but at output the block is treated as if it were for a CODE=≠1 except that no testing is made for redundant G,F, or S commands. The block is printed and punched.

CODE=-5

The block is for a RESET statement. This statement calls for a resetting and reinitialization of the postprocessor program, and as such, is not an item for output. The block is neither printed nor punched.

CODE=+6

The command block is for an INSERT statement and is already set up in BCD form. The block is immediately printed and punched.

CODE=-6

The block is for a BREAK statement. This CODE signals the postprocessor to produce the "breaking" of the output tape; and, if the machine tool recognizes an OPSTOP command (TABLEG(2) ≠ DMBITS), the BREAK statement also issues an OPSTOP block. Hence, processing can follow two possible routes: (1) if TABLEG(2) ≠ DMBITS, the block is processed as for a CODE = -1 and is printed and punched; (2) if TABLEG(2) = DMBITS, the block is disregarded and is neither printed nor punched. See Part Programmer's Manual for further information on the BREAK statement.

3.5.3.2 OUTPUT PROCESSING (cont'd)

CODE=+7

The command block is for a PPRINT statement and is already in BCD form. The block is immediately printed but is not punched.

CODE=-7

The command block is for a PARTNO statement and is already in BCD form. The first programmed PARTNO is converted and made output in readable format; it is also saved in coded (non-readable) form in the vector PART (dimensioned at 11 and in COMMON) from where it is printed as a title on each page. The command block is printed and punched.

CODE=+8

The block is for a TMARK statement. No sequence number N is issued in a TMARK block, since the TMARK is stored in DBFSEG(1) for output. The title TMARK is printed at the left side of the page for each TMARK; the BCD value of TMARK (option 65) is punched.

CODE=-8

The block is for a LEADER statement. The requested amount of leader length is given in DBFSEG(3) so the postprocessor produces a length of at least this amount. The generated length will usually be slightly larger than requested since the postprocessor issues leader codes on a full card of 72 columns. For example, the statement LEADER/48 was given. The number N of full cards produced is

$$N= < 48*0.14 + 1 > = 7 \text{ cards.}$$

The actual leader length L produced is

$$L = 7.2 * 7 - 1 = 49.4 \text{ inches.}$$

The actual leader length is printed in the statement, "49.4 INCHES OF LEADER HERE"; the same amount is punched using the BCD value at option 64 for the leader code.

3.5.3.2 OUTPUT PROCESSING (cont'd)

CODE=+9

The block is for a postprocessor warning or error comment block, and as such, is already in BCD form. The comment is printed but is not punched. If the Multiple Printouts (GEOUT3) is used, the comment is printed only on the Incremental Print.

CODE=-9

The block is for a postprocessor information block; see Section 5.5. The information in the block is used for output purposes, but the block is neither printed nor punched.

CODE=±10,±11±12

The command block is for a circular interpolation move and is processed as a motion block as described above. The block is printed and punched.

CODE=±13

The command block is for a thread. The block is processed exactly as for a motion block except that subroutine CONTUR is not called since a feed command need not be generated. The block is printed and punched.

CODE=±14

The command block is for a multihead turret corrective move. +14 indicates a turret correction is made on head 1 while there is a combined, normal motion on head 2; -14 is the inverse effect for the heads. The head with the turret corrective move is processed as if for a CODE = -3, while the other head is processed as for a regular motion. The turret corrective moves are printed and punched but are not summated nor printed on the Absolute Print. The regular motion is both printed and punched.

CODE=+15

The command block is for a motion which retains its feed in IPM as the feedrate command. It is processed as a regular motion block except that subroutine CONTUR is bypassed since there is no need for generating a feedrate command. The block is both printed and punched.

3.5.3.2 OUTPUT PROCESSING (cont'd)

CODE=±16

The command block is for a positioning move. It is processed essentially the same as for an incremental move except that subroutine POSIT is called instead of subroutine CONTUR, and there is no accumulation of increments to produce an Absolute Print. The block is printed and punched.

CODE=±17

The command block is for a multihead combined motion. This type block is processed only in GEOUT3; an error is assumed in the other GEOUT's. The block is processed as a normal motion block except that frequent tests are made on the head flag HEADGB to decide the proper disposition of DBFSEG data relative to each head. For example, when the test is made for possible suppression of redundant G codes, the postprocessor must know from which head the G code it is processing has come before it can legitimately effect a suppression. Since DBFSEG (1 to 20) is set up the same for each head, the test on the HEADGB flag is the only means of knowing which head is being processed.

Subroutine CONTUR is also bypassed for a CODE of ±17 since the multihead feedrate has already been predetermined. The command block is both printed and punched.

CODE=18

The block is for a FINI statement, and signals the postprocessor to conclude its processing. The punch buffers are emptied, the total cut and dwell times and tape footage is printed, and a general wrap-up of the program occurs. The block is neither printed nor punched.

3.5.4 GEOUT1 (SUMMARY PRINTOUT)

The GEOUT1 printout is the fastest and simplest output segment in the GECENT III postprocessor. It is designed to handle the vast majority of non-multiaxis, non-multihead machine tools, and is especially adaptable to positioning machines, lathes, and two-or three-axis milling machines.

Its print format provides the reader with an easy-to-read summary of the control data in its tape coded form and an interpretation of the data in absolute terms. On the left side of a printed page appear the punched tape image data of each command block. The sequence number (if any) identifies each command block. The column headings identify each of the registers utilized by the particular machine. The feedrate column gives the command value of the feedrate. Similarly, the spindle speed column gives the command value.

On the right side of the page are given the absolute values of the same point. (Actually, the data can be either Absolute Data or Operator Data; but for convenience, reference is made only to the Absolute Data.) Feedrates are given in IPM and spindle speeds in RPM. If the machine has a rotary table, the absolute data reflects the absolute table location in degrees.

Actually, two types of print are developed in GEOUT1: one for an incremental system and one for an absolute system.

When the machine tool is of an incremental system, the data printed on the left side of the page are the incremental values x, y, or z as the case may be. On the right side of the page are the summated absolute values of these increments.

When the machine tool is of an absolute system, the x, y, z data are already given in their absolute form; hence, there is no need for a double representation of these values.

The accumulated cut and dwell times are given at the bottom of each page, and on the final page the total tape footage is also given.

See Diagram 3.5.4.1 for a sample printout of GEOUT1.

3.5.4 GEOUT1 (SUMMARY PRINTOUT) (cont'd)

GEOUT1 is selected when option 164 is 1, the standard value. Since GEOUT1 summarizes all the register data on one page, it is impossible to apply GEOUT1 for all machine tools, i.e., to completely represent all the register data for machine tools which have a large number of registers with large formats. For example, a three-axis mill with circular interpolation, programmable spindle, and tool changer would have the registers N G X Y Z I J K F S T M. Representing all of these registers plus the Absolute Values X, Y, Z, F, and S all on one line across a page would be either impossible or an extremely tight squeeze. The postprocessor, therefore, makes some decisions when developing the print layout. First of all, the postprocessor will attempt to produce all of the registers on one line. If this is not possible, the postprocessor drops the absolute value column of spindle speed in RPM. If there still is not enough room, it drops the feedrate column in IPM. If this also fails, the postprocessor rejects the option of using GEOUT1, and uses GEOUT2 after first printing a warning comment to the user.

These decisions are normally made in subroutine CALCP1 where the column indices are determined from the REGFOR table.

```
  P3882   DATE- 10/06/69       CUTTER LOCATION LISTING(SECTION III ).       TIME OF DAY- 23HR.   33MIN.
*** COMMANDS ************************************************************* CARD NO    INT SEQNO  CLREC NO

 EXAMPLE OF GEOUT 1 PRINTOUT                                                                1         2
MACHIN/GECENT,   24.0000,  OPTAB,  132.0000,    2.0000,  164.0000,$                         2         4
          1.0000
LEADER/  24.0000                                                           2                4         6
 FROM /                                                                    4                5         8
                          X               Y               Z
                       0.              0.              0.
FEDRAY/   2.0000,    IPM                                                   5                6        10
SPINDL/ 210.0000                                                           6                7        12
TOOLNO/  12.0000,     5.0000                                               7                8        14
 DS IS/                                                                    8                9        16
                          X               Y               Z
                     2.0000000       4.0000000       0.
 CYCLE/  FACE,     2.0000,    3.0000,     6.0000                           10              10        18
 DS IS/                                                                    11              11        20
                          X               Y               Z
                     8.0000000       8.0000000       0.
 CYCLE/   OFF                                                              12              12        22
 DS IS/                                                                    13              13        24
                          X               Y               Z
                     4.0000000       4.0000000       0.
 CYCLE/  DEEP,     2.0000,    3.0000,     8.0000,     IPM                  14              14        26
 CYCLE/ DRILL,     7.0000,    1.0000,    10.0000                           14.1            15        28
 CYCLE/   TAP,     2.0000,    4.0000,     6.0000                           15              16        30
 CYCLE/  BORE,     4.0000,    5.0000,     0.0200,     IPR                  16              17        32
 DS IS/                                                                    17              18        34
                          X               Y               Z
                     8.0000000       8.0000000       0.
 CYCLE/  MILL,     5.0000,    6.0000,     0.0100                           18              19        36
 CYCLE/   OFF                                                              19              20        38
 TRANS/   2.0000,     2.0000,     0.                                       20              21        40
 DS IS/                                                                    21              22        42
                          X               Y               Z
                     9.0000000       9.0000000       0.
 CYCLE/  THRU,     6.0000,    7.0000,     8.0000,     IPM                  22              23        44
 CYCLE/   OFF                                                              23              24        46
 DS IS/                                                                    24              25        48
                          X               Y               Z
                     8.0000000       8.0000000       0.
TOOLNO/   2.0000,     6.0000                                               25              26        50
 CYCLE/   OFF                                                              27              27        52
 DS IS/                                                                    28              28        54
                          X               Y               Z
                     6.0000000      10.0000000       0.
 CYCLE/  DEEP,     5.0000                                                  29              29        56
REWIND/   1.0000                                                           30              30        58
LEADER/  24.0000                                                           31              31        60
 FINI                                                                      32              32        62

                                ****** END OF SECTION III ******
  P3882   DATE- 10/06/69                          G.E. APT III 10/01/67  TIME OF DAY= 23HR.   33MIN.

                     1  PARTNO EXAMPLE OF GEBUT 1 PRINTOUT
                     2        MACHIN/GECENT,24,OPTAB,132,2,164,1
                     3        CLPRNT
  2                  4         LEADER/24
  4                  5         FROM/0,0,0
  5                  6        FEDRAT/2,IPM
  6                  7        SPINDL/210
  7                  8        TOOLNO/12,5
  8                  9        GO TO/2,4,0
  10                10        CYCLE/FACE,2,3,6
  11                11         GO TO/8,8,0
  12                12         CYCLE/OFF
  13                13         GO TO/4,4,0
  14                14        CYCLE/DEEP,2,3,8,IPM
  14.1              15        CYCLE/DRILL,7,1,10
  15                16        CYCLE/TAR,2,4,6
  16                17        CYCLE/BORE,4,5,0.02,IPR
  17                18         GO TO/8,8,0
  18                19         CYCLE/MILL,5,6,0.01
  19                20         CYCLE/OFF
  20                21         TRANS/2,2,0
  21                22         GO TO/9,9,0
  22                23         CYCLE/THRU,6,7,8,IPM
  23                24         CYCLE/OFF
  24                25         GO TO/8,8,0
  25                26         TOOLNO/2,6
  27                27         CYCLE/OFF
  28                28         GO TO/6,10,0
  29                29         CYCLE/DEEP,5
  30                30         REWIND/1
  31                31         LEADER/24
  32                32         FINI
```

```
                                                   ***GENERAL ELECTRIC POSTPROCESSOR GECENT-3 ***
                                        MACHINE    24,
                            EXAMPLE OF GEOUT 1 PRINTOUT
  N      G          X              Y             Z          F      S      T      M        R               F-IPM        S-RPM
 EXAMPLE OF GEOUT 1 PRINTOUT
      27.8   INCHES OF LEADER HERE
FROM          +0000.0000    +0000.0000    +0000.0000
LOWEST RANGE THAT SPINDLE SPEED FALLS IN IS ASSUMED
  014    80                                                03     12                                          +0200
  016            +02.0000      +04.0000                 04                                      +0002.        +0200
OPTION FEEDRATE MODE ASSUMED
  018    82                                 +02.0000    12                          +03.00          +0006.        +0200
  020            +08.0000      +08.0000                                                             +0006.        +0200
  024    80      +04.0000      +04.0000                                                             +0006.        +0200
  026    83                                 +02.0000    16                          +03.00          +0008.        +0200
  028    81                                 +07.0000    20                          +01.00          +0010.        +0200
  030    84                                 +02.0000    12                          +04.00          +0006.        +0200
  032    85                                 +04.0000    08                          +05.00          +0004.        +0200
  034            +08.0000      +08.0000                                                             +0004.        +0200
  036    86                                 +05.0000    04                          +06.00          +0002.        +0200
  042    80      +11.0000      +11.0000                                                             +0002.        +0200
  044    87                                 +01.0000    16                          +02.00          +0008.        +0200
  048    80      +10.0000      +10.0000                                                             +0008.        +0200
  050                                                          03     02                            +0008.        +0200
  054            +08.0000      +12.0000                                                             +0008.        +0200
  056    83                                 -01.0000                                -06.00          +0008.        +0200
  058                                                                        30                     +0008.        +0200
*FROM* MUST BE GIVEN AFTER AN *END* OR *RESET*
      27.8   INCHES OF LEADER HERE
NO END STATEMENT HAS BEEN GIVEN BEFORE THE FINI STATEMENT
    TAPE FOOTAGE      8
```

3.5.4.1 DETAILED DESCRIPTION OF CALCP1

As described in Section 3.5.3.1, subroutine CALCP1 is called from subroutine GEPRE when GEOUT is in core. The column vectors NIP, NFP, NPR, and NPT are set up for both the Incremental and Absolute printouts.

One of the first items that subroutine CALCP1 determines is whether to produce absolute coordinate data or operator data; option 172 dictates which to use. If option 172 is zero, the title heading

```
ABS-X   ABS-Y   ABS-Z   ABS-A
```

is printed; if the option is +1, the title heading

```
OPR-X   OPR-Y   OPR-Z   OPR-A
```

is printed. If any of the register addresses is unavailable, i.e., REGSTR(Y) = DBLNKS, that register is deleted from the title. For example, if the NC machine has only the X and Z registers, references to Y and A do not appear.

```
REGFOR(4) = REGFOR(6) = 0

REGSTR(4) = REGSTR(6) = DBLNKS
```

The title appears as:

```
N   G   X   Z   ----M    ABS-X  ABS-Z  F-IPM  S-RPM
```

Optimum and equal spacing is provided between the register columns.

A final change to the title is made if the NC machine utilized the metric system. If option 138 is +1 (indicating the metric system), the absolute title reference, F-MPM, for feedrate in millimeters per minute is used instead of F-IPM.

The derived absolute title is temporarily stored in BCD form in the vector ABWORD. Later on in the subroutine it is used to form the permanent complete title that appears on each page.

Section 3.5.2 describes the manner in which the column vectors NIP, NFP, NPR, and NPT are each determined and set up in accordance with the given values in the REGFOR table. These vectors are next extended to include the column format data for the Absolute Printout.

3.5.4.1 DETAILED DESCRIPTION OF CALCP1 (cont'd)

The feedrate in IPM column index is at vector location 25; thus NPR(25) is set to -103 indicating that the printed value of feedrate in IPM must show three places to the right of the decimal point, must drop trailing zeroes, and the decimal point must be printed. The feedrate in IPM is printed with this large decimal format so as to embrace very small programmed values. The other column vectors NPT, NIP, and NFP are also set up for the feedrate in IPM. NPT and NFP are expanded by one if the metric system is indicated.

A similar setup for the spindle speed in RPM is also performed on the column vectors at vector location (26).

Knowing the number of print columns which are to be used and knowing the number of registers to be printed, the postprocessor determines and adds the space increment to each element of NIP and NFP to obtain optimum spacing between each printed column.

At this point subroutine CALCP1 has determined all the information necessary to print and punch the output. The final task is to set up the print title as a permanent BCD image. This is done by using subroutine SETLIN and the vectors NIP and NFP. The subroutine SETLIN places a right justified BCD word into a given array; it stores the right-most BCD characters of the given work into the given array beginning at the given initial position through the given final position. For example, the

```
   CALL SETLIN (REGSTR(2), 4,4, BCDREG)
```

stores the BCD character at REGSTR(2), which is

```
   1   1   1   1   1   G,
```

into location (4) of the array BCDREG. Actually, the location value 4 derives from the calculation of

$$\frac{NIP(2) + NFP(2)}{2} = \frac{3 + 5}{2} = 4$$

The average is taken in order to place the register title centrally over the print column.

By this technique the print title for both the Incremental and Absolute Printouts is set up and permanently stored into the array BCDREG. A call to subroutine TITLE1 prints the contents of BCDREG to produce the title.

3.5.4.1 DETAILED DESCRIPTION OF CALCP1 (cont'd)

Section 3.5.3 details the overall general flow that occurs in GEOUT; therefore, little more need be added to that description since the program flow of GEOUT1 is substantially the same. The only minor difference is in printing the title (using subroutine TITLE1) and printing each BCD converted command block BCDIMG (using subroutine GEPRN1).

3.5.5 GEOUT2 (COMBINED PRINTOUT)

GEOUT2 is selected when option 164 is set equal to 2, or when GEOUT1, though called, cannot be used (see Section 3.5.4). The chief advantage of GEOUT2 is that it presents the Incremental, Absolute, and Operator data on consecutive lines all in one combined printout on each page. This makes it especially attractive for checkout and debugging purposes since the output data can be easily checked in its various output forms. Next to GEOUT, it is the fastest processing output sequence and is especially adaptable to multiaxis processing though it may be used for any machine tool type except multihead machines. It is not recommended for positioning machines because the Absolute and Operator Data Printouts are redundant with the regular printout and are a waste of computer time.

The print format of GEOUT2 provides in sequential order a representation of printed output as it exists for the Incremental, Absolute, and Operator Printouts in that order. The printed Incremental data is an exact copy (without letter addresses) of the punched output; decimal points and blanks are not punched.

Across the top of the page is printed the machine registers title which is derived from the NC machine's related REGFOR and REGSTR tables. Each line printed is preceded at its left most side by a title identifying the printout type for that line, as INC for the Incremental Print, ABS for Absolute, and OPR for Operator's Printout. These three type lines are printed only for motion records; for non-motion records only the INC line is printed except when the non-motion block contains a spindle speed, in which case the ABS line is also printed.

The INC line is the true reflection of what appears on the control system tape.

The ABS line represents the summated motion values and gives the feedrate in IPM and spindle speed in RPM. Rotary values are given in degrees.

3.5.5 GEOUT2 (COMBINED PRINTOUT) (cont'd)

The OPR line represents the motion data in terms of machine orientation, i.e., the summated motion values are modified by the given ORIGIN values.

If a sequence number (usually N) exists, GEOUT2 automatically makes it a unit increasing number irrespective of the setting of option 143. This value appears in the INC and OPR lines. However, the ABS line carries the CL tape record number as its sequence number, thereby making it easy to correlate each output line with its source CL tape record.

A sample printout could be as follows.

```
       N     G     X     Y     Z      F     S     T     M

INC   040   01    2.4  -0.1   1.12   075   28    02    08

ABS   234        13.6   0   -22.6    20   80

OPR   040   01    3.6  -10  -32.6    075   28    02    08

INC   041   04    0.4

INC   042                                              01

INC   043   04    0.8                      26

ABS   246                                  60
```

The accumulated cut and dwell times are given at the bottom of each page, and on the final page the total tape footage is also given.

See Diagram 3.5.5 for a sample printout of GEOUT2.

Processing of GEOUT2 begins in subroutine CALCP2 in GEINIT where the column vectors NIP, NFP, NPR, and NPT are determined and set up per the description of Section 3.5.2. In addition to these column vectors, the vectors NIPA and NPTA for the Absolute data are also determined and set up. These vectors are used for the Operator data also.

3.5.5 GEOUT2 (COMBINED PRINTOUT) (cont'd)

The methods and techniques used in subroutine CALCP2 are the same as delineated in Section 3.5.4.1 for subroutine CALCP1 except that the additional vectors NIPA (initial print position for Absolute Printout) and NPTA (total number of digits per register) are developed by modifying the NIP and NPT vector elements for registers XYZABIJKF and S so as to expand the printout of these registers. As explained earlier, it is essential to make these formats broader because the absolute algebraic references will normally be numerically larger than permitted by the incremental data format.

Section 3.5.3 details the overall general flow that occurs in GEOUT; therefore, little more need be added to that description since the program flow of GEOUT2 is substantially the same. The only minor differences are the following:

(1) The title is printed by subroutine TITLE2.

(2) Each BCD converted command block BCDIMG is printed by subroutine GEPRN2.

(3) The array ABSVAL carries the Absolute data. while the array OPRVAL carries the operator data.

(4) The BCDIMG for the Absolute and Operator Printout is set up using the column vectors NIPA and NPTA.

(5) To print the title INC, ABS, or OPR, the related BCD equivalent is stored into the parameter IDLINE which then is printed by subroutine GEPRN2.

(6) The setting up and printing of each command block BCDIMG for the INC, ABS, and OPR lines is done in three independant looping areas of GEPRO2.

```
P3882   DATE- 10/06/69                G.E. APT III 10/01/67           TIME OF DAY- 23HR.    35MIN.

         1  PARTNO EXAMPLE OF GEOUT 2 PRINTOUT
         2  MACHIN/GECENT,23,OPTAB,132,1,164,2
         3          CLPRNT
         4          CUTTER/0
         5          ORIGIN/1,1,1
         6          FROM/1,1,1
         7          FEDRAT/60,IPM
         8          GO TO/4,5,6
         9          STOP
        10          GO TO/6,7,8
        11          GO TO/6,7,12
        12          GO TO/6,7,8
        13          GO TO/6,0,0
        14          GO TO/0,0,0
        15          RAPID
        16          GO TO/20,20,20
        17          GO TO/0,0,0
        18          STOP
        19          C1=CIRCLE/5,3,1
        20          L1=LINE/(POINT/0,2),RIGHT,TANTO,C1
        21          L2=LINE/(POINT/1,1),LEFT,TANTO,C1
        22          INDIRP/(POINT/0,2,0)
        23          GO/L1
        24          TLRGT,GORGT/L1
        25          GOFWD/C1
        26          GOFWD/L2,ON,L1
        27          GO TO/0,0,0
        28          LEADER/24
        29          END
        30          FINI
```

```
  P3882   DATE- 10/06/69       CUTTER LOCATION LISTING(SECTION III ).        TIME OF DAY= 23HR.    35MIN.

*** COMMANDS ************************************************************** CARD NO    INT SEQNO  CLREC NO

 EXAMPLE OF GEOUT 2 PRINTOUT                                                                 1          2
MACHIN/GECENT,   23.0000,  OPTAB,  132.0000,    1.0000,  164.0000,$                          2          4
          2.0000
CUTTER/   0.                                                                                 4          6
ORIGIN/   1.0000,    1.0000,    1.0000                                                       5          8
 FROM /                                                                                      6         10
                   X               Y                Z
               1.0000000       1.0000000        1.0000000
FEDRAT/  60.0000,    IPM                                                                     7         12
 DS IS/                                                                                      8         14
                   X               Y                Z
               4.0000000       5.0000000        6.0000000
  STOP                                                                                       9         16
 DS IS/                                                                                     10         18
                   X               Y                Z
               6.0000000       7.0000000        8.0000000
 DS IS/                                                                                     11         20
                   X               Y                Z
               6.0000000       7.0000000       12.0000000
 DS IS/                                                                                     12         22
                   X               Y                Z
               6.0000000       7.0000000        8.0000000
 DS IS/                                                                                     13         24
                   X               Y                Z
               6.0000000       0.               0.
 DS IS/                                                                                     14         26
                   X               Y                Z
               0.              0.               0.
 RAPID                                                                                      15         28
 DS IS/                                                                                     16         30
                   X               Y                Z
              20.0000000      20.0000000       20.0000000
 DS IS/                                                                                     17         32
                   X               Y                Z
               0.              0.               0.
  STOP                                                                                      18         34
 DS IS/    L1                                                                               23         37
                   X               Y                Z
               0.              2.0000000        0.
 DS IS/    L1                                                                               24         39
                   X               Y                Z
               5.0000000       2.0000000        0.
    C1(   0) = CIRCLE/   5.0000   3.0000   0.       1.0000                                  25         41
 DS IS/    C1                                                                               25         42
```

X	Y	Z
5.0317488	2.0000039	0.
5.0939257	2.0039186	0.
5.1557384	2.0116955	0.
5.2169473	2.0233045	0.
5.2773149	2.0387005	0.
5.3366073	2.0578239	0.
5.3945945	2.0806005	0.
5.4516517	2.1069420	0.
5.5057600	2.1367463	0.
5.5585073	2.1698977	0.
5.6090890	2.2062678	0.
5.6573089	2.2457155	0.
5.7029803	2.2880878	0.
5.7459259	2.3332206	0.
5.7859793	2.3809387	0.
5.8229851	2.4310572	0.
5.8567998	2.4833818	0.
5.8872923	2.5377094	0.
5.9143445	2.5938296	0.
5.9378514	2.6515247	0.
5.9577218	2.7105789	0.
5.9738787	2.7707394	0.
5.9862596	2.8317968	0.
5.9948142	2.8935064	0.
5.9995156	2.9556289	0.
6.0003394	3.0179235	0.
5.9972845	3.0801486	0.
5.9903628	3.1420629	0.
5.9796010	3.2034263	0.
5.9650408	3.2640010	0.
5.9467388	3.3235521	0.
5.9247659	3.3818486	0.
5.8992073	3.4386645	0.
5.8701671	3.4937796	0.
5.8377429	3.5469800	0.
5.8020754	3.5980596	0.
5.7632980	3.6468202	0.
5.7215609	3.6930729	0.
5.6770260	3.7366382	0.
5.6298661	3.7773473	0.
5.5802639	3.8150423	0.
5.5284117	3.8495770	0.
5.4745107	3.8808176	0.
5.4187698	3.9086429	0.
5.3614052	3.9329450	0.
5.3026392	3.9536297	0.
5.2426999	3.9706167	0.
5.1818194	3.9838403	0.
5.1202340	3.9932491	0.
5.0581824	3.9988067	0.

```
                     4.9959052          4.0004916          0.
                     4.9336439          3.9982970          0.
                     4.8716398          3.9922317          0.
                     4.8101335          3.9823191          0.
                     4.7493634          3.9685977          0.
                     4.6895650          3.9511205          0.
                     4.6309704          3.9299555          0.
                     4.5738046          3.9051847          0.
                     4.5182953          3.8769041          0.
                     4.4646519          3.8452233          0.
                     4.4130841          3.8102653          0.
                     4.3641101          3.7717798          0.
 DS IS/    L2                                                                                  26          44
                          X                  Y                  Z
                     2.2137003          2.0000000          0.
 DS IS/                                                                                        27          46
                          X                  Y                  Z
                     0.                 0.                 0.
LEADER/  24.0000                                                                               28          48
   END                                                                                         29          50
 FINI                                                                                          30          52

                                 ****** END OF SECTION III ******
```

```
                                      ***GENERAL ELECTRIC POSTPROCESSOR GECENT-3 ***                          PAGE   1
                            MACHINE   23,
                   EXAMPLE OF GEOUT 2 PRINTOUT
           N      G         X            Y            Z            A            I           J           K         F       M
      EXAMPLE OF GEOUT 2 PRINTOUT
INC       001     17
     FROM              +0001;       +0001.       +0001;
ABS                    +0001;       +0001.       +0001;
OPR       002          +0000;       +0000.       +0000,

     NO SPINDLE STATEMENT HAS BEEN GIVEN PRIOR TO THE FIRST MOTION REQUEST
INC       003     01     +03;         +04.         +05,                                                         085;
ABS       014          +0004;       +0005.       +0006;      +000;                                          060;     IPM
OPR       003     01   +0003;       +0004.       +0005,       +00;                                              085;

INC       004     04                                                                                                      00
ABS       016          +0004;       +0005.       +0006;      +000;                                          060;     IPM
OPR       004     04   +0003;       +0004.       +0005;       +00;                                                        00

INC       005     01     +02;         +02.         +02;                                                         173;
ABS       018          +0006;       +0007.       +0008;      +000;                                          060;     IPM
OPR       005     01   +0005;       +0006.       +0007;       +00.                                              173;

INC       006     01                               +04;                                                         150;
ABS       020          +0006;       +0007.       +0012,      +000;                                          060;     IPM
OPR       006     01   +0005;       +0006.       +0011,       +00.                                              150;

INC       007     01                               =04;
ABS       022          +0006;       +0007.       +0008,      +000;                                          060;     IPM
OPR       007     01   +0005;       +0006.       +0007,       +00;

INC       008     01                  =07.         =08;                                                         056;
ABS       024          +0006;       +0000.       +0000;      +000;                                          060;     IPM
OPR       008     01   +0005;       -0001.       =0001,       +00,                                              056;

INC       009     01     -06;                                                                                   100;
ABS       026          +0000;       +0000.       +0000;      +000;                                          060;     IPM
OPR       009     01   -0001;       =0001.       =0001;       +00.                                              100;

INC       010     10     +20;         +20.         +20,                                                         173;
ABS       030          +0020;       +0020.       +0020,      +000;                                          060;     IPM
OPR       010     10   +0019;       +0019.       +0019,       +00.                                              173;

INC       011     10     -20;         =20.         =20;
ABS       032          +0000;       +0000.       +0000,      +000;                                          060;     IPM
OPR       011     10   -0001;       =0001.       =0001,       +00.

INC       012     04                                                                                                      00
ABS       034          +0000;       +0000.       +0000,      +000;                                          060;     IPM
OPR       012     04   -0001;       =0001.       =0001;       +00.                                                        00
```

```
                                  ***GENERAL ELECTRIC POSTPROCESSOR RECENT-3 ***
                          MACHINE    23.
                   EXAMPLE OF GEOUT 2 PRINTOUT
             N      G        X         Y          Z          A          I          J          K         F      M
    INC     013    01                 +02.                                                                 300.
    ABS     037          +0000.      +0002.     +0000.      +000.                                     060.      IPM
    OPR     013    01    -0001.      +0001.     -0001.       +00.                                         300.

    INC     014    01      +05.                                                                            120.
    ABS     039          +0005.      +0002.     +0000.      +000.                                     060.      IPM
    OPR     014    01    +0004.      +0001.     -0001.       +00.                                         120.

    INC     015    03      +01.        +01.                                       01.                      500.
    ABS     042          +0006.      +0003.     +0000.      +000.                                     050.      IPM
    OPR     015    03    +0005.      +0002.     -0001.       +00.        00.        01.                    500.

    INC     016    03      -01.        +01.                              01.
    ABS     042          +0005.      +0004.     +0000.      +000.                                     050.      IPM
    OPR     016    03    +0004.      +0003.     -0001.       +00.        01.        00.

    INC     017    03      -00.6359    -00.2282                                   01.
    ABS     042          +0004.3441  +0003.7718 +0000.      +000.                                     050.      IPM
    OPR     017    03    +0003.3441  +0002.7718 -0001.       +00.        00.        01.

    INC     018    01      -02.1504    -01.7718                                                            215.
    ABS     044          +0002.2137  +0002.     +0000.      +000.                                     060.      IPM
    OPR     018    01    +0001.2137  +0001.     -0001.       +00.                                          215.

    INC     019    01      -02.2137    -02.                                                                201.
    ABS     046          +0000.      +0000.     +0000.      +000.                                     060.      IPM
    OPR     019    01    -0001.      -0001.     -0001.       +00.                                          201.

       24.00    INCHES  OF LEADER HERE
        REQUESTED MISCELLANEOUS FUNCTION CODE IS NOT AVAILABLE ON THIS MACHINE
    INC     020    04
    ABS     050          +0000.      +0000.     +0000.      +000.                                     060.      IPM
    OPR     020    04    -0001.      -0001.     -0001.       +00.

      CUT TIME    2.04  MIN.     DWELL TIME  0.  MIN.
      TAPE FOOTAGE       3
```

3.5.6 GEOUT3 (MULTIPLE PRINTOUT-MULTIHEAD)

GEOUT3 is called when option 164 is set to 3. It is used only for multihead machines since it is the only print sequence capable of handling the many registers common to this type of NC machine. GEOUT3 produces in accordance with the setting of option 17 all or any one of the Incremental, Absolute, or Operator Printouts. Each output is complete, i.e., all of the Incremental is printed, then all of the Absolute, and then the Operator. Processing time is greatest when all three outputs are requested.

Across the top of the page is printed the machine register's title which is derived from the NC machine's related REGFOR and REGSTR tables. Each page is identified with the identifying title of INCREMENTAL, ABSOLUTE, or OPERATOR.

The Incremental data is the true reflection of what appears on the output tape (without letter addresses but with decimal point for clarity).

The Absolute data represents the summated motion values, and gives the feedrate in IPM and spindle speed in RPM. Rotary values are given in degrees.

The Operator data represents the motion data in terms of machine orientation, i.e., the summated motion values are modified by the given ORIGIN values.

With multihead machines the register title printed at the top of each page consists of at least two lines of title, each line giving the registers for each head. The Head 1 title is always given first followed by the Head 2 title, and so on.

Preceding each line of output on all three printout types is the title HEAD1 or HEAD2 to identify the processing head. A sample case might be:

```
          N      G      X      Z      F      S      T      M
                 H      U      W      E

HEAD 1   023     01     2.4    3.6    280     21     01     08

HEAD 2           10    12.6    0.2    500     21
```

The accumulated cut and dwell times are given at the bottom of each page, and on the final page the total tape footage is also given.

3.5.6 GEOUT3 (MULTIPLE PRINTOUT - MULTIHEAD) (cont'd)

See Diagram 3.5.6 for a sample printout of GEOUT3 with and without multihead processing.

Option 17 is used to select any one or combination of the three print types. The standard setting of 111 produces all three printouts. See Part Programmer's Manual option 17 for a full description of its use.

The processing sequence of GEOUT3 is considerably more complex than the other GEOUT's, however, the program flow is analagous. Basic initialization begins in GEOUT when subroutine GEPRE calls subroutine CALCP3 where the column vectors are set up per the description of Section 3.5.2. In addition to these column vectors, the vectors NIPA and NPTA for the Absolute data are also determined and set up. These vectors are used for the Operator data also.

The methods and techniques used in subroutine CALCP3 are the same as delineated in Section 3.5.4.1 for subroutine CALCP1 except that the additional vectors NIPA (initial print position for Absolute Printout) and NPTA (total number of digits per register) are developed by modifying the NIP and NPT vector elements for registers XYZABIJKF and S so as to expand the printout format of these registers. As explained earlier, it is essential to make these formats broader because the absolute algebraic references will normally be numeric ally larger than permitted by the incremental data format.

With multihead processing designated, the column vectors mentioned above are used for head 1, and an analogous set of vectors are used for head 2. These are the column vectors NIP2, NFP2, NPR2, NPT2, NIPA2, and NPTA2.

All of these vectors are determined by subroutine CALCP3 through a double call from subroutine GEPRE. The calling sequence of subroutine CALCP3 includes the input tables REGSTR and REGFOR for head 1, and GMWORD and GMFORM for head 2. See Section 2.4.5 on multihead processing for information on how these tables are derived. The output items in the calling sequence of subroutine CALCP3 are the column vectors which are set up in accordance with the input tables.

Note also that the array BCDRG2 is constructed for the head 2 page title. See Section 2.4.5.

3.5.6 GEOUT3 (MULTIPLE PRINTOUT - MULTIHEAD) (cont'd)

Section 3.5.3 details the overall general flow that occurs in GEOUT as related to the processing and treatment of each command block, and this description holds well for GEOUT3.

The major problem when processing a multihead sequence is knowing for which head the command block should be made output, and this is especially true for a merged block, i.e., a combined motion of the heads.

As explained in Section 2.4.5 for multihead processing, head 1 data are stored on TAPES2 and head 2 data are stored on TAPES3. Link GEMULT then overlays all of the postprocessor except for overlay GEMON. GEMULT now becomes the main processing element in core where the command data of each head is read from its respective tape and either merged or made separate output. In either event, the command data is stored not in DBFSEG, but in GMHBUF, which is dimensioned at 40, the first 20 cells for head 1 and the next 20 for head 2. GEMULT passes on to GEOUT3 the setup command block buffer GMHBUF for output. The buffer can be for three possible conditions:

(1) For head 1 only - GMHBUF(21) to (40) is DMBITS;

(2) For head 2 only - GMHBUF(1) to (20) is DMBITS;

(3) Merged for heads 1 and 2 - GMHBUF (1) to (40) is not DMBITS.

Since GEOUT3 is written in terms of DBFSEG, the data in GMHBUF is first restored into DBFSEG before calling subroutine GEPRO3. Thus, subroutine GEOUT first scans GMHBUF to determine which of the above three conditions exists, then it sets the flags HEAD, HEAD1, HEAD2, and NOW accordingly, stores that portion of GMHBUF into DBFSEG, and calls subroutine GEPRO3 to output the block. A non-zero setting of flags HEAD1 and HEAD2 indicates current selection of that head; a non-zero setting of flag NOW directs GEOUT to conclude the command block with an EOB. Whichever head has been selected for output, the COMMON flag HEAD is set to the value 1 or 2 to designate which head is currently operative. The flags are set according to these conditions:

3.5.6 GEOUT3 (MULTIPLE PRINTOUT - MULTIHEAD (cont'd)

(1) Head 1 data only - HEAD1 = 1, HEAD2 = 0, NOW = 1; DBFSEG is set up with GMHBUF(1) through (20).

(2) Head 2 data only - HEAD1 = 0, HEAD2 = 1, NOW = 1; DBFSEG is set up with GMHBUF(21) through (40).

(3) If GMHBUF is for a merged block, then HEAD1 = 1 and HEAD 2 = 1. DBFSEG is first set up with GMHBUF(1) through (30), NOW = 0, and DBFSEG is made output. Afterwards, DBFSEG is again set up with GMHBUF(21) through (40), and NOW = 1, and DBFSEG is made output.

When each head's data is made output, subroutine GEOUT sets up the proper REGFOR and REGSTR tables for that head. The buffer GMWORD for head 2 corresponds to REGSTR for head 1, and, likewise, GMFORM corresponds to REGFOR.

Multihead output processing of the command block DBFSEG follows the normal flow through GEOUT3 as described above in Section 3.5.6 for non-multihead machines except that at certain key junctions a test is made on the HEADGB flag to branch accordingly for the sequence for that head. This usually is done, for example, in setting up the Absolute data vectors DABVAL for head 1, or ABSVL2 for head 2; and similarly, for the Operator data vectors OPRVAL and OPRVL2.

The HEADGB flag is also tested when suppressing redundancies, i.e., redundant G, F, and S codes are suppressed by head only.

```
  A3863   DATE- 10/13/69                          G.E. APT III 10/01/67            TIME OF DAY- 17HR,     9MIN,

                 1  PARTNO EXAMPLE OF GEOUT 3 PRINTOUT
                 2  MACHIN/GECENT,51,OPTAB,164,3
 2.1             3          $$ CHANGE OPTIONS
                 3          CLPRNT
                 4  COMBIN/2
                 5  SELECT/HED,1
                 6  OP/1
 11              7          FROM/0,0,0
 5               8          TOLER/0.0005
 6               9          LEADER/24
 7              10          FEDRAT/10,IPM
 8              11          TMARK/1
 9              12  TURRET/2,3,2,2,FRONT  $$ NO M-CODE WILL BE AVAILABLE FOR TURRET
 10             13  INSERT N000G06H1M2**
 12             14          COOLNT/ON
 13             15          SPINDL/100,RPM
 14             16          GO TO/3,4,0
 17             17          GO TO/20,4,0
 23             18          SPINDL/170,RPM,CCLW,RANGE,2
 24             19          GO TO/17,6,0
 25             20          TURRET/4,5,6,8,FRONT      $$ TURRET CHANGE
 26             21          FEDRAT/40
                22          GO TO/10,10,0
                23          SPINDL/OFF
                24  SELECT/HED,2
                25  OP/2
                26  FROM/0,0,0
                27  SPINDL/60,RPM,RANGE,1
                28          TURRET/2,4,3,3
 30             29          GO TO/10,8,0                $$ GEARS SHIFT UP
 31             30          GO TO/12,10,0                $$ GEARS SHIFT DOWN
 32             31          RAPID
 33             32          GO TO/12,12,0
 34             33          RAPID
 35             34          GO TO/18,14,0                $$ GEARS STAY IN RAPID
 38             35          GO TO/20,20,0
 41             36          GO TO/22,24,0
 43             37          GO TO/24,24,0
 52             38          GO TO/9,9,0
 60             39          GO TO/10,11,0
 66             40          FEDRAT/0,2,IPR               $$ IPR NOT MANDATORY
 68             41          GO TO/10,20,0
 69             42          SPINDL/OFF
 77             43          GO TO/5,5,0
                44          OP/3
 83             45          REWIND/1
 84             46          FINI
 A3863   DATE- 10/13/69       CUTTER LOCATION LISTING(SECTION III ),       TIME OF DAY- 17 HR,    9MIN.

*** COMMANDS ************************************************************************ CARD NO    INT SEQ NO  CLREC NO

 EXAMPLE OF GEOUT 3 PRINTOUT                                                                        1           2
MACHIN/GECENT,   51,0000,  OPTAB,  164,0000,    3,0000                                              2           4
COMBIN/   2,0000                                                                                    4           6
SELECT/   HED,    1,0000                                                                            5           8
    OP/   1,0000                                                                                    6          10
 FROM /                                                                            11               7          12
                         X                    Y                     Z
                      0,                    0,                   0,
OUTTOL/   0,0005,    0,0005,     0,0005                                             5               8          14
 INTOL/   0,     ,   0,     ,    0,                                                 5               8          15
LEADER/  24,0000                                                                    6               9          17
FEDRAT/  10,0000,    IPM                                                            7              10          19
 TMARK/   1,0000                                                                    8              11          21
TURRET/   2,0000,    3,0000,     2,0000,     2,0000,  FRONT                          9              12          23
 N000G06H1M28$                                                                     10              13          25
COOLNT/    ON                                                                      12              14          27
SPINDL/ 100,0000,    RPM                                                           13              15          29
 DS IS/                                                                            14              16          31
                         X                    Y                     Z
                      3,0000000         4,0000000              0,
 DS IS/                                                                            17              17          33
                         X                    Y                     Z
                     20,0000000         4,0000000              0,
SPINDL/ 170,0000,    RPM,    CCLW,   RANGE,     2,0000                             23              18          35
 DS IS/                                                                            24              19          37
                         X                    Y                     Z
                     17,0000000         6,0000000              0,
TURRET/   4,0000,    5,0000,     8,0000,     8,0000,  FRONT                         25              20          39
FEDRAT/  40,0000                                                                   26              21          41
 DS IS/                                                                                            22          43
                         X                    Y                     Z
                     10,0000000        10,0000000              0,
SPINDL/   OFF                                                                                      23          45
SELECT/   HED,    2,0000                                                                           24          47
    OP/   2,0000                                                                                   25          49
 FROM /                                                                                            26          51
                         X                    Y                     Z
                      0,                    0,                   0,
SPINDL/  60,0000,    RPM,  RANGE,     1,0000                                                       27          53
TURRET/   2,0000,    4,0000,     3,0000,     3,0000                                                28          55
 DS IS/                                                                            30              29          57
                         X                    Y                     Z
                     10,0000000         8,0000000              0,
 DS IS/                                                                            31              30          59
                         X                    Y                     Z
                     12,0000000        10,0000000              0,
RAPID
DS IS/                                                                        32          31       61
                                                                              33          32       63
                         X                 Y                 Z
                    12.0000000        12.0000000        0.
RAPID
DS IS/                                                                        34          33       65
                                                                              35          34       67
                         X                 Y                 Z
                    14.0000000        14.0000000        0.
DS IS/                                                                        38          35       69
                         X                 Y                 Z
                    20.0000000        20.0000000        0.
DS IS/                                                                        41          36       71

                         X                 Y                 Z
                    22.0000000        24.0000000        0.
DS IS/                                                                        43          37       73
                         X                 Y                 Z
                    24.0000000        24.0000000        0.
DS IS/                                                                        52          38       75
                         X                 Y                 Z
                     9.0000000         5.0000000        0.
DS IS/                                                                        60          39       77
                         X                 Y                 Z
                    10.0000000        11.0000000        0.
FEDRAT/    0.2000,     IPR                                                    66          40       79
DS IS/                                                                        68          41       81
                         X                 Y                 Z
                    10.0000000        20.0000000        0.
SPINDL/    OFF                                                                69          42       83
DS IS/                                                                        77          43       85
                         X                 Y                 Z
                     5.0000000         6.0000000        0.
    OP/    3.0000                                                                         44       87
REWIND/    1.0000                                                             83          45       89
  FINI                                                                        84          46       91

                              ****** END OF SECTION III ******
```

```
                                                ***GENERAL ELECTRIC POSTPROCESSOR GECENT-3 ***
                                                                *** INCREMENTAL ***
                                              MACHINE   51, SAMPLE MULTIHEAD MACHINE
                         EXAMPLE OF GEOUT 3 PRINTOUT
        N        G           X               Z               I             K            F         S        T        M
        N        G           U               W               H             J            E         S        T        M
 EXAMPLE OF GEOUT 3 PRINTOUT

HEAD          1
FROM                   +00000,        +00000,

       24,00      INCHES  OF LEADER HERE
  TMARK
HEAD          1
      023       04                                                                                          23

  N000G06H1M281

HEAD          1
      029       04                                                                                                   08

LOWEST RANGE THAT SPINDLE SPEED FALLS IN IS ASSUMED

HEAD          1
      029       04                                                                                 09                03

HEAD          1
      031       04       +000.05                                                                                     41

HEAD          1
      031       01       +003,           +004,                                               0020,

HEAD          1
      033       10       +017,                                                               0059,

RANGE REQUESTED IS NOT AVAILABLE, USE HIGHEST

WARNING -- SPINDLE DIRECTION HAS CHANGED

HEAD          1
      035       04                                                                                 14                04

HEAD          1
      037       01       -003,           +002,                                               0028,

HEAD          1
      039       04                                                                                          45

HEAD          1
      039       01       -006,           -006,                                               0012,

HEAD          1
      043       01       -007.           +004,                                               0056,
                                      ***GENERAL ELECTRIC POSTPROCESSOR GECENT-3 ***
                                                   *** INCREMENTAL ***
                                    MACHINE   51, SAMPLE MULTIHEAD MACHINE
                    EXAMPLE OF GEOUT 3 PRINTOUT
      N      G         X            Z            I           K          F       S      T      M
      N      G         U            W            H           J          E       S      T      M
HEAD       1
     045     04                                                                                05

HEAD       2
FROM              +00000,      +00000,

HEAD       2
     053     04                                                                 05            04

HEAD       2
     055     04                                                                        24

HEAD       2
     057     04     +000.05                                                                   45

HEAD       2
     057     10     +010.       +008,                                0312,

HEAD       2
     059     01     +002.       +002,                                0141,

HEAD       2
     063     01                 +002,                    004,        0380,

HEAD       2
     067     01     +006.       +002,                                0240,

HEAD       2
     069     01     +002.       +006,                                0063,

HEAD       2
     071     01     +002.       +004,                                0089,

HEAD       2
     073     01     +002.                                            0200,

HEAD       2
     075     10     -015,       -015,                                0189,

HEAD       2
     077     01     +001.       +002,                                0179,

HEAD       2
     081     01                 +009,                                0013,

HEAD       2
     083     04                                                                                05
```

```
                                                  ***GENERAL ELECTRIC POSTPROCESSOR GECENT-3 ***
                                                                  *** INCREMENTAL ***
                                               MACHINE    51, SAMPLE MULTIHEAD MACHINE
                              EXAMPLE OF GEOUT 3 PRINTOUT
      N        G          X             Z             I              K             F         S        T        M
      N        G          U             W             H              J             E         S        T        M
HEAD         2
     085      10       -005.         -014,                                        0081.

HEAD         2
     089      04                                                                                               30

NO END STATEMENT HAS BEEN GIVEN BEFORE THE FINI STATEMENT


     CUT TIME    6.10  MIN,      DWELL TIME  0,02MIN,
     TAPE FOOTAGE      5
```

```
                                        ***GENERAL ELECTRIC POSTPROCESSOR GECENT-3 ***
                                                       *** ABSOLUTE ***
                                      MACHINE   51, SAMPLE MULTIHEAD MACHINE
                    EXAMPLE OF GEOUT 3 PRINTOUT
       N      G         X            Z            I            K          F       S       T      M
       N      G         U            W            H            J          E       S       T      M
 EXAMPLE OF GEOUT 3 PRINTOUT                                             $

HEAD        1
FROM              +00000.       +00000,
    24 INCHES OF LEADER HERE
 TMARK

HEAD        1
     023     04   +00000,       +00000,                                                        23
 N000G06H1M28$

HEAD        1
     029     04   +00000,       +00000,                                                               08

HEAD        1
     029     04   +00000.       +00000,                                         100.0                 03

HEAD        1
     031     04   +00000,05                                                                           41

HEAD        1
     031     01   +00003,       +00004,                                 000010,

HEAD        1
     033     10   +00020,       +00004,                                 000010,

HEAD        1
     035     04   +00020,       +00004,                                         150.0                 04

HEAD        1
     037     01   +00017,       +00006,                                 000010,

HEAD        1
     039     04   +00017,       +00006,                                                        45

HEAD        1
     039     01   +00017,       +00006,                                 000010,

HEAD        1
     043     01   +00010.       +00010,                                 000040,

HEAD        1
     045     04   +00010,       +00010,                                                               05
```

```
                                        ***GENERAL ELECTRIC POSTPROCESSOR GECENT-3 ***
                                                       *** ABSOLUTE ***
                                      MACHINE   51, SAMPLE MULTIHEAD MACHINE
                     EXAMPLE OF GEOUT 3 PRINTOUT
       N      G         X             Z            I           K          F       S       T      M
       N      G         U             W            H           J          E       S       T      M
HEAD        2
FROM               +00000,      +00000,

HEAD        2
     053     04    +00000,      +00000,                                         060,0          04

HEAD        2
     055     04    +00000,      +00000,                                                  24

HEAD        2
     057     04    +00000,05                                                                   45

HEAD        2
     057     10    +00010,      +00008,                               000040,

HEAD        2
     059     01    +00012,      +00010,                               000040,

HEAD        2
     063     01    +00012,      +00012,       00000,      00004,      000152,

HEAD        2
     067     01    +00018,      +00014,                               000152,

HEAD        2
     069     01    +00020,      +00020,                               000040,

HEAD        2
     071     01    +00022,      +00024,                               000040,

HEAD        2
     073     01    +00024,      +00024,                               000040,

HEAD        2
     075     10    +00009,      +00009,                               000040,

HEAD        2
     077     01    +00010,      +00011,                               000040,

HEAD        2
     081     01    +00010,      +00020,                               000012,

HEAD        2
     083     04    +00010,      +00020,                                                        05
```

```
                             ***GENERAL ELECTRIC POSTPROCESSOR GECENT-3 ***
                                         *** ABSOLUTE ***
                            MACHINE   51, SAMPLE MULTIHEAD MACHINE
               EXAMPLE OF GEOUT 3 PRINTOUT
     N    G       X        Z         I        K        F     S    T    M
     N    G       U        W         H        J        E     S    T    M
HEAD     2
    085   10  +00005,  +00006,                    000012,

HEAD     2
    089   04  +00005,  +00006,                                          30

   CUT TIME    6.10  MIN,   DWELL TIME  0.02MIN,
   TAPE FOOTAGE      5
```

```
                                        ***GENERAL ELECTRIC POSTPROCESSOR GECENT-3 ***
                                                       *** OPERATOR ***
                                      MACHINE   51, SAMPLE MULTIHEAD MACHINE
                    EXAMPLE OF GEOUT 3 PRINTOUT
       N      G         X            Z            I           K         F       S       T      M
       N      G         U            W            H           J         E       S       T      M
 EXAMPLE OF GEOUT 3 PRINTOUT                                           $

HEAD        1
FROM               +00000,      +00000,
    24 INCHES OF LEADER HERE
 TMARK

HEAD        1
     023     04                                                                        23
 N000G06H1M28$

HEAD        1
     029     04                                                                               08

HEAD        1
     029     04                                                                 09            03

HEAD        1
     031     04    +00000,05                                                                  41

HEAD        1
     031     01    +00003,      +00004,                              000020,

HEAD        1
     033     10    +00020,      +00004,                              000059,

HEAD        1
     035     04                                                                 14            04

HEAD        1
     037     01    +00017,      +00006,                              000028,

HEAD        1
     039     04                                                                        45

HEAD        1
     039     01    +00011,      -00000,                              000012,

HEAD        1
     043     01    +00004,      +00004,                              000050,

HEAD        1
     045     04                                                                               05
```

```
                                              ***GENERAL ELECTRIC POSTPROCESSOR GECENT-3 ***
                                                             *** OPERATOR ***
                                          MACHINE   51, SAMPLE MULTIHEAD MACHINE
                        EXAMPLE OF GEOUT 3 PRINTOUT
      N         G            X               Z               I               K              F          S        T        M
      N         G            U               W               W               J              E          S        T        M
HEAD          2
FROM                    +00000.         +00000,

HEAD          2
     053         04                                                                                     05                04

HEAD          2
     055         04                                                                                              24

HEAD          2
     057         04     +00000,05                                                                                         45

HEAD          2
     057         10     +00010,         +00008,                                        000312,

HEAD          2
     059         01     +00012,         +00010,                                        000141,

HEAD          2
     063         01     +00012,         +00012,         00000,          00004,         000380,

HEAD          2
     067         01     +00018,         +00014,                                        000240,

HEAD          2
     069         01     +00020,         +00020,                                        000063,

HEAD          2
     071         01     +00022,         +00024,                                        000089,

HEAD          2
     073         01     +00024,         +00024,                                        000200,

HEAD          2
     075         10     +00009,         +00009,                                        000189,

HEAD          2
     077         01     +00010,         +00011,                                        000179,

HEAD          2
     081         01     +00010.         +00020,                                        000013,

HEAD          2
     083         04                                                                                                       05
```

```
                                         ***GENERAL ELECTRIC POSTPROCESSOR GECENT-3 ***
                                                        *** OPERATOR ***
                                       MACHINE   51, SAMPLE MULTIHEAD MACHINE
                     EXAMPLE OF GEOUT 3 PRINTOUT
       N      G          X             Z            I            K          F        S      T       M
       N      G          U             W            H            J          E        S      T       M
HEAD        2
     085     10    +00005,       +00006,                                000081,

HEAD        2
     089     04                                                                                     30

    CUT TIME     6.10  MIN,      DWELL TIME  0.02MIN,
    TAPE FOOTAGE       5
```

3.5.7 GEOUT4 (MULTIPLE PRINTOUT - NON-MULTIHEAD)

GEOUT4 is called when option 164 is set to 4. The multiple printout of GEOUT4 was formerly the standard printout of the GECENT II postprocessor. Like GEOUT3, it yields the separate production of all or any one of the Incremental, Absolute, or Operator's Printout - as selected by the setting of option 17. GEOUT4 may be used with all types of NC machines except multihead machines which must use GEOUT3.

Processing for GEOUT4 (non-multihead machines) deviates only slightly from the general description found in Section 3.5.3. These differences are as follows:

(1) The register title is printed by subroutine TITLE3.

(2) Each BCD converted command block BCDIMG is printed by subroutine GEPRN3.

(3) The array DABVAL carries the Absolute data while the array ORPVAL carries the Operator data.

(4) The BCDIMG for the Absolute and Operator Printout is set up using the column vectors NIPA and NPTA.

(5) The setting up and printing of each command block BCDIMG for each of three printout types is done in three independent looping areas.

(6) After each command block of Incremental Printout is processed and made output, the Absolute and Operator data, if called by option 17, are processed for output and written on a tape by using subroutine GMWRIT. The Absolute output is written on TAPES1, while the Operator output is written on TAPES4.

(7) Subroutine PAGE prints the page number.

(8) Subroutine TIMES prints the total cut and dwell times and tape footage at the end of each printout.

3.5. GEOUT4 (MULTIPLE PRINTOUT - NON-MULTIHEAD) (cont'd)

(9) On a FINI subroutine ABSOPR is called to output the Absolute and Operator Printout. This is accomplished by rewinding TAPES1 and TAPES4, and then opening them for reading. TAPES1 is first processed for the Absolute Printout by reading a block of data using subroutine GMREAD, and printing it out by calling subroutine GEPRN3. Subroutine TITLE3 is called for each new page. TAPES4 is similarly processed for the Operator Printout.

See Diagram 3.5.7 for sample printout of GEOUT4.

```
   P3882   DATE= 10/06/69                      G.E. APT III 10/01/67              TIME OF DAY= 23HR.   42MIN.

                     1   PARTNO EXAMPLE OF GEOUT 4 PRINTOUT
                     2   MACHIN/GECENT,22,OPTAB,164,4
 2.1                 3           $$ CHANGE OPTIONS
                     3           CLPRNT
 5                   4           TOLER/0.0005
 6                   5           LEADER/24
 7                   6           FEDRAT/10,IPM
 8                   7           TMARK/1
 9                   8   TURRET/2,3,2,2,FRONT  $$ NO M-CODE WILL BE AVAILABLE FOR TURRET
 10                  9   INSERT N000G06H1M28$
 11                 10           FROM/0,0,0
 12                 11           COOLNT/ON
 13                 12           SPINDL/100,RPM
 14                 13           GO TO/3,4,0
 15                 14   PPRINT  EXAMPLE OF A SEGMENTATION DUE TO A
 16                 15   PPRINT  PATH WHICH IS GREATER THAN THE MAXIMUM DEPARTURE
 17                 16           GO TO/20,4,0
 18                 17           PREFUN/88
 19                 18           AIR/ON
 20                 19           COOLNT/OFF
 22                 20           OPSTOP
 23                 21           SPINDL/170,RPM,CCLW,RANGE,2
 24                 22           GO TO/17,6,0
 25                 23           TURRET/4,5,8,8,FRONT      $$ TURRET CHANGE
 26                 24           FEDRAT/40
 27                 25           STOP
 28                 26   PPRINT  EXAMPLE OF RAPID TRAVERSE SHIFTING
 29                 27           RAPID
 30                 28           GO TO/10,8,0              $$ GEARS SHIFT UP
 31                 29           GO TO/12,10,0              $$ GEARS SHIFT DOWN
 32                 30           RAPID
 33                 31           GO TO/12,12,0
 34                 32           RAPID
 35                 33           GO TO/18,14,0              $$ GEARS STAY IN RAPID
 37                 34           SPINDL/8000$$ SPINDLE SPEED TOO LARGE
 38                 35           GO TO/20,20,0
 41                 36           GO TO/22,24,0
 43                 37           GO TO/24,24,0
 46                 38           AUXFUN/99
 47                 39           AUXFUN/3
 52                 40           GO TO/9,9,0
 53                 41   PPRINT EXAMPLE OF THREADING
 55                 42           PITCH/10
 56                 43          COUPLE/ON   $$ COUPLE ENCODER
 57                 44           SPINDL/85,RANGE,1
 58                 45           DELAY/10
 59                 46          THREAD/TURN
 60                 47           GO TO/10,11,0
 61                 48           DELAY/2,REV
 62                 49          COUPLE/OFF
```

```
65          50  PPRINT  EXAMPLE OF AN SFM SEQUENCE
66          51          FEDRAT/0.2,IPR          $$ IPR NOT MANDATORY
67          52          SPINDL/100,SFM,RADIUS,XCOORD,MAXRPM,90,MAXIPM,15
68          53          GO TO/10,20,0
69          54          SPINDL/OFF
77          55          GO TO/5,6,0
83          56          REWIND/1
84          57          FINI
```

```
 P3882   DATE= 10/06/69      CUTTER LOCATION LISTING(SECTION III ).       TIME OF DAY= 23HR.   42MIN.

*** COMMANDS ********************************************************* CARD NO    INT SEQNO  CLREC NO

 EXAMPLE OF GEOUT 4 PRINTOUT                                                        1         2
MACHIN/GECENT,   22.0000,  OPTAB,  164.0000,    4.0000                              2         4
OUTTOL/   0.0005,     0.0005,     0.0005                         5                  4         6
 INTOL/   0.    ,     0.    ,     0.                             5                  4         7
LEADER/  24.0000                                                 6                  5         9
FEDRAT/  10.0000,     IPM                                        7                  6        11
 TMARK/   1.0000                                                 8                  7        13
TURRET/   2.0000,     3.0000,     2.0000,     2.0000,  FRONT     9                  8        15
 N000G06H1M28$                                                  10                  9        17
 FROM /                                                         11                 10        19
                     X            Y            Z
                  0.           0.           0.
COOLNT/    ON                                                   12                 11        21
SPINDL/ 100.0000,     RPM                                       13                 12        23
 DS IS/                                                         14                 13        25
                     X            Y            Z
                  3.0000000    4.0000000    0.
  EXAMPLE OF A SEGMENTATION DUE TO A                            15                 14        27
  PATH WHICH IS GREATER THAN THE MAXIMUM DEPARTURE              16                 15        29
 DS IS/                                                         17                 16        31
                     X            Y            Z
                 20.0000000    4.0000000    0.
PREFUN/  88.0000                                                18                 17        33
   AIR/    ON                                                   19                 18        35
COOLNT/   OFF                                                   20                 19        37
OPSTOP                                                          22                 20        39
SPINDL/ 170.0000,     RPM,   CCLW,  RANGE,    2.0000            23                 21        41
 DS IS/                                                         24                 22        43
                     X            Y            Z
                 17.0000000    6.0000000    0.
TURRET/   4.0000,     5.0000,     8.0000,     8.0000,  FRONT    25                 23        45
FEDRAT/  40.0000                                                26                 24        47
  STOP                                                          27                 25        49
  EXAMPLE OF RAPID TRAVERSE SHIFTING                            28                 26        51
 RAPID                                                          29                 27        53
 DS IS/                                                         30                 28        55
                     X            Y            Z
                 10.0000000    8.0000000    0.
 DS IS/                                                         31                 29        57
                     X            Y            Z
                 12.0000000   10.0000000    0.
 RAPID                                                          32                 30        59
 DS IS/                                                         33                 31        61
                     X            Y            Z
                 12.0000000   12.0000000    0.
 RAPID                                                          34                 32        63
 DS IS/                                                         35                 33        65
                     X            Y            Z
                 18.0000000   14.0000000    0.
```

```
SPINDL/8000.0800                                                                                   37            34        67
 DS IS/                                                                                            38            35        69
                              X               Y                Z
                         20.0000000      20.0000000       0.
 DS IS/                                                                                            41            36        71
                              X               Y                Z
                         22.0000000      24.0000000       0.
 DS IS/                                                                                            43            37        73
                              X               Y                Z

                         24.0000000      24.0000000       0.
AUXFUN/  99.0000                                                                                   46            38        75
AUXFUN/   3.0000                                                                                   47            39        77
 DS IS/                                                                                            52            40        79
                              X               Y                Z
                          9.0000000       9.0000000       0.
 EXAMPLE OF THREADING                                                                              53            41        81
 PITCH/  10.0000                                                                                   55            42        83
COUPLE/    ON                                                                                      56            43        85
SPINDL/  85.0000,  RANGE,    1.0000                                                                57            44        87
 DELAY/  10.0000                                                                                   58            45        89
THREAD/  TURN                                                                                      59            46        91
 DS IS/                                                                                            60            47        93
                              X               Y                Z
                         10.0000000      11.0000000       0.
 DELAY/   2.0000,     REV                                                                          61            48        95
COUPLE/   OFF                                                                                      62            49        97
  EXAMPLE OF AN SFM SEQUENCE                                                                       65            50        99
FEDRAT/   0.2000,     IPR                                                                          66            51       101
SPINDL/ 100.0000,     SFM, RADIUS, YCOORD, MAXRPM,   90.0000, MAXIPM,$                             67            52       103
         15.0000
 DS IS/                                                                                            68            53       105
                              X               Y                Z
                         10.0000000      20.0000000       0.
SPINDL/   OFF                                                                                      69            54       107
 DS IS/                                                                                            77            55       109
                              X               Y                Z
                          5.0000000       6.0000000       0.
REWIND/   1.0000                                                                                   83            56       111
 FINI                                                                                              84            57       113

                                        ****** END OF SECTION III ******
```

```
                                             ***GENERAL ELECTRIC POSTPROCESSOR GECENT-3 ***
                                                          *** INCREMENTAL ***
                                           MACHINE   22.
                           EXAMPLE OF GEOUT 4 PRINTOUT
    N      G       X          Z          I          K          F        S     T      M        W
 EXAMPLE OF GEOUT 4 PRINTOUT
    24.00    INCHES  OF LEADER HERE
 TMARK
   015    04               +0.3                                               23
 N800G06H1M28$
FROM          +000.     +000.
   023    04                                                                         08
LOWEST RANGE THAT SPINDLE SPEED FALLS IN IS ASSUMED
   023    04                                                            10           03
   025    04               +0.05
   025    04                                                                         43
   025    04               +0.1                                                      41
   025    01     -4.       +3.                                 020.
  EXAMPLE OF A SEGMENTATION DUE TO A
  PATH WHICH IS GREATER THAN THE MAXIMUM DEPARTURE
   031    01               +8.5                                011.76
   031    01               +8.5
   033    88
   039    04                                                                         09
   039    04                                                                         01
WARNING -- SPINDLE DIRECTION HAS CHANGED
   041    04               +0.3                                         17           04
   043    01     -2.       -3.                                 027.74
   045    04               +0.3                                               45
   045    01     +6.       -6.                                 010.
   049    04                                                                         00
  EXAMPLE OF RAPID TRAVERSE SHIFTING
   055    04               +0.05
   055    04                                                                         42
   055    04               +0.2                                                      40
   055    01     -2.       -7.                                 200.
   057    04               +0.05
   057    04                                                                         43
   057    04               +0.1                                                      41
   057    01     -2.       +2.                                 070.71
   061    04               +0.05
   061    04                                                                         42
   061    04               +0.2                                                      40
   061    01     -2.                                           500.
   065    01     -2.       +6.                                 233.33
   067    04                                                            11           04
   069    04               +0.05
   069    04                                                                         43
   069    04               +0.1                                                      41
   069    01     -6.       +2.                                 031.62
   071    01     -4.       +2.                                 044.72
```

```
                                                        ***GENERAL ELECTRIC POSTPROCESSOR GECENT-3 ***
                                                                   *** INCREMENTAL ***
                                                   MACHINE   22.
                                     EXAMPLE OF GECUT 4 PRINTOUT
    N      G          X          Z          I          K          F        S      T      M        W
   073    01                   +2.                               100.
   075    04                                                                               99
   077    04                                                                               03
   079    01      +7.5         -7.5                              018.86
   079    01      +7.5         -7.5
 EXAMPLE OF THREADING
   085    04                   +0.2                                                        50
   089    04                   +1.                                          05
   093    33      -2.          +1.         2.          1.
   095    04                   +0.1412
   097    04                   +0.2                                                        51
  EXAMPLE OF AN SFM SEQUENCE
 SFM MODE IS ESTABLISHED
 LOWEST SPEED IN RANGE IS OUTPUT
   105    01      -9.                                            006.       27
   107    04                                                                               05
   109    01      +7.          -2.5                              007.26
   109    01      +7.          -2.5
   111    04                                                                               30

     CUT TIME     8.93  MIN.      DWELL TIME  0.57MIN.
     TAPE FOOTAGE       8
```

```
                                                  ***GENERAL ELECTRIC POSTPROCESSOR RECENT-3 ***
                                                                   *** ABSOLUTE ***
                                               MACHINE    22.
                             EXAMPLE OF GEOUT 4 PRINTOUT
    N      G         X          Z          I          K          F        S      T      M         W
 EXAMPLE OF GEOUT 4 PRINTOUT                                                            $
    24 INCHES OF LEADER HERE
 TMARK
   015    04 +000.3     +000.3                                                   23
 N000006H1M28$
FROM         +000.      +000.
   023    04 +000.      +000.                                                           08
   023    04 +000.      +000.                                         100.0             03
   025    04 +000.05    +000.05
   025    04 +000.      +000.                                                           43
   025    04 +000.1     +000.1                                                          41
   025    01 -004.      +003.                               00010.
  EXAMPLE OF A SEGMENTATION DUE TO A
  PATH WHICH IS GREATER THAN THE MAXIMUM DEPARTURE
   031    01 -004.      +011.5                              00010.
   031    01 -004.      +020.
   033    88 -004.      +020.
   039    04 -004.      +020.                                                           09
   039    04 -004.      +020.                                                           01
   041    04 +000.3     +000.3                                        170.0             04
   043    01 -006.      +017.                               00010.
   045    04 +000.3     +000.3                                                   45
   045    01 -006.      +017.
   049    04 -006.      +017.                                                           00
  EXAMPLE OF RAPID TRAVERSE SHIFTING
   055    04 +000.05    +000.05
   055    04 -006.      +017.                                                           42
   055    04 +000.2     +000.2                                                          40
   055    01 -008.      +010.                               00145.6
   057    04 +000.05    +000.05
   057    04 -008.      +010.                                                           43
   057    04 +000.1     +000.1                                                          41
   057    01 -010.      +012.                               00020.
   061    04 +000.05    +000.05
   061    04 -010.      +012.                                                           42
   061    04 +000.2     +000.2                                                          40
   061    01 -012.      +012.                               00100.
   065    01 -014.      +018.                               00147.57
   067    04 -014.      +018.                                         110.0             04
   069    04 +000.05    +000.05
   069    04 -014.      +018.                                                           43
   069    04 +000.1     +000.1                                                          41
   069    01 -020.      +020.                               00020.
   071    01 -024.      +022.                               00020.
   073    01 -024.      +024.                               00020.
```

```
                                                         ***GENERAL ELECTRIC POSTPROCESSOR RECENT-3 ***
                                                                          *** ABSOLUTE ***
                                                   MACHINE   22.
                                   EXAMPLE OF GEOUT 4 PRINTOUT
   N     G     X             Z           I          K          F        S     T     M        W
  075   04  -024.        +024.                                                      99
  077   04  -024.        +024.                                                      03
  079   01  -016.5       +016.5                           00020.
  079   01  -009.        +009.
EXAMPLE OF THREADING
  085   04  +000.2       +000.2                                                     50
  089   04  +001.        +001.                                     085.0
  093   33  -011.        +010.       002.       001.
  095   04  +000.1412   +000.1412
  097   04  +000.2       +000.2                                                     51
 EXAMPLE OF AN SFM SEQUENCE
  105   01  -020.        +010.                            00005.4  027.0
  107   04  -020.        +010.                                                      05
  109   01  -013.        +007.5                           00005.4
  109   01  -006.        +005.
  111   04  -006.        +005.                                                      30

   CUT TIME     8.93  MIN.      DWELL TIME  0.57MIN.
   TAPE FOOTAGE       8
```

```
                                                        ***GENERAL ELECTRIC POSTPROCESSOR RECENT-3 ***
                                                                       *** OPERATOR ***
                                                     MACHINE   22.
                                       EXAMPLE OF GEOUT 4 PRINTOUT
      N      G       X         Z          I         K          F         S      T      M          W
  EXAMPLE OF GEOUT 4 PRINTOUT                                                        $
     24 INCHES OF LEADER HERE
  TMARK
    015    04 +000.3    +000.3                                                 23
  N000006H1M28$
 FROM         +000.     +000.
    023    04                                                                        08
    023    04                                                        10              03
    025    04 +000.05   +000.05
    025    04                                                                        43
    025    04 +000.1    +000.1                                                       41
    025    01 -004.     +003.                             00020.
   EXAMPLE OF A SEGMENTATION DUE TO A
   PATH WHICH IS GREATER THAN THE MAXIMUM DEPARTURE
    031    01 -004.     +011.5                            00011.78
    031    01 -004.     +020.
    033    88 -004.     +020.
    039    04                                                                        09
    039    04                                                                        01
    041    04 +000.3    +000.3                                       17              04
    043    01 -008.     +017.                             00027.74
    045    04 +000.3    +000.3                                                45
    045    01 +000.     +011.                             00010.
    049    04                                                                        00
   EXAMPLE OF RAPID TRAVERSE SHIFTING
    055    04 +000.05   +000.05
    055    04                                                                        42
    055    04 +000.2    +000.2                                                       40
    055    01 -002.     +004.                             00200.
    057    04 +000.05   +000.05
    057    04                                                                        43
    057    04 +000.1    +000.1                                                       41
    057    01 -004.     +006.                             00070.71
    061    04 +000.05   +000.05
    061    04                                                                        42
    061    04 +000.2    +000.2                                                       40
    061    01 -006.     +006.                             00500.
    065    01 -008.     +012.                             00233.33
    067    04                                                        11              04
    069    04 +000.05   +000.05
    069    04                                                                        43
    069    04 +000.1    +000.1                                                       41
    069    01 -014.     +014.                             00031.62
    071    01 -018.     +016.                             00044.72
    073    01 -018.     +018.                             00100.
```

```
                                                  ***GENERAL ELECTRIC POSTPROCESSOR GECENT-3 ***
                                                                    *** OPERATOR ***
                                              MACHINE    22.
                              EXAMPLE OF GEOUT 4 PRINTOUT
   N     G        X          Z          I          K          F        S     T     M        W
  075   04                                                                         99
  077   04                                                                         03
  079   01 -010.5     +010.5                              00018.86
  079   01 -003.      +003.
EXAMPLE OF THREADING
  085   04 +000.2     +000.2                                                       50
  089   04 +001.      +001.                                                 85
  093   33 -005.      +004.      002.       001.
  095   04 +000.1412 +000.1412
  097   04 +000.2     +000.2                                                       51
 EXAMPLE OF AN SFM SEQUENCE
  105   01 -014.      +004.                               00006.       27
  107   04                                                                         05
  109   01 -007.      +001.5                              00007.28
  109   01 +000.      -001.
  111   04                                                                         30

   CUT TIME     8.93  MIN.      DWELL TIME  0.57MIN.
   TAPE FOOTAGE       8
```

4.0 SPECIAL SEQUENCES

In the sections which follow are given some further details on special items which are treated as separate entities within the postprocessor. Normally, at least one of these items will be used on any NC machine.

4.1 FEEDRATE

The Mark Century control considers feedrates from two aspects, one for contouring machines and the other for positioning machines.

For contouring machines the feedrate command is a computed code derived by one of three methods, each of which are fully explained below. The relations of multiaxis or circular interpolation to the feedrate command are also detailed.

For positioning machines the feedrate command is obtained from one of several feed types. These types vary considerably from one another, but each type is detailed according to its structure and use.

The feedrate of a machine tool is the travel velocity of the tool along the path of movement and is usually measured at the tool tip. Feedrate is normally measured in IPM, but occasionally it is measured in IPR. Whenever IPR is the mode, the feedrate is directly related to the spindle speed, and the relation between IPM and IPR is:

$$F_{IPM} = F_{IPR} * S,$$

where F_{IPR} is the feedrate in IPR, S is the spindle speed in RPM, and F_{IPM} is the feedrate in IPM.

The postprocessor works internally with feedrate in IPM only, i.e., all feedrates in IPR are converted to IPM. The postprocessor keeps feedrates in IPM primarily for SFM operation and for acceleration-deceleration (A/D) testing. Prior to output, the feedrates are converted to their feedrate command form, but only after the feedrates have been tested for a variety of conditions. For example, all feedrates are tested versus the minimum and maximum allowable feedrates in IPM (options 48 and 25), and feedrate command maximum and minimums (options 24 and 49). If the feedrate is a rapid traverse feedrate, other actions are taken as explained in Section 4.1.5.4.

4.1 FEEDRATE (cont'd)

The postprocessor can also leave the feedrate in IPM, i.e., not convert it to the command form, whenever it is needed, as for an auxiliary saddle. When the feedrate is kept in IPM form, all the below described tests do not apply. The only testing performed is to insure that the given feedrate lies within the minimum and maximum feedrate range.

4.1.1 CONTOURING FEEDRATE COMMANDS

The postprocessor can output the feedrate in one of three different command forms, viz., as a function of a calculated feedrate number, as an EIA 3 digit number, or as 1/T (inverse time). The specification of the desired form is given in option 10. Contouring feedrate commands are all determined in subroutine CONTUR.

The resultant feedrate command is always compared with the minimum and maximum feedrate command values; and when it transgresses a bound, it is set to that bound. The new feedrate in IPM is then redetermined. For example, assume the feedrate command (F_C) as a function of F_{IPM} becomes greater than the feedrate command maximum (FCOMAX):

$$F_C = f(F_{IPM}) > \text{FCOMAX}.$$

Then, $F_C = \text{FCOMAX}$, and

$$F_{IPM} = f^{-1}(F_C)$$

4.1.1.1 FEEDRATE NUMBER COMMAND

This type of conforming feedrate command is selected by setting option 10 equal to zero. For machines having up to three linear axes, the feedrate along straight line paths is converted to a feedrate command by the relation:

$$F_c = \frac{D * F_{IPM}}{S}$$

where D is a constant called the dimension multiplier and is a function of the preparatory function G code, F_{IPM} is the feedrate in IPM, and S is the path length as determined by

$$S = \sqrt{\Delta X^2 + \Delta Y^2 \ + \Delta Z^2}$$

where ΔX, ΔY, ΔZ are the linear machine coordinate departures along their respective axes.

The dimension multiplier, D, obtains its constant value in subroutine SELG where the preparatory function G code is selected for the linear departures; see Section 3.4.6.1. The value of D for a given departures is dependant also upon the units system, i.e., inch or metric.

Consider the example of a linear move as shown in Diagram 4.1.1.1A. The tool has a feedrate of 40 IPM.

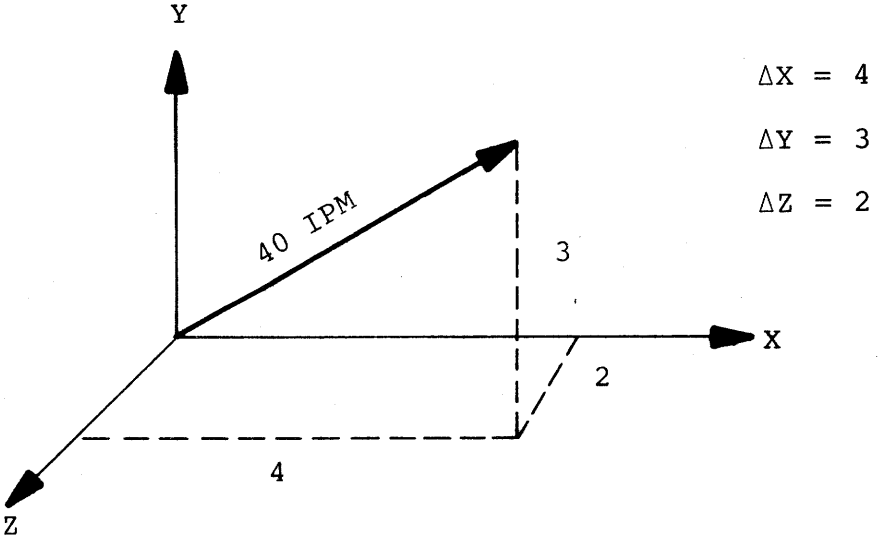


Diagram 4.1.1.1A

$$S = \sqrt{16 + 9 + 4} \ = \ 5.39\text{inches}$$

4.1.1.1 FEEDRATE NUMBER COMMAND (cont'd)

The G code would be G01 and the dimension multiplier (GDIMUL) is 10. Therefore,

$$F_C = \frac{10*40}{5.39} = 74.8$$

Since this is well within the feed command range ($1 \le F_C \le 500$), the computed value of F_C is accepted and made output.

It should be noted that in general the feedrate command range for this type is usually

$$1 \le F_C \le 500;$$

however, this is not always true, so the specifications of each machine must be carefully checked for this item.

Dimensionally, it can be seen that this type feedrate command is a frequency since its units are the reciprocal of time, T^{-1}.

$$F_C = \frac{F_{IPM}}{S} = \frac{|S/T|}{|S|} = \frac{1}{T}.$$

For circular interpolation moves, the formula is

$$F = \frac{D * F_{IPM}}{R},$$

where R is the circle radius and is always less than the maximum departure. The dimension multiplier, D , is a function of the radius length and is assigned its constant value in subroutine SELGCR; see Section 3.4.6.2.

A point to be noted here is that the length R is not usually the circle radius of the part, but rather is the distance from the part circle center to the tool control point. In Diagram 4.1.1.1B the true radius of the part is P; but since the postprocessor computes the radius from the given CL data, and since the CL tape passes on the cut data from the tool control point, the postprocessor actually uses

$$R = P + r,$$

where r is the radius of the tool.

4.1.1.1 FEEDRATE NUMBER COMMAND (cont'd)

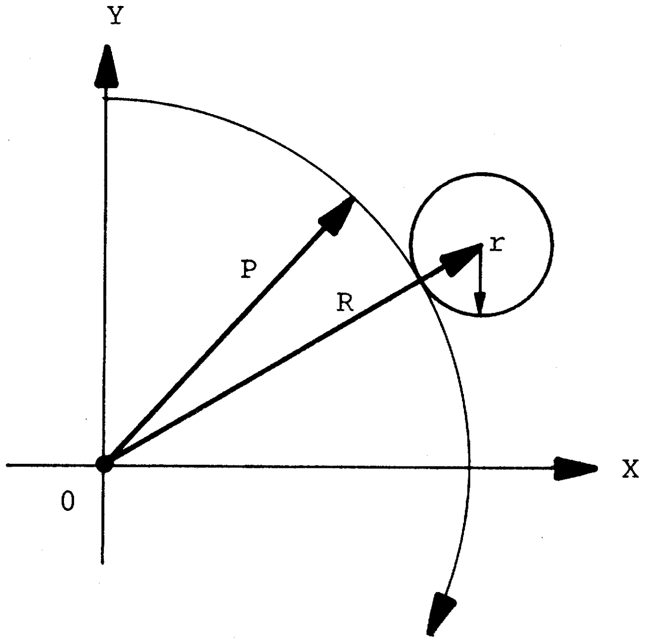


Diagram 4.1.1.1B

This causes a lower feedrate to be used than need be, but for large circles (P>>r)this feedrate variation is negligible. See the Part Programmers Manual for the use of the SELECT/RADIUS statement for cases when r≥P.

The radius R used in the feedrate command formula is computed from the arc center offsets, as:

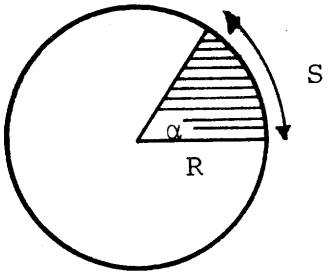
$$R = \sqrt{I^2 + J^2 + K^2}$$

A rotary table feedrate for an incremental system requires a preparatory function G code which specifies the dimension of the increment; see Section 3.4.6.3. As with a linear move, the rotary move feedrate is determined from the relation

$$F_C = \frac{D * F_{IPM}}{S}$$

4.1.1.1 FEEDRATE NUMBER COMMAND (cont'd)

where D is the dimension multiplier selected in subroutine SELGRO, and S is the effective tool path length in inches. The length S is a function of the table radius R since $S = \alpha R$, where α is the incremental rotary move in radians. The part radius R will usually vary with the part program; hence, R must be given with the part program. Option 112 specifies the probable part radius which is used in the feedrate command formula. During the course of the part program, the radius becomes larger or smaller; the radius can be changed by the MACHIN statement as:



MACHIN/GECENT, n, OPTAB, 112, r.

unless the option is changed, the radius is assumed to be the standard value of 6 inches.

Because of the variable nature of the part radius R, the table feedrate minimum, maximum, and rapid traverse must also be determined by the postprocessor for each part program. These rotary speeds are given in RPM and are converted to IPM as $F_{IPM} = 2\pi R * RPM$, where R is the radius. Options 133, 114 and 115 specify the minimum, maximum and rapid traverse speeds respectively.

A multiaxis linear motion obtains its feedrate command from the same relation as for non-multiaxis motions, namely,

$$F_c = \frac{D * F_{IPM}}{S}$$

However, S in this case is not the length of the space curve which results from the combined linear and rotary motions in machine coordinates, but rather is the part path length. For example, assume the part coordinates:

$x_1 = 0,\ y_1 = 1,\ z_1 = 2,\ i_1 = 0,\ j_1 = 0,\ k_1 = 1;$

$x_2 = 3,\ y_2 = 5,\ z_2 = 2,\ i_2 = 0,\ j_2 - 1,\ k_2 = 0.$

The resultant machine coordinates are then, say:

$X_1 = 0,\ Y_1 = 0,\ Z_1 = 9,\ A_1 = 0,\ B_1 = 90;$

$X_2 = 6,\ Y_2 = 8,\ Z_2 = 12,\ A_2 = 10,\ B_2 = 180.$

4.1.1.1 FEEDRATE NUMBER COMMAND (cont'd)

The determination of the space curve for a multiaxis move is a highly complex relation, so a close approximation is used instead. The path length is determined from the part coordinates, which, in the example, gives:

$$S = \sqrt{3^2 + 4^2 + 0^2} = 5.$$

The machine coordinate departures ΔX, ΔY, ΔZ, ΔA, or ΔB are not used at all in the determination of the feedrate command.

A multiaxis circular interpolation move uses a relation similar to but slightly different from the non-multiaxis relationship, for in the multiaxis case, F_{IPM} in the feed command formula is not actually the path velocity, hence, it must be corrected as:

$$F_{IPM} = \sqrt{F_P^2 - F_N^2},$$

where F_P is the desired path velocity, and F_N is the velocity in the axial direction perpendicular to the plane of the circular arc. Thus,

$$F_C = \frac{D * \sqrt{(F_P^2 - F_N^2)}}{R} \quad .$$

this can be reduced to known terms in the following expression:

$$F = \frac{D*F_P}{\sqrt{\frac{\Delta z^2}{\Theta_{xy}^2} + R^2_{xy}}},$$

where ΔZ is the departure along the Z axis, R_{xy} is the radius of arc in the XY plane, and Θxy is the angle of arc in radians. The relationship shown is for the case when the circle lies in the XY plane, but similar relationships exist for the YZ and ZX planes.

Subroutine PROCQD determines Θ and stores it into the parameter ARCANG. The radius is determined from the arc center offsets as

$$R = \sqrt{I^2 + J^2 + K^2} .$$

4.1.1.2 INVERSE TIME FEEDRATE COMMAND

This type of contouring feedrate command is selected by setting option 10 to a negative value.

The format for inverse time is essentially the same as for the dimension multiplier type except that in all cases, regardless of the departure lengths, the dimension multiplier, D, is always 1.

Thus, from the linear relation given in Section 4.1.1.1,

$$F_C = \frac{D * F_{IPM}}{S}$$

it can be seen that when D = 1,

$$F_C = \frac{F_{IPM}}{S} = \frac{F_{IPM}}{T*F_{IPM}} = \frac{1}{T}$$

where T is time in minutes.

The command maximum and minimum range for this type of feedrate command varies as a function of the preparatory function G code; the feedrate register format also changes with the G code, thereby necessitating a change in the column print vectors. The feedrate format can also be one of four possible kinds; the kind is specified by option 10. All of these variables are summarized in the following chart.

4.1.1.2 INVERSE TIME FEEDRATE COMMAND (cont'd)

	G Code	REGFOR(11)	F_{COMIN}	F_{COMAX}
option	G11	31	1.0	999.9
10 = -1	G01	32	0.1	99.99
	G10	33	0.01	9.999
	G11	30	1.0	999.0
option	G01	31	0.1	99.9
10 = -2	G10	32	0.01	9.99
	G12	40	1.0	9999.0
	G11	41	.1	999.9
option	G01	42	.01	99.99
10 = -3	G10	43	.001	9.999
	G12	42	1.0	9999.99
	G11	43	.1	999.999
10 = -4	G01	44	.01	99.9999
	G10	45	.001	9.99999
	G23	46	.0001	.999999
	G26	47	.00001	.0999999

Since the feedrate command maximum for this feed command type is larger than with the dimension multiplier feedrate command type, it is possible to obtain higher feedrates and shorter execution times with the 1/T type.

The postprocessor must check the G code of each command block in order to redetermine and reset the command maximum and minimum values, F_{COMAX} and F_{COMIN}, respectively; this is all done in subroutine FVARGO which is called from subroutine CONTUR.

4.1.1.2 INVERSE TIME FEEDRATE COMMAND (cont'd)

Subroutine FVARGO also redetermines and resets the column print vectors NPR, NPT, NPTA, and NFP as a function of the changed feedrate register format; see Section 3.5.4.1. These column print vectors are used also for producing the punched output. The vectors must change when the decimal format of the register changes because the number of places to the right of the decimal point must be exactly specified or else an error results.

For example, suppose the format is left to be 30 as for a G11 when option 10 = -2. Suppose now we get a feedrate command of 8.76 for a G10. The postprocessor outputs the feedrate command according to the decimal format, so the postprocessor would output the erroneous value F876 instead of F00876, a considerably different value.

Subroutine FVARGO sets the dimension multiplier parameter GDIMUL to 1 so that upon returning to subroutine CONTUR, the same program flow is followed as for the dimension multiplier type.

4.1.1.3 EIA MAGIC 3 FEEDRATE COMMAND

This type of contouring feedrate command is selected by setting option 10 to +1.

This rarely used format simply converts the feedrate in IPM to the EIA "Magic 3" format. For example, the feedrate 40 IPM when converted to the feedrate command becomes 540; see Section 7.1 of the Appendix for an explanation of the "Magic 3" conversion method.

Note that because of the resultant integer value of the "Magic 3" number that the feedrate register format <u>must</u> be 30, i.e., REGFOR(11) = 30.

Also with this type of feedrate format, the postprocessor cannot use the sequence which ratioes the axes feedrate through the IJK registers whenever the feedrate command exceeds the feedrate command maximum; therefore, option 26 must be set to 1.

4.1.2 POSITIONING FEEDRATE COMMANDS

The feedrate command for a positioning machine can evolve from one of a variety of different methods. A different method is used by nearly every NC machine mode. Since the number of probable methods is unlimited, the positioning feedrate commands are defined as types; the type is designated by option 78.

Subroutine POSIT calls subroutine POSFED which branches to the subroutine for the designated feedrate type; the type subroutine is usually named according to its type, thus, a type 2 feedrate command is handled in subroutine FTYPE2. If there is no separate subroutine for the type, the type is generated exclusively in subroutine POSFED.

The only test made on positioning feedrates is to ensure that the feedrate in IPM (or IPR) and the feedrate commands are within the range extremums.

Nearly all positioning feedrate types have a set of discrete values in one or more ranges. These discrete values, when available, are stored in the FRTAB section of table SRTAB; see Section 5.6.

Option 174, 62, 63, 144, and 78 must be set in accordance with the requirements of the specified feedrate type.

In all the examples given below, the feedrate command is always given as the value which would appear on the printed output; the punched value would not have a decimal point but might have leading zeroes.

Row numbering always begins at zero and increases monotonically by one. Thus, if FRTAB has forty speeds, they are said to be stored at rows 0 through 39.

4.1.2.1 FEED TYPE 0

This type is selected by a zero or negative setting for option 78.

The command is generated by multiplying the feedrate in IPM times some constant which is given in option 78. The command is rounded to the closest integer.

$$F_{COM} = F_{IPM} * K,$$

where K = |Option 78|.

If option 78 is zero, K = 2.

For example: option 78 = -3.

Therefore, K = 3.

Feedrate	F_{COM}
7.123	21
18.2694	55
1.74	5.0
94.926	285.0

This feedrate command type is programmed within subroutine POSFED.

4.1.2.2 FEED TYPE 1

This type is selected by setting option 78 to +1. The feedrate in IPM is also the feedrate command; there is no conversion necessary.

For example: Option 78 = 1, FORMAT(11) = 21.

Feedrate	F_{COM}
0.1	00.1
2.63	02.6
35.17	35.2
40.0	40.0

This feedrate command type is programmed with subroutine POSFED.

4.1.2.3 FEED TYPE 2

This type is selected by setting option 78 to + 2.

The feedrate command is obtained with the feedrate value in IPR; the current spindle range value specifies which feedrate range to use in the table of discrete IPR values. The feedrate command is the resultant row number of the table position containing the required feedrate in IPR.

The table of discrete IPR values is scanned whenever a feedrate is given to ensure that the programmed feedrate is actually available on the NC machine. Thus, whenever a feedrate is programmed in IPM, it is first converted to IPR by

$$F_{IPR} = \frac{F_{IPM}}{S} \quad .$$

where S is the current spindle speed. The resulting feedrate in IPR is sought in FRTAB, and the closest value to the given IPR value is used; the feedrate in IPM is then recomputed. This is illustrated in the example below.

4.1.2.3 FEED TYPE 2 (cont'd)

This feedrate command type is programmed in subroutine FTYPE2 which is called from subroutines POSFED and FEDRAT. In the calling sequence of subroutine FTYPE2, the K flag indicates which operation is to be performed. When called from subroutine FEDRAT (K = 0), subroutine FTYPE2 is called upon to obtain the exact IPR value, as explained above; when called from subroutine POSFED (K = 1), subroutine FTYPE2 obtains the feedrate command. These operations are exemplified by the following case.

Assume there are 15 IPR feedrates in three ranges; there are also three spindle ranges.

FRTAB

Row	F_{COM}	Range 1	Row	F_{COM}	Range 2	Row	F_{COM}	Range 3
0	11	0.01	5	16	0.04	10	21	0.10
1	12	0.02	6	17	0.06	11	22	0.14
2	13	0.03	7	18	0.08	12	23	0.18
3	14	0.04	8	19	0.10	13	24	0.22
4	15	0.05	9	20	0.12	14	25	0.26

For these conditions the pertinent options are: option 78 = 2; option 62 = 3; option 63 = 5; option 144 = 11. Example: Spindle speed = 100 RPM, spindle range = 2, programmed feedrate is 7 IPM.

Subroutine FTYPE2 is called from subroutine FEDRAT to ensure that 7 IPM is available.

$$F_{IPR} = 7/100 = 0.07 \text{ IPR}$$

Scanning the FRTAB table we can see that in range 2 the closest and next lowest value is 0.06 IPR, therefore,

$$F_{IPR} = 0.06, \text{ and}$$

$$F_{IPM} = 0.06 * 100 = 6\text{IPM},$$

which becomes the programmed feedrate.

4.1.2.3 FEED TYPE 2 (cont'd)

To convert to the feedrate command, subroutine FTYPE2 is called from subroutine POSFED. Scanning range 2 of FRTAB (since spindle range is 2), the feedrate IPR value is found at row 6. Therefore,

$$F_{COM} = \text{Row Number} + \text{Option } 144$$

$$= 6 + 11 = 17.$$

It is important to note that the row numbering begins at zero.

4.1.2.4 FEED TYPE 3

This type is selected by setting option 78 = +3.

The feedrate command is obtained by converting the feedrate in IPM directly to the EIA "Magic 3" code equivalent; see Section 7.1 for an explanation of this technique.

For example: $F_{IPM} = 40$; $F_{COM} = 540$.

$F_{IPM} = 2$; $F_{COM} = 420$.

$F_{IPM} = 0.1$; $F_{COM} = 210$.

This feedrate command type is programmed within subroutine POSFED wherein subroutine EIACOM is called to convert the feedrate to the command form.

4.1.2.5 FEED TYPE 4

This type is selected by setting option 78 to +4.

This type feedrate utilizes two or less feedrate ranges. Only the range one feedrate values in IPR are stored in FRTAB. If there are two feedrate ranges, the range two feedrates are assumed to be five times the range one values.

4.1.2.5 FEED TYPE 4 (cont'd)

Assume the following conditions:

	Range 1		Range 2	
Row	F_{COM}	Feed IPR	F_{COM}	Feed IPR
0	F0	.001	F0	.005
1	F1	.002	F1	.010
2	F2	.003	F2	.015
3	F3	.004	F3	.020
4	F4	.006	F4	.030
5	F5	.009	F5	.045

Note that in practice only Range 1 would be stored in FRTAB. Before scanning the table for comparison selection, the feedrate in IPR is first converted to IPR, i.e.,

$$F_{IPR} = F_{IPM}/\text{Spindle Speed.}$$

For these conditions option 78 = 4, option 62 = 1, option 63 = 6, option 144 = 0.

Example 1: F_{IPM} = 0.2, Spindle Speed = 100, Feed Range = 1.

F_{IPR} = 0.2/100 = 0.002.

Therefore,

$$F_{COM} = \text{Row Number + Option 144}$$

$$= 1 + 0 = \underline{1}.$$

Example 2; F_{IPM} = 4, Spindle Speed = 200, Feed Range = 2.

F_{IPR} = 4/200 = 0.02

The values of Range 1 times 5 are scanned comparing F_{IPR} . Therefore,

$$F_{COM} = \text{Row Number + Option 144}$$

$$= 3 + 0 = \underline{3}.$$

This feedrate command type is programmed in subroutine FTYPE4.

4.1.2.6 FEEDRATE TYPE 5

This type is selected by setting option 78 to 5.

The characteristic feature of this type feedrate command is that the XY axes have their own separate set of feedrate values and so does the Z axis. The feed command for XY is programmed in the XY motion block, and the feed command for Z is programmed in the Z motion block.

There is one feedrate table with N ranges for Z motions and another feedrate table with M ranges for X-Y motion.

The total number of feedrate ranges "N + M" is stored in option 62, and the number Z feedrate ranges "N" is stored option 201. The number of feedrates per range is stored in option 63. The Z feedrate command minimum is stored in option 49, and the X-Y feedrate command minimum is stored in option 202. The increment between ranges is stored in option 144. The Z feedrate ranges are stored first in FRTAB, then followed by the X-Y ranges. Assume the following conditions:

Z Feedrates						X-Y Feedrates			
Range 1		Range 2		Range 3		Range 1		Range 2	
F_{COM}	IPM	F_{COM}	IPM	F_{COM}	IPM	F_{COM}	IPM	F_{COM}	IPM
F11	4.2	F21	10.2	F31	21.3	F21	6.3	F31	28.6
F12	6.8	F22	11.4	F32	25.6	F22	10.9	F32	35.9
F13	7.9	F23	12.6	F33	28.9	F23	15.6	F33	50.1
F14	8.5	F24	13.8	F34	31.7	F24	18.9	F34	57.8
F15	9.1	F25	15.1	F35	38.9	F25	22.5	F35	65.2

The pertinent options are set as follows:

option 78 = 5, option 49 = 11,

option 62 = 5, option 63 = 5,

option 201 = 3, option 202 = 21,

option 144 = 6.

4.1.2.6 FEEDRATE TYPE 5 (cont'd)

Note that in this case option 144 refers not to the incremental adder to the row number, but rather to the difference in F_{COM} between ranges, i.e., between 15 and 21, 25 and 31. The subroutine counts rows beginning at 11 for Z and 21 for XY.

FRTAB would be set up as follows:

FRTAB(276)	=	4.2	FRTAB(289)	=	31.7
FRTAB(277)	=	6.8	FRTAB(290)	=	38.9
FRTAB(278)	=	7.9	FRTAB(291)	=	6.3
FRTAB(279)	=	8.5	FRTAB(292)	=	10.9
FRTAB(280)	=	9.1	FRTAB(293)	=	15.5
FRTAB(281)	=	10.2	FRTAB(294)	=	18.9
FRTAB(282)	=	11.4	FRTAB(295)	=	22.5
FRTAB(283)	=	12.6	FRTAB(296)	=	28.6
FRTAB(284)	=	13.8	FRTAB(297)	=	35.9
FRTAB(285)	=	15.1	FRTAB(298)	=	50.1
FRTAB(286)	=	21.3	FRTAB(299)	=	57.8
FRTAB(287)	=	25.6	FRTAB(300)	=	65.2
FRTAB(288)	=	28.9			

Example: F_{IPM} = 40, Feed Range = 2.

For the XY move,

F_{COM} = Row Number + Option 144-1
= 27 + 6 -1 = 32.

When the exact value cannot be found, the next lowest is taken, hence, F_{IPM} = 35.9.

For the Z move,

F_{COM} = Row Number + Option 144 - 1
= 20 + 6 - 1 = 25, and
F_{IPM} = 15.1

When Range 3 is programmed for Z, Range 2 is used for XY. This feedrate command type is programmed in subroutine FTYPE5.

4.1.2.7 FEED TYPE 6

This type is selected by setting option 78 to + 6.

The available feedrates are a set of discrete values in IPM and are assembled in two ranges, a low and a high feedrate range. The low feedrate range is used only for milling and is normally selected by programming a CYCLE/MILL statement followed by a FEDRAT/RANGE, 1 statement. Otherwise, regardless of the range selected range 2 (or the high range) is always modal and is cancelled only by the CYCLE/OFF or another CYCLE statement used for both milling and drilling operations. The feed command is formed from the relation

$$F_{COM} = K * F_{IPM},$$

where K = 12 for Range 1, and K = 2 for Range 2.

Since there are only discrete IPM feeds available, the postprocessor first ensures that a programmed feedrate is truly available, and, if not, it selects the next lowest available feedrate. Subroutine FEDRAT calls subroutine FTYPE6 with K = 0 in the calling sequence which directs subroutine FTYPE6 to obtain the proper feedrate. Subroutine POSFED calls subroutine FTYPE6 with K = 1 to obtain the feedrate command. These operations are illustrated in the example below.

Assume the following conditions:

Low Feedrate-Range 1		High Feedrate-Range 2	
F_{COM}	IPM	F_{COM}	IPM
F6	.5	F4	2.0
F12	1.0	F6	3.0
F18	1.5	F8	4.0
F24	2.0	F10	5.0
F30	2.5	F12	6.0
F36	3.0	F14	7.0
F42	3.5	F16	8.0
F48	4.0	F18	9.0
F54	4.5	F20	10.0
F60	5.0	F24	12.0

The pertinent options are set as follows:

option 78 = 6, option 62 = 2, option 63 = 10.

4.1.2.7 FEED TYPE 6 (cont'd)

Example 1: F_{IPM} = 3.6, Feed Range 1,

CYCLE/MILL programmed (ICTYP = 6).

The FRTAB is scanned for an exact comparison, and not being found, the next lowest value 3.5 is selected; therefore, F_{IPM} = 3.5 and F_{COM} = 12 * 3.5 = 42.

Example 2: F_{IPM} = 7.9, Feed Range 2.

After scanning, F_{IPM} = 7, and

F_{COM} = 2 * 7 = 14.

4.1.2.8 FEED TYPE 7

This type is selected by setting option 78 to +7.

The feedrate command for this type is a one for one output of the value given in the feedrate statement. The machine tool has a number of manually set feedrate combinations which are selected by a code number; neither IPM or IPR. The code number is given in the feedrate statement and the postprocessor outputs this code as the feedrate command.

Example: Feedrate potentiometer No. 1 set manually to desired feedrate. Part program statement is FEDRAT/1. Postprocessor will output f_{COM} = 1

This feedrate command type is programmed within subroutine POSFED.

4.1.2.9 FEED TYPE 8

This type is selected by setting option 78 to + 8.

The feedrate command for this type is generated in a manner similar to Feed Type 2; see Section 4.1.2.3 for full details. The feedrates consist of a discrete set of IPR values in three ranges; the feedrate range is selected as a function of the current spindle range. The selected IPR value is converted to an EIA "Magic 3" code to become the feedrate command; see Section 6.1 for a description of this conversion method.

4.1.2.9 FEED TYPE 8 (cont'd)

Using the feed tables in the example of Feed Type 2, the following example illustrates Feed Type 8.

Example: F_{IPM} = 7, Spindle Speed = 100 RPM, spindle range =2.

F_{IPR} = 7/100 = 0.07.

Scanning FRTAB in range 2 we obtain 0.06 IPR; therefore, F_{IPR} = 0.06, F_{IPM} = 6, and F_{COM} = 260.

This feedrate command type is programmed within subroutine FTYPE2.

4.1.2.10 FEED TYPE 9

This type is selected by setting option 78 to +9.

The generation and use of the feedrate command for this type is identical to Feed Type 2 in every respect except that the values stored in FRTAB are in IPM rather than IPR; see Section 4.1.2.3 for full details on Feed Type 2.

The feedrates consist of a discrete set of IPM values in three ranges; the feedrate range is selected as a function of the current spindle range. The feed command is derived from the row number of the table position containing the required feedrate.

Assume the following conditions:

FRTAB

Row	F_{COM}	Range 1	Row	F_{COM}	Range 2	Row	F_{COM}	Range 3
0	11	1	5	16	14	10	21	50
1	12	3	6	17	18	11	22	60
2	13	6	7	18	26	12	23	70
3	14	9	8	19	30	13	24	80
4	15	11	9	20	40	14	25	90

The pertinent options are option 78 = 9; option 62 = 3, option 63 = 5, option 144 = 11.

4.1 2.10 FEED TYPE 9 (cont'd)

Example: F_{IPM} = 45, spindle range = 2.

Closest value in range 2 to programmed value is 40, therefore,
F_{IPM} = 40, and
F_{COM} = Row Number + option 144
= 9 + 11 = 20.

It is important to note that the row numbering begins at zero. This feedrate command type is programmed within subroutine FTYPE2.

4.1.3 POSITIONING MACHINE ROTARY FEEDRATE COMMANDS

Positioning machines which have a rotary axis may have a separate feedrate register exclusively for the rotary axis. In such cases the rotary feedrate command can be of a format entirely different from the feedrate command format for the linear axes. Therefore, the rotary feedrate commands are defined as a set of types as were the feedrate commands for the positioning linear axes.

The rotary feedrate command may or may not have its own feedrate register, or it may use the linear axes feedrate register. If there is a separate rotary feedrate register, then option 139 must be set accordingly.

Option 141 specifies the rotary feedrate command type. Each of the rotary feedrate command types are defined below.

4.1.3.1 ROTARY FEED TYPE 1

This rotary feed type is selected by setting option 141 to +1. It requires the use of a separte register for the rotary feedrate command, e.g., an E register. Since this is an extra register (not one of the permanent assignments), the register is assigned to DBFSEG(16); therefore, REGFOR(16) and REGSTR(16) must be set accordingly, and option 139 is set to 16. For example:

```
REGSTR(16) = E
REGFOR(16) = 10.0
```

option 139 = 16, option 141 = 1.

Subroutine ROTYP1 processes the rotary feedrate type 1 operation.

4.1.3.1 ROTARY FEED TYPE 1 (cont'd)

This type has a table of discrete rotary feeds which must be used as the feedrate for the rotary table. In other words if a feedrate is programmed which is a value not exactly found in the table, the postprocessor selects the next lowest exact feedrate. Corresponding to the row of the table is the rotary feedrate command value.

Subroutine ROTYP1 has the parameter K in it's calling sequence; when K = 0, the subroutine selects the closest IPM value from the table of values to the given feedrate value; when K = 1, the subroutine obtains the feed command corresponding to the given feedrate value. These functions are illustrated in the example below.

The table of IPM feedrates are generated from the relations:

1) $$F_{RPM} = \frac{RF_{MAX}}{RK^{(9-n)}}, \quad n = 1,2,3,-----9,$$

where RF_{MAX} is the rotary maximum feedrate in RPM, RK is a constant equal to 1.43, and F_{RPM} is the feedrate in RPM.

2) $F_{IPM} = 2\pi \; F_{RPM} * R_T$, where R_T is the table (or part) radius, and F_{IPM} is the rotary feedrate in IPM.

Option 112 is the table (or part) radius R_T, and option 114 is the rotary maximum feedrate RF_{MAX}.

With the specified conditions of options 112 and 114, subroutine ROTYP1 generates the IPM table upon initial entry into the subroutine.

Assume the following conditions:

Row	F_{COM}	Feedrate in IPM	Feedrate in RPM
0	E0	0.754	.020
1	E1	1.0801	.0287
2	E2	1.5419	.0409
3	E3	2.2009	.059
4	E4	3.1479	.084
5	E5	4.5051	.115
6	E6	6.4476	.171
7	E7	9.1986	.244
8	E8	13.1570	.349
9	E9	18.8496	1.0

4.1.3.1 ROTARY FEED TYPE 1 (cont'd)

Example: The programmed rotary feedrate is 7 IPM.

When it is time for the feedrate to be stored for use, subroutine ROTABA calls subroutine ROTYPE which, because of option 141, calls for subroutine ROTYP1 with K = 0. The subroutine scans the IPM table, and finding no feedrate exactly equal to 7, selects the next lowest value of 6.4476.

At output time, subroutine ROTYP1 is called form subroutine POSIT with K = 1 to obtain the rotary feedrate command, which for the IPM value of 6.4476, is in row 6 (row numbering begins at zero); therefore, the rotary feedrate command is 6.

4.1.4 RAPID TRAVERSE

Although a rapid traverse is nothing more than a high feedrate, the postprocessor gives special treatment to such moves. A rapid traverse is normally used for repositioning a tool for a tool change or when moving to a new cut point; and since it is a non-cutting motion, the path is traversed usually at the maximum feedrate so as to minimize the machining time. But in all cases except for some positioning machines which have a separate rapid traverse register, a rapid traverse motion is only a regular motion. In order to differentiate between a rapid traverse and a regular feed motion, the postprocessor identifies a rapid traverse command block with a negative feedrate. Thus, at output time the rapid traverse blocks can be easily singled-out for special optimizing treatment (See Section 4.1.5.4.)

When a RAPID is called for, the postprocessor uses the maximum feedrate value (option 42) on the next motion block if RAPID is one-shot, which it normally is. However, if option 109 is set for a modal condition, then the rapid feedrate value is used on all motion blocks until the rapid traverse mode is cancelled.

Some NC machines, usually lathes, require an M code to enter into and out of a rapid traverse gear setting. The postprocessor automatically outputs these M codes and any requisite dwells that are required; see options 16, 37, 39, 42, 43, 44, 45, 46, 81, and 109.
A brief description of a typical part program example will clarify the postprocessor's method of handling rapid traverses.

```
(1)  RAPID
(2)  GOTO/x, y, z
(3)  GOTO/x, y, z
```

4.1.4 RAPID TRAVERSE (cont'd)

On statement 1 subroutine RAPID* is called; this subroutine simply sets the rapid flag RAPFLG to 1 indicating that a rapid condition has been called for.

On statement 2 subroutine MOTION calls subroutine TSTFLG to interrogate RAPFLG; since RAPFLG is non-zero, subroutine RAPIDO* is called wherein RAPFLG is set to zero, and the rapid-on flag FLRPON is set non-zero. Subroutine RAPIDO proceeds to output any required gear shifting M codes; sets the rapid feed flag FRAPID to the negative maximum feedrate value (option 42); and returns to subroutine TSTFLG. The motion block can now be output.

When outputting the motion block, subroutine OUTPUT tests the FRAPID flag, and finding it negative, stores the value of FRAPID into DBFSEG(11) to become the feedrate for that block. The negative sign is retained in order to identify the move as being a rapid traverse.

On statement 3, subroutine TSTFLG checks RAPFLG, and finding it zero, then checks flag FLRPON, and finding it non-zero, calls subroutine RAPIDX* to remove the rapid traverse condition. This subroutine sets flag FRAPID and FLRPON to zero; outputs any required gear shifting M codes, and returns to subroutine TSTFLG. The motion block is now made output, but subroutine OUTPUT, now that FRAPID is zero, uses the current feedrate FEDIPM to store in DBFSEG(11), thereby achieving a return to the feedrate mode.

On statement 3, had there been another RAPID statement, RAPFLG would be 1, therefore causing a call to subroutine RAPIDO. But subroutine RAPIDO, upon testing FLRPON, finds it already non-zero; therefore, the postprocessor knows that a rapid mode already exists; hence, there is no need to reestablish it. This avoids the redundant output of gear shifting M codes.

Subroutine RAPIDO makes some preliminary checks before it decides to enter into a rapid mode. The minimum path length (option 37) is checked to see if it is larger than the given motion. If it is, there is no point in shifting gears or otherwise entering into the high feedrate range since the move is too short to warrant the time required. In this case, the postprocessor uses the highest feedrate of the current feed range (option 39).

* This subroutine is actually a multiple entry subroutine with one of the entries so titled.

4.1.4 RAPID TRAVERSE (cont'd)

Subroutine RAPIDO then checks for a tape reader limitation using the resulting rapid feedrate. If the block is tape reader limited, flag FRAPID is set to the positive highest value of feedrate which does not cause a tape reader limitation. Hence, in subroutine OUTPUT, since FRAPID is positive, it's value is used for the feedrate, but the value is made negative to identify the command block as being one of a rapid traverse.

4.1.5 FEEDRATE OPTIMIZATION

There are many conditions which during the course of a part program can cause a lowering or limiting value to the programmed feedrate. But there are also several special techniques which it may be possible to apply in order to obtain the programmed feedrate, or barring this possibility, at least to obtain the highest possible feedrate. Each of these several techniques are discussed in detail in the following sections.

4.1.5.1 G CODE SEGMENTATION

Consider the following case where a motion has the increments ΔX=30 and ΔY = 40 inches, and the programmed feedrate is 300 IPM. The feedrate command (assume option 10 = 0) for this motion is: (See Section 4.1.1.1)

$$F_{COM} = \frac{D * F_{IPM}}{S} = \frac{100 * 300}{\sqrt{900 + 1600}} = \frac{30000}{50} = 600.$$

This value is greater than the feedrate command maximum of 500; therefore, F_{COM} would be made equal to 500. But this is highly restricting, for now the feedrate has been reduced to 250 IPM;

$$F_{IPM} = \frac{F_{COM} * S}{D} = \frac{500 * 50}{100} = 250 \text{ IPM.}$$

This reduction is undesirable and can be eliminated.

If we take the original path of 50 inches and segment it to path sizes such that the increments are less than 10 inches, then it becomes possible to use a smaller dimension G code with the possibility of obtaining an acceptable feedrate command.

4.1.5.1 G CODE SEGMENTATION (cont'd)

Accordingly then, we get five segments such that

$$\Delta X = \frac{30}{5} = 6, \text{ and } \Delta Y = \frac{40}{5} = 8$$

Each of the segment paths now has an

$$F_{COM} = \frac{10 * 300}{\sqrt{36 + 64}} = \frac{3000}{10} = 300,$$

which is considerably below the feedrate command maximum of 500; therefore, the programmed feedrate of 300 IPM can now be used.

This G code segmentation procedure exists in subroutine TSTFCM which is called from subroutine SELG when the linear preparatory function G code is being selected. (See Section 3.4.6.1)

Option 170 must be set to 1 to call for this sequence. Note that this G code segmentation cannot be used for multiaxis moves, nor for circular interpolation or rotary moves; and obviously, it cannot be used if only one G code is available. Also, the feedrate command must be of the dimension multiplier type, i.e., option 10 = 0.

In attempting to segment the path, the subroutine begins with the initially selected dimension G code, and works its way down to each successively small and available dimension G code. Subroutine COMPFC computes the feedrate command.

4.1.5.2 VARIABLE MAXIMUM FEEDRATE ON EACH AXIS

The most common condition for NC machines is to have the same feedrate maximum on all axis. Such a condition requires no special concern since all axes respond identically when under the same limitations. However, some NC machines, usually those of the very large variety, have different feedrate limitations on one or more of their axes. This restriction can affect the programmed feedrate, that is, cause a reduction of the programmed value if the resulting component axis feedrate is greater than the allowable axis feedrate maximum.

4.1.5.2 VARIABLE MAXIMUM FEEDRATE ON EACH AXIS (cont'd)

The postprocessor must interrogate each motion block and its feedrate, and compute each component axis feedrate to compare versus the allowable limits. These limits are specified in FRTAB and ordered according to axes and feedrate ranges. The example below best illustrates the usage and operation of this sequence.

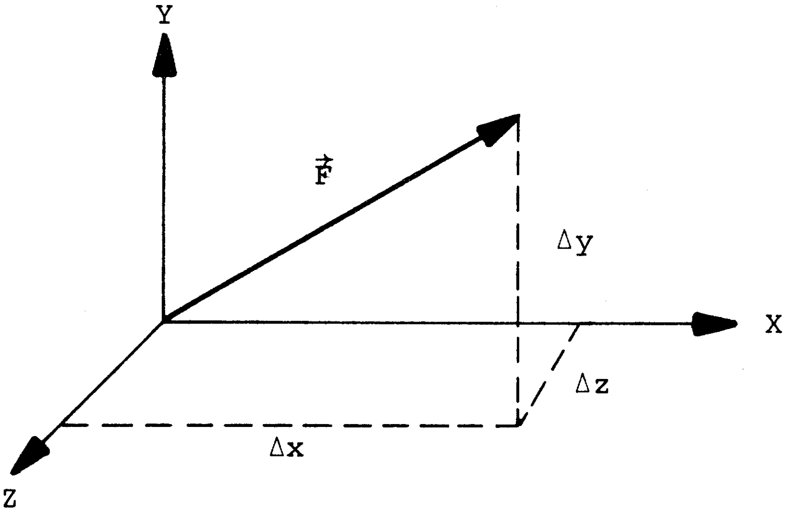


Diagram 4.1.5.2A

For a part programmed feedrate $\vec{F}$, the postprocessor determines the axis component feedrate;

$$F_x = \frac{\Delta x \vec{F}}{\sqrt{\Delta x^2 + \Delta y^2 + \Delta z^2}}$$

similary for F_y and F_z.

Each axis component feedrate is then compared versus its allowable maximum and minimum feedrate, and where a bound is exceeded, the feedrate is set to that bound. The axis feedrate which has the most limiting condition is used to then redetermine the allowable programmed feedrate.

4.1.5.2 VARIABLE MAXIMUM FEEDRATE ON EACH AXIS (cont'd)

The axes maximum and minimum feedrates are stored in the feedrate table FRTAB as:

FRTAB(289)	=	X axis min	Range 1
FRTAB(290)	=	X axis max	
FRTAB(291)	=	Y axis min	
FRTAB(292)	=	Y axis max	
FRTAB(293)	=	Z axis min	
FRTAB(294)	=	Z axis max	
FRTAB(295)	=	X axis min	Range 2
FRTAB(296)	=	X axis max	
FRTAB(297)	=	Y axis min	
FRTAB(298)	=	Y axis max	
FRTAB(299)	=	Z axis min	
FRTAB(300)	=	Z axis max	

In the example above, only two ranges are used, but as many ranges can be used as room in FRTAB permits.

The tabled values must also be ordered according to the axes setting of option 59. In the example above, option 59 was assumed to be ordered as XYZ, but if the machine were for a lathe, the order could be ZX, (Y is implied but disregarded). In this case then, the storage would be:

FRTAB(295)	=	Z axis min	(Assume 1 Range)
FRTAB(296)	=	Z axis max	
FRTAB(297)	=	X axis min	
FRTAB(298)	=	X axis max	
FRTAB(299)	=	(Y axis min)	
FRTAB(300)	=	(Y axis max)	(can be set to zero)

4.1.5.2 VARIABLE MAXIMUM FEEDRATE ON EACH AXIS (cont'd)

The postprocessor utilizes the above feedrate table in the following example.

Assume the machine is a lathe (ordered as +Z-X) and has the following feedrate table:

	FRTAB (289)	=	0.03	Z		FRTAB (295)	=	0.13	Z
	FRTAB (290)	=	3.4			FRTAB (296)	=	13.5	
Range 1	FRTAB (291)	=	0.01	X	Range 2	FRTAB (297)	=	0.36	X
	FRTAB (292)	=	1.0			FRTAB (298)	=	36.0	
	FRTAB (293)	=	0	Y		FRTAB (299)	=	0	Y
	FRTAB (294)	=	0			FRTAB (300)	=	0.	

The programmed feedrate is 100 IPM and feedrate range 2 is used. The departures are ΔX = 2, ΔZ = 3.

Then,

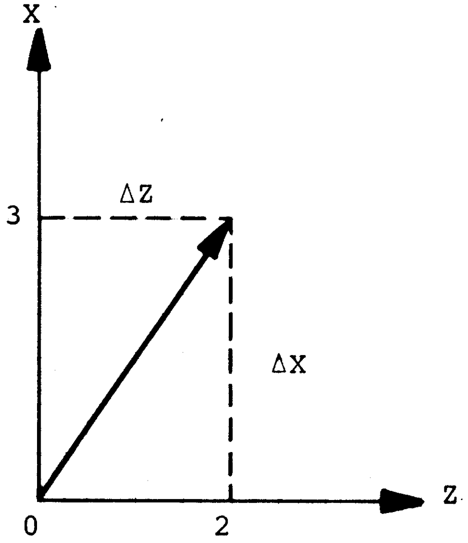


Diagram 4.1.5.2B

4.1.5.2 VARIABLE MAXIMUM FEEDRATE ON EACH AXIS (cont'd)

$$F_x = \frac{2*100}{\sqrt{4+9}} = \frac{200}{3.61} = 55.5 \text{ IPM},$$

$$F_z = \frac{3*100}{\sqrt{4+9}} = \frac{300}{3.61} = 83.3 \text{ IPM}.$$

F_x >36, therefore, F_x is made 36 IPM. Similarly, since F_z>13.5, F_z is made 13.5 IPM.

$\vec{F}$ must be recomputed to determine which condition is the more limiting.

Using F_x, $\vec{F} = 36 * \frac{3.61}{2} = 65$ IPM, indicating that the

programmed feedrate could be as much as 65 IPM without exceeding the X axis maximum feedrate. Computing for the Z axis;

using F_z, $\vec{F} = 13.5 * \frac{3.61}{3} = 16.2$ IPM, indicating that the

programmed feedrate on the Z axis is the more limiting. Therefore, $\vec{F}$ must be reduced to 16.2 IPM. The component feedrates are now within acceptable limits.

$$F_x = \frac{2}{3.61} * 16.2 = 9.0 \text{ IPM}$$

$$F_z = \frac{3}{3.61} * 16.2 = 13.5 \text{ IPM}.$$

A final test is made on $\vec{F}$ to make sure $\vec{F}$ is less than or equal to the absolute value of option 25. If $\vec{F}>|$ OPTAB(25)$|$, $\vec{F}$ is set equal to OPTAB(25).

This sequence resides in subroutine FEDLIM which is called from subroutine SELG. Although machines which have different feedrate maximums on each axis must utilize the feedrate optimizing sequence of subroutine FEDLIM, this does not pertain to varying rapid traverse maximums. See Section 4.1.5.4 for this function.

To call for the use of subroutine FEDLIM, option 25 must be set negative. If multiple feedrate ranges exist, option 18 must be set accordingly.

4.1.5.2.1 VARIABLE MAXIMUM FEEDRATE ON MULTIAXIS MACHINES

Subroutine FEDLIM is also used for multiaxis machines which have different feedrate maximums on each axis. For multiaxis machines the maximum and minimum feedrates for the rotary motions are stored (in RPM) in SRTAB. The linear feedrates are ordered as specified by option 59, followed by the rotary feedrates as ordered in DBFSEG; head first, table second, followed by the third rotary axis if there is one.

In the following example only two feedrate ranges are used, but as many ranges can be used as room in SRTAB permits.

Range 1	Range 2
SRTAB(281) = X axis minimum	SRTAB(291) = X axis minimum
SRTAB(282) = X axis maximum	SRTAB(292) = X axis maximum
SRTAB(283) = Y axis minimum	SRTAB(293) = Y axis minimum
SRTAB(284) = Y axis maximum	SRTAB(294) = Y axis maximum
SRTAB(285) = Z axis minimum	SRTAB(295) = Z axis minimum
SRTAB(286) = Z axis maximum	SRTAB(296) = Z axis maximum
SRTAB(287) = head axis minimum	SRTAB(297) = head axis min.
SRTAB(288) = head axis maximum	SRTAB(298) = head axis max.
SRTAB(289) = table axis minimum	SRTAB(299) = table axis min.
SRTAB(290) = table axis maximum	SRTAB(300) = table axis max.

Option 25, the maximum feedrate in IPM, must be set negative to indicate that each axis has its own maximum and minimum feedrate per range. Option 112, the radius in inches of the part, and option 128, the head tool swing radius in inches, must also be set.

The APT program computes the XYZ coordinates and the direction cosines of the tool vector offset from the part surface by the radius of the cutter. For multiaxis machines the postprocessor uses transformation (class) equations to relate part geometry to machine geometry and slide motion. The motions of the rotary axes are derived from the direction cosine data.

Since the APT program calculates only linear cut vector, the part program tool tip path length is determined from the relationship

$$L \text{ (part program path)} = \sqrt{\Delta X^2 + \Delta Y^2 + \Delta Z^2}$$

where ΔX, ΔY, and ΔZ are the increments between CL data points.

The feedrate number is determined using this path length L and any of the conventional methods given by option 10.

4.1.5.2.1 VARIABLE MAXIMUM FEEDRATE ON MULTIAXIS MACHINES (cont'd)

When a given cut vector has been resolved by the transformation equations into machine slide motions, it is necessary to determine that no axis feedrate constraints have been violated. The following example will illustrate the method:

Problem statement: To cut a spiral groove 1/4 inch wide and 1/8" deep in the face of a cone. (See Figure 4.1.5.2.1A)

Given: (1) Cone with 4 inch base diameter, 8 inches in height.
(2) Option 112 = 2.0 inches, option 128 = 3 inches.
(3) Groove to have one inch lead along major axis of cone (Y axis of machine).
(4) Tool axis to be normal to cone surface at all times.
(5) Desired feedrate is 40 IPM in range 1.
(6) SRTAB is set thus:

Range 1			Range 2		
SRTAB(281)	= .05	X	SRTAB(291)	= .1	X
SRTAB(282)	= 4.0		SRTAB(292)	= 8.0	
SRTAB(283)	= .1	Y	SRTAB(293)	= .2	Y
SRTAB(284)	= 6.0		SRTAB(294)	= 12.0	
SRTAB(285)	= .1	Z	SRTAB(295)	= .2	Z
SRTAB(286)	= 6.0		SRTAB(296)	= 12.0	
SRTAB(287)	= .5	A	SRTAB(297)	= 1.0	A
SRTAB(288)	= 2.0		SRTAB(298)	= 4.0	
SRTAB(289)	= .5	B	SRTAB(299)	= 1.0	B
SRTAB(290)	= 3.0		SRTAB(300)	= 6.0	

Cone 1/2 angle $= \tan^{-1} \frac{2}{8}$

$= 14.03°$

Lead (Y axis) = 1 in. or $\frac{1}{360}$ inch per degree of rotation

Tool advance (Z axis) = (1" X sin 14.03°) cos 14.03° in/revolution
= .234 inches/revolution
= .234/360 inches/degrees

4.1.5.2.1 VARIABLE MAXIMUM FEEDRATE ON MULTIAXIS MACHINES (cont'd)

It can be shown that for a given tolerance t, the maximum linear cut vector (L) at radius R is given by

$$L = 4\sqrt{RT}.$$

Assuming equal inside and outside tolerances, T = .005 at

radius R, $$L = 4\sqrt{2 * .005} = .4''$$

The table will rotate $2 \sin^{-1} \frac{L}{2(R+t)}$ degrees.

$$\text{Table rotation } B = 2 \sin^{-1} \frac{.4}{2(2.005)} = 2 \sin^{-1} .0998 = 11.46°$$

The machine data is

$\Delta X = 0$ (no motion of X axis)

$\Delta Y = \frac{1''}{360} * 11.46° = .0318''$ (one inch spiral lead)

$\Delta Z = \frac{.234}{360} * 11.46° = .00745$ (in feed of Z axis)

$\Delta A = 0$ (assuming previously set at 14.03°)

$\Delta B = 11.46°$ (rotation of table for each cut vector)

For a given feedrate $\vec{F}$ the axis component feedrate is computed for each axis by the formula:

$$F_n = \frac{\Delta n \vec{F}}{L}$$

where the factor $\vec{F}/L$ is the inverse of the cut time.

4.1.5.2.1 VARIABLE MAXIMUM FEEDRATE ON MULTIAXIS MACHINES (cont'd)

Each axis component feedrate is compared with its allowable maximum and minimum values in SRTAB, where a bound is exceeded, the feedrate is set to that bound. The axis feedrate which has the most limiting condition is used to redetermine the allowable programmed feedrate.

$$F_Y = \frac{.0318*40}{.4} = 3.18 \text{ IPM}$$

$$F_z = \frac{.00745*40}{.4} = .745 \text{ IPM}$$

The table move is converted to inches

$$\Delta B = 2 * 3.14159 * \frac{11.46°}{360°} * 2 \text{ in} = .4 \text{ inches}$$

$$F_B = \frac{.4 * 40}{.4} = 40 \text{ IPM}$$

The maximum allowable feedrate in IPM for the table is

$$F_B \text{ (allow)} = 2 * 3.14159 * 2 * 3 = 37.7 \text{ IPM}$$

Since $F_B > F_B$ (allow), therefore, F_B is set to 37.7.

Recomputing $\vec{F}$ gives

$$\vec{F} = \frac{37.7 * .4}{.4} = 37.7$$

All component axis feedrates will be correspondingly reduced since the 40 IPM requested feedrate cannot be achieved due to the limitation of the rotary table (B axis)

$$F_Y = \frac{.318 * 37.7}{.4} = 3.0 \text{ IPM}$$

$$F_z = \frac{.00745 * 37.7}{.4} = .702 \text{ IPM}$$

4.1.5.2.1 VARIABLE MAXIMUM FEEDRATE ON MULTIAXIS MACHINES (cont'd)

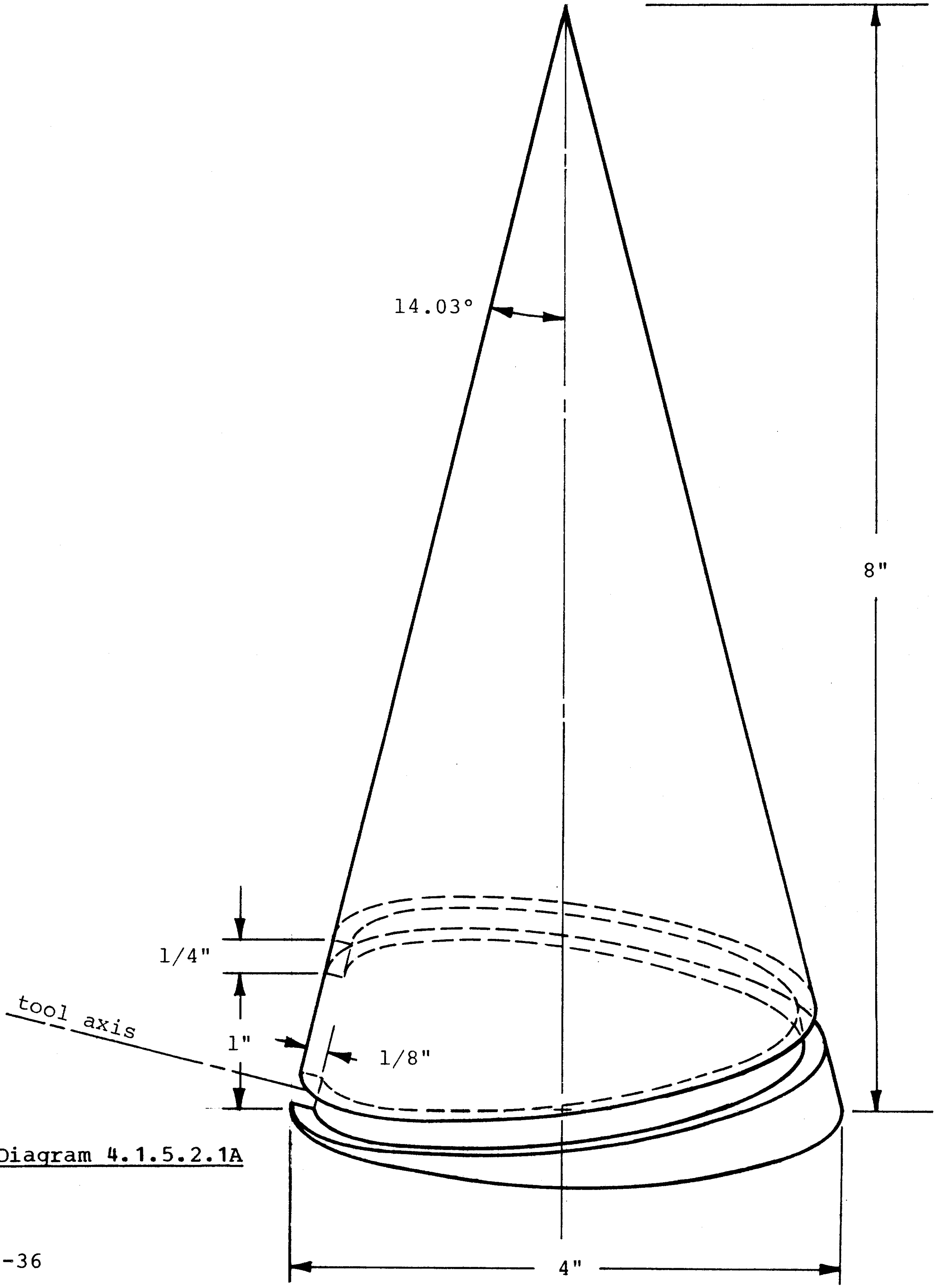


Diagram 4.1.5.2.1A

4.1.5.3 FEEDRATE MULTIPLIER CONSTANT

Whenever a feedrate command (dimension multiplier or inverse time type) for a linear move exceeds the command maximum, and unless the postprocessor can use some optimizing method which retains the programmed feedrate, the feedrate command is set to the command maximum which necessarily causes a reduction in the feedrate; see the example in Section 4.1.5.1.

One of the optimizing methods that the postprocessor can use requires the availability of the IJK registers which are normally used for circular interpolation and threading. This method, in effect, proportionately reduces the feedrate command by ratioing the incremental moves by some arbitrary constant.

Whenever the calculated feedrate command exceeds the maximum, an arbitrary constant can be chosen which, when divided into the feedrate command, reduces it to its maximum value or below. The XYZ axis departures are then multiplied by this same constant, and the resultant products are programmed in their respective IJK registers.

This method can be applied to both the dimension multiplier and inverse time types; see Sections 4.1.1.1 and 4.1.1.2. The relations are:

$$F_{COM} = \frac{D * F_{IPM}}{S * C}$$

where C is the arbitrary multiplier constant.

$$\Delta I = X * C,$$

$$\Delta J = Y * C,$$

$$\Delta K = Z * C,$$

Example: Assume the inverse time feedrate command type, therefore, D = 1. $\Delta X = 0.01$ and $\Delta Y = 0.01$ inches, and the programmed feedrate is 30 IPM.

$$F_{COM} = \frac{30}{0.01414} = 2120,$$

which is greater than the feedrate command maximum of 999.9 This value can be brought down to an acceptable quantity by adopting an arbitrary constant which scales down F_{COM}.

.1.5.3 FEEDRATE MULTIPLIER CONSTANT (cont'd)

Adopt a value of 5 for C. Therefore,

$$F_{COM} = \frac{2120}{5} = 424,$$

which is now well below the maximum.

Then,

$$I = 0.01 * 5 = 0.05,$$
$$J = 0.01 * 5 + 0.05.$$

The programmed block would then contain the values $\Delta X = 0.01$, $\Delta Y = 0.01$, $I = 0.05$, $J = 0.05$, and $F_{COM} = 424$.

This scaling method cannot be used with circular interpolation because it would alter the length of the circle radius, but it may be applied to rotary motions by an analagous method, i.e., the values D and E are determined by multiplying the rotary motions by the multiplier constant; see Section 3.4.7.1.

$$D = \Delta A * C,$$

$$E = \Delta B * C.$$

The major restriction on this scaling method is that the resultant product of multiplying the departures by the multiplier constant must not exceed the storage capacity of the I,J,K,D, or E registers.

Note that the feedrate command will no longer be equal to 1/T when this method is used.

4.1.5.4 RAPID TRAVERSE OPTIMIZATION

Since a rapid traverse is a non-cutting type move, it should normally be made at the highest feedrate possible. The maximum feedrate for an NC machine may be 100 IPM, but it is possible to legitimately exceed this value by obtaining an optimim vector feedrate.

Options 42, 43, and 44 provide the maximum rapid traverse feedrate for each axis, and these values are used to optimize the rapid traverse. In the following example we assume a two-axis machine and want to rapid traverse over the path S

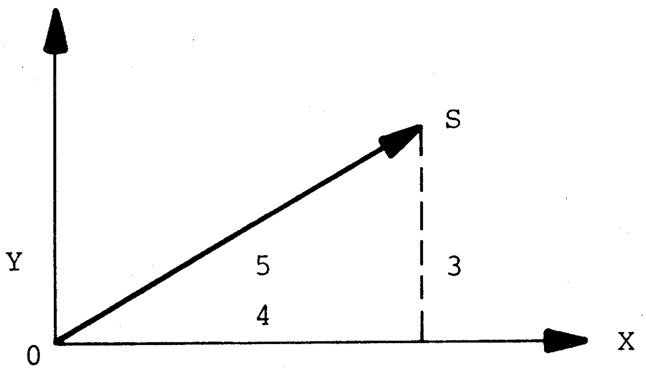


Diagram 4.1.5.4A

(Diagram 4.1.5.4A) whose length is 5 and whose component lengths are 3 and 4. Assume also that the maximum rapid traverse feedrate for each axis is 100 IPM. If we compute the feedrate command for the rapid traverse path S, we get:

$$F_{COM} = \frac{D * F_{IPM}}{S}$$

$$F_{COM} = \frac{10 * 100}{5} = 200;$$

(See Section 4.1.1.1).

This is an acceptable feedrate command which maintains the programmed feedrate of 100 IPM; but, as will be seen, we can get a yet higher value.

4.1.5.4 RAPID TRAVERSE OPTIMIZATION (cont'd)

Using the resultant speeds on each axis we can compute:

$$X \text{ rapid traverse} = \frac{3}{5} * 100 = 60 \text{ IPM;}$$

$$Y \text{ rapid traverse} = \frac{4}{5} * 100 = 80 \text{ IPM.}$$

We can get greater speeds if we select the optimum ratio value of the component path length to maximum rapid traverse feedrate for the component's axis, that is, we select the maximum of the three values:

$$\frac{\Delta x}{\text{option } 42'} \qquad \frac{\Delta y}{\text{option } 43'} \qquad \frac{\Delta z}{\text{option } 44'}$$

In our example these ratios are $\frac{3}{100}$, $\frac{4}{100}$; the maximum is $\frac{4}{100}$.

The feedrate command relation for rapid traverse is now

$$F_{COM} = \frac{D}{M},$$

where R_M is the maximum of the ratios. In our example,

$$F_{COM} = \frac{10 * 100}{4} = 250.$$

This value is higher than the previously calculated F .
Using this value, the resultant feedrate over S is then:

$$F_{IPM} = \frac{F_{COM}}{D} * 5 = \frac{250*5}{10} = 125 \text{ IPM.}$$

The resultant speeds on each axis are now:

$$X \text{ rapid traverse} = \frac{3}{5} * 125 = 75.$$

$$Y \text{ rapid traverse} = \frac{4}{5} * 125 = 100.$$

Neither axis traverse exceeds the maximum, but each traverse attains its optimum value.

4.1.5.4 RAPID TRAVERSE OPTIMIZATION (cont'd)

If the rapid traverse occurs in a low range, option 39 is used for all axes, thereby replacing options 42, 43, and 44.

This rapid traverse optimization is done in subroutine CONTUR.

Option 36 must be set to 0 to call for this optimizing sequence. NC machines which have an inadequate hydraulic power supply to move each axis at it's maximum feedrate cannot use this optimizing feature.

4.2 MULTIAXIS TRANSFORM CLASSES

The APT system passes multiaxis information on to the postprocessor through the CL tape. Cutter location data for the programmed part are in the form of algebraic points (x, y, z) along the cutter path, and direction cosines (i, j, k) of the tool axis. Before the postprocessor can properly process data for control output, it must first transform the linear motions into the nonlinear motions of the multiaxis machine. In general, the nonlinearity is due to one or more rotary axes which alter the position value of at least one of the linear axes. Hence, the postprocessor's geometric function is two-fold: one, to relocate the affected linear axis point as a function of the nonlinear motion; and, two, to transform the direction cosine data into the angular results required for rotary motion.

According to the NAS 938 description of standard machines there are approximately twenty existing types of machines which could be classed as multiaxis machines. This includes machines which have at least one translatory axis, one rotary axis, or both translatory and rotary axes. A translatory axis is defined as a secondary or tertiary axis operating parallel to one of the primary axes. Machines which have ancillary translatory axes are not normally considered part of the multiaxis problem. Specifically, only those machines which have one or more rotary motions about the primary axes are considered.

The following notation is used for descriptive purposes in order to differentiate between the various types of multiaxis machines: I(P,T,R), where I is the total number of tape controlled axes, P is the number of primary axes, T the number of translatory axes, and R the number of rotary axes about primary axes. Thus, a five axis machine 5(3,2,0) is easily discernible from 5(3,0,2) as being a type machine not considered to be a multiaxis machine. An extension of the notation gives other axis information; thus,9(3,2-1,2-1) refers to a 9 axis machine with 3 primary axes, 2 secondary and 1 tertiary translatory axes, and 2 rotary and 1 secondary rotary axes. Multiple spindles are indicated by a final number outside the parentheses, as 9(3,2-1,2-1)2 to indicate two heads.

4.2 MULTIAXIS TRANSFORM CLASSES (cont'd)

In general the GECENT III postprocessor can easily handle multiaxis machines of type 4(3,0,1) and 5(3,0,2) which, happily, are the most common configurations. However, the postprocessor can also handle machines which have translatory axes but not as a multiaxis move. If, for a 6-axis machine 6(3,1,2), the translatory axis is independant of the other five axes, i.e., is essentially a positioning motion, then the postprocessor is applicable. This implies that after the five axis slides have been set for some given point X, Y, Z, A, B, that the translatory axis is then moved. With this definition the multiaxis GECENT postprocessor is capable of handling 13 of the 20 multiaxis types. This includes types 5(3,0,2), 5(2,0,3) 4(3,0,1), 5(2,1,2) 5(3,1,1), 6(3,1,2), and 4(3,1,0). Each of the types discussed here are one of these.

The most basic feature of a multiaxis postprocessor is its transformation equations for converting the data from the part program coordinate system to the machine coordinate system. Since there are so many possible configurations, it is not economically possible to generalize a sequence for all possible combinations. The postprocessor uses one of several defined configurations during any particular run. This method, in a generalized sense, processes a multiaxis motion which is yet particularized to a given machine geometry.

Briefly stated, the method involves a classification of the geometric transform equations of existing multiaxis machines, to program these equations and identify them by their class, and to allow an option value to select the class for the given machine tool. For example, if a multiaxis machine has a rotating and tilting table and an orthogonal system, it would have a Class 3 geometric configuration. The Machine Subroutine merely assigns the appropriate option values for the class, and the CL data is transformed according to the relations given for Class 3. Option 116 specifies the geometry class.

Each class of equations is programmed in its own subroutine, and usually involves no more than ten equations. The number of classes, however, will vary. As new machine configurations occur, their classes are added to the postprocessor. Thus, there is no limitation to the adaptability of the postprocessor, and as new classes are added, the overall generality of the postprocessor is enhanced.

4.2 MULTIAXIS TRANSFORM CLASSES (cont'd)

Present multiaxis machine structures do not vary too much, and the same can probably be said for future machines. It is probable, therefore, that the number of classes will be no more than twelve, which is the approximate number of logical combinations of table and tool rotating and/or tilting machines. There will be special cases which can increase the number. In any event, a large number of classes can be handled quite adequately in the postprocessor.

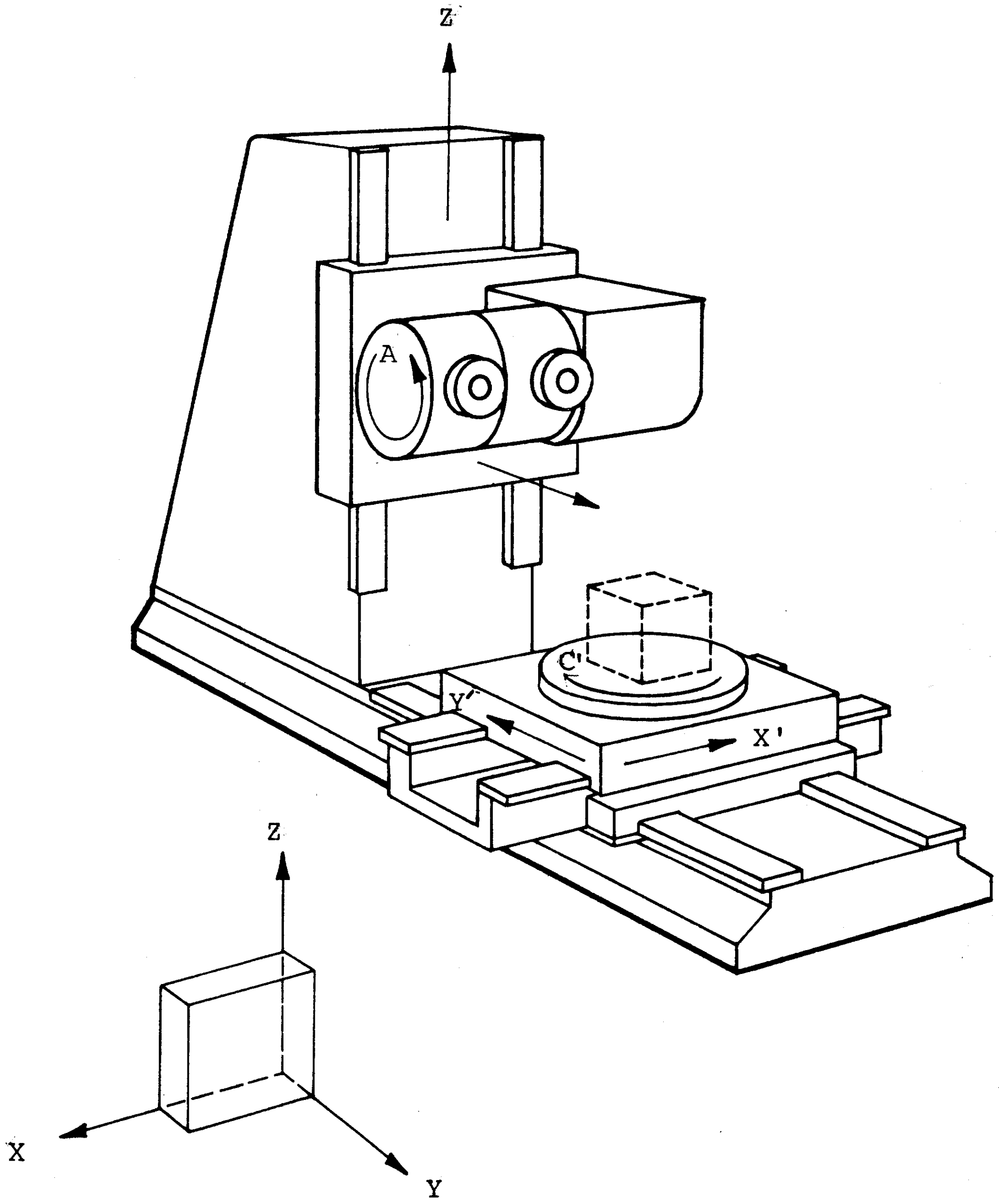
Each of the various classes of multiaxis machines defined in this section are illustrated by a diagramatic sketch which gives the basic relationships of the linear and rotary axes. The sketch is not meant to typify any particular NC machine.

The direct and inverse transforms are given for each class. The notation X Y Z ABC refer to the machine coordinates, whereas x y z ijk refer to the part coordinates. ABC are the machine rotary motions, and ijk are the backward directed direction cosines of the tool.

Options 100 through 105 are used as the input source for equation constants which may be needed by the transforms. These constants are identified and given in the Machine Subroutine.

4.2.1 CLASS 1 5(3,0,2)

This class is for five axis machine with a rotary table about the Z axis and a rotary head about the X axis.



Class 1

Diagram 4.2.1A

4.2.1 CLASS 1 5(3,0,2) (cont'd)

Direct Transforms: Part to Machine Coordinates

$$X = x \frac{j}{\sqrt{1 - k^2}} - y \frac{i}{\sqrt{1 - k^2}}$$

$$Y = x \frac{i}{\sqrt{1 - k^2}} + y \frac{j}{\sqrt{1 - k^2}} + (R+T_L)\sqrt{1+k^2}$$

$$Z = z + (R+T_L)k$$

$$A = \tan^{-1} \frac{k}{\sqrt{1 - k^2}} \text{ (Head)}$$

$$C = \tan^{-1} \frac{j}{i} \text{ (Table)}$$

Inverse Transforms: Machine to Part Coordinates

$$x = X \sin C + (Y - (R+T_L) \cos A) \cos C$$

$$y = (Y - (R+T_L) \cos A) \sin C - X \cos C$$

$$z = Z - (R+T_L) \sin A$$

$$i = \cos A \cos C$$

$$j = \cos A \sin C$$

$$k = \sin A$$

Definition of Terms

x y z	The part coordinate plus its respective TRANS values
X Y Z	The machine coordinates
A	Head rotation angle in the machine coordinate system
C	Table rotation angle in the machine coordinate system
ijk	The direction cosines of the tool axis

4.2.1 CLASS 1 5(3,0,2) (cont'd)

Definition of Terms (cont'd)

T_L Tool length in inches

R Distance in inches between the rotary head axis and spindle face

The value of R is a function of the head and gripper; for example:

	Small Gripper	Large Gripper
Head 1	R = 4	R = 9.5
Head 2	R = 3	R = 8.5

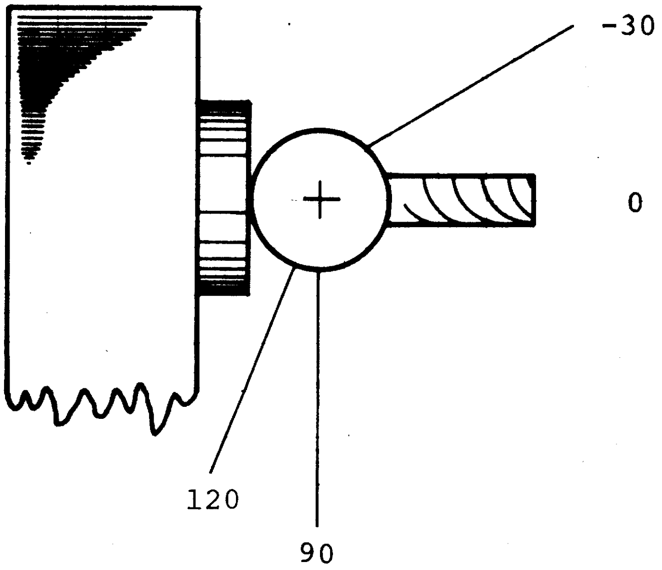


Diagram 4.2.1B

4.2.1 CLASS 1 5(3,0,2) (cont'd)

An indeterminate value for the table rotation occurs when the tool is perpendicular to the table since it cannot be clearly defined as to which way the rotation should go.

Thus, when k = 1, A is indeterminate. For this case, A is made 90 degrees and C is given the same value as at the previous point. Then,

$$X = x \sin C - y \cos C$$

$$Y = x \cos C + y \sin C$$

$$Z = z + R + T_L$$

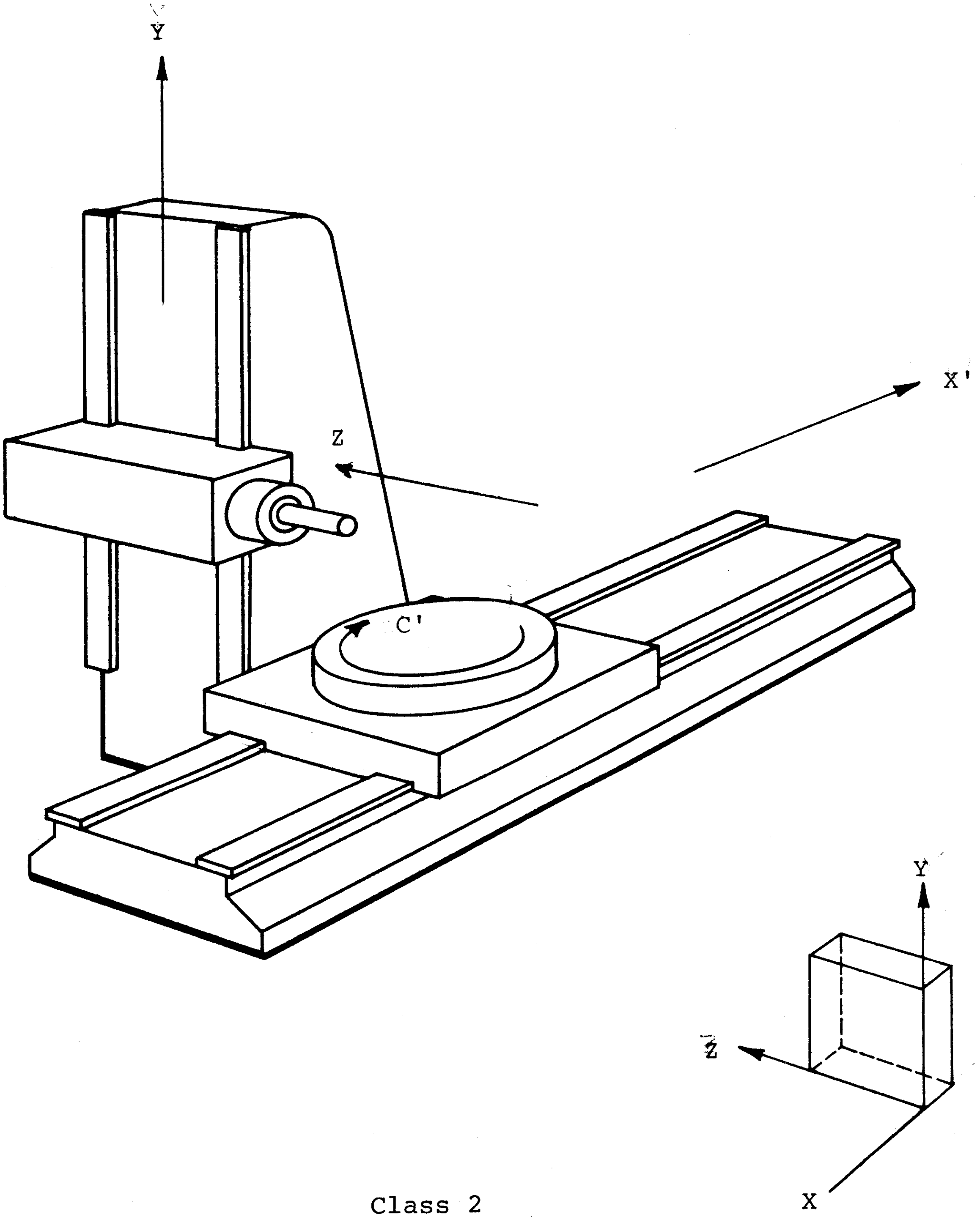
$$A = \pi/2$$

$$C = \text{Previous } C$$

Linearity testing is disregarded over this move.

4.2.2 CLASS 2 4(3,0,1)

This class is for a four axis machine with a rotary table about the Y axis.



Class 2

Diagram 4.2.2A

4.2.2 CLASS 2 4(3,0,1) (cont'd)

Direct Transforms: Part to Machine Coordinates

$X = x\,k - z\,i$

$Y = y$

$Z = x\,i + z\,k + (R + T_L)$

$B = \tan^{-1} \frac{i}{k}$

Inverse Transforms: Machine to Part Coordinates

$x = [Z - (R + T_L)] \sin B + X \cos B$

$y = Y$

$z = [Z - (R + T_L)] \cos B - X \sin B$

$i = \sin B$

$j = 0$

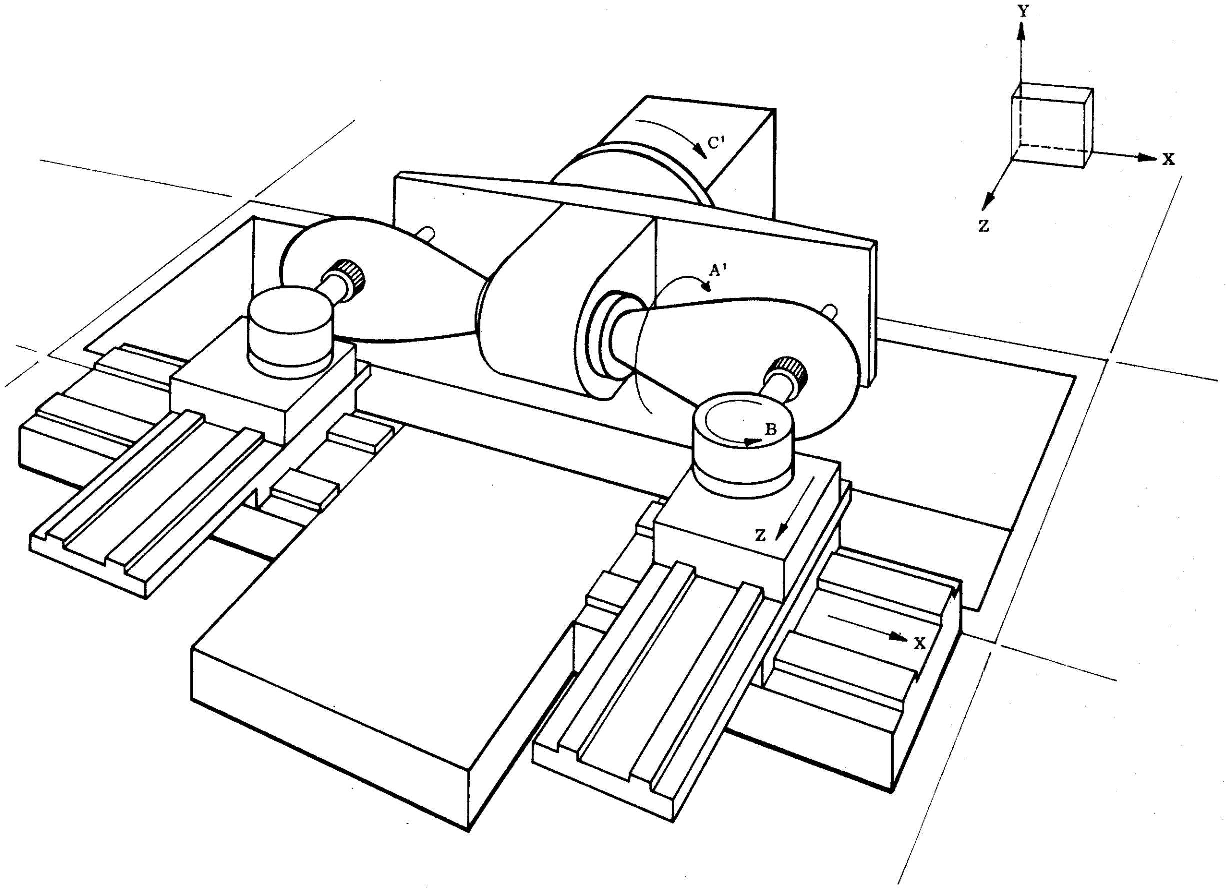
$k = \cos B$

Definition of Terms

xyz	The part coordinates plus its TRANS value
XYZ	The machine coordinates
B	Table rotation angle in the machine coordinate system
i,j,k	The direction cosines of the tool axis
T_L	Tool length in inches
R	Distance in inches between the rotary head axis and spindle face. The value of R has the standard value of 3 (option 102).

4.2.3 CLASS 3 5(2,0,3)

This class is for a five-axis machine which has two linear and three rotary motions.



Class 3

Diagram 4.2.3A

4.2.3 CLASS 3 5(2,0,3) (cont'd)

Direct Transforms - Part to Machine Coordinates

$$X_C = x + iT_L$$

$$Y_C = y + jT_L$$

$$Z_C = z + kT_L$$

$$H = \sqrt{X_C^2 + Y_C^2}$$

$$i' = (iX_C + jY_C)/H$$

$$j' = (jX_C - iY_C)/H$$

$$k' = k$$

$$G = \sqrt{(j')^2 + (k')^2}$$

$$X = H$$

$$Y = 0 \text{ (by definition)}$$

$$Z = Z_C$$

$$A = \tan^{-1} \frac{(-j' \text{ sign } k)}{|k|}$$

$$B = \tan^{-1} \frac{i'}{G \text{ sign } k}$$

$$C = \tan^{-1} \frac{(-XC)}{YC}$$

Inverse Transforms - Machine to Part Coordinates

$$i = - \sin C \sin B + \cos C \cos B \sin A$$

$$j = \cos C \sin B + \sin C \cos B \sin A$$

$$k = \cos B \cos A$$

$$x = -X \sin C - T_L i$$

$$y = X \cos C - T_L j$$

$$z = Z-T_L k$$

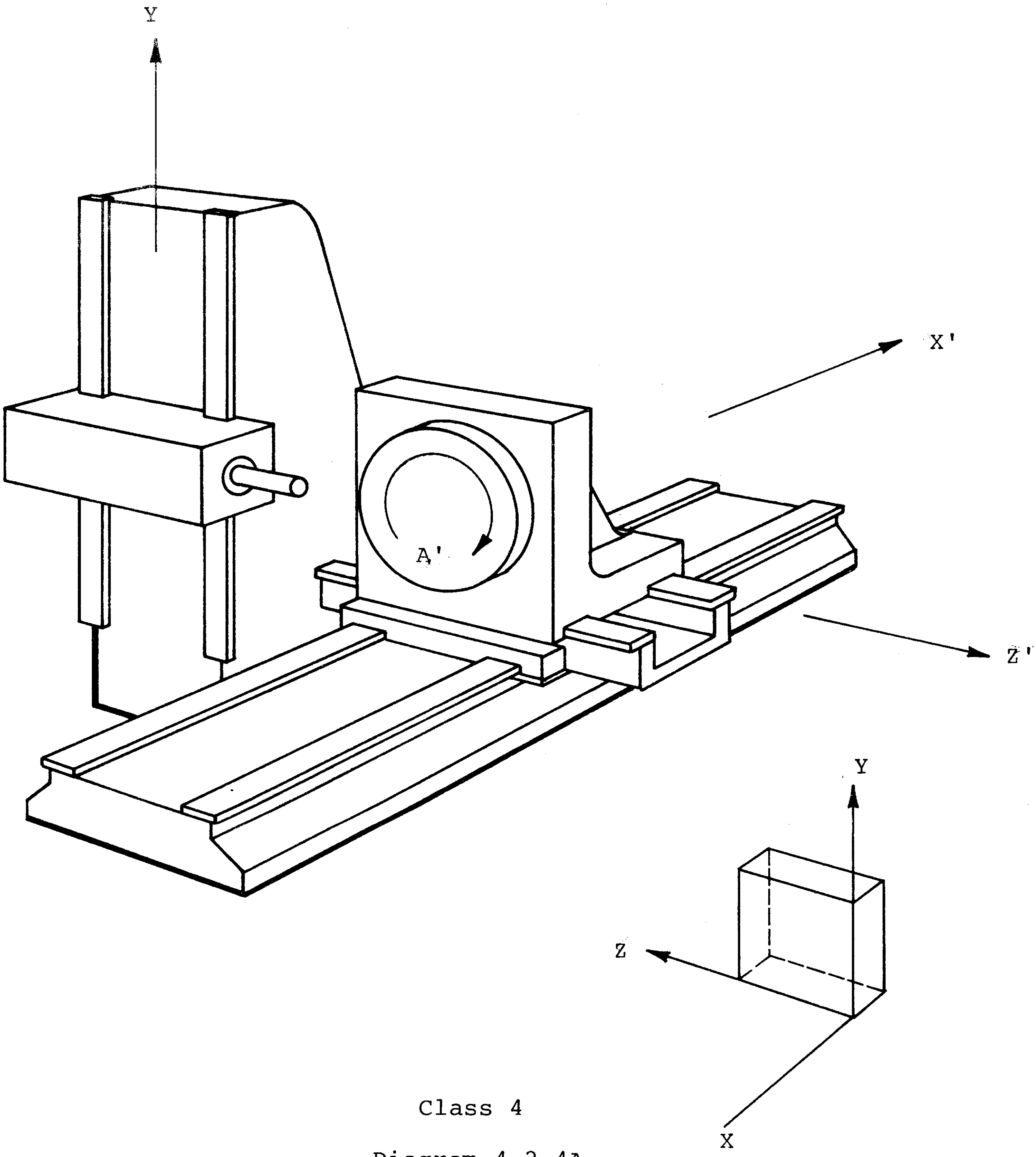
4.2.3 CLASS 3 5(2,0,3) (cont'd)

Definition of Terms

xyz	The part coordinates
XYZ	The machine coordinates
A	Head rotation angle in the machine Coordinate system
B	Head rotation angle in the machine Coordinate system
C	Table rotation angle in the machine coordinate system
ijk	The direction cosines of the tool axis
T_L	Tool length in inches

4.2.4 CLASS 4 4(3,0,1)

This is for a four-axis machine with a rotary table about the X axis.



Class 4

Diagram 4.2.4A

4.2.4 CLASS 4 4(3,0,1) (cont'd)

Direct Transforms Part to Machine Coordinates

$X = x$

$Y = yk - zj$

$Z = yj + zk$

$A = \tan^{-1} \frac{k}{j} \quad 0 \leq A \leq 2\pi$

Inverse Transforms Machine to Part Coordinates

$x = X$

$y = Y \sin A + Z \cos A$

$z = -Y \cos A + Z \sin A$

$j = \cos A$

$k = \sin A$

$i = 0$

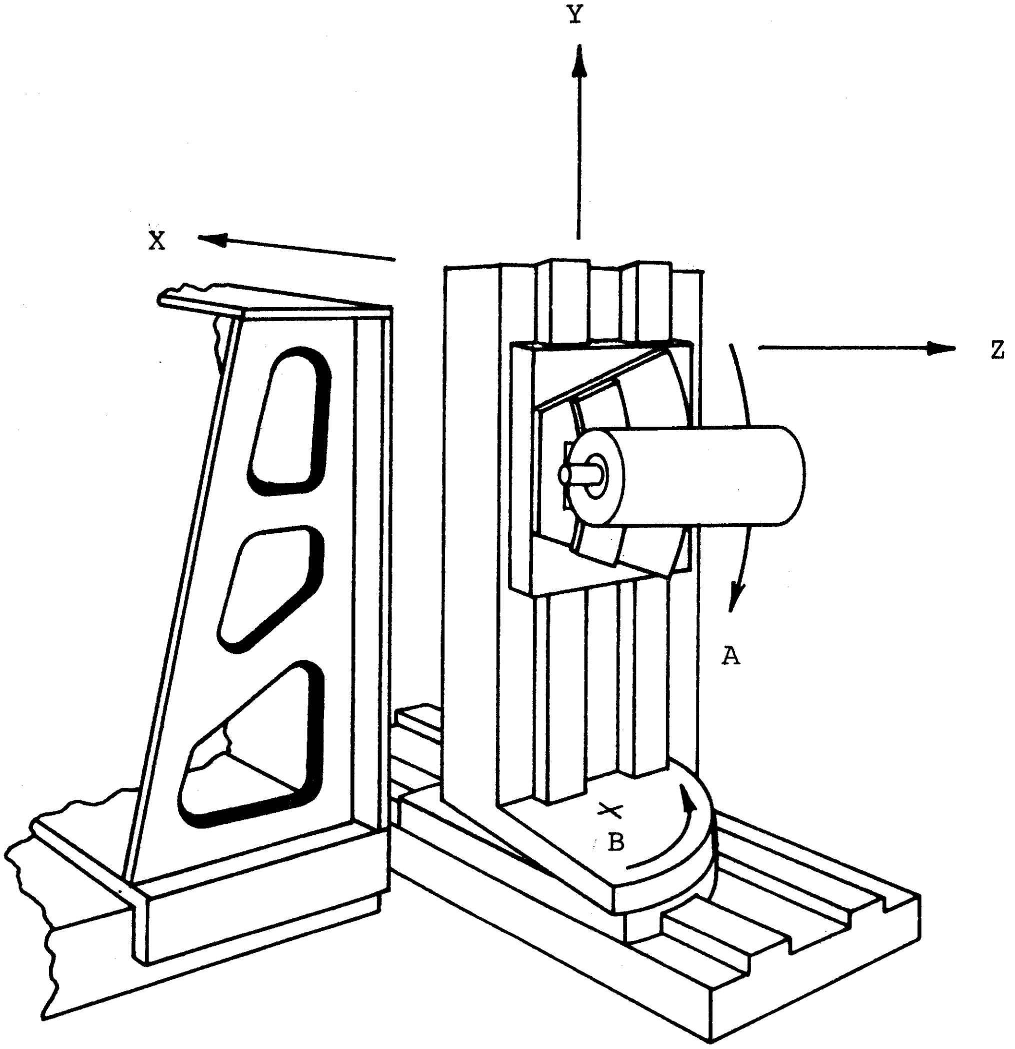
4.2.5 CLASS 5 4(2,0,2)

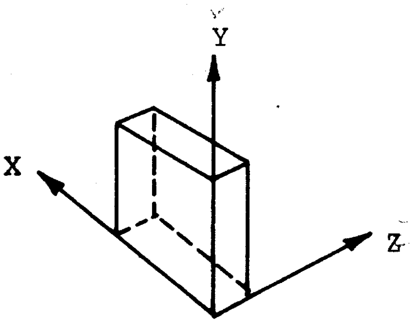
This class is essentially a dummy class subroutine for a filament winder. It satisfies the postprocessor necessity of providing a class subroutine within the multiaxis sequence. In effect, all that it does in to set DPRESM equal to DPRESP.

4.2.6 CLASS 6 5(3,0,2)

This class is for a five-axis machine with a rotating head about the X axis and a pivoting column about the Y axis.

Diagram 4.2.6A illustrates a single-head NC machine while Diagram 4.2.6B illustrates a multi-head machine.

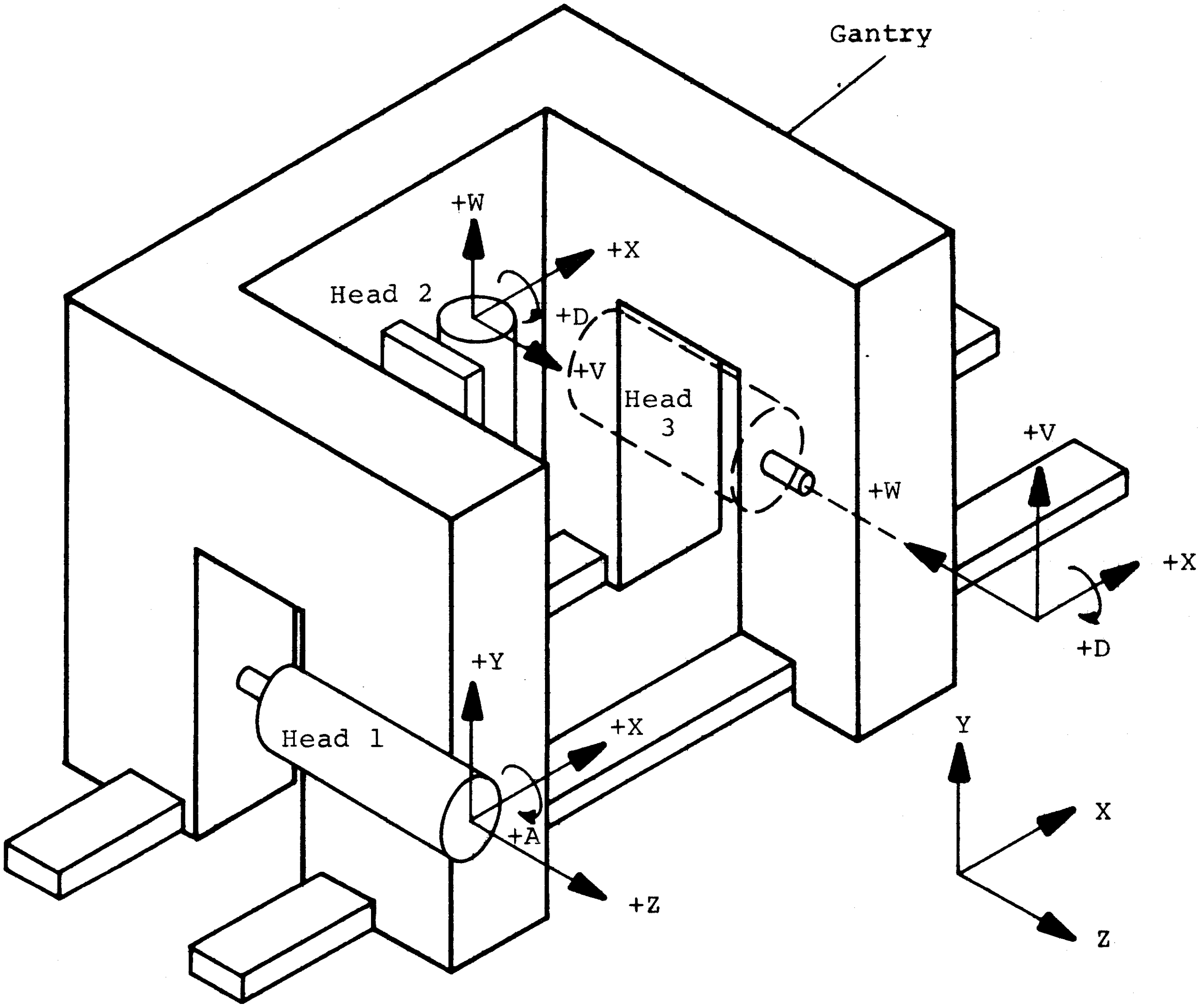




Class 6

Diagram 4.2.6A

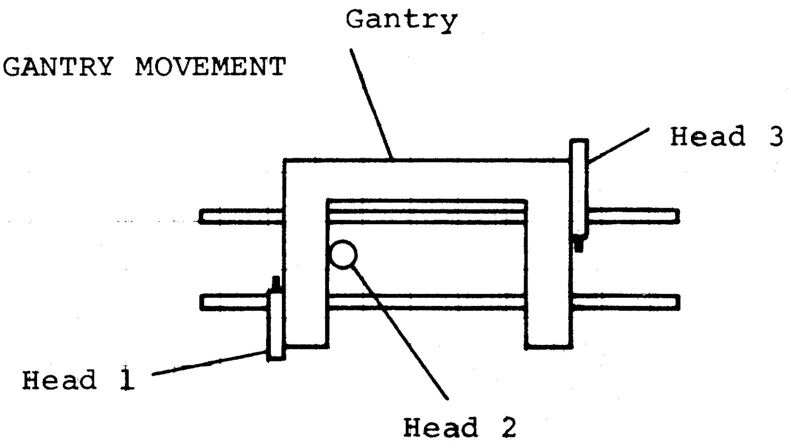
4.2.6 CLASS 6 5(3,0,2) (cont'd)



X MOTION IS GANTRY MOVEMENT

Axis Nomenclature

Head 1	+X, +Y, +Z, ±A
Head 2	+X, +V, +W, ±D
Head 3	+X, +V, +W, ±D



TOP VIEW

Class 6

Diagram 4.2.6B

4.2.6 CLASS 6 5(3,0,2) (cont'd)

Direct Transforms Part to Machine Coordinates

$$X = x + \frac{i}{k}(-Q-Z)$$

$$Y = y + \frac{j}{k}(-Q-Z)$$

$$Z = -Q - \frac{1}{K}(-Q-Z)$$

$$A = \tan^{-1} \frac{-j}{\sqrt{1-j^2}}$$

$$B = \tan^{-1} \frac{i}{K}$$

Inverse Transforms Machine to Part Coordinates

$$x = X + (Z+Q)i$$

$$y = Y + (Z+Q)j$$

$$z = Q + (Z+Q)k$$

$$i = \cos A \sin B$$

$$j = -\sin A$$

$$k = \cos A \cos B$$

Definition of Terms

x,y,z The Part Coordinates plus the TRANS value

X,Y,Z The Machine Coordinates

A Head Rotation Angle (Tilt)

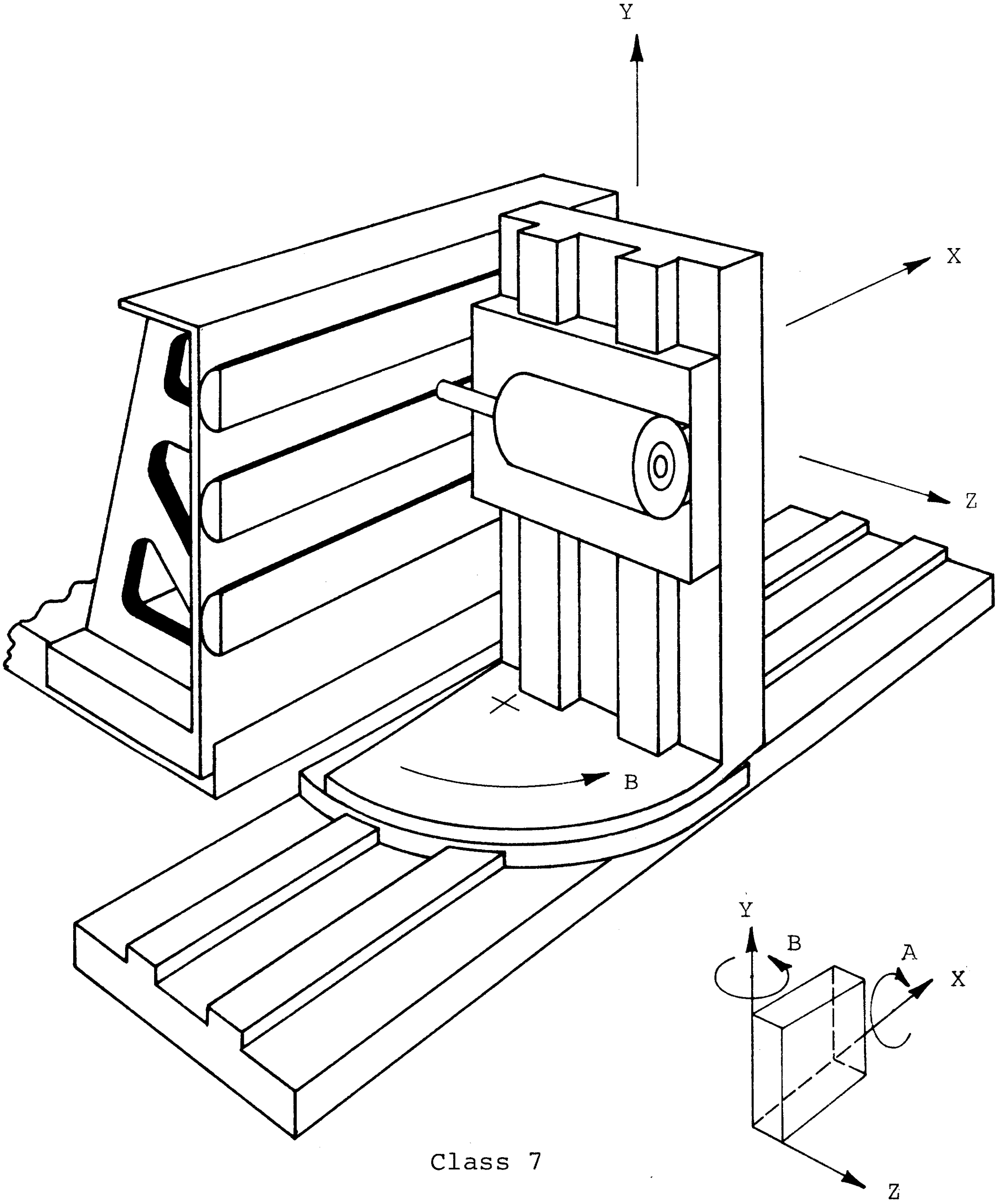
B Column Rotation (Swivel)

Q Directed distance from pivot plane to part origin on Z axis.

The pivot plane is the plane parallel to the XY plane and contains the column rotary axis and the head rotary axis.

4.2.7 CLASS 7 5(3,0,2)

This class is for a five axis machine with a rotating column about the Y axis and rotating "venetian blinds" about the X axis.



Class 7

Diagram 4.2.7A

4.2.7 CLASS 7 5(3,0,2) (cont'd)

In actuality the angle plate shown in Diagram 4.2.7A has three "venetian blinds", and the machine has a triple spindle. The A axes venetian blinds are work holding fixtures mounted on the angle plate.

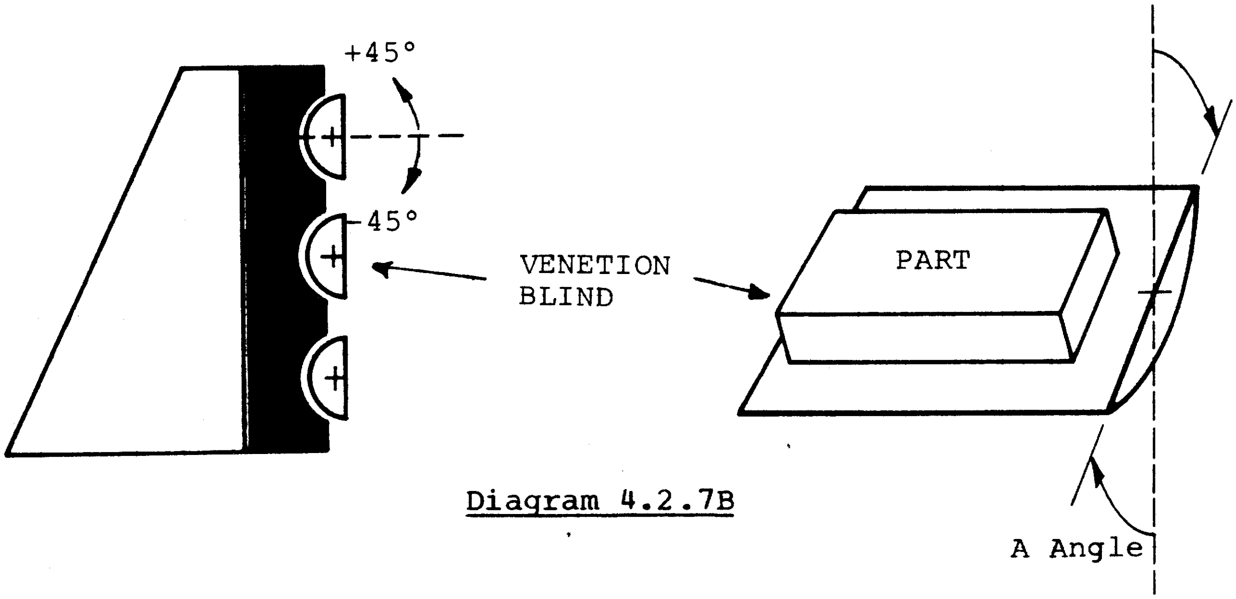


Diagram 4.2.7B

Direct Transforms Part to Machine Coordinates

$$X = x + i\frac{[(S1-Q)(\sqrt{j^2+k^2})] - [j(y+S2) + k(z+S1)]}{\sqrt{j^2+k^2}}$$

$$Y = -S2 + \frac{k(y+S2) - j(z+S1)}{\sqrt{j^2+k^2}}$$

$$Z = -Q - \frac{[(S1-Q)(\sqrt{j^2+k^2})] - [j(y+S2) + k(z+S1)]}{\sqrt{j^2+k^2}}$$

$$A = \tan^{-1}\left[-\frac{j}{k}\right]$$

$$B = \tan^{-1}\left[\frac{i}{j^2+k^2}\right]$$

Inverse Transforms: Machine to Part Coordinates

$$x = X + (Z + Q) \sin B$$

$$y = -S2 + [(Y + S2) \cos A)] - [(S1-Q) + (Z + Q \cos B) \sin A]$$

$$z = -S1 + [(Y + S2) \sin A] + [(S1 - Q) + (Z + Q \cos B) \cos A]$$

$$i = \sin B$$

$$j = -\sin B \cos B$$

$$k = \cos A \cos B$$

4.2.7 CLASS 7 (5,0,2) (cont'd)

Defination of Terms

x,y,z, Part Coordinates

X,Y,Z, Machine Coordinates

A Venetian Blinds Rotation Angle

B Column Rotation

Q Directed distance from pivot plane to part origin on Z axis

S1 Directed distance from axis of venetian blind to part origin on Z axis

S2 Directed distance from axis of venetion blind to part origin on Y axis

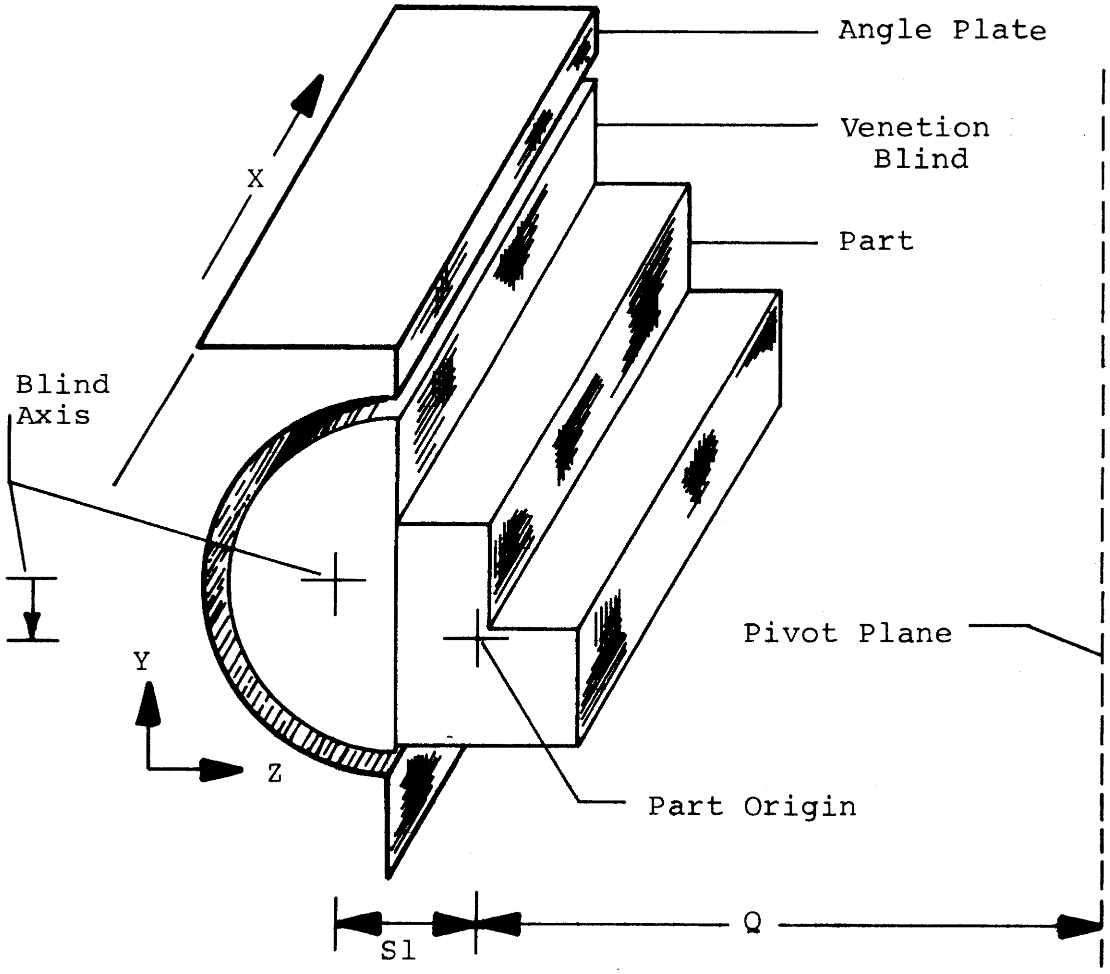
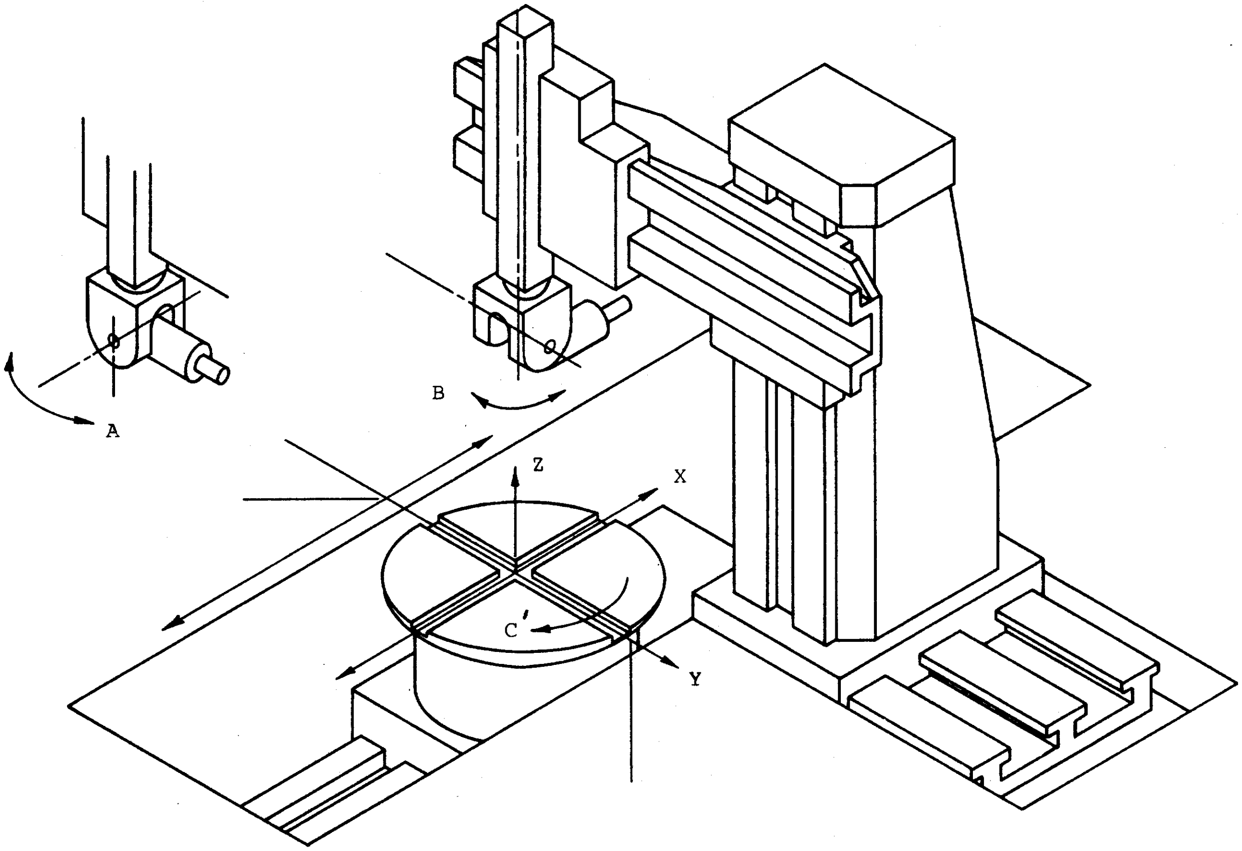


Diagram 2.4.7C

4.2.8 CLASS 8 5(3,0,2)

This class is for a five-axis machine with a rotating table about the Z axis and a rotating head about the Y axis.



Class 8

Diagram 4.2.8A

4.2.8 CLASS 8 5(3,0,2) (cont'd)

Direct Transforms

$$A = \tan^{-1} \frac{k}{\sqrt{i^2 + j^2}}$$

$$C = \tan^{-1} \left[\frac{i}{j}\right]$$

$$X = x * \cos C + y * \sin C$$

$$Y = (z - ZMO) * \mathrm{Sin}\ (\mathrm{Ram\ Angle}) + T_L * \mathrm{Cos}\ A$$
$$-x * \mathrm{Sin}\ C + y * \mathrm{Cos}\ C$$

$$Z = [z - T * (1-\sin A)]/\mathrm{Cos}\ (\mathrm{Ram\ Angle})$$

Inverse Transforms

$$i = - (\mathrm{Cos}\ A * \mathrm{Sin}\ A)$$

$$j = \mathrm{Cos}\ A * \mathrm{Cos}\ C$$

$$k = \mathrm{Sin}\ A$$

$$x = -TIVAL * \mathrm{Sin}\ C + X * \mathrm{Cos}\ C$$

$$y = TIVAL * \mathrm{Cos}\ C + X * \mathrm{Sin}\ C$$

$$z = Z * \mathrm{Cos}\ (\mathrm{Ram\ Angle}) + T_L * (1-\mathrm{Sin}\ A)$$

Notes: $TIVAL = Y-[(Z-ZMO) * \mathrm{Sin}\ (\mathrm{Ram\ Angle}) + T_L * \mathrm{Cos}\ A]$

$ZMO = ((\mathrm{Option}\ 125 + 50.3389) - T_L) / \mathrm{Cos}\ (\mathrm{Ram\ angle})$

4.2.9 CLASS 9 5(3,0,2)

This class is for a five axis machine with a rotating table about the Z axis and a rotating head about the X axis. Refer to Diagram 4.2.8A.

The NC machine illustrated for Class 8 is the same as for Class 9 except that the head is positioned so that it pivots about the Y axis.

Direct Transforms

$$A = \tan^{-1} \frac{k}{\sqrt{i^2 + j^2}}$$

$$C = \tan^{-1} (i/j)$$

$$X = x + \text{Cos } C + y * \text{Sin } C$$

$$Y = (z - ZMO) * \text{Sin (Ram Angle)} + T_L * \text{Cos } A - x * \text{Sin } C + y * \text{Cos } C$$

$$Z = [\, z - T_L * (1 - \text{Sin } A)\,]/\text{Cos (Ram Angle)}$$

Inverse Transforms

$$i = -\,(\text{Cos } A * \text{Sin } C)$$

$$j = \text{Cos } A * \text{Cos } C$$

$$k = \text{Sin } A$$

$$x = -TIVAL * \text{Sin } C + X * \text{Cos } C$$

$$y = TIVAL * \text{Cos } C + X * \text{Sin } C$$

$$z = Z * \text{Cos (Ram Angle)} + T_L * (1 - \text{Sin } A)$$

Notes: $TIVAL = Y - [\,(Z - ZMO) * \text{Sin(Ram Angle)} + T_L * \text{Cos } A)\,]$

$ZMO = [\,(\text{Option } 125 + 50.3389) - T_L)\,] / \text{Cos (Ram Angle)}$

4.2.8 CLASS 8 5(302) (cont'd)

Definition of Terms for Class 8 and Class 9

x,y,z The part coordinates plus the respective TRANS values

X,Y,Z, The machine coordinates

A Head rotation angle in the machine coordinate system.

C Table rotation angle in the machine coordinate system.

i,j,k The direction cosines of the tool axis.

T_L Tool length in inches

4.3 ACCELERATION-DECELERATION TESTING (A/D)

The acceleration-deceleration sequence of the GECENT III postprocessor reduces feedrates below the values specified by the part programmer in circumstances where maintaining the programmed rate would cause an excessive deviation of the tool center path from the commanded path. The need for such feedrate reduction is found in the vicinity of corners and small arcs, where the programmed feedrate would demand large accelerations from the machine servo drives and cause the path error to exceed the required tolerance.

In the interest of fast machining, it is desirable to interfere with the programmed feedrate as little as will suffice to hold the tolerance. The function of the postprocessor is similar to that of a racing driver in the Monaco Grand Prix. Although he might like to go at full throttle continuously, he necessarily must slow down at the corners to stay within the tolerances of the roadway. To save time, he brakes hard just ahead of a corner to get down to a safe cornering speed and then uses all the resources of his machine to get his speed back up to the value best suited for the next part of the path, whether it be wide open for a straightway, or some lower value for a curved path. His mental computer weighs knowledge of terrain and tire adhesion to tell him what speed he should maintain under each condition. The postprocessor does not do anything quite as spectacular as the racing driver. But similar principles are employed to the extent that the limiting speed for each condition is computed, and locations for required slowdowns are established just far enough ahead of critical points to permit the slowdown to be made in the space available.

The basis for the calculation of feedrates is an analysis of the performance of servo drives which expresses tool center maximum path error as a function of servo characteristics, path geometry, and feedrate. The relationship is then solved for feedrate in terms of the given tolerance, path geometry, and appropriate servo constants. Different kinds of paths have to be analyzed separately, and the use of more than one feedrate along the path must also be considered. Some of these details are described later.

In the discussion which follows, consideration is first given to the physical and theoretical implications of servo control reaction and effect. A complete description of the servo system of the Mark Century numerical control is given, and each dynamic effect is highlighted as to its cause and the theoretical method applied for resolution of the problem.

4.3 ACCELERATION - DECLERATION TESTING (A/D) (cont'd)

Later sections discuss the postprocessor programmed sequence for the theoretically derived relations and principles which constitute the GECENT III solution to A/D dynamic problems.

4.3.1 SERVO ANALYSIS OF A/D PROBLEM

By assuming that the servos on all axes of a three-dimensional contouring system have identical characteristics and are linear, some surprisingly simple relationships are found. The assumption of linearity can be justified by noting that when the postprocessor reduces feedrates to maintain tolerable path errors, a tendency of the servos toward saturation at large errors is reduced. Furthermore, all Mark Century servos are designed for a good steady-state linearity between position error and velocity over the full range of contouring velocities in order to obtain low path errors on slopes. The effects of friction and backlash, neglected in the analysis, are expected to be small for most machines. Note, however, that the servo constants used in the postprocessor are entered in a table of values for each machine (OPTAB) and can be changed as necessary if experience indicates. The philosophy regarding these constants is to fix them initially on a theroetical linear basis, yet retaining for the user the capability of modifying or adjusting them as his own experience warrants, simply by exchanging values of constants in the table.

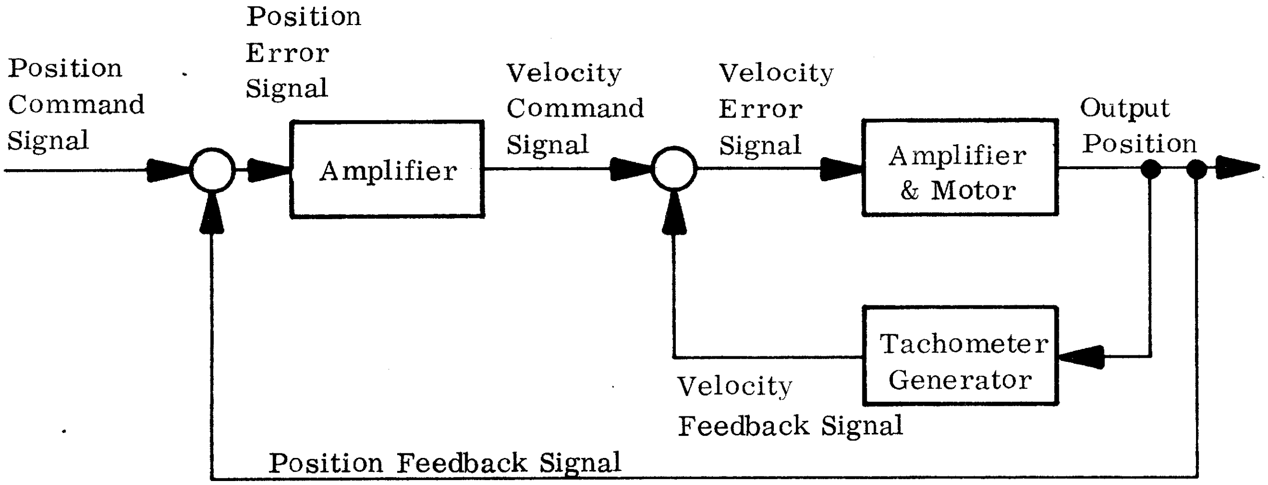
The assumption that the servos on all the machine axes have identical characteristics is attractive to the analyst because it greatly simplifies his results. For machines with drives all of a given type having similar horsepower rating, it is justifiable. For greatly different power ratings on different axes, transient responses may not be entirely similar, in which case, conservative constants corresponding to the drive with highest transient errors can be used. Even when horsepowers are different, Mark Century servos are designed with identical steady-state gains in order to obtain low path errors on slopes, and this characteristic, combined with the "naked system" type of servo used in most Mark Century controls, tends to insure that the different servos will have very similar response characteristics.

As background for discussion of tool center path errors, the essential characteristics of Mark Century servos will be described. These remarks apply equally to drives with DC motors controlled by thyratrons, silicon controlled rectifiers, or amplidynes, and to hydraulic drives employing either cylinder or hydraulic motors.

4.3.1 SERVO ANALYSIS OF A/D PROBLEM (cont'd)

In common with all servomechanisms, the Mark Century servo drives utilize a position-feedback signal which algebraically combined with the input position command signal to obtain a position error signal which actuates the motor to move in the direction tending to reduce the error signal toward zero. In addition to this so-called position feedback loop, the Mark Century servos are provided with an inner velocity loop employing a velocity feedback signal from a DC tachometer generator or other equivalent means. This signal is subtracted from an amplified version of the position error signal, and the resultant signal serves as a velocity command, or a velocity error signal which is amplified to actuate the motor. The use of the inner loop insures that the motor velocity is, to a high degree of accuracy, proportional to the position error signal, and in effect it improves the performance of the physical motor so that it behaves more nearly like an ideal motor whose velocity would be totally unaffected by machine inertia and friction, and by variations in characteristics, with temperature and aging. The Mark Century servo with its inner velocity loop is shown in block diagram form in Diagram 4.3.1A.

To the extent that stability considerations permit the gain around the velocity loop to be made high and the velocity error signal to be kept small, the velocity feedback signal is maintained equal to the velocity command signal; and hence, it is proportional to the position error signal. For linear motions, the proportionality constant is designed typically in the range of 0.5 to 3 inches per minute for .001 inch of position error. At a velocity of 60 IPM, a Mark Century servo might have .120" to .020" of position error or velocity lag. It should be noted that this lag does not cause an equal error in the workpiece. On the contrary, at constant velocity



Mark Century Servo Drive

Diagram 4.3.1A

4.3.1 SERVO ANALYSIS OF A/D PROBLEM (cont'd)

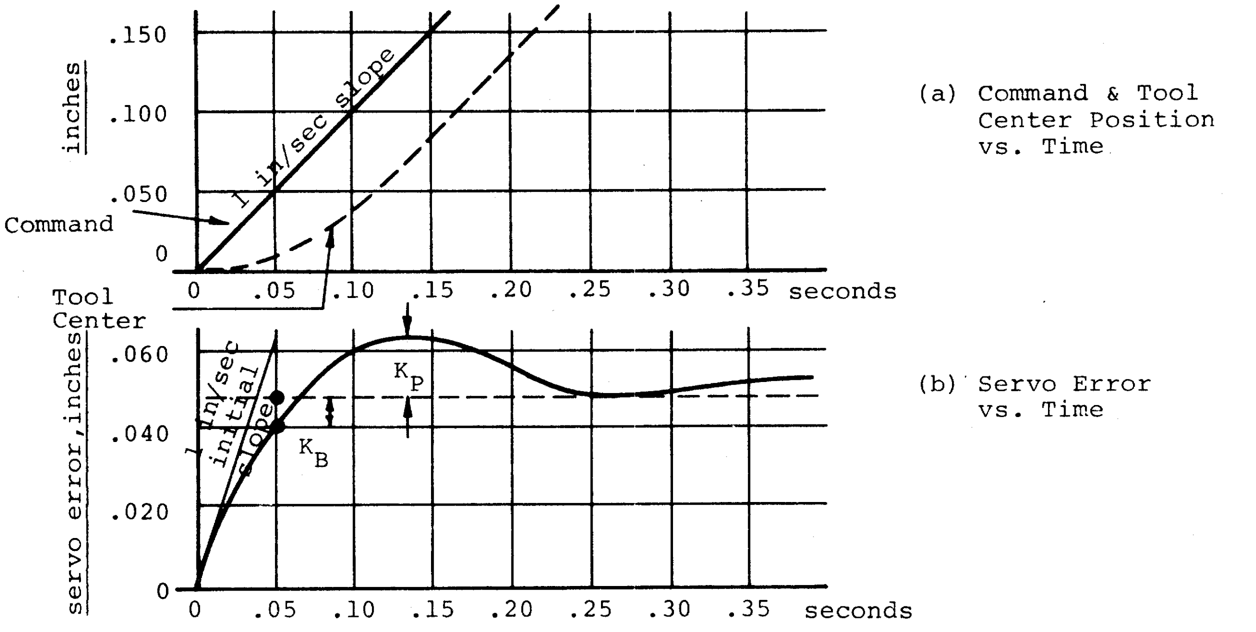
on sloping paths, the error is theoretically zero. The tool center trails along behind the commanded point, on a path which is exactly on the commanded path, so long as the servo gains are exactly equal. If, contrary to the design objective, the gain of one servo only should change by 1%, for example, it is true that a small error is introduced. The maximum path error in such a case, on a 45° slope, theoretically becomes 1/2 of 1% of the normal velocity lag of one servo operating at the feedrate commanded along the slope. For a 1 IPM/.001" system running at 60 IPM on a 45° slope, the error associated with a 1% gain change in one servo thus is 1/2% of .060", or .0003", and is proportionally smaller at lower feedrates. Satisfactory constancy of gain dependent only on stable quantities such as the tachometer volts/rpm characteristic and resistor values, and essentially independent of unstable quantities such as friction or transistor gain.

Although servos of this kind are sometimes characterized by the term "low gain" to distinguish them from servos which operate at full speed with much smaller position errors, it should be clear that the low velocity gain characteristic is achieved deliberately by the use of velocity feedback around the velocity error amplifier for the sake of the benefits which such feedback bestows and not because of any skimping on amplifiers. For moving against static friction in response to a small command, the entire gain of both amplifiers in Diagram 4.3.1A is effective, and a quantitative analysis of a typical hydraulic system shows that only .000010 inch position error is required to develop full system torque.

From the standpoint of A/D routines, the most significant characteristic of the velocity loop system is its ability to follow large changes in commanded velocity without excessive overshoot. In general terms, because the tool center lags behind the moving commanded point, the tool has adequate space in which to stop without significant overshoot.

Diagrams 4.3.1B and 4.3.1C will clarify this. In Diagram 4.3.1B(a), the solid line plotted against time shows an assumed command to a Y axis servo drive. The command is initially stationary, and then suddenly starts to move upward at 60 IPM, or 1 inch per second. The dotted line shows the tool center position, obtained by assuming that the servo has a gain of 1.2 IPM/.001" (20 in/sec per inch of error) and a transient response as indicated in Diagram 4.3.1B(b), where the servo error (the difference between input command and servo position) is plotted against time with an expanded vertical scale.

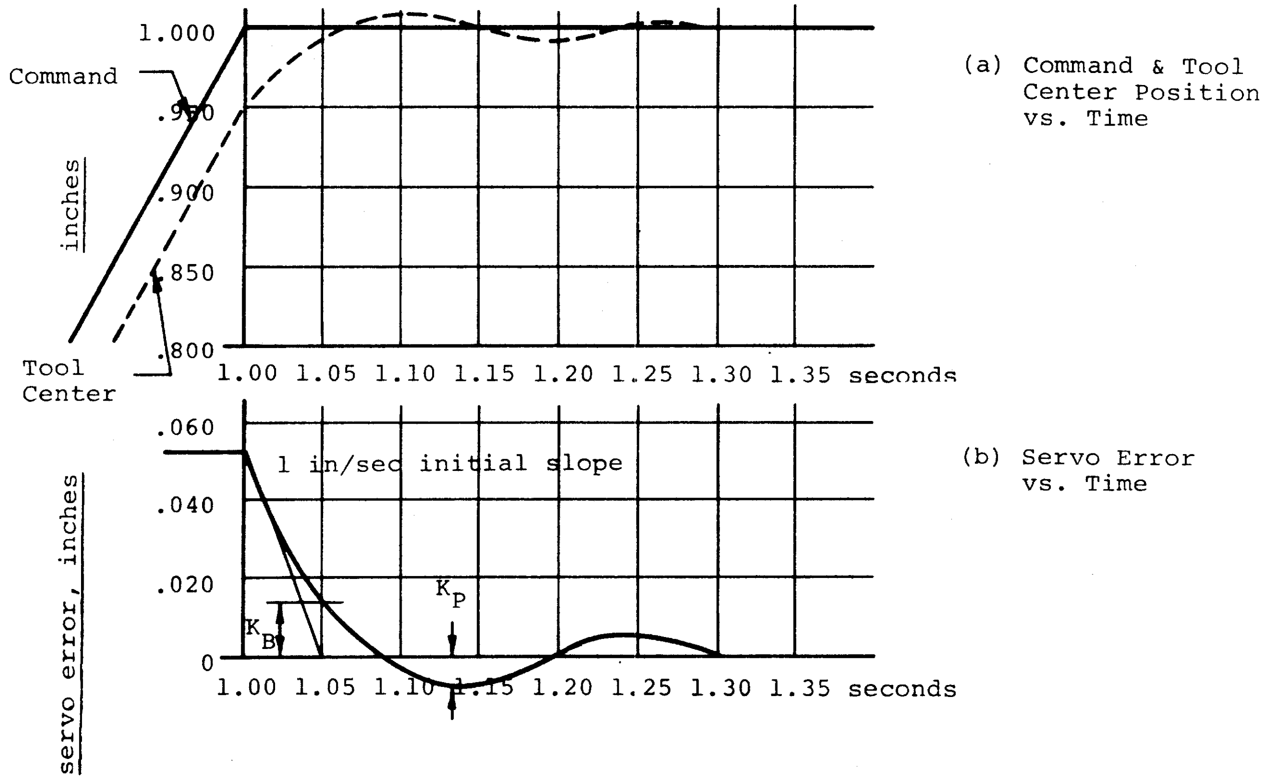
4.3.1 SERVO ANALYSIS OF A/D PROBLEM (cont'd)



(a) Command & Tool Center Position vs. Time

(b) Servo Error vs. Time

Diagram 4.3.1B 0 to 60 IPM



(a) Command & Tool Center Position vs. Time

(b) Servo Error vs. Time

Diagram 4.3.1C 60 to 0 IPM

4.3.1 SERVO ANALYSIS OF A/D PROBLEM (cont'd)

After steady conditions are reached, the tool center moves at the same velocity as the command but trails by .050" in distance, or .05 second in time. In Diagram 4.3.1C(a), the command is assumed to stop suddenly after 1 inch of motion, and the tool center stops with a small overshoot as shown. The corresponding plot of error is shown in Diagram 4.3.1C(b). With Coulomb friction neglected, this curve of error is theoretically identical to that shown in Diagram 4.3.1B(b) except that it is inverted and displaced upward. It is inverted because the step change in commanded velocity is negative in Diagram 4.3.1C(a) compared with the positive change of Diagram 4.3.1B(a) and the plot is displaced because of the initial .050" steady error. Step changes in velocity of different magnitudes would produce similar plots of error versus time, larger or smaller in magnitude in proportion to the magnitude of the command velocity change, and displaying an initial value proportional to the velocity existing before the command change. Plotted as it is for the particular case of 1 inch per second feedrate, Diagram 4.3.1B(b) can serve as a generally useful response curve which, after appropriate scaling and shifting, describes the servo error following any step change of command velocity, when steady-state conditions are assumed to exist before the change. The shape of Diagram 4.3.1B(b) depends on servo adjustments and normally shows an overshoot in the range of 0 to 20% beyond the final value

4.3.1.1 PATH ERRORS ON CORNERS FORMED BY TWO CONNECTING STRAIGHT LINES

While contouring along a straight line 3-dimensional path at constant velocity, the command to each axis servo moves at a constant velocity which is the appropriate component of the space vector velocity. A step change to a new feedrate, or a sudden change in direction of the path, or both, will in general cause a step change in the velocity of the command to each axis. Each servo position could be plotted by first plotting the error from Diagram 4.3.1B(b) using an appropriate scale factor and initial value in each case, and subtracting this error curve from the known command. Geometrically combining the servo position curves would yield a plot of tool center path, and its deviation from the commanded path could be measured. A simple 2-dimensional example will make this more clear. Suppose the commanded path consists of a 90° turn with the two lines forming the angle being parallel with the machine axes as shown by the solid lines in Diagram 4.3.1D. Suppose futher that the tangential feedrate is a constant 60 IPM. The Y-axis servo is initially at rest, and its command calls for a step change to 60 IPM, as already sketched in Diagram 4.3.1B (a). The X - axis servo is

4.3.1.1 PATH ERRORS ON CORNERS FORMED BY TWO CONNECTING STRAIGHT LINES (cont'd)

simultaneously commanded to stop from 60 IPM, the condition that was plotted in Diagram 4.3.1C(a). If the servo positions are transferred point by point to an X-Y plot, the dotted tool center path of Diagram 4.3.1D is obtained. The small numbers show the time in seconds after the command leaves the corner. At .050 seconds, the tool center path crosses the 45° bisector of the angle at point B, and is distant from each line of the command path at this moment by the amount K_B indicated on Diagrams 4.3.1B(b) and 4.3.1C(b). The undershoot path error E_u measured along the bisector is the distance BO = 1.414K_B . At about .125 seconds the path exhibits its maximum overshoot. The overshoot error E_O in Diagram 4.3.1D is clearly the overshoot of the X servo indicated as K_p in Diagram 4.3.1C(b) or 4.3.1B(b). K_B and K_p are servo constants stored in the postprocessor.

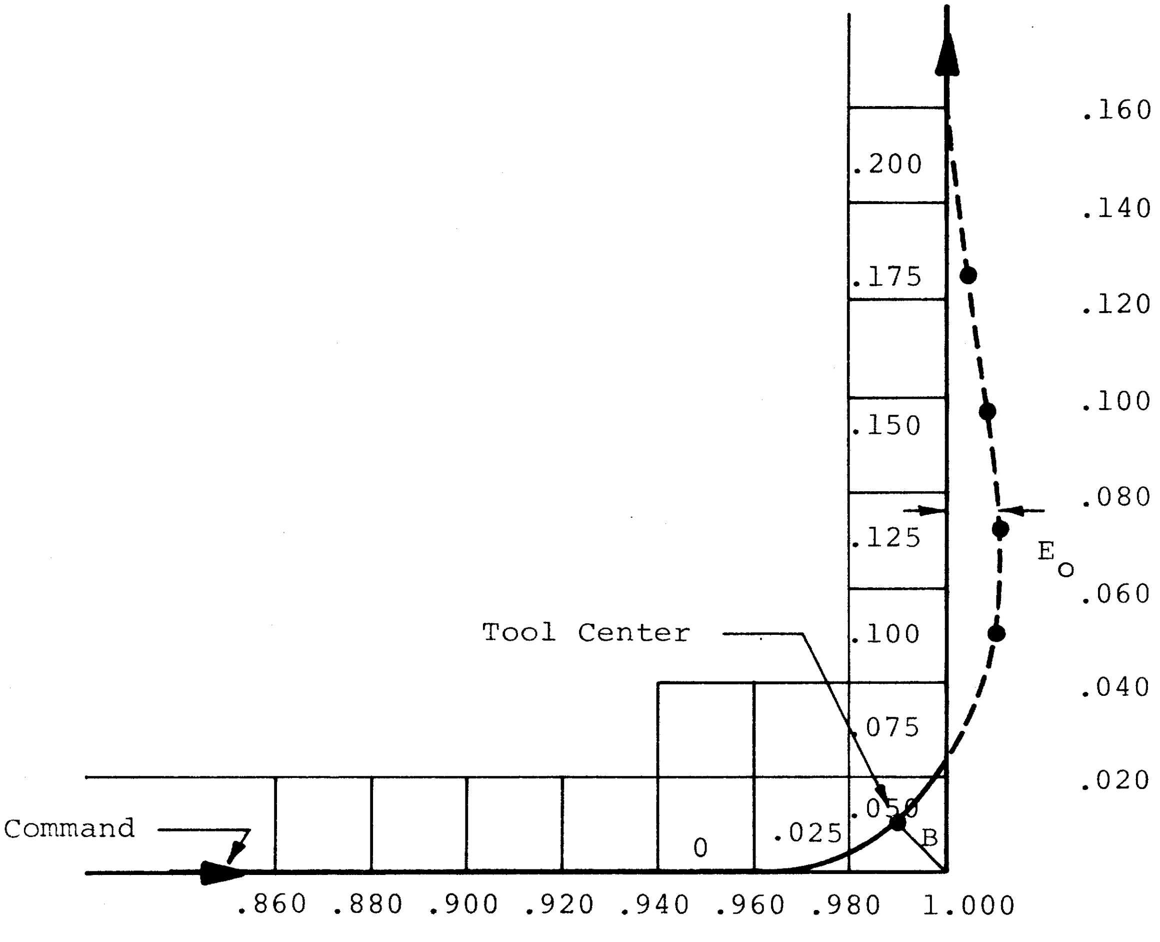


Diagram 4.3.1D

4.3.1.1 PATH ERRORS ON CORNERS FORMED BY TWO CONNECTING STRAIGHT LINES (cont'd)

E_u would be affected by changing the feedrate at the corner. An increase in the feedrate,for instance,would cause the tool center. path to be stretched in the vertical direction, the angle bisector would be crossed sooner, and point B would be further from 0. On the other hand, the overshoot of the X servo, E_o, is independent of commands to the Y servo and would be unchanged, even if a dwell were programmed at point 0.

A general analysis of the corner error for any angle of turn 0 and a general feedrate F_1 inches per minute yields the following results:

$$E_u = 2K_B\frac{F_1}{60}\sin\frac{\theta}{2} \quad \text{or} \quad F_1 = 60\frac{E_u}{2K_B \sin\frac{\theta}{2}} \qquad (1)$$

$$E_o = K_P\frac{F_1}{60}\sin\theta \quad \text{or} \quad F_1 = 60\frac{E_o}{K_P \sin\theta} \qquad (2)$$

It can be shown that these results are applicable regardless of the orientation of the path on the machine. That is, the servos will cooperate to produce the tool center path of Diagram 4.3.1D even if the commanded paths are not parallel to the machine axes. Although the individual servos will receive quite different commands for different orientations of the 90° corner, the path errors turn out to be identical, and the feedrate calculation is, therefore, made without regard to individual component velocity changes.

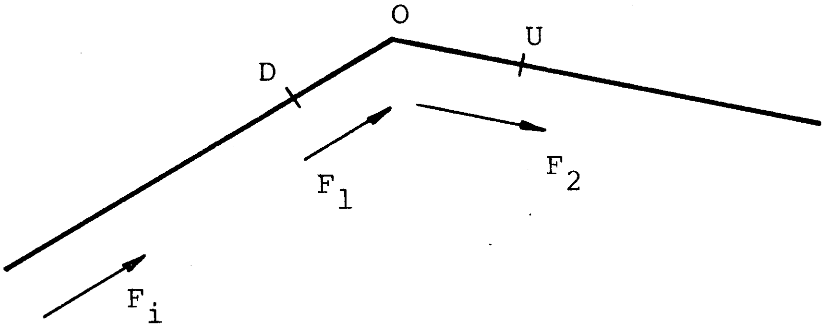
Equations (1) and (2) apply to the special case of a fixed feedrate F_1 for both approaching and leaving the corner. The postprocessor actually is not restricted to this special case, but uses a more elaborate equation for the departing feedrate F_2. It calculates the approaching feedrate F_1 from the allowable E_o, (which is unaffected by F_u) and then calculates F_2 from F_1 and the allowable E_u. In the case of well-damped servos with little or no overshoot, this procedure gives a fast approach to the corner, a short slow segment after the corner which holds the undershoot to the tolerance, and a prompt speed-up after the corner, if space permits.

4.3.1.2 LOCATION OF SLOWDOWN POINT

In Diagram 4.3.1E a slowdown point D where the feedrate is changed from its initial value F_i to a required lower approach value F_1 is located far enough ahead of the corner O to give the servos time to settle down between the disturbance at D and the corner O. For this purpose, $DO=T_sF_1$, in which T_s is the servo settling time. If F_1 were very small, however, DO might be calculated so small that the tool center could overshoot O in the process of slowing down at D from the higher feedrate F_i. A second tentative value of DO is therefore calculated from the expression

$$DO_2=k_{ao}K_PF_i,$$

and the larger of the two values selected for use as DO. The product K_PF_i gives the overshoot beyond D for a commanded stop at D, and the factor k_{ao}, in the range of 1.2 to 1.4 increases the result to allow for extra overshoot for values of F_1 greater than zero. For all but very small values of F_1, $DO=T_sF_1$. T_s is selected rather arbitrarily as the time for the error curve of Diagram 4.3.1B(b) to settle to within a few per cent of the final value, and depending on the damping, is normally in the range of 4 to 8 times $1/K_v$, where K_v is the velocity error coefficient, or gain in inches per second per inch of error.



Slowdown and Speedup Points

Diagram 4.3.1E

4.3.1.3 LOCATION OF SPEEDUP POINT

In Diagram 4.3.1E if F_2 is less than the programmed rate, the speedup point U is separated from O by the distance T_sF_2, to permit the servos to settle down after the disturbance at the corner. If the next corner is close, there may not be space for speeding up.

4.3.1.4 FEEDRATE LIMITATION ON ARCS

Because the Mark Century control provides circular interpolation, servo errors during operation on circular arcs must be considered. Contouring around a circle in a coordinate plane causes the two axis servos to be given sinusoidal input commands. The tool center path will be a circle which can be slightly oversize if the closed loop response of the servos exceeds unity at the frequency of operation ("Frequency" is measured in revolutions or radians per second of servo sinusoidal operation), or can be slightly undersize if the closed-loop response is less than unity. By equating the difference between the tool center circular path radius and the commanded radius to the tolerance, a limiting feedrate in IPM is determined from the expression,

$$F = 60\ K_C \sqrt{E * r} \qquad (3)$$

In equation (3), K_C is a constant ranging from 25 to a very large value for well-damped servos adjusted for flat response; E is the error tolerance in inches; and r is the radius of the circle in inches. This expression is derived in Section 4.3.2 and will normally set a practical limit on feedrate only for very small circles. To allow for transient peaks in error experienced when changing feedrate on an arc, or proceeding between tangent arcs or between a straight line and a tangent arc, appropriate slowdown factors based on computer tests of various situations are applied to equation (3).

4.3.1.5 NON-TANGENT ARCS

Non-tangent intersections between arcs and between straight lines and arcs are handled like corners formed by straight lines. On the basis that for all but the very shortest radius arcs, the region of cornering error will be so small in relation to the curvature of the arcs that they will be the practical equivalent of straight lines.

4.3.2 DERIVATIONS OF FORMULAE

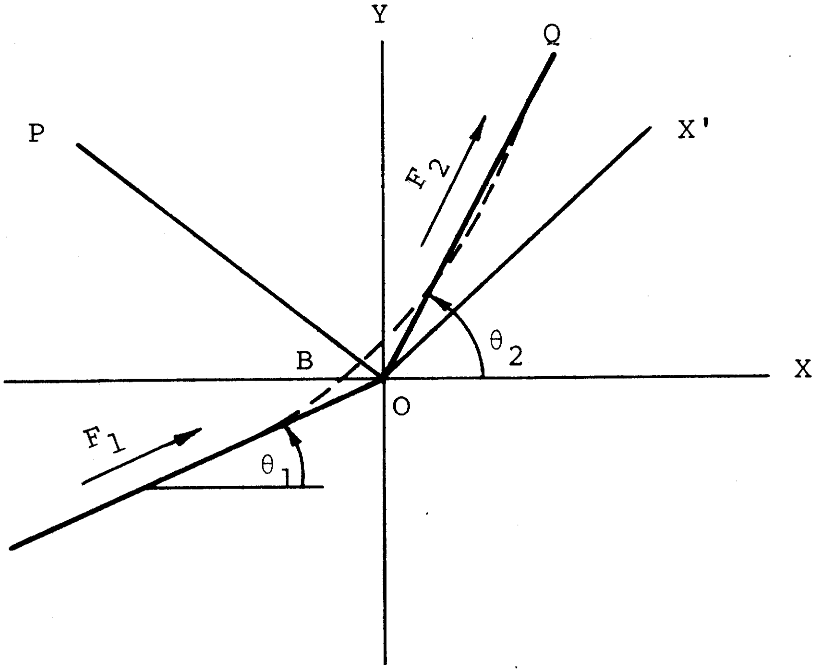
4.3.2.1 UNDERSHOOT ERROR ON CORNER WHEN $F_2 \neq F_1$

Referring to the illustration in Diagram 4.3.2A it can be shown that the tool center path as a function of time is given by

$$X = (F_2 \cos \theta_2)t-(F_1/K_V) \cos \theta_1-(F_2 \cos \theta_2-F_1 \cos \theta_1)f(t) \qquad (4)$$

$$Y = (F_2 \sin \theta_2)t-(F_1/K_V) \sin \theta_1-(F_2 \sin \theta_2-F_1 \sin \theta_1)f(t) \qquad (5)$$

in which f(t) is the response curve of Diagram 4.3.1B(b) for a step change from 0 to 1 inch per second. In equations (4) through (9), feed rates are in inches per second.



Corner With $F_2 \neq F_1$

Diagram 4.3.2A

Let PO bisect < MOQ.

To locate point B where the tool center path crosses PO, the axes will be rotated to the position X', Y', with the Y' axis lying along OP. After this rotation, the tool center coordinates X' and Y' will be given as functions of time. The value of time which makes X'=0 will be found and substituted into the expression for Y' to obtain the distance OB.

angle POY = $\theta_1 + (1/2)\theta$ where $\theta = \theta_2 - \theta_1$.

4.3.2.1 UNDERSHOOT ERROR ON CORNER WHEN $F_2 \neq F_1$ (cont'd)

After rotating the axes through the angle POY,

$$X' = F_1 \cos (\theta/2)[(F_2/F_1)t-1/k_v-(F_2/F_1-1) f(t)] \qquad (6)$$

$$Y' = F_1 \sin (\theta/2)[(F_2/F_1)t+1/k_v-(F_2/F_1+1) f(t)] \qquad (7)$$

$$\text{For } X' = 0, (F_2/F_1)t-1/k_v-(F_2/F_1-1)f(t)=0,$$

$$\text{or } f(t) = \frac{(F_2/F_1)t- 1/K_v}{F_2/F_1-1} \qquad (8)$$

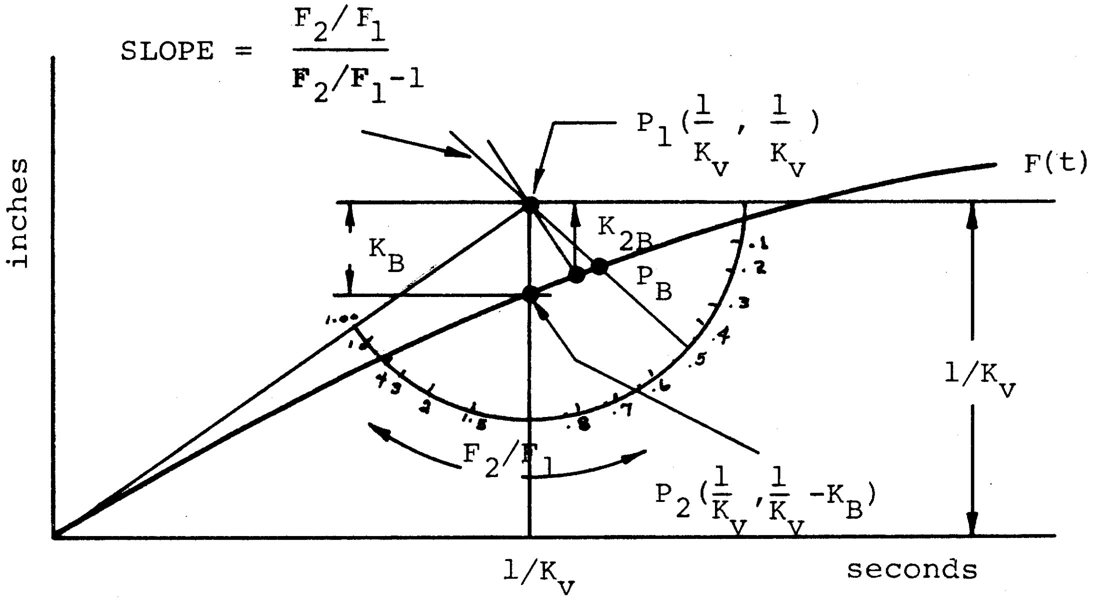
The right side of equation (8) is a linear function of t, and becomes equal to $1/K_V$ if t is set equal to $1/K_V$. Geometrically, equation (8) states that the value of t for which X' is zero can be found by the construction of Diagram 4.3.2B where a plot of f(t) has superimposed on it a line drawn through the point $P_1(1/k_v, 1/K_v)$ with a slope

$$\frac{F_2/F_1}{F_2/F_1-1}$$

This line intersects the f(t) curve at a point P_B. The line shown as an example in Diagram 4.3.2B is drawn with a slope of -1, corresponding to $F_2/F_1=0.5$. The circular scale indicates the location of the line for other values of F_2/F_1. If the time corresponding to P_B is called t_B and this value is combined with equation (8) and substituted for t in equation (7),

$$Y' = OB = E_B = 2F_1 \sin \theta/2 \, [1/K_V - f(t_B)] = 2K_{2B} F_1 \sin \theta/2 \qquad (9)$$

4.3.2.1 UNDERSHOOT ERROR ON CORNER WHEN $F_2 \neq F_1$ (con'd)


SLOPE = $\frac{F_2/F_1}{F_2/F_1-1}$
$P_1(\frac{1}{K_V}, \frac{1}{K_V})$
F(t)
inches
K_B
K_{2B}
P_B
$1/K_V$
.1
.2
.3
.4
.5
.6
.7
.8
1.5
2
3
F_2/F_1
$P_2(\frac{1}{K_V}, \frac{1}{K_V}-K_B)$
$1/K_V$
seconds

Graphical Interpretation of Equation 8

Diagram 4.3.2B

The quantity $K_{2B} = 1/K_V - f(t_b)$ is indicated on Diagram 4.3.2B.

To obtain an approximate value of K_{2B} analytically, we can replace f(t) with a parabola of the form $f_p(t) = t - Ct^2$. This will pass through the origin with unity slope and by letting $C = K_B K_V{}^2$, it will pass through the point $P_2(1/K_V, 1/K_V - K_B)$, which is a known point on the f(t) curve.

Solving the straight line and this parabola for the location of P_B leads to the following result for K_{2B}:

$$K_{2B} = K_B \frac{2(F_2/F_1)}{1 + 2K_BK_V(F_2/F_1-1) + \sqrt{1 + 4K_BK_V(F_2/F_1-1)}}$$

When this is substituted into (9), the resulting equation may be solved for F_2 in terms of F_1, E_b, and Θ, and the servo constants K_B and K_V:

4.3.2.1 UNDERSHOOT ERROR ON CORNER WHEN $F_2 \neq F_1$ (cont'd)

$$F_2 = 60 \quad \frac{-B_2 \pm \sqrt{B_2{}^2 - 4A_2C_2}}{2A_2}$$

where

$$A_2 = K_B(K_VE_B - 2\ \frac{F_1}{60}\ \sin\ \frac{\Theta}{2}\ )^2$$

$$B_2 = 2\ \frac{F_1}{60}\ E_B \left[-\ \frac{F_1}{60}\ \sin\frac{\Theta}{2} \quad + \quad 2\ \frac{F_1}{60}\ K_BK_V\ \sin\frac{\Theta}{2}\ -\ K_BK_V{}^2E_B \right]$$

$$C_2 = \left[\frac{F_1}{60}\right]^2 \quad K_BK_V{}^2E_B{}^2$$

4.3.2.2 FEEDRATE ON CIRCLE DUE TO STEADY STATE SERVO ERRORS

The open loop response of the naked system servo whose Bode diagram is given in Diagram 4.3.2C is

$$\frac{C}{E} = \frac{w_c}{p(1+p/w_3)\ (1+p/w_4)\ (1+p/w_5)} = \frac{w_cw_3w_4w_5}{p(p+w_3)\ (p+w_4)\ (p+w_5)}$$

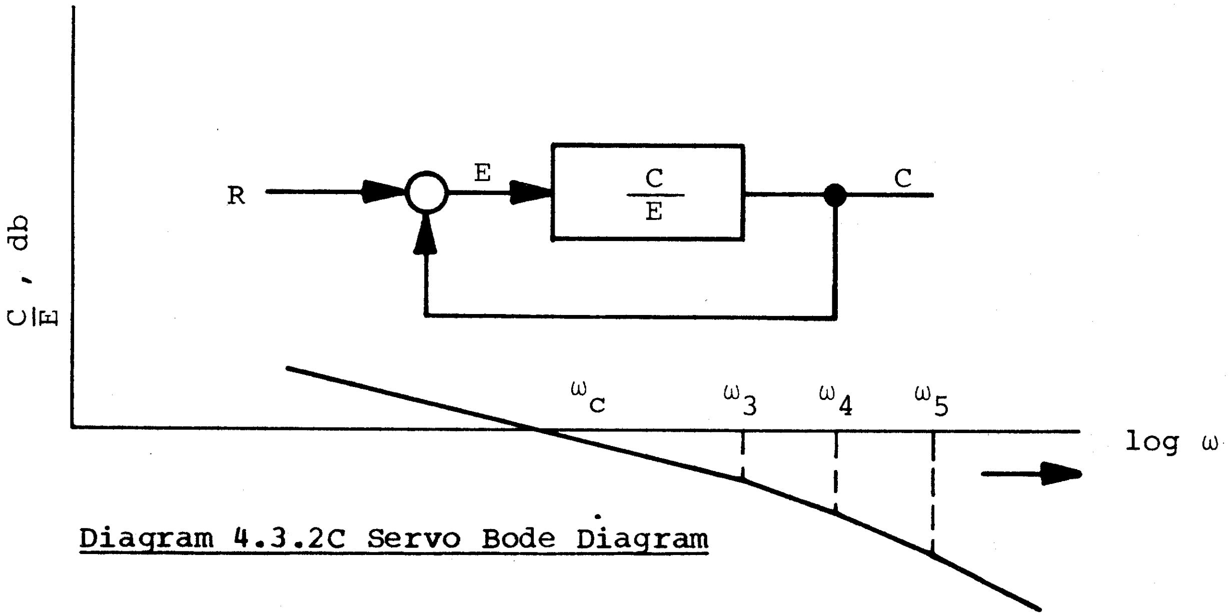


Diagram 4.3.2C Servo Bode Diagram

The closed loop response,

$$\frac{C}{R} = \frac{C/E}{1+C/E} = \frac{w_cw_3w_4w_5}{p^4 + (w_3 + w_4 + w_5)\,p^3 + (w_3\,w_4 + w_3\,w_5 + w_4\,w_5)\,p^2 + w_3\,w_4\,w_5\,p + w_c\,w_3\,w_4\,w_5}$$

4.3.2.2 FEEDRATE ON CIRCLE DUE TO STEADY STATE SERVO ERRORS (cont'd)

Letting p=jw, and collecting real and imaginary parts of the denominator,

$$\frac{C}{R} = \frac{w_c w_3 w_4 w_5}{(w_3w_4+w_3w_5+w_4w_5)w^2 + w_c w_3 w_4 w_5 + j -[(w_3+w_4+w_5)w^3 + w_3 w_4 w_5 w[} =$$

$$\frac{1}{1-w_c\left[\frac{1}{w_3}+\frac{1}{w_4}+\frac{1}{w_5}\right]\left[\frac{w}{w_c}\right]^2 + \frac{w_c^3}{w_3w_4w_5}\left[\frac{w}{w_c}\right]^4 + j\left[\frac{w}{w_c} - w_c^2\left(\frac{1}{w_3w_4}+\frac{1}{w_3w_5}+\frac{1}{w_4w_5}\right)\left(\frac{w}{w}\right)^3\right]}$$

Taking the square root of the sum of the squares of the real and imaginary parts of the denominator to obtain the magnitude of C/R,

$$\frac{C}{R} = \frac{1}{[1 + C_2(w/w_c)^2 + C_4(w/w_c)^4 + C_8(w/w_c)^8]^{1/2}}$$

where

$$C_2 = 1 - 2w_c\left(\frac{1}{w_3}+\frac{1}{w_4}+\frac{1}{w_5}\right)$$

$$C_4 = w_c^2\left(\frac{1}{w_3^2}+\frac{1}{w_4^2}+\frac{1}{w_5^2}\right) + 2w_c^3\left(\frac{1}{w_3w_4w_5}\right)$$

$$C_6 = w_c^4\left(\frac{1}{w_3^2w_4^2}+\frac{1}{w_3^2w_5^2}+\frac{1}{w_4^2w_5^2}\right)$$

$$C_8 = w_c^6\left(\frac{1}{w_3^2w_4^2w_5^2}\right)$$

If the same analysis is made of a servo with another down break at w_6, the result corresponding to equation (11) is of 10th order in w/w_c, and the coefficients are very similar in form:

4.3.2.2 FEEDRATE ON CIRCLE DUE TO STEADY STATE SERVO ERRORS (cont'd)

$$C_2 = 1 - 2w_c \left(\frac{1}{w_3} + \frac{1}{w_4} + \frac{1}{w_5} + \frac{1}{w_6} \right)$$

$$C_4 = w_c^2 \left(\frac{1}{w_3^2} + \frac{1}{w_4^2} + \frac{1}{w_5^2} + \frac{1}{w_6^2} \right) + 2w_c^3 \left(\frac{1}{w_3 w_4 w_5} + \frac{1}{w_3 w_4 w_6} + \frac{1}{w_3 w_5 w_6} + \frac{1}{w_4 w_5 w_6} \right)$$

$$C_6 = w_c^4 \left(\frac{1}{w_3^2 w_4^2} + \frac{1}{w_3^2 w_5^2} + \frac{1}{w_3^2 w_6^2} + \frac{1}{w_4^2 w_5^2} + \frac{1}{w_4^2 w_6^2} + \frac{1}{w_5^2 w_6^2} \right)$$

$$C_8 = w_c^6 \left(\frac{1}{w_3^2 w_4^2 w_5^2} + \frac{1}{w_3^2 w_4^2 w_6^2} + \frac{1}{w_3^2 w_5^2 w_6^2} + \frac{1}{w_4^2 w_5^2 w_6^2} \right)$$

$$C_{10} = w_c^8 \left(\frac{1}{w_3^2 w_4^2 w_5^2 w_6^2} \right)$$

Note that all coefficients except C_2 are inherently positive and that their terms will contribute to a decrease in $|C/R|$ as the frequency variable w/w_c increases. In (12), C_2 will be positive if w_3, w_4, and w_5 are sufficiently large. With a positive C_2, $|C/R|$ will never exceed unity. If w_3, w_4, and w_5 are sufficiently small, C_2 can become negative and its term can cause $|C/R|$ to exceed unity for some values of w/w_2. This is the algebraic mechanism which gives rise to a peak in the closed loop response. For the critical condition $C_2 = 0$, the closed loop response stays at essentially unity as w/w_c increases, until the higher order terms cause it to drop off. The condition for $C_2=0$ is

$$\frac{w_c}{w_3} + \frac{w_c}{w_4} + \frac{w_c}{w_5} = \frac{1}{2}$$

or approximately, $\Theta_{CH} = 1/2$ radian or 28.6°

4.3.2.2 FEEDRATE ON CIRCLE DUE TO STEADY STATE SERVO ERRORS (cont'd)

Let us calculate a typical set of constants from (12) for a system with a resonant peak.

Let $\frac{w_c}{w_3} = \frac{16}{40}$, $\frac{w_c}{w_4} = \frac{16}{64}$, and $\frac{w_c}{w_5} = \frac{16}{128}$

Then $C_2 = -.550$

$C_4 = .263$

$C_6 = .0135$

$C_8 = .000156$

For $w/w_c = 1/2$, the terms in (11) have the values,

$C_2 (1/2)^2 = -.138$

$C_4 (1/2)^4 = .0164$

$C_6 (1/2)^6 = .0002$

$C_8 (1/2)^8 = .000006$

Clearly, for $w/w_c = 1/2$ or less, the 6th and 8th order terms are neglibible. The 4th order term is 12% of the 2nd order term, and becomes rapidly of lesser importance as w/w_c decreases. Therefore, we neglect it also and write approximately,

$$\left|\frac{C}{R}\right| = \frac{1}{\left[1 + C_2 \frac{w}{w_c}^2\right]^{1/2}}$$

4.3.2.2 FEEDRATE ON CIRCLE DUE TO STEADY SERVO ERRORS (cont'd)

Since $C_2 \left(\frac{w}{w_c}\right)^2 << 1$,

$$\left|\frac{C}{R}\right| = \frac{1}{1 + (1/2)C_2\left(\frac{w}{w_c}\right)^2} = 1 - (1/2)C_2 \left(\frac{w}{w_c}\right)^2$$

The term $-(1/2)C_2\left(\frac{w}{w_c}\right)^2$ represents the excess of the closed loop response over unity. If the radial oversize error of the contoured circle in inches is called E,

$$\frac{E}{r} = -(1/2)C_2\left(\frac{w}{w_c}\right)^2, \qquad (13)$$

where r is the radius of the circle. If F is the feedrate in

in in./sec., the angular velocity on the circle is

$$F/r = w \qquad (14)$$

Substituting (14) into (13) and solving for F in terms of E and r,

$$F = w_c\sqrt{\frac{2}{-C_2}}\sqrt{E \cdot r}$$

$$F = K_c \sqrt{E\ r} \quad \text{in/sec., where} \qquad (15)$$

$$K_c = w_c\sqrt{\frac{2}{-C_2}} = w_c \sqrt{\frac{1}{\frac{w_c}{w_3} + \frac{w_c}{w_4} + \frac{w_c}{w_5} - \frac{1}{2}}} \qquad (16)$$

Equations (15) and (16) are valid as long as

$$C_2 \left(\frac{w}{w_c}\right)^2$$

in (11) is the dominant term, as it will be for servos with at least a moderate overshoot.

4.4 SURFACE FEET PER MINUTE (SFM)

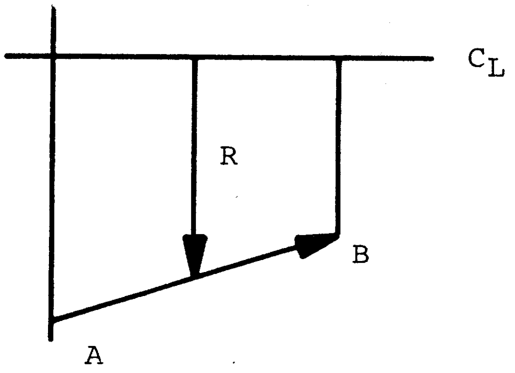


Diagram 4.4A

Suppose the part shown in Diagram 4.4A is machined on a lathe with the cutter starting at A and working to B. Suppose further that there is a constant feedrate and that the spindle rotates at a constant speed. Since the part radius decreases from A to B, the tool cutting speed varies along the path AB, and the cutting speed is high for the larger radii and low for the smaller radii. This varying cutter speed does not create a smooth finish, and the chip removal is not the optimum amount possible for the life of the tool. These undesirable effects can be reduced if the spindle speed is increased proportionately to the decreasing radii, that is, maintain a constant cutting value of surface feet per minute. The part programmer can plan his program so that the spindle changes speed at certain points, and essentially, keeps a constant cutting speed. However, this is an enormous task and subject to many errors. But, the APT postprocessor statement, SPINDL/f, SFM provides the part programmer with an easy method for accomplishing this work. He merely calls for the SFM he desires and the postprocessor does all the work necessary to maintain the requested SFM.

The following example illustrates the postprocessor's method of generating the SFM condition. The example illustrates the method for spindle types which depend upon a table of discrete spindle speeds. See the paragraphs at the end of this section for the SFM method used for variable type spindles.

4.4 SURFACE FEET PER MINUTE (SFM) (cont'd)

Diagram 4.4B is the table of spindle speeds in RPM for the machine tool.

Spindle Speeds
15
20
30
40
50
60
70
80

Diagram 4.4B

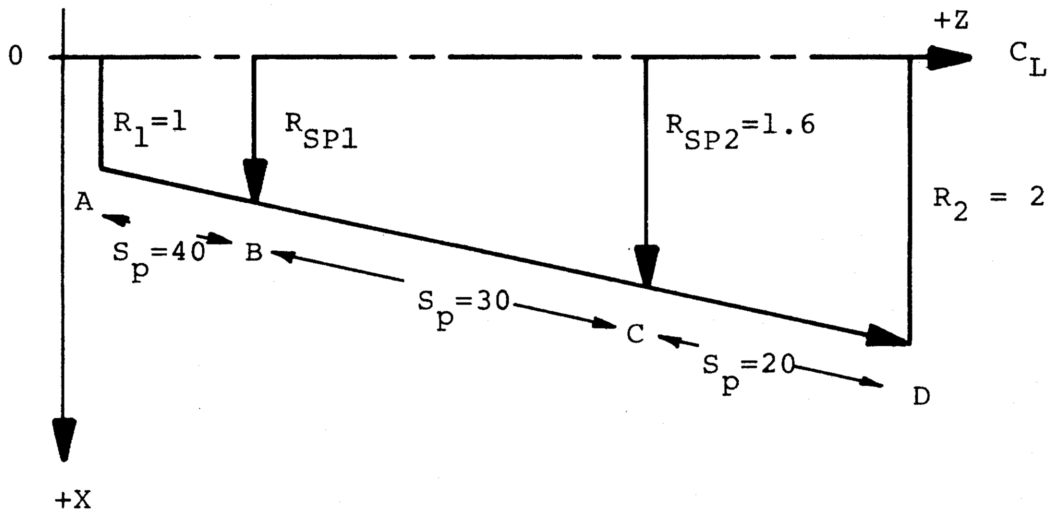


Diagram 4.4C

4.4 SURFACE FEET PER MINUTE (SFM) (cont'd)

Diagram 4.4C represents the part which the tool is to cut from A to D with an SFM of 20.

The starting spindle speed is computed from the equation

$$S_p = \frac{12}{2\pi} \frac{SFM}{R_1},$$

where SFM is the desired SFM, R_1 is the radius at the beginning point, and S_p is the resultant spindle speed. For this example, we have:

$$S_p = \frac{1.9 * 20}{1} = 38 \text{ RPM.}$$

This value is bracketed in the spindle speed table by the speeds 30 and 40. Using these values we next compute the optimum shift point, i.e., the point at which the spindle speed changes so that the SFM variation is a minimum. The value determined is the radius at the optimum shift point and is given by the equation:

$$R_{SP} = \frac{12}{4\pi} \quad SFM(\frac{1}{S_1} + \frac{1}{S_2}),$$

where S_1 and S_2 are the speeds which are selected from the spindle speed table, and R_{SP} is the radius at the optimum shift point. The present example gives

$$R_{SP} = 0.95 * 20 * (\frac{1}{40} + \frac{1}{30}) \cong 1.1$$

Hence, at the radius of 1.1 (point B), the spindle must change from speed 40 to speed 30. Since the radius of the workpiece is increasing, we must select decreasing values of spindle speeds from the spindle table. Therefore, for the next determination of the optimum shift point, S_1 = 30 and S_2 = 20, and

$$R_{SP} = 0.95 * 20 * (\frac{1}{20} + \frac{1}{15}) \cong 2.2.$$

4.4 SURFACE FEET PER MINUTE (SFM) (cont'd)

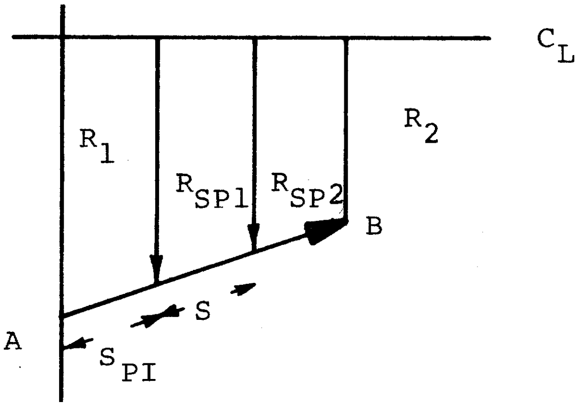
But the radius 2.2 is beyond the end of the workpiece; therefore, the determination of shift points ends, and the speed 20 is used for the remaining path. A similar case occurs when there are no new values to select from the table; the last selected value is used for the remainder of the path. Note that if the feedrate mode is IPR, the feedrate in IPM is computed by

$$F_{IPM} = F_{IPR} * S_P,$$

where F_{IPR} is the feedrate in IPR.

The postprocessor segments the path AD into the subsegments AB, BC, and CD. Each segment will have the proper value of spindle speed and feedrate necessary to produce the requested SFM for the given path.

The SFM technique is essentially the same for variable spindle types except that each succeeding shift point is determined by the spindle speed which is a certain percentage of the preceding spindle speed. The percentage used is specified in option 15. The following steps in the example illustrate the method.



(1) Determine the initial spindle speed at A from the relation

$$S_{P_I} = \frac{12}{2\pi} * \frac{SFM}{R} .$$

(2) Determine the limiting spindle speed at B by

$$S_{P_L} = \frac{12}{2\pi} * \frac{S_{FM}}{R_2} .$$

(3) Determine the shift point spindle speed from $S = OPTAB(15) * S_P$.

OPTAB(15) is standardly 0.1%.

(4) Determine the radius at the optimum shift point from the relation

$$R_{SP_1} = \frac{12}{4\pi} SFM \left(\frac{1}{S_1} + \frac{1}{S_2} \right)$$

where S_1 and S_2 are the two determined speeds; S_1 is the previous speed, and S_2 is the newly determined speed.

4.4 SURFACE FEET PER MINUTE (SFM) (cont'd)

(5) The previous speed is used over the segment determined by the optimum shift point. Thus, initially S_{P_I} is the speed from A to R_{SP_1}, whereas the speed S is used from R_{SP_1} to R_{SP_2} and so on. Each succeeding spindle speed is determined as at step 3.

(6) The SFM sequence discontinues whenever S falls outside the bound of S_{P_L} , or whenever the radius R_{SP_M} exceeds the bound of R_2. The postprocessor segments the path AB into the computed SFM subsegments as described above.

When a SPINDL/n, SFM statement is given, subroutine SPINDL sets the flag SFMFLG to 1 to indicate that an SFM mode has been established. All subsequent motions except rapid traverses, threads, or tool corrective moves are subjected to the SFM influence. Subroutines GOLINE, PROCQD, and SEGMNT interrogate the SFMFLG, and when finding it non-zero, branch to subroutine SFMO which generates the spindle speeds and path segments to obtain the requested SFM.

When the SFM mode is cancelled, the SFMFLG is set to zero and the SFM sequence discontinues. A number of parameters is set for a given SFM statement. Consider a full statement such as

```
SPINDL/40,SFM,CLW,RANGE,2,RADIUS,YCOORD,MAXIPM,10,MAXRPM,100.
```

The following flags are accordingly set:

SFMFLG = 1,	SFMDES = 40,	SPNDIR = 1,
ISRNGE = 2,	ISFMOD = 1(for X axis), 2(for Y axis), 3(for Z axis),	
SFMAXI = 10,	SFMAXR = 100,	SFMLOK = 1,
SFMRPM = 1,	FLONSP = 1,	

All of these parameters are used and referred to in subroutine SFMO, the subroutine which produces the requested SFM effect.

Since Subroutine SFMO generates its own segments, it must first save the current beginning and end points DPREVM and DPRESM; these points are saved in the local arrays PREVM and PRESM, respectively. Now the arrays DPREVM and DPRESM can be used in their normal manner when the subroutine produces new segments.

4.4 SURFACE FEET PER MINUTE (SFM) (cont'd)

One of the first duties performed by subroutine SFMO is to determine the sense of inclination of the cutter path. This is easily done by comparing the beginning and ending radii of the path, which for the statement given above, means comparing the Y coordinate values from DPREVM and DPRESM.

The flag ISENSE is then set as:

-1 = increasing radius

0 = constant radius

+1 = decreasing radius

The SFM sequence over a particular path ends whenever:

(1) the path end point is reached;

(2) a spindle speed is generated which produces a radius beyond that of the end point;

(3) a spindle speed is generated which exceeds the SFMAXR limitation;

(4) the selection of spindle speeds ceases because the range bound has been reached;

(5) the radius is essentially constant.

For all of these conditions, the flag SFMLIM is set to 1 which directs the subroutine to conclude the SFM sequence over this path.

It is sometimes very difficult to start an SFM sequence for a given path if the coordinate points and available spindle speeds are incompatible. In Diagram 4.4E is illustrated a case where the starting speed cannot produce a radius which falls on the path. The table of discrete speeds is also shown.

4.4 SURFACE FEET PER MINUTE (SFM) (cont'd)

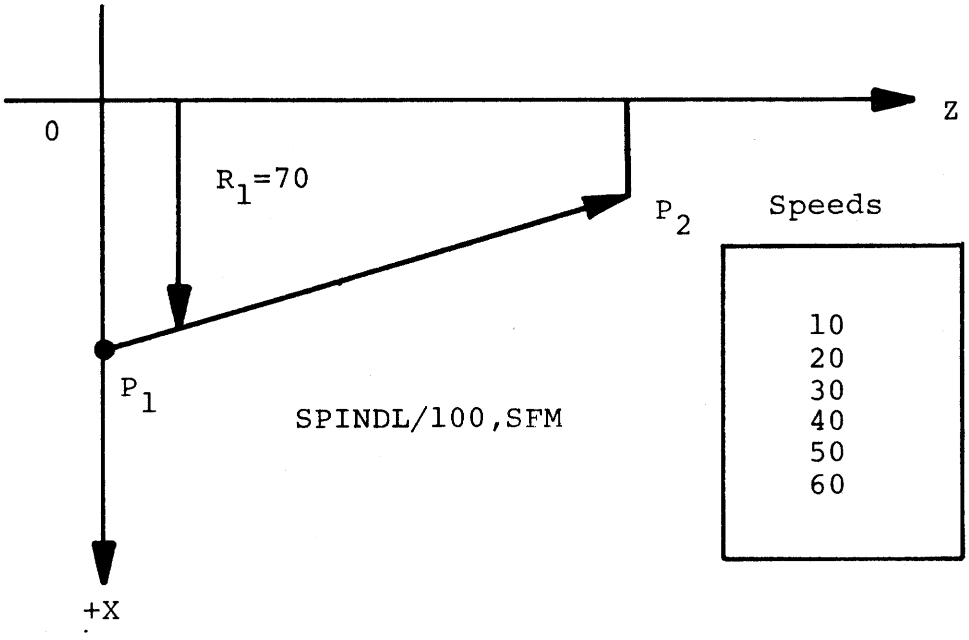


Diagram 4.4E

The sense of inclination ISENSE is +1 which means we must select from the table those values which become increasingly larger. But when the starting speed is attempted to be found, as

$$S = \frac{12 * SFMDES}{2 \pi \ R1} \cong \frac{2 * 100}{70} = 2.86,$$

it can be seen that such a speed is not available in the table. Hence, the SFM sequence cannot even be started for this path with these given conditions. The postprocessor outputs the path P_1P_2 with the current spindle speed.

4.5 SPINDLE TYPES

Spindles are typed, i.e., classified, according to the manner in which their spindle speed commands are formed. This formation usually results in a coded value which, according to the manner of the NC machine, leads to the obtaining of the desired spindle speed.

It is important to note that the spindles are not typed according to whether or not the spindle is AC or DC motor driven, or whether the spindle requires gear shifting for range changes, or whether the gears are shifted electrically or hydraulically. The only consideration is the make-up of the spindle speed command.

4.5 SPINDLE TYPES (cont'd)

Spindle types can be classified into two groups: one group consists of those types whose speeds are selected from a preset list of discrete values; the other group consists of those types which have a range of variable speeds. For proper spindle operation of types of the first group, the postprocessor must have available the list of preset discrete spindle speeds. These speeds are given in ordered form in the spindle speed table, SRTAB.

A maximum of 300 spindle speeds can be given in the SRTAB table; the actual number is specified in options 7 and 8. If the speeds are grouped into ranges, the speeds as given in SRTAB must be in increasing range order, i.e., range 1 values must precede range 2 values, and so on. The spindle speeds are ordered this way because the programmed intent is to select a speed from a range which gives the highest motor speed; thus, if a requested speed can be selected from one of two ranges, the lowest range is chosen since its spindle speed will have a higher motor speed. The terms "low, medium, and high range" refer to the motor speed.

If a speed is called for that is within the range of, but not listed in the spindle table, SRTAB, the postprocessor selects the next lowest, closest, or next highest speed, depending upon the value of option 90. If the specified speed is outside the table, the postprocessor outputs the appropriate maximum or minimum listed speed.

SRTAB may have any number of ranges, but each range <u>must</u> have the same number of speeds. Therefore, it is possible to have 30 ranges of 10 speeds in each range, or, 15 ranges of 20 speeds, and so on. The total number of speeds cannot exceed 300*. Unless otherwise specified, all the spindle types described below assume that SRTAB is given with preset discrete speeds, and that the speeds are grouped in several ranges.

* These statements assume that there is no feedrate table FRTAB. If there is, the total number of speeds permissible is consequently reduced. See Section 5.6.6 for a description of FRTAB.

4.5.1 TYPE 0: COMBINATION RANGE AND ROW

The speed command is formed by using the range number as its first digit, and the row number of that range as its second digit plus some increment which is given by option 47. Note that the row number in this example begins at 0 and ends at 9. Each range is numbered separately.

	Range 1		Range 2		Range 3
0	2	0	12	0	26
1	4	1	14	1	28
2	6	2	16	2	30
3	8	3	18	3	32
4	10	4	20	4	34
5	12	5	22	5	36
6	14	6	24	6	38
7	16	7	26	7	40
8	18	8	28	8	42
9	20	9	30	9	44

Example: speed = 12; speed command =S15 or S20. Option 47 in this case is 0.

If there are more than ten rows per range, the speed command has three digits.

S622 means range 6, 22nd row. If there are more than ten ranges, the speed command has four digits. S1204 means range 12, 4th row; S1011 means range 10, 11th row.

4.5.2 TYPE 1: EIA 3-DIGIT CODE NUMBER (VARIABLE SPINDLE)

The spindle may have any speed which falls within a maximum and minimum value. No table of speeds is required, and no ranges are used in the coding. The spindle speed is converted to the standard 3-digit EIA command number.

Example: Speed = 137.2; speed command = S614. Speed = 0.0123; speed command = S212. See Section 7.1 of the Appendix for the EIA conversion method.

The only information necessary to the postprocessor for this type spindle is the minimum and maximum values that the spindle can assume. To be consistent with the other type spindles, the minimum and maximum values are given in the SRTAB. The SRTAB, then has only the two values, and SRTAB is considered as representing one range with two values in the range. Option 7,therefore, is set to 1, add option 8 to 2. The Standard Machine assumes this type spindle, hence, these options are set accordingly.

4.5.2 TYPE1: EIA 3-DIGIT CODE NUMBER (VARIABLE SPINDLE) (cont'd)

As a condition of this type spindle, the machine tool control must either produce every possible coded speed within the specified range, or search for the appropriate speed if not all speeds are possible.

4.5.3 TYPE 2: ASSOCIATED SPEED CODE

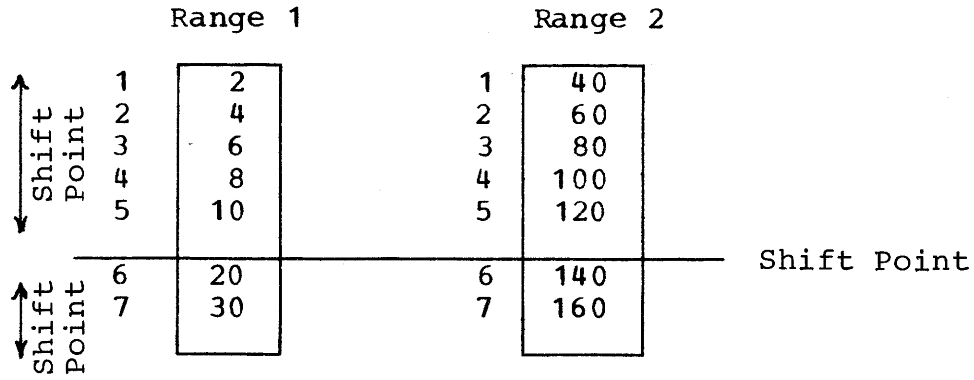
For a given spindle speed there is an associated code number. These code numbers may come in ranges or be all in one range, but there must be the same number of codes in each range. The code number is assumed to be related to the row number plus an incremental value, e.g., if the speed falls in row 12, the code number is assumed to be 12 plus some increment. Hence, the incremental variation from a code number to an adjoining code number must be constant within each range. The increment is given in option 47. A variation in the speed codes is permissible between ranges provided the same variation exists between all ranges. In the example below, the variation between ranges is 3 since the last code in range 1 is 17 while the first code in range 2 is 20. This incremental variation between ranges is given in option 31.

	Range 1			Range 2			Range 3	
	Code	Speed		Code	Speed		Code	Speed
0	10	5	0	20	14	0	30	30
1	11	6	1	21	16	1	31	32
2	12	8	2	22	18	2	32	34
3	13	10	3	23	20	3	33	36
4	14	12	4	24	22	4	34	38
5	15	14	5	25	24	5	35	40
6	16	16	6	26	26	6	36	42
7	17	18	7	27	28	7	37	44

Example: speed = 14; speed command = S15 or S20. The code increment (option 47) for this example is 10; the range increment (option 31) is 3. The speed command S20 is derived from the relation of the row number modified by the range number, option 47, and option 31. 119

4.5.3.1 TYPE 2: PROTECTIVE MULTIPLE SHIFTING

Some machine tools require a multiple shifting sequence when going from one spindle range to another. For example, it may be damaging to some machine tools to change spindle ranges if the two speeds are high values in each range. In fact, there is a shift point common to each range above which direct shifting from one range to another is damaging. In the example below, the shift point is at row 5.



	Range 1		Range 2
1	2	1	40
2	4	2	60
3	6	3	80
4	8	4	100
5	10	5	120
6	20	6	140
7	30	7	160

If the spindle speed is 30 RPM in Range 1 and we wish to shift to 160 RPM in Range 2, the proper way to obtain this speed is to output the sequence 10 RPM, 120 RPM, and lastly, 160 RPM. That is, the speed is first brought down to the shift point in Range 1, the range is changed to Range 2 with the spindle speed coming from the shift point of Range 2, and then the spindle is brought up to the new speed. A similar path is followed in going from 160 to 30 RPM.

No multiple shifting is required as long as the ranges do not change. Thus in Range 1 we can go directly from 2 RPM to 30 RPM.

Only two spindle speeds are made output when changing ranges from above the shift point to a speed below the shift point. To go from 160 RPM in Range 2 to 4 RPM in Range 1 requires only the output of two values, namely, 60 RPM and 4 RPM. In this sequence the shift from one range to the other is made at the same row; this is always permissible as long as the old and new spindle speeds fall below the shift point.

Direct shifting is also permissible between ranges when both the old and new spindle speeds occur below the shift point. For example, we can go directly from 8 RPM in Range 1 to 100 RPM in Range 2 since both values lie below the shift point.

For a Type 2 spindle, option 137 specifies the shift point row number beginning with the lowest speed row and counting toward the highest speed row.

4.5.4 TYPE 3: ASSOCIATED SPEED CODE WITH RANGE AND/OR DIRECTION M CODES

The spindle command is given by both an S code and an auxilliary function M code. The S code is related to the row number plus an incremental value which is given in option 47. The incremental variation between adjacent code numbers must be constant. The S word selection is independent of range; i.e., the same for all ranges.

The range is selected by auxiliary function M code in one of two ways. In the first, a single M code is assigned to each range, M_1 for range 1, M_2 for range 2, etc. M_1 is assigned to TABLEM locations 71 and 72 for range 1, M_2 to locations 73 and 74, etc. Spindle direction is obtained by M03 and M04 which are stored at TABLEM locations 4 and 5.

In the second case, auxiliary function M codes determine both range number and spindle direction. Two M codes are assigned to each range, one for CLW rotation and the other for CCLW. These codes are stored in TABLEM beginning at location 71 and continuing up to location 82 for a maximum of 6 ranges.

A spindle speed command including RPM, range and direction will interrogate TABLEM in the 70 series for proper range number. It will interrogate direction M code stored at TABLEM location 4 or 5 unless these latter locations are set to DMBITS. Machines using both range and direction combination M codes will not use M03 and M04, and locations 4 and 5 should be set to DMBITS. A range only M code should be stored in both CCW and CCLW locations for the appropriate range.

Range 1

M_1

	Code	Speed
0	2	100
1	3	200
2	4	300
3	5	400
4	6	500

Range 2

M_2

	Code	Speed
0	2	300
1	3	400
2	4	500
3	5	600
4	6	700

Range 3

M_3

	Code	Speed
0	2	600
1	3	700
2	4	800
3	5	1000
4	6	

4.5.4 TYPE 3: ASSOCIATED SPEED CODE WITH RANGE AND/OR DIRECTION M CODES (con'd)

Example: Speed=400; speed command=S5 with M_1 or S3 with M_2. The incremental option 47 for this example is 2. Note that the rows are numbered separately for each range, and that the rows begin numbering with 0. In this case spindle direction will come from TABLEM location 4 or 5.

Alternatively -

	Range 1			Range 2			Range 3	
	CCW	CCLW		CCW	CCLW		CCW	CCLW
	M_1	M_2		M_3	M_4		M_5	M_6
	Code	Speed		Code	Speed		Code	Speed
0	2	100	0	2	300	0	2	600
1	3	200	1	3	400	1	3	700
2	4	300	2	4	500	2	4	800
3	5	400	3	5	600	3	5	900
4	6	500	4	6	700	4	6	1000

Example: speed = 400, clockwise; speed code S5 with M_1 from range 1, or S3 with M_3 from range 2. The postprocessor will always select the speed from the lowest numbered range in which it can be found, unless a specific range has been called.

4.5.5 TYPE 4: QUASI EIA 3-DIGIT CODE NUMBER WITH RANGE AND DIRECTION M CODE

The spindle speed command is given by both a miscellaneous function M code which is chosen according to the range and spindle direction, and by the EIA 3-digit coded number for the spindle speed <u>in the lowest range</u> which occupies the same row as the desired spindle speed. See Section 7.1 of the Appendix for the EIA conversion method.

Range 1	Range 2	Range 3
M_1	M_2	M_3
Low	Medium	High
100	400	600
200	500	700
300	600	800
400	700	900
500	800	1000
600	900	1100

Example: speed = 600; speed command could be:

S660 with M_1, or
S630 with M_2, or
S610 with M_3.

It is apparent that the EIA code number is correct for those speeds found only in the lowest range (range 1). They are meaningless, and therefore arbitrary numbers, for the speeds in all other ranges.

4.5.6 TYPE 5: DISCRETE EIA 3-DIGIT CODE NUMBER

The speed command is derived by selecting the spindle speed from SRTAB, and converting it into the EIA 3-digit number. See Section 7.1 of the Appendix for the EIA conversion method. With this type spindle there are no multiple ranges but only discrete speeds. Furthermore, the table may not include all possible EIA coded speeds. If a speed given by the part programmer does not appear exactly in the spindle table SRTAB, the postprocessor selects the next lower value. This is extremely important for those machines which have an AC motor drive and use the discrete EIA values, since this type machine may produce an incorrect spindle speed if the speed called for is not exactly available in the machine's spindle table.

Speeds
2
3
5
7
10
12
15
17
-
-
-
193
197
200

Example: speed = 13; speed command = S512. The closest value, 12, is selected.

4.5.7 TYPE 6: DISCRETE EIA 3-DIGIT CODE NUMBER - SELECTIVE SEARCH

This type spindle is similar to Type 5, except that the machine tool control system automatically searches its spindle table until it finds the next lowest available EIA-coded speed. The postprocessor considers this type spindle to be identical with Type 5.

Although it is possible to have the postprocessor simply convert the speed to the EIA 3-digit number, and then let the machine tool control system find the proper value, it is actually simpler to let the postprocessor find the exact value. In fact, if SFM is used, it is essential that the exact variations of spindle speeds be used.

4.5.8 TYPE 7: (PRESENTLY UNDEFINED)

4.5.9 TYPE 8: ASSOCIATED SPEED CODE INDEPENDENT OF RANGE

For a given spindle speed there is an associated speed code related to the row number of the range in which the speed falls. The speed code number related to each row is the same regardless of the range number. Thus, the code number for row 2 of range 1 is the same as for row 2 of range 2. The code number is assumed to be derivable from the row number plus an incremental value, e.g., if the speed falls in row 12, the code number is assumed to be 12, plus some increment; the increment must be given in option 47.

Range 1

	Code	Speed
0	1	10
1	2	20
2	3	30
3	4	40
4	5	50

Range 2

	Code	Speed
0	1	40
1	2	50
2	3	60
3	4	70
4	5	80

Range 3

	Code	Speed
0	1	70
1	2	80
2	3	90
3	4	100
4	5	110

Example: speed = 40; speed command = S04 if in range 1; speed command = S01 if in range 2. If the speed is 70 and range 3 is in use, then S01 is the command code. The increment option 47 for this example is 1. Note that the rows are numbered separately for each range, and that the numbering begins with 0. Shifting between ranges is not tape controlled.

4.5.10 TYPE 9: ASSOCIATED SPEED CODE RELATED TO TOOL NUMBER

This is similar to Type 8 in that the speed code is independent of the range, but this type spindle specifies the range by the tool number in use.

	Range 1			Range 2			Range 3	
	Tools 1 & 4			Tools 2 & 5			Tools 3 & 6	
	Code	Speed		Code	Speed		Code	Speed
0	0	10	0	0	40	0	0	70
1	1	20	1	1	50	1	1	80
2	2	30	2	2	60	2	2	90
3	3	40	3	3	70	3	3	100
4	4	50	4	4	80	4	4	110

Example: speed = 50; speed command = S04 if tool 1 is in use, or S01 if tool 5 is in use.

The number of tools on the machine is given in option 88. Regardless of the number of tools, the range relation is assumed to be:

Range 3 for tools 1, 4, 7, 10, 13

Range 2 for tools 2, 5, 8, 11, 14

Range 1 for tools 3, 6, 9, 12, 15

4.5.11 TYPE 10: VARIABLE SPEED WITH RANGE AND DIRECTION M CODE

The spindle speed command is given by both a miscellaneous function M code which is chosen according to the range and the spindle direction, and the spindle speed which is converted to a 3-digit EIA code. See Section 7.1 of the Appendix for the EIA conversion method. The speeds in each range are variable and are only limited by a maximum and minimum value for each range.

Range 1	
CLW	CCLW
M60	M61
1	1.2
2	13

Range 2	
CLW	CCLW
M62	M63
1	3.1
2	31

Range 3	
CLW	CCLW
M64	M65
1	6.6
2	69

Example: speed = 10; speed command could be S510 with M60 for range 1 and a CLW spindle direction, or S510 with M63 for range 2 and a CCLW spindle direction. The M codes are specified in TABLEM. Each range must be specified as having two rows which are the range minimum and maximum values, hence option 8 must be set to 2, and SRTAB must carry the range values.

4.5.11.1 TYPE 10 PROTECTIVE MULTIPLE SHIFTING

This technique is very similar to the method of the Type 2 spindle (See Section 4.5.3.1); but since the speed command is determined differently for the Type 10 spindle, a slightly different approach is used.

In the example below, the shift point is at 25% of maximum RPM in the current range.

	Range 1	Range 2
1	6.5	28
2	270	1200

Assuming option 137 = -0.25, the shift speed in Range 1 is 67 RPM and the shift speed in Range 2 is 300 RPM.

No multiple shifting is required as long as the ranges do not change. Thus, in Range 1 we go directly from 10 to 200 RPM, and conversely. When changing ranges from a speed below the shift point to a speed above the shift point, no multiple shifting is required. Thus, in Range 1 at 10 RPM, a direct shift is made to 1000 RPM in Range 2.

When changing ranges from a speed above the shift point to any speed in another range, multiple shifting must take place. The speed at the shift point current range is output with a 6 second dwell, then the new spindle speed is output. Thus, if in Range 2 at 700 RPM, it is desired to change to 20 RPM in Range 1, the output will be as follows: output 300 RPM in Range 2 with a 6 second dwell, then output the new speed of 20 RPM in Range 1.

A negative value for option 137 specifies the percentage of the maximum RPM in a range to use as a shift point. Note that a negative value for option 137 must be used for a Type 10 spindle when specifying the need for the multiple shifting sequence.

4.5.12 TYPE 11: TABLE LOOKUP

The spindle speed command is produced by issuing the code value at a table position which corresponds to a similar position for the spindle speed. Thus, the commands are produced irrespective of any specified range or any other condition. The speed command table is set up to correspond one-to-one with the spindle speed values. The setup is made in table SRTAB, hence, the maximum number of spindle is 150 and not 300. The table is regarded as one range consisting of twice the number of available spindle speeds, therefore, it is essential to not ever call for Range 2 in any SPINDLE statement.

	Range 1	
1	5	
2	10	
3	15	
4	20	Spindle Speeds
5	25	
6	30	
7	35	
1	8	
2	20	
3	13	
4	14	Spindle Commands
5	1	
6	10	
7	15	

Example: speed = 20; speed command is S14. Speed is 35; speed command is S15. This example has one range of seven values even though in fact there are fourteen values stored in SRTAB. Therefore, option 7 is made one and option 8 is made seven. In setting up SRTAB the spindle speeds must precede the spindle commands.

4.5.13 TYPE 12 SPINDLE: COMMAND EQUAL TO SELECTED SPEED

This type spindle issues a spindle command equivalent to the spindle speed. The postprocessor sets the spindle speed to the next lowest integer whenever the spindle speed is not an exact value in the range from 25 RPM to 199 RPM. For example: S = 28.4; the postprocessor issues S28. For S = 28.7, the postprocessor issues S28.

Similarly, in the range from 200 RPM to 600 RPM, the postprocessor sets the spindle speed to the next lowest increment of 10. For example: S = 418; the postprocessor issues S410; for S = 422, the postprocessor issues S420.

Whenever there is a change in spindle speed which crosses over the 199 RPM speed, the postprocessor issues a dwell block of time as given in option 54. For example, a dwell (if option 54 is nonzero) is issued when going from a speed of 70 RPM to 200 RPM, and also when going from 300 RPM to 180 RPM.

If option 54 is zero, the S code is output in a block by itself.

4.5.14 TYPE 13: EXPANDED QUASI-EIA 3-DIGIT CODE

This type is very similar to Type 4 except that it has been expanded to include more speeds and to be operative for two heads. In brief, the Type 13 spindle has the same S code for the row for all ranges. The code is the 3-digit EIA code for the actual RPM value in Range 1.

Code	Range 1	Range 2
510	10	88
512	12	100
520	20	200
552	52	300
577	77	450

Example: speed = 100 RPM in range 2. The S code is 512 which is the EIA code for the second row of range 1. See Section 7.1 for a description of the EIA conversion method.

4.5.14 TYPE 13: EXPANDED QUASI-EIA 3-DIGIT CODE (cont'd)

The Type 13 spindle requires special attention when setting up the Machine Subroutine, namely:

(1) Make option 7 be the number of ranges per head; for example, two ranges; therefore, option 7 = 2.

(2) Make option 8 be 2.0; put the Range 1 minimum in SRTAB(1), the maximum in SRTAB(2); put the Range 2 minimum in SRTAB(3), and the maximum in SRTAB(4), and so on. This keeps all postprocessor testing consistent.

(3) Set up SRTAB as follows:

SRTAB(1)	=	Range 1 Min.	Head 1
(2)	=	Range 1 Max.	
(3)	=	Range 2 Min.	
(4)	=	Range 2 Max.	
(5)	=	Range 1 Min.	Head 2
(6)	=	Range 1 Max.	
(7)	=	Range 2 Min.	
(8)	=	Range 2 Max.	
STRTPT(9)	=	Number of actual speeds in Range 1, Head 1	
(10)	=	Multiplying Factor Range 1, Head 1	
(11)	=	Multiplying Factor Range 2, Head 1	
(12)	=	Multiplying Factor Range 1, Head 1	
(13)	=	Multiplying Factor Range 2, Head 2	
(14)	=	First Speed Range 1	
(15)	=	Second Speed Range 1	
.			
.			
.			
(179)	=		
(180)	=	SRTAB row number where starting point STRTPT is stored.	

If an NC machine has more speeds than can be stored in SRTAB, Type 13 (instead of Type 4) can be used since each range is some multiple of the corresponding speed of Range 1, therefore, we simply store the values of the first range in SRTAB. The multiple factors are also stored as are other pertinent data. Thus, if a value of Range 2 is to be used, the postprocessor refers to the related value of Range 1 multiplied by the appropriate multiple factor. For example:

$$S_2 = S_1 * R_2$$

4.5.14 TYPE 13: EXPANDED QUASI-EIA 3-DIGIT CODE (cont'd)

where S_1 is the Range 1 speed and R_2 is the Range 2 multiple. The spindle command for all ranges is the 3-digit EIA "Magic Three" code for the range 1 value. Example: say 198 is in Range 2. The corresponding speed in Range 1 is 94. (The Range 2 multiple here is 2.2 since 94 * 2.2 = 198.) The S command made output for the Range 2 spindle speed of 198 is, therefore, S594.

4.5.15 TYPE 14: SPEED CHANGES BY MODE

The Type 14 spindle is a spindle type that is coded as a Type 2 spindle except that M codes provide a mode of speeds consisting of several ranges. Each of the individual modes is programmed as a separate Type 2 spindle. All modes have the same identical S codes. The ranges are consecutively numbered beginning with the low mode ranges. This subroutine tests the current mode and determines if the new range falls within the same mode. If so, no M code is output. If the mode has changed, the appropriate M code for the new mode is output. The table below shows how the modes, ranges, and coding are tied together.

Mode #1	Range 1		Range 2		Range 3	
	Code	Speed	Code	Speed	Code	Speed
M26	10	5	20	14	30	30
	11	6	21	16	31	32
	12	8	22	18	32	34
	13	10	23	20	33	36
	14	12	24	22	34	38
	15	14	25	24	35	40
	16	16	26	26	36	42
	17	18	27	28	37	44

Mode #2	Range 1		Range 2		Range 3	
	Code	Speed	Code	Speed	Code	Speed
M27	10	20	20	56	30	120
	11	24	21	64	31	128
	12	32	22	68	32	136
	13	40	23	80	33	144
	14	48	24	88	34	152
	15	56	25	96	35	160
	16	64	26	104	36	168
	17	68	27	112	37	176

4.5.15 TYPE 14: SPEED CHANGES BY MODE (cont'd)

For the above example, option 47 = 10 and option 31 = 3. When Mode 1 is in effect, a spindle speed of 40 RPM in Range 3 uses a command of S35. The same spindle speed and range value with Mode 2 produces S30 since 40 RPM is below the minimum value of Range 3.

4.5.16 TYPE 15: RATIO BETWEEN RANGES

The Type 15 spindle is designed to accomodate spindle speed tables where there is a direct ratio between spindle speeds and spindle commands. The minimum and maximum speeds for each range are stored in SRTAB(1) through (6), while the ratio between spindle commands and spindle speeds of Range 1, 2 and 3 are stored in SRTAB(7, 8 and 9). No other speeds need be placed in memory. When a speed is requested, it is first checked against minimum and maximum speeds for that particular range. If it is outside these limits, it is set at the appropriate limit. The speed requested is then altered in accordance with option 90. The speed command is then multiplied by the ratio factor to obtain the speed. TABLEM 71 through 76 may also be used for range and direction changes.

Code	Range 1	Range 2	Range 3
S01	5		
S02	10		
S03	15		
S04	20	30.	
S05	25	37.5	100
S06	30	45	120
S07	35	52.5	140
S08	40	60.	160

4.5.16 TYPE 15: RATIO BETWEEN RANGES (cont'd)

Example:

The ratio between spindle command and spindle speeds is for each range as follows:

Range	Ratio
1	5
2	7.5
3	20

Speed requested = 35 RPM.

Range 1
Spindle Command = 35/5 = 7
Spindle Speed = 7 x 5 = 35

Range 2
Spindle Command = 35/7.5 = 4.67

(a) If the closest or next higher speed is desired as expressed by option 90 being set at zero or one, then
Spindle Command = 5.
Spindle Speed = 5 x 7.5 = 37.5

(b) If the next lower speed is desired as expressed by option 90 being set to a -1, then
Spindle Command = 4
Spindle Speed = 4 x 7.5 = 30.

Range 3
In subroutine SPINDL the speed requested is raised to the minimum of that range, i.e., 100 RPM. Therefore,
Spindle Command = 100/20 = 5
Spindle Speed = 5 x 20 = 100.

The Type 15 spindle require special attention in setting up the Machine Subroutine as follows:

(1) Set option 7 = the number of ranges.

(2) Set option 8 = 2.0 the number of testing limits for each range, i.e., minimum and maximum.

4.5.16 TYPE 15: RATIO BETWEEN RANGES (cont'd)

(3) Set SRTAB as follows:

SRTAB(1) = Range 1 Min. Speed
SRTAB(2) = Range 1 Max. Speed
SRTAB(3) = Range 2 Min. Speed
SRTAB(4) = Range 2 Max. Speed
SRTAB(5) = Range 3 Min. Speed
SRTAB(6) = Range 3 Max. Speed
SRTAB(7) = Ratio between Spindle Command and Speed Range 1
SRTAB(8) = Ratio between Spindle Command and Speed Range 2
SRTAB(9) = Ratio between Spindle Command and Speed Range 3

4.5.17 TYPE 16: SPINDLE

This spindle type is like Type 3 except that the speed codes are not unit increasing. For example:

	SPEEDS	
S-CODE	RANGE 1	RANGE 2
00	50	250
02	100	500
05	150	750
15	200	1000
20	250	1250
30	300	1500

The spindle speed is determined by the S code and an associated M code based on spindle range. The following options must be set:

```
OPTAB(7)    =  Number of Ranges
OPTAB(8)    =  Number of Speeds in each Range
OPTAB(19)   =  16.0
```

The spindle table (SRTAB) would be set up as follows where N = Number of speeds/range

```
SRTAB(1)    }
   .        }   Speeds for Range 1
   .        }
SRTAB(N)    }
SRTAB(N+1)  }
   .        }   Speeds for Range 2
   .        }
SRTAB(2N)   }
SRTAB(2N+1) }
   .        }   Speed Codes
   .        }
SRTAB(3N)   }
```

4.5.18 TYPE 17: SPINDLE

Speeds are selected within each range by a three (3) digit S code (s000 thru s999). The S code for any required spindle speed in any range is obtained from the formula:

$$S_N = \frac{S_D - S_{MIN}}{S_{MAX} - S_{MIN}} \text{ X } 1000$$

where:

S_N = S Code Number

S_D = Desired Spindle Speed

S_{MIN} = Minimum Speed in Range

S_{MAX} = Maximum Speed in Range

Some things to consider when setting up the machine subroutine:

(1) Set the maximum and minimum values for each range in SRTAB as follows:

SRTAB(1) = Mimimum value for range 1

SRTAB(2) = Maximum value for range 1

SRTAB(3) = Minimum value for range 2

SRTAB(4) = Maximum value for range 2

. . . etc.

(2) Set Option 7 = number of ranges (No need to set Option 8).

(3) Speed changes within a range are made without stopping the spindle. If it is required that the spindle be stopped before changing ranges, set Option 216 = 1.

4.5.19 TYPE 18: SPINDLE

Type 18 spindle is designed to accommodate spindle tables where the speeds have an associated code number and is a variable spindle type when in SFM, i.e., a percentage of change in speed is considered rather then the next speed in the table. This percent of change should be placed in option 15. If, for example, option 15 is set to 10.0. the speed will be changed by 10%, regardless if that is the next speed in the table or not.

To store the various speeds of the table in memory, the highest speed of range 1 should be stored in SRTAB(1), with the percent of change for range 1 in SRTAB(2). The highest speed for range 2 in SRTAB(3) and percent change for range 2 in SRTAB(4), etc. Options 7, 8, 19, 31, and 47 should also be set. The postprocessor will then store the number of speeds per range as given in option 8, starting with SRTAB(1) and changing by the SRTAB(2) factor and continue for the number of ranges as given in option 7.

Example:

RANGE 1		RANGE 2	
CODE	SPEED	CODE	SPEED
S300	340	S500	1500
S299	337	S499	1485
S298	333	S498	1470
↓	↓		↓
S203	10	S403	45

Set:

SRTAB(1)	=	340.0	Highest speed Range 1
SRTAB(2)	=	.01	% of change between speeds in Range 1
SRTAB(3)	=	1500.0	Highest speed Range 2
SRTAB(4)	=	.01	% of change between speeds in Range 2
OPTAB(7)	=	2.0	Number of ranges
OPTAB(8)	=	98.0	Speeds per range
OPTAB(19)	=	18.0	Spindle type
OPTAB(31)	=	103.0	Increment between code between ranges
OPTAB(47)	=	203.0	First code in Range 1

4.5.20 TYPE 19: SPINDLE

Each time the spindle changes ranges, the postprocessor issues three command blocks. The first block is for a stop M code, the second block is a postprocessor comment which states,

"A SPINDLE SPEED CHANGE OCCURS AT THIS POINT,"

and the third block is a non-motion block which carries the new spindle speed. For example.

(1) N123G04M00

(2) A SPINDLE SPEED CHANGE OCCURS AT THIS POINT

(3) N125G04545

This type spindle is the old type 7 spindle in GECENT II.

4.6 THREADING PROCEDURES

The processing of a thread in the GECENT III postprocessor is not too dissimilar from a normal linear move. In fact, one of the restrictions upon threading is that it be a linear path; furthermore, the thread path cannot exist concurrently with a SFM nor a RAPID mode.

A threading path is segmented when it exceeds the maximum departure--see the comments below regarding the changing maximum departure for extended lead threads.

Although the threading function is basically the same, it is handled in a different manner by the different control systems. The programming instructions for the Mark Centruy 100S is unique from other series 100 controls, and there are programming differences in the 7500 series which are different from each other also. Consult the programming manuals for the particular installation to determine what these restrictions may be. The postprocessor attempts to treat threading in a general way with as few restrictions as possible to the programming method used.

There are three postprocessor statements which control the use and operation of a thread, viz., PITCH, COUPLE, and THREAD. The Part Programming Manual should be consulted for the description of their general applications. In the paragraphs which follow, a brief description of the above postprocessor statements is given as related to postprocessor operation. The use of the word COUPLE is confined to very early versions of the Mark Century line and is not used with controls whose encoder is permanently coupled to the spindle.

The PITCH statement calls for the desired number of threads per inch. The postprocessor converts this value to a lead which is the reciprocal of threads per inch. Lead, measured in inches per thread, is the number of inches a screw will advance when turned through 360 degrees. If the given number of threads per inch is less than 10, the postprocessor uses a five-digit lead for IJK registers which accept 5 digits. For fine threads, more than 10 per inch, the lead consists of six digits, thus reducing the round-off error (if any) by a factor of 10 or more in most cases. This is possible in the control because the leading zero is passed over and only the last five digits are used.

Machines which have a 6-digit IJK register can similarly accept a 7 digit lead for a pitch greater than 10.

4.6 THREADING PROCEDURES (cont'd)

Lead is defined as the distance from any point on the thread of a screw to the corresponding part on an adjacent thread measured parallel to the screw's axis of rotation. Hence, when the number of threads changes from 12 threads per inch to 6 threads per inch, the lead is increasing, since the distance between adjacent threads increases. The words INCR or DECR, as used in the PITCH statement, refer to the increasing or decreasing pitch for linearly variable leads only. It should be remembered that thread lead is the inverse of pitch, therefore an increading pitch will result in a decreasing lead, and vice versa.

The threading rate or rate of change of lead in inches per thread is computed from the equation:

$$F_T = \frac{(L_F^2 - L_I^2)}{2S}$$

where L_F is the final lead, L_I is the initial lead, S is the screw length, and F_T is the threading rate in inches per thread.

Before threading can begin, the encoder must be coupled. Unless this coupling occurs automatically with the threading G code or unless the encoder is permanently coupled to the spindle, the part programmer must couple the encoder device by use of the statement, COUPLE/ON. While the encoder is coupled, the spindle cannot go above the RPM value of option 175, therefore, it is important to uncouple the encoder (for those systems which require it) after a threading sequence is completed. The postprocessor will always print a comment if the spindle speed is ever greater than the option 175 value while the encoder is coupled. If the part programmer should forget to couple the encoder before calling for a thread, the postprocessor will print a warning, but will also continue with the program. However, no warning is given if the part programmer should forget to uncouple the encoder, because the postprocessor is unable to determine whether or not the part programmer wishes to keep the encoder coupled. Since the THREAD statement is one-shot, the encoder should be kept in its coupled position if there are other THREAD statements to be given.

If a dwell time is required while coupling or uncoupling, option 92 must be set to the required dwell time. Note that if the spindle speed must be reduced to the lowest speed before coupling, option 92 must be set negative. The postprocessor then automatically reduces the speed of the spindle but brings it back up to speed after coupling.

4.6 THREADING PROCEDURES (cont'd)

Facing threads are cuts in the workpiece face; in this case the part programmer uses the abscissa axis as the threading axis of rotation, otherwise, part programming is the same for facing threads as it is for turning threads.

The programmer should refer to his particular NC machine threading manual for the exact threading requirements.

For constant lead, no feedrate is issued in the thread block since the threading feedrate is a function of the lead and spindle speed. Sequences such as

```
RAPID
THREAD
```

are not permitted since one condition overrides the other.

Some additional considerations might be required for extended lead threading or other special threading capabilities. These functions are largely dependant upon the control model and NC machine.

There are four major considerations that the postprocessor must recognize:

(1) Lead Type: Is the pitch constant, increasing, or decreasing?

(2) Lead Range: What is the maximum number of inches per thread?

(3) Path Type: Is the path for a constant or tapered thread?

(4) Path Range: What is the maximum departure for the thread?

Each of these items are considered separately in the postprocessor, and the resulting output is contingent upon which items apply.

There are a variety of restrictions which exist for any given threading condition. Among these restrictions are spindle speed maximums which are a function of the number of spindle speed ranges, type of lead, and size of lead. Path departure maximums can also vary as a function of the lead type and size.

4.6 THREADING PROCEDURES (cont'd)

These variations and concomitant restrictions are so numerous that only by descriptive tables can they best be summarized.

In the tables which follow, a summary of threading capability is presented by control tape. Data is given for resolver feedback gearing of 0.1 inch per revolution and 1 mm per revolution only.

TABLE 1 Mark Century Custom 100 Series Control

Resolver Gearing	Preparatory Function	Maximum Head	Maximum Departure	System Format	Maximum Spindle RPM
0.1 $\frac{\text{inch}}{\text{rev.}}$	g36,g37, g38	9.9999 in	9.9999 in 99.9999 in	14 23	60

	g33,g34
	g35
		.99999 in			370(2 range)
					600(3 range)

	g33,g34
	g35
		.099999 in			370(2 range)
					600(3 range)

1 $\frac{\text{mm}}{\text{rev.}}$	g36,g37 g38	99.999 mm	99.999 mm 999.999 mm	23 33	60

	g33,g34,
	g35
		9.9999 mm			370(2 range)
					600(3 range)

	g33,g34,
	g35
		.99999 mm			370(2 range)
					600(3 range)

4.6 THREADING PROCEDURES (cont'd)

TABLE 2 Mark Century 100S Control

Resolver Gearing	Preparatory Function	Maximum Lead	Maximum Departure	System Format	Maximum Spindle RPM
0.1 inch rev.	g38 g33 g39	1.99998 in 0.99999 in .099999 in	9.9999 in	14	200 400 2500
	g28 g23 g29	1.99998 in .99999 in .099999 in	19.9999 in	14	200 400 2500
1 mm rev.	g38 g33 g39	59.998 mm 29.999 mm 2.9999 mm	999.99 mm	32	200 400 2500

TABLE 3 Mark Century 7582 Control

Resolver Gearing	Preparatory Function	Maximum Lead	Maximum Departure	System Format	Maximum Spindle RPM
0.1 inch rev.	g36,g37, g38	9.9999 in	9.9999 in	44 variable	100
	g33,g34, g35	.99999 in			1000*
1 mm rev.	g36,g37 g38	99.999 mm	99999.999mm	53	100
	g36,g37 g38	9.9999 mm			1000*

* When spindle speed is in the range of 450 to 1000 rpm, the maximum programmable lead must be less than .50000 in or 5.0000 mm.

4.6 THREADING PROCEDURES (cont'd)

TABLE 4 Mark Century 7542, 7543-4, and 7544 Controls

Resolver Gearing	Preparatory Function	Maximum Lead	Maximum Departure	System Format	Maximum Spindle RPM
0.1 $\frac{\text{inch}}{\text{rev.}}$	g33	9.99999 in	99.9999 in	24	100
		.99999 in			1000
		.09999 in			5000
1 $\frac{\text{mm}}{\text{rev.}}$	g33	99.9999 mm	999.999 mm	33	
	g33	9.9999 mm			1000
	g33	.9999 mm			5000

Referring to Table 2 for departures on the axis which is parallel to the axis of symmetry and which are greater than 9.9999 inches, the postprocessor outputs the amount by which the departure is in excess of 10 inches. The G code (usually 28, 23, 29 instead of 38, 33, 39) communicates the distinction between the two departure types.

For the extended lead cases (G38 or G28), the postprocessor divides the programmed lead by 2 and outputs the result. The G code communicates the distinction.

When a THREAD statement is given, the postprocessor sets the flag THFLAG to 1, and sets flag ITHTYP to 1 for TURN and to 2 for FACE. The parameter THLEAD is set in subroutine PITCH, and carries the lead value, i.e., PITCH. The flag THMODE is set in subroutine PITCH, and has the values;

= +1, increasing lead

THMODE = 0, constant lead

= -1, decreasing lead

4.6 THREADING PROCEDURES (cont'd)

When the thread motion record is encountered, subroutine GOLINE (or subroutine SEGMNT) tests THFLAG and branches to subroutine THREDO (double entry with THREAD) to generate and output the required threading command block.

Subroutine THREDO is the basic processing subroutine for all lathe threads; (the special subroutine THREDM processes threads for milling machine.)

In order to set up the thread command block, the postprocessor first computes the departures ΔX and ΔY. The threading lead register values are determined as:

TURN	FACE
$I = \lvert THLEAD \rvert *M$	$J = \lvert THLEAD \rvert *M$
$J = \left\lvert \frac{\Delta Y*I}{\Delta X} \right\rvert$	$I = \left\lvert \frac{\Delta X*J}{\Delta Y} \right\rvert$

where M is some multiple of 10 which scales the value of I and J so that leading zeros are shifted out, thereby, allowing a proportionate increase in accuracy. ΔX and ΔY are stored in DBFSEG(3) and (4), respectively, and I and J are stored in DBFSEG(8) and (9), respectively.

The tool tip velocity in IPM is determined as:

$$F_T = S \sqrt{I^2 + J^2} ,$$

where S is the threading spindle speed. It is important to note that this feedrate in IPM is used only for printout purposes. As such, it is stored in DABVAL(11) for printing out as an Absolute value; see Section 3.5.3.2.

An F command is required only for a non-constant lead, i.e., for an increasing or decreasing lead. For these cases the parameter THRATE (which is set in subroutine PITCH), carries the required threading rate, and this value is stored in DBFSEG(11) when the THMODE is non-zero.

Depending on the conditions of extended or non-extended leads, variable departures, and THMODE, the G code is selected from TABLEG and is stored in DBFSEG(2) to complete the setup of the threading command block.

Again, the programmer is referred to the programming instructions furnished with his control system for a more complete definition of threading procedures, restrictions, and limitations.

4.7 AUTOMATIC REINSTATEMENT OF PROGRAM CONDITIONS

Subject to the control of option 145 the postprocessor will automatically reinstate the part program status of previously cancelled functions which may occur on the statements STOP, OPSTOP, and BREAK. Depending upon the setting of option 145, the postprocessor can reinstate the functions in various manners.

When an OPSTOP is encountered, the machine control unit may automatically turn off the spindle and coolant and enter into the lowest feedrate range. The part programmer would therefore have to reprogram all of these conditions after each OPSTOP. However, by use of option 145, the postprocessor can be directed to perform these reinstating chores.

The functions which are automatically reinstated are the tool or turret T code, the spindle speed command, the spindle condition (CLW, CCLW, OFF), the spindle range, the SFM mode and value, the feed or rapid range, and the coolants condition.

Since each of the above items are reinstated, they must obviously be saved for reinstatement whenever a condition is changed. Thus, in subroutines SPINDL, COOLNT, RAPIDO, RAPIDX, TURRET, TOOLNO, and FEDRAT, each time a related part program statement is processed, the postprocessor stores away the condition in the STATE vector which is dimensioned at 12 and is in basic COMMON. The STATE vector has the present assignments:

STATE(1) = T code

STATE(2) = Spindle Command

STATE(3) = Spindle Condition (CLW, CCLW, OFF)

STATE(4) = Spindle Range

STATE(5) = SFM Value or Mode

STATE(6) = M code for Feed or Rapid Range

STATE(7) = Second M code for Feed or Rapid Range

STATE(8) = Coolant Number 1

STATE(9) = Coolant Number 2

4.7 AUTOMATIC REINSTATEMENT OF PROGMAN CONDITIONS (cont'd)

For example, on the statement

```
SPINDL/20, SFM, RANGE, 2, CLW
```

subroutine SPINDL stores in STATE as follows:

```
STATE(3) = 3 (for M03)

STATE(4) = 2

STATE(5) = 20
```

The fact that a non-DMBITS value is in STATE(5) indicates the existence of an SFM mode. Thus, on the statement

```
SPINDL/40, RPM
```

subroutine SPINDL stores in STATE as follows:

```
STATE(2) = 540 (assume 40 RPM = 540 spindle
                command)

STATE(5) = DMBITS.
```

Hence, the STATE vector at any point in time can give the condition status of the part program.

When an OPSTOP, STOP, or BREAK statement is given , the flag STOPON is set to 1 indicating that a "stop condition" exists. Thus, when a motion statement is encountered, subroutine MOTION calls subroutine TSTFLG which tests flag STOPON, and finding it non-zero, calls subroutine RESTAT which outputs the pending conditions of the STATE vector per the specification of option 145.

The postprocessor must consider one special case; it must allow for a new respecification of a condition after a STOP (or OPSTOP or BREAK) and before the motion statement. For example:

```
STOP

SPINDL/40,RPM

COOLNT/MIST

FEDRAT/RANGE,2

GOTO/X,Y,Z
```

4.7 AUTOMATIC REINSTATEMENT OF PROGMAN CONDITIONS (cont'd)

Prior to the GOTO/X,Y,Z statement the postprocessor must output these conditions as given, and yet must not reinstate them again when subroutine RESTAT is called. The argument may be raised: instead of outputting the conditions when given, why not simply store the conditions in STATE and output them automatically when subroutine RESTAT is called? The reasons are: first, the subroutines (SPINDL, COOLNT, FEDRAT, etc) would have to be altered to not output data when a "stop condition" exists; and, second, flushing the conditions out of the STATE vector will most probably output them in a different order than programmed.

To circumvent this problem the postprocessor, therefore, always processes and outputs the data for every statement; but when storing away the condition into the STATE vector, the subroutines test the STOPON flag, and if non-zero, make the stored value in the STATE vector negative. Hence, subroutine RESTAT, when outputting the conditions of the STATE vector, first checks for a negative value of the condition, and, if found to be negative, the subroutine restores the positive value back into STATE, and disregards outputting it.

Subroutine RESTAT also outputs any necessary dwells when reinstating the feed or rapid mode.

4.8 VARIABLE FORMAT BY G CODE

The Mark Century 100M control (and those similar to it) have motion registers whose decimal format changes as a function of the linear G code. For example, the format for the XYZ registers may be 14.0, i.e., REGFOR(3) = 14 for X, REGFOR(4) = 14 for Y, and REGFOR(5) = 14 for Z. However, when a G10 is used, the format changes to 23; the total number of digits is the same as before, but now there are only <u>three</u> digits to the right of the decimal point. Thus, there is a loss of data in the fourth decimal place which, as will be shown, is recoverable but only at the cost of changing the programmed cut path. This may be an intolerable situation making it physically impossible on NC machines with this feature to cut accurately with a G10 code.

The presence of this variable format feature is designated by option 41 being set to zero.

Because of the resultant altered path the postprocessor uses the G10 and maximum departure of 99.999 only for rapid traverse moves. A move greater than 9.9999 inches is automatically segmented so as to be able to use the G01, G11, or G12 codes and maximum departure of 9.9999 inches.

Example: (1) FROM/0,0,0

(2) GOTO/40,40,40

(3) RAPID

(4) GOTO/100,100,100

Statement 2 is segmented into five 8 inch departures using G01. Statement 4 is output with departures of 60 inches using G10.

The technique of recovering the "lost" decimal data can be illustrated by the following example. In effect, the technique follows two basic steps:

(1) Process the incremental data in the normal manner except for a G10 block. For these blocks truncate, but not round, on the 0.001 factor. The XYZ data are issued with this truncation.

(2) The truncation remnant lost in the 0.0001 position is then subtracted from the present point. Hence, when the next point increments are computed, the previously "lost" amount is now included in the new increments.

4.8 VARIABLE FORMAT BY G CODE (cont'd)

Example:

	CL point	Rounded to 0.001	
A.	2.6789231	2.6789	
			G10
B.	23.786802	23.7868	
			G01
C.	30.236685	30.2367	

True Δ_{BA} = 21.1079, but we output a Δ_{BA} = 21.107, and subtract 0.0009 from the present point 23.7868, making it 23.7859 instead of 23.7868. Hence, instead of Δ_{CB} = 6.4499, it is Δ_{CB} = 6.4508. This avoids the accumulative error and places the tool back on the correct path. In this example, if we redetermine the lost CL point from the produced increments we get: 2.6789 + 21.1070 + 6.4508 = 30.2367, the exact point.

Although this technique arrives at the correct point, it's path to get there has been changed from the programmed path. This is demonstrated in Diagram 4.8A where a grossly exaggerated two axes correction is made by the method explained above

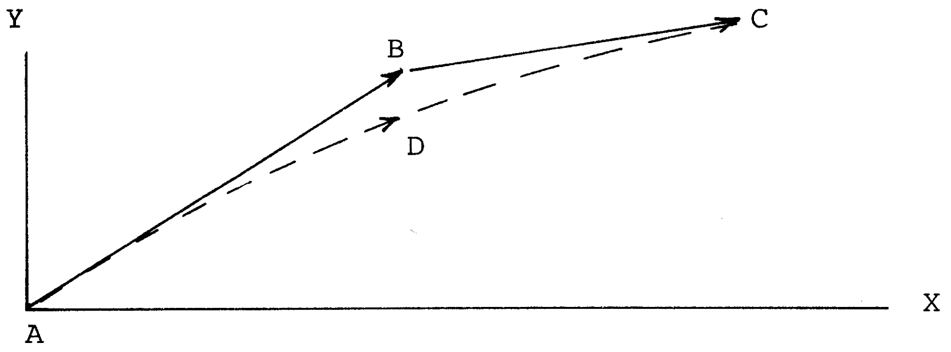


Diagram 4.8A

Paths ABC are the true paths, but path ADC is the result of the use of G10 and subsequent loss and regain of the fourth decimal place data.

4.8 VARIABLE FORMAT BY G CODE (cont'd)

The postprocessor achieves this corrective procedure by changing the departure maximum BIGDEP and STEP size as a function of the current G code. Thus, for a RAPID move greater than 9.9999 inches, BIGDEP remains as 99.999, but step is changed to 10 times its previous value. Assume STEP is 0.0001; it is then changed to 0.001 to conform to the changed decimal requirements of the XYZ format.

In subroutine SELG when option 41 = 0, STEP is changed to 0.001 and the present machine point vector DPRESM is modified to reflect the loss of the fourth decimal place data. Before the next CL tape record is read, DPREVM is set to DPRESM; therefore, the new DPRESM(from the CL record) will create departures that recover the lost data.

When a non-RAPID move greater than 9.9999 inches is given, subroutine GOLINE makes BIGDEP = 9.9999, thereby automatically forcing a segmentation of the path.

In all cases when BIGDEP or STEP are changed, their original values are first saved and then restored before exiting from the subroutine.

The postprocessor assumes an XYZ format of -14.0 for G11 and G01, and a format of -23.0 for G10. Machines which have six-digit registers or greater should ignore this option.

To use this feature the Machine Subroutine must have TABLEG(11) = 10.0, REGFOR(XYZ) =-14.0, OPTAB(41) = 0, and OPTAB(4) = 99.999. IF the G10 is not to be used, then option 4 should be set to 9.9999.

The postprocessor is not structured to accept circular interpolation departures with this type of shifting or floating format. Control systems using circular interpolation must have fixed formats.

When option 41 is set to -1, the postprocessor has the variable format and no corrective action as described above takes place. Departures processed normally for both normal departures (format = 14) and long departures (format = 23) for both RAPID and FEED moves.

When option 41 is set to -2 the variable format takes on a slightly different aspect. No corrective action is required; both RAPID and FEED moves are processed normally, but the normal departure has a format of 14 where as the long departure has a format of 24.

4.9 LEADING ZERO SUPPRESSION

A normal function of the Mark Century numerical control is to drop trailing zeroes, but leading zero suppression is a capability of the 100M control system and certain of the 7500 series control systems.

For the 100M control, leading zero suppression is available only in the motion registers X Y Z, and only when in a G11 or G12 (supersmall) mode. Suppression is obtained by setting option 51 = -2.0.

In a G11 mode, leading zeros to the left of the decimal point are suppressed, while in a G12 mode leading zeros to the left and one zero to the right of the decimal point are suppressed. There is no suppression in the G01 or G10 mode.

Leading zeroes are deleted in the GEPRON routines. Trailing zeroes (for all incremental systems) are always deleted through subroutine CONBCD.

For the 7582 contouring control the pattern of leading zero suppression is extended for all departure G codes greater than G12 up to and including G26, G27, and G28 (format 0744). Option 52 is set equal to -2.

4.10 DWELL BLOCKS

An NC machine dwell can be produced in a variety of ways, but the most common method is through the use of the preparatory function G code G04. This method outputs a command block with a G04 and the required delay time which is normally stored in one of the motion registers, i.e., XYZ registers. For example, the statement

```
DELAY/4
```

may output a command block

```
NxxxG04X4
```

which should produce a 4 second dwell.

In general, milling machines use the X register for the dwell time whereas lathes use the Z register. Since any register might possibly be used for storing the dwell time, the postprocessor refers to option 57 since it specifies the location of the dwell register.

4.10 DWELL BLOCKS (cont'd)

A control which uses this G04 method usually considers the value in the motion register to be ten times smaller than it really is. In the example above, although we call for a 4 second dwell, the control interprets the command block (as given) to be a 40 second dwell. Therefore, the postprocessor scales down the dwell time before outputting the dwell block. This scaling value is given in option 56 whose standard value is 10. Thus, the command block would appear as (assuming REGFOR(3) = 14):

NxxxG04X04

It is important to note, therefore, that a printed dwell time is actually ten times larger than shown.

Dwell times can also be effected by miscellaneous function M codes or other preparatory function G codes. These alternate methods usually produce a fixed or preset dwell time value. An M31 for example, may select preset timer number 1 to produce a 1 second dwell, while an M32 may select preset timer number 2 to produce a 3 second dwell, and so on.

Another fairly common method with linear control systems is to use the F register for producing the dwell time. This method does not normally use G04 or any other code, but simply outputs a command block with an F value which is obtained by dividing the requested dwell time into 60. For example:

DELAY/40

F = 60/40 = 1.5

Assuming REGFOR(11) = 31, the command block is:

NxxxF0015

A precaution must be observed with this method however. Machine tools which produce dwell times through the F-register (no G04), and which also have several dimensional G codes (G01, G10, G11, G12,) require special treatment in the postprocessor. Since the G codes are modal, their respective dimension multiplier can affect the actual dwell time. For example: say the dwell time is to be 3 seconds. If a G10 is in mode, the actual time will be 300 seconds. G01 would give 30 seconds, G11, 3 seconds, etc. Hence, to avoid the undesired long dwell times, the postprocessor selects the smallest range G code available (down to a G11) and outputs it in the dwell block.

4.9 LEADING ZERO SUPPRESSION (cont'd)

Another simple solution if one G code is preferred, e.g., G11, is to set the preparatory function dwell code location accordingly, viz., TABLEG(5) = 11.

All dwell blocks have the command block identification code of +4.

4.11 OPSKIP PROCESSING

NC machines which have the optional skip feature (block delete) must have option 30 set to 1.

When an OPSKIP statement is given, subroutine OPSKIP sets the flag SKPFLG to 1 for ON and 0 for OFF. To indicate that a command block is to be an OPSKIP block, the postprocessor in subroutine OUTPUT makes the sequence number value (BUFSEG(1)) negative.

At output time the postprocessor tests either for a negative sequence number (in GEOUT3 and GEOUT4) or tests the SKPFLG. When the OPSKIP condition is on, the postprocessor in subroutine SETUP adds the skip code SKPCOD to the second location of BCDIMG which is subsequently printed and punched. The parameter SKPCOD is in basic COMMON and is set in subroutine GEPRE. The normal code is a / which is stored in BCD form into SKPCOD.

4.12 CUT AND DWELL TIMES

The cut and dwell times in minutes for incremental systems are printed at the bottom of each page. The parameters CUTIME and DWTIME which are in output COMMON contain the accumulative cut and dwell times, respectively.

The dwell times are accumulated in subroutine GEPRON, but the cut times are computed and accumulated in subroutine CONTUR.

The cut times are computed from the basic relationship

$$T = \frac{S}{60*F},$$

where T is the path cut time in minutes, F is the feedrate in IPM, and S is the path length in inches.

4.12 CUT AND DWELL TIMES (cont'd)

S is computed according to its type of move and interpolation mode:

Non-multiaxis Linear: $S = \sqrt{\Delta X^2 + \Delta Y^2 + \Delta Z^2}$, where the deltas are the machine coordinate departures.

Circular: $S = 0.972D + \frac{0.1737D^2}{R}$,

where R is the circle radius, and D is the departure path length; see Diagram 4.1.12A.

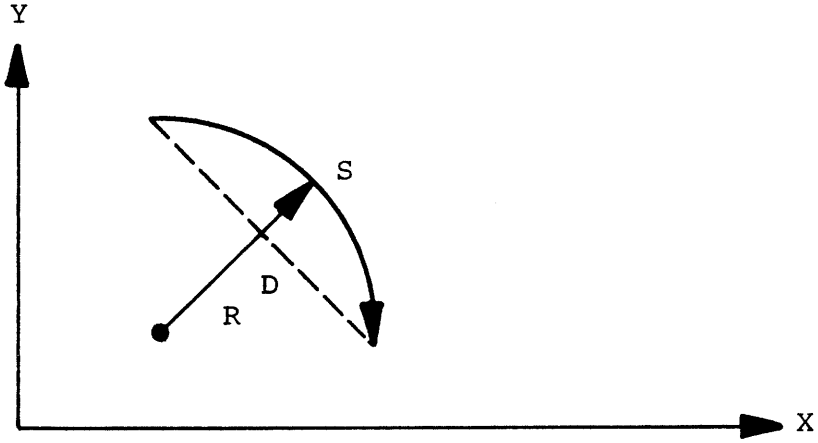


Diagram 4.1.12A

Multiaxis: $S = \Delta x^2 + \Delta y^2 + \Delta z^2$, where Δx, Δy, Δz are the part coordinate departures.

Rotary: $S = \gamma R$, where γ is the angular rotation in radians, and R is the part radius in inches.

Threading: In order to compute the threading time, the following relations are used:

Feedrate = Lead * Spindle Speed,

$$\frac{in}{min} = \frac{in}{rev} * \frac{rev}{min}$$

where Lead is the inverse of the given pitch.

4.12 CUT AND DWELL TIMES (cont'd)

Therefore, thread time T in seconds is

$$T = \frac{\text{Path Length}}{60*\text{Feedrate}}$$

$$= \frac{\sqrt{\Delta X^2 + \Delta Y^2}}{60\,(\text{Lead} * \text{Spindle Speed})}$$

For tapered threads, Lead is the maximum of the given lead and the determined ratioed lead for the other axis.

5.0 POSTPROCESSOR PROGRAM DETAILS

This section covers the programming conventions, techniques, and anatomy of the postprocessor. The internal structure of the GECENT III postprocessor is defined by function and usage, and details are given regarding the location of subroutines, parameters, errors, and so on.

Especial reference is made to the heart of the postprocessor - the Machine Subroutine. Details are given concerning it's make-up, theory of use, and method of application.

This section is essential reading to anyone who plans to work with the GECENT III postprocessor.

5.1 LABELED COMMONS

Each major overlay of the postprocessor has its own labeled COMMON. This has the advantage in that only those COMMON areas which are necessary to a given overlay structure are loaded into memory. Blank (or block) COMMON is not as convenient to use as is a labeled COMMON. However, the APT System COMMON is a blank COMMON.

The labeled COMMON for each major overlay is given below; see a GECENT III postprocessor listing for the parameters included in the labeled COMMON, or see Section 5.1.1, Parameter Definitions.

GECOM *-- this COMMON is the general one used for and by all overlays. It contains the parameters which are used for communication between overlays, and it contains all of the parameters used for general application.

GECBAS* -- this is the basic COMMON required for all types of NC machines, and, in particular, applies mainly to the GEBASE overlay. It is used also in the following overlays: GEINIT, GETERP, GEPOS, GELATH, GEMILL, GEMAXS, and all of the GEOUT overlays. The special overlays GESPIN, GECLAS, GEMFUN, GEFLAM, GEWIND, GEWELD, and so on, also use GECBAS COMMON.

The GECBAS COMMON provides communication between the GEBASE overlay and the NC machine type and output overlays.

GEOUT* -- this labeled COMMON is the communicating region for subroutines in the GEOUT overlays. It contains the basic parameters which any GEOUTn overlay requires. It is also used in some subroutines in GEBASE in order to provide contact between GEBASE and GEOUT.

* These common regions are called D1,D2,I1,I2,S1 and S2 in the 360 version of GECENT. D1,D2,S1 and S2 contain double precision parameters and I1 and I2 contain integer parameters.

5.1 LABELED COMMONS (cont'd)

GECOT3 -- this COMMON is used exclusively for communication between the subroutines of GEOUT3, and to a lesser extent, for communication between certain GEMULT subroutines with GEOUT3.

COMBIN -- this is the basic COMMON required for the GEMULT link. It is used as a communicating area between multihead subroutines throughout GEMULT.

FXAXCM -- this labeled COMMON is used only for the common-axis sequence in the multihead segment of the postprocessor; see Section 3.4.8.2.1.3. It is used for an area of communication between subroutines of the common-axis and other multihead sequences.

The labeled COMMON assignments by overlay are illustrated in Diagram 5.1A.

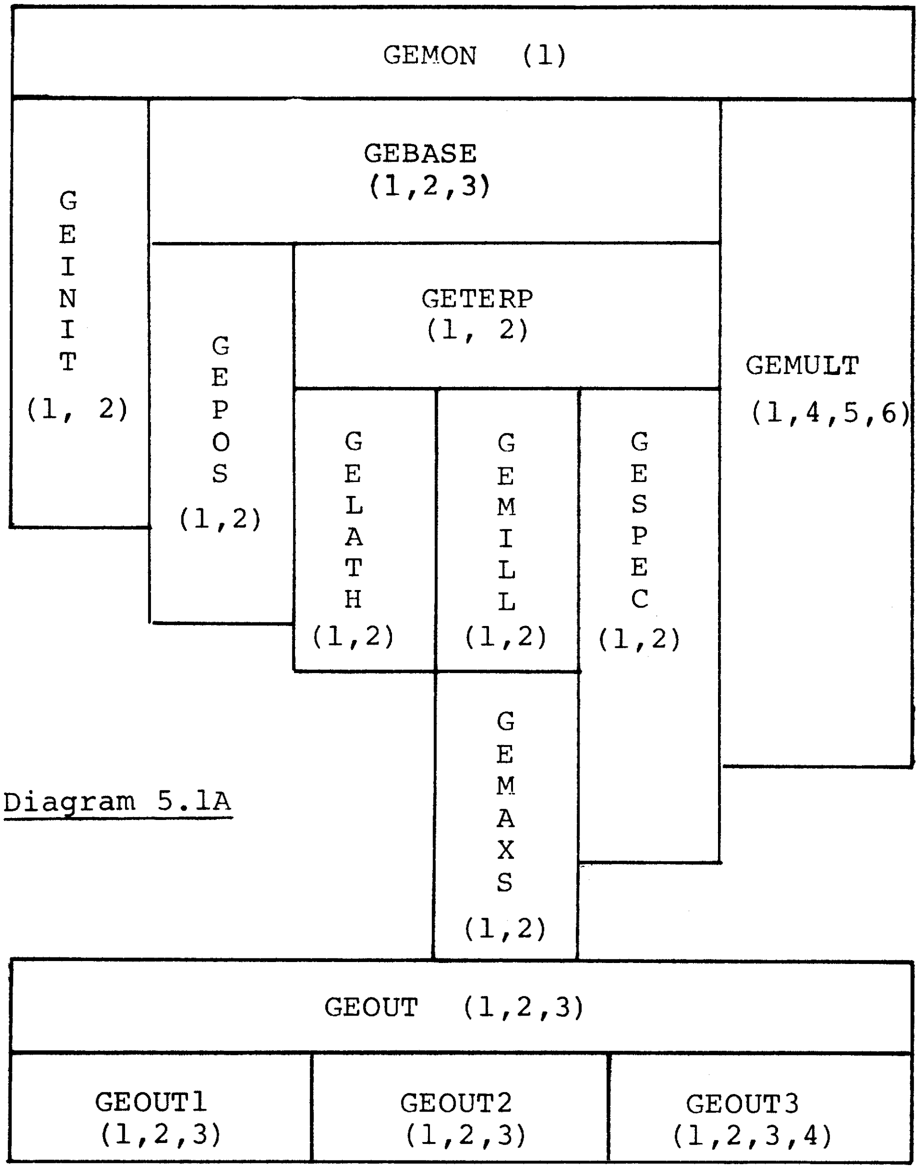


Diagram 5.1A

1. - GECOM

2. - GECBAS

3. - GECOUT

4. - GECOT3

5. - COMBIN

6. - FXAXCM

5.1.1 PARAMETER DEFINITIONS

In this section the postprocessor's program parameters in COMMON are defined according to their purpose, usage, or function. Those parameters not in COMMON are local variables and are defined in the subroutine listing. Dimensioned parameters are given with their dimensions in parentheses; dimensioned parameters, which are ordered, have their symbols and order given. Each COMMON parameter is identified as to which labeled COMMON it belongs. The first labeled common refers to all versions of GECENT except the 360 version; the second name is the labeled common for the 360 version of GECENT.

The subroutine which sets the parameter value is given when possible, but some parameters are set and reset in several subroutines and, hence, all the subroutines cannot be given.

For convenience and easy reference the parameters are listed in alphabetical order.

ABCF1 (FXAXCM)

The flag used in subroutine FXMULT which indicates whether or not the incremental moves for Head 1 have been added to the Head 1 absolute coordinate values:

0 = increments not added,

1 = increments have been added.

ABCF2 (FXAXCM)

The flag used in subroutine FXMULT which indicates whether or not the incremental moves for Head 2 have been added to the Head 2 absolute coordinate values:

0 = increments not added,

1 = increments have been added.

ABS2(3) (COMBIN)

The absolute coordinate system for Head 1. Ordered as X,Y,Z.

5.1.1 PARAMETER DEFINITIONS (cont'd)

ABS3(3) (COMBIN)

The absolute coordinate system for Head 2. Ordered as X,Y,Z.

AMINTL (COMBIN, S2)

The minus tolerance restriction for the secondary head.

ANGLIN (GECBAS, S2)

The tolerance half-angle of the cone used in linearity testing; set up in subroutine LINTOL.

ANGSEL (GECBAS, S2)

The angle selected by the SELECT/ANGLE statement. Set in subroutine SELANG.

ARCANG (GECBAS, S2)

The angle of arc of the circle that a multiaxis circular interpolation motion makes; computed in subroutine PROCQD.

AS2(20,2) (COMBIN)

The first row of AS2 is used for storing the current DBFSEG command block of Head 1; the second row of AS2 is used as storage for segmentation in a combined cut.

AS3(20,2) (COMBIN)

The first row of AS3 is used for storing the current DBFSEG command block of Head 2; the second row of AS3 is used as storage for segmentation in a combined cut.

5.1.1 PARAMETER DEFINITIONS (cont'd)

AXMULT (GECOM)

The flag indicating that a multiaxis mode exists; set in subroutine GEBASE for a Type 9000 record.

BCDIMG(34) (GECOUT)

The array into which a row of DBFSEG is stored after being converted to BCD. This array is then set up for printing and punching.

BCDRG2(20) (GECOT3)

The array containing the BCD title of the NC machine registers for Head 2; the title is printed at the top of each page. Set in subroutine CALCP3.

BCDREG(20) (GECOUT, S2)

The array containing the BCD title of the NC machine registers; the title is printed at the top of each page. Set in subroutine CALCPn. For multihead machines BCDREG is used for Head 1.

BIGDEP (GECBAS, S2)

The linear departure maximum value. It is given in option 4 and set in subroutine ASSIGN.

BUFPRE(20) (COMBIN)

Preliminary buffer which is set up for one head prior to dumping onto the scratch device; ordered the same as DBFSEG.

CDEP (GECBAS, D2)

Departure of the C axis; computed from DPRESM(18) - DPREVM(18).

5.1.1 PARAMETER DEFINITIONS (cont'd)

CIRDIR (GECBAS, S2)

The direction of the circle motion:

0 = CLW

1 = CCLW.

CIRFLG (GECBAS, S2)

Path condition flag:

0 = linear path,

1 = circular path.

Set in subroutine SRFCHK.

CIRRAD (GECBAS, S2)

The radius of the circle. Normally, it is the distance from the part circle center to the tool control point.

CIRSEQ (GECBAS, S2)

Flag indicating that a circle motion is being processed in the circular interpolation mode; set in subroutine GOCIRC.

CLERP (GECBAS, S2)

The clearance plane value as given by a CLRSRF statement: set in art CLRSRF.

CLMPEX (GECBAS, S2)

The flag set in subroutine CLAMP which indicates that an axis has been clamped. If motion is attempted with the clamped axis, IWAVEN is set to 1 and a warning comment is issued.

5.1.1 PARAMETER DEFINITIONS (cont'd)

CLMPFL (GECBAS, S2)

Clamp flag indicating the condition of the clamp:

0 = off,

1 = on.

Set in subroutine CLAMP.

CODE (GECBAS, S2)

The command block identifying code stored in DBFSEG(15); see Section 2.3.2 for details.

CONDFL (GECBAS, S2)

The condition flag.

CRCODE (GECBAS, S2)

The CODE for the current circle motion being processed; it is always ≠ 10, 11, or 12 if on; otherwise, it is 0.

CTRLIN (GECOUT, S2)

The line counter parameter. The number of lines printed are counted so as to know when to complete a page and begin a new one.

CURCYG (GECBAS, S2)

The current canned cycle G code; set in subroutine CYCLE.

CURMAC (GECOM, S1)

The current machine number n as given in the MACHIN/GECENT, n statement. Set in subroutine MACHIN.

5.1.1 PARAMETER DEFINITIONS (cont'd)

CURMOD (GECBAS, S2)

The current positioning mode.

CURNGE (GECBAS, S2)

The current spindle speed range; normally set in subroutine SPINDL.

CURNTZ (GECBAS, S2)

The current Z value; used during the positioning mode when RETRCT with a CLERP is used. The retained z value is the given, unaltered value.

CUST(5) (GECOM, S1)

The customer parameters in COMMON: see Section 5.1.2.

CUTTER (GECBAS, S2)

Cutter flag designating current cutter in use:

1 = ball tool,

0 = non-ball tool.

CUTRAD (GECBAS, S2)

The radius of the current cutter; set in subroutine GEBASE for a Type 6000 record.

CYCFLG (GECBAS, S2)

Cycle flag as used for positioning machines:

0 = off,

1 = on, which means a programmed cycle is in mode.

5.1.1 PARAMETER DEFINITIONS (cont'd)

DABVAL(20) (GECOUT)

The print vector set up with the Absolute Values of the registers; ordered the same as DBFSEG.

DATACL(246) (GECBAS, D2)

Input storage array for one record read from the CL tape; primarily used for floating point references.

DATCIR(3) (GECBAS, D2)

The x,y,z coordinates of the part circle center; set in subroutine SRFCHK.

DBFSEG(30) (GECBAS, D2)

The buffer set up for each command block which is to become output. The fixed order of DBFSEG is:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	30
N	G	X	Y	Z	A	B	I	J	K	F	S	T	M	CODES	E	E	C	IPM	RPM	

DBLNKS (GECOM, D1)

Parameter containing a full word of BCD blank characters; set up in subroutine INIT.

DBUFER(6,6) (GECBAS, D2)

A buffer for general usage. It is also used for containing the quadrant intersection points for a circular interpolation move; setup in subroutine QUADET, it then has the order x,y,z,i,j,k.

DCRPT1(3) (GECBAS, D2)

The beginning point of the circle to be processed with circular interpolation; ordered as x,y,z. Normally set in subroutine GOCIRC.

5.1.1 PARAMETER DEFINITIONS (cont'd)

DCRPT2(3) (GECBAS, D2)

The final point of the circle to be processed with circular interpolation; ordered as x,y,z. Normally set in subroutine GOCIRC.

DEPA (GECBAS)

The departure of the A axis; computed form PRESMP(4) - PREVMP(4) in subroutine ROTMOV.

DEPB (GECBAS, D2)

Departure of the B axis; computed from DPRESM(5) - DPREVM(5) in subroutine ROTMOV.

DEPX (GECBAS, D2)

The departure of the X axis; computed from DPRESM(1) - DPREVM(1) in subroutine DEPART.

DEPY (GECBAS, D2)

The departure of the Y axis; computed from DPRESM(2) - DPREVM(2) in subroutine DEPART.

DEPZ (GECBAS, D2)

The departure of the Z axis; computed from DPRESM(3) - DPREVM(3) in subroutine DEPART.

DIR (FXAXCM)

The direction flag which indicates the direction of motion of multiheads which share a common axis:

+1 = positive direction,

-1 = negative direction.

5.1.1 PARAMETER DEFINITIONS (cont'd)

DMBITS (GECOM, Dl)

Parameter which indicate that an empty or null setting exists; equal to -40404040.0; set in subroutine INIT.

DMS (GECBAS, S2)

Parameter used as a linear departure calculation tolerance; it is determined by (BIGDEP-STEP) in subroutine ASSIGN.

DPBITS (GECOM, D1)

The parameter containing the plus value of DMBITS. Set in subroutine INIT.

DPATH (GECBAS, D2)

The computed length of the path.

DPRESM(6) (GECBAS, D2)

The present point in machine coordinates, ordered as XYZABC.

DPRESP(6) (GECBAS)

The present point in part coordinates, ordered as xyzijk.

DPREVM(6) (GECBAS, D2)

The previous point in machine coordinates, ordered as XYZABC.

DPRESP(6) (GECBAS, D2)

The previous point in part coordinates, ordered as xyzijk.

5.1.1 PARAMETER DEFINITIONS (cont'd)

DPARTNO(6) (GECOUT, D2)

The array which contains the first six words of the first PARTNO for printing as a title on each page.

DTEMP(10) (GECOM, D1)

Temporary storage area for floating point numbers; must be used exclusively within a subroutine.

DTRANS(3) (GECBAS, S2)

The array of incremental values used for translating the part as given by a TRANS statement. Set in subroutine TRANS. Ordered as xyz.

ENCODE (GECBAS, S2)

The flag which gives the condition of the threading encoder:

0 = encoder uncoupled,

1 = encoder coupled.

ENDFLG (GECBAS, S2)

The flag which indicates that an END has been given:

0 = no END given,

1 = END given,

-1 = no FROM point was given after the END.

EPSLON (GECOM, S1)

The epsilon value used as a tolerance for equality in various tests. Given in option 5 and set in subroutine ASSIGN.

5.1.1 PARAMETER DEFINITIONS (cont'd)

ERROR (GECOM, S1)

The error number which identifies the type and condition of a fatal error; see Section 5.7.

FACDEP (GECOM, S1)

The unit system departure constant: it is 10 for the inch and 100 for the metric system; set in subroutine ASSIGN.

FCOMAX (GECOM, S1)

The feedrate command maximum value. Given in option 24 and set in subroutine ASSIGN.

FDHOLD (GECBAS, S2)

The fiven feedrate value either in IPM or IPR. The programmed feedrate value is retained free of any other effects. Set in subroutine FEDRAT.

FDMHD(2) (FXAXCM)

The saved M code for Head 1 or 2. The M code is saved only if it is for a gear shift to the feedrate or rapid ranges.

FEDIPM (GECBAS, S2)

The current feedrate in IPM: normally set in subroutine FEDRAT.

FEDIPR (GECBAS, S2)

The current feedrate in IPR: normally set in subroutine FEDRAT.

5.1.1 PARAMETER DEFINITIONS (cont'd)

FEED(2) (COMBIN)

FIRST (GECBAS, S2)

The "first time" flag indicating that the first command block has been processed for output.

FL1 (GECOM, S1)

Floating point one; set in subroutine INIT.

FL2 (GECOM, S1)

Floating point two; set in subroutine INIT.

FL3 (GECOM, S1)

Floating point three; set in subroutine INIT.

FL4 (GECOM, S1)

Floating point four; set in subroutine INIT.

FL5 (GECOM, S1)

Floating point five; set in subroutine INIT.

FL10 (GECOM, S1)

Floating point ten; set in subroutine INIT.

FL100 (GECOM, S1)

Floating point one hundred; set in subroutine INIT.

5.1.1 PARAMETER DEFINITIONS (cont'd)

FL360 (GECOM, S1)

Floating point 360; set in subroutine INIT.

FLGCOM (GECBAS)

FLGDIR (GECBAS, S2)

The flag specifying the direction of rotation to use.

FLM1 (GECOM, S1)

Floating point minus one; set in subroutine INIT.

FLZ (GECOM, S1)

Floating point zero; set in subroutine INIT.

FLONKL (GECBAS, S2)

The flag indicating the coolant condition:

0 = off

1 = on.

Set in subroutine COOLNT.

5.1.1 PARAMETER DEFINITIONS (cont'd)

FLONSP (GECBAS, S2)

The flag indicating the spindle condition:

0 = off,

1 = on.

Normally set in subroutine SPINDL.

FLRPON (GECBAS, S2)

The RAPID Mode-on flag:

0 = off;

+1 = on, when RAPID is one-shot;

-1 = on, when RAPID is modal.

Set in subroutines RAPIDO and RAPIDX.

FLSFON (GECBAS, S2)

The flag indicating the existence of a SAFETY mode condition:

0 = off,

1 = on.

Set in subroutine SAFETY.

FRAPID (GECBAS, S2)

The rapid traverse feedrate in IPM. It normally is negative to indicate the existence of a rapid mode, but it is positive when a rapid mode does exist but no gear shifting is required. Given in option 46.

5.1.1 PARAMETER DEFINITIONS (cont'd)

FRMAX (GECOM, S1)

The feedrate maximum in IPM. Given in option 25 and set in subroutine ASSIGN.

FRMIN (GECOM, S1)

The feedrate minimum in IPM. Given in option 48 and set in subroutine ASSIGN.

FRMOD (GECBAS, S2)

The feedrate mode:

0 = IPM,

1 = IPR.

Set in subroutine FEDRAT.

GDIMUL (GECBAS, S2)

The dimension multiplier associated with the currently selected G code. Given in option 178 and set in subroutines SELG. SELGCR, or SELGRO.

GMFORM(30) (GECOT3)

The table of register format values for Head 2, given in the Machine Subroutine , are analogous to and ordered in a manner similar to the REGFOR table.

GMHBUF(40) (GECOT3)

The storage array of the merged heads (or single head only) when being set up as a command block for output by GEOUT3. The order is the same as DBFSEG. Head 1 uses GMHBUF (1-20) and Head 2 uses GMHBUF(21-40).

5.1.1 PARAMETER DEFINITIONS (cont'd)

GMWORD(20) (GECOT3)

The BCD table of register codes for Head 2, given in the Machine Subroutine, are analogous to and ordered in a manner similar to the REGSTR table.

GRPLOD (GECBAS, S2)

The gripper which is to be used when loading a tool.

GRPSLC (GECBAS, S2)

The gripper which is selected for the next loading of the tool.

H1FLAG (FXAXCM)

Head 1 flag which indicates that part of the original Head 1 tape record remains to be processed.

H2FLAG (FXAXCM)

Head 2 flag which indicates that part of the original Head 2 tape record remains to be processed.

H1X1 (FXAXCM)

The X value of the beginning point of the Head 1 path.

H1X2 (FXAXCM)

The X value of the end point of the Head 1 path.

H1Y1 (FXAXCM)

The Y value of the beginning point of the Head 1 path.

5.1.1 PARAMETER DEFINITIONS (cont'd)

H1Y2 (FXAXCM)

The Y value of the end point of the Head 1 path.

H1Z1 (FXAXCM)

The Z value of the beginning point of the Head 1 path.

H1Z2 (FXAXCM)

The Z value of the end point of the Head 1 path.

H2X1 (FXAXCM)

The X value of the beginning point of the Head 2 path.

H2X2 (FXAXCM)

The X value of the end point of the Head 2 path.

H2Y1 (FXAXCM)

The Y value of the beginning point of the Head 2 path.

H2Y2 (FXAXCM)

The Y value of the end point of the Head 2 path.

H2Z1 (FXAXCM)

The Z value of the beginning point of the Head 2 path.

H2Z2 (FXAXCM)

The Z value of the end point of the Head 2 path.

5.1.1 PARAMETER DEFINITIONS (cont'd)

HEADGB (GECOT3)

The head flag which indicates the current operative head.

HEDALL (GECBAS, S2)

Flag set in subroutine SELALL to designate feed and rapid on all axes;

0 = not set,

1 = set.

HSTEP (GECOM)

The parameter containing the half-step size. Determined by (STEP * 0.499999) and set in subroutine ASSIGN.

IADRET (GECBAS, I2)

The return flag which indicates the condition of A/D sequence.

IBLANK (GECOUT)

The parameter containing blanks.

ICLDAT(20) (GECBAS, I2)

The integer array containing the first twenty items of data of a CL record. Used primarily for non-floating point references.

5.1.1 PARAMETER DEFINITIONS (cont'd)

ICIRLN (COMBIN)

The Circle-Line flag:

-1 = Head 1 is linear and Head 2 is circular,

0 = both heads are linear,

+1 = Head 1 is circular and Head 2 is linear,

+2 = both heads are circular.

ICODE (GECBAS) - also (COMBIN)

The integer equivalent of CODE plus one for Head 1.

ICYTYP (GECBAS, I2)

The current cycle environment or type as called for by a CYCLE statement:

0 = cycle OFF

1 = FACE,

2 = BORE,

3 = TAP,

4 = THRU,

5 = DEEP,

6 = MILL,

7 = OUT,

8 = IN,

9 = DRILL,

10 = BORE,MANUAL

11 = BORE, DWELL

12 = BORE, DRAG

13 = BORE, DWELL, DRAG

14 = BORE, DWELL, MANUAL

Set in subroutine CYCLE.

5.1.1 PARAMETER DEFINITIONS (cont'd)

IDAB(10) (COMBIN)

The general vector used for flags:

IDAB(1) Flag used by the SAFETY command when the secondary head is removed from the combined cut.

IDAB(2) Flag used in subroutine GMOUT for Head 1.

IDAB(3) Flag used in subroutine GMOUT for Head 2.

IDAB(4 thru 10) (Not presently used).

IDLINE (GECOUT, I2)

The parameter used in GEOUT2 which is set up with the ABS or OPR line BCD identification title; set up in subroutine GEPRO2.

IDWLFL (GECBAS, I2)

Dwell flag which is set to 1 when a SPINDL/DWELL is given; set in subroutine SPINDL. The flag, when = 1, indicates that DBFSEG(2) contains the spindle-dwell G code.

IFDRNG (GECBAS, I2)

The number of the current feedrate range. Normally set in subroutine FEDRAT.

IFIX (FXAXCM)

The index which identifies the common axis shared by both heads; set in subroutine GMINIT.

IFIX = 1 for X, 2 for Y, 3 for Z.

5.1.1 PARAMETER DEFINITIONS (cont'd)

IGEFLG (GECOM, I1)

A general flag for one-time communication between overlays. In communicating between GEMON and GEINIT, the IGEFLG indicates whether or not to call in GEPLAD:

0 = no,

1 = yes.

IHD1(6) (COMBIN)

The shuffle vector for relocating XYZIJK for Head 1 according to options 59 and 60.

IHD2(6) (COMBIN)

The shuffle vector for relocating XYZIJK for Head 2 according to options 59 and 60.

IHEAD (GECBAS, I2)

The integer equivalent of the HEADGB flag.

IKLMOD (GECBAS, I2)

The coolant mode flag set in subroutine COOLNT:

1 = TAP,	5 = SADDLE,
2 = MIST,	6 = FRONT,
3 = FLOOD,	7 = REAR.
4 = Normal On,	

Set in subroutine ASSIGN.

5.1.1 PARAMETER DEFINITIONS (cont'd)

INDFR (GECOM, I1)

The index location of SRTAB where the first feedrate value is stored; see Section 5.6. Set in subroutine ASSIGn. See option 174.

INDIC (GECBAS, I2)

The integer flag equivalent of CURMAC, the current Machine Number.

INDPTI (GECBAS, I2)

Loop index for selecting the XYZIJK values from DATACL. Set in subroutine MOTION.

INDPTS (GECBAS, I2)

Index for selecting points of motion records (Type 5000) from CLOATA. Set in subroutine GEBASE.

INTZ (GECOM, I1)

Integer zero; set in subroutine INIT.

INT1 (GECOM, I1)

Integer one; set in subroutine INIT.

INT2 (GECOM, I1)

Integer two; set in subroutine INIT.

INT3 (GECOM, I1)

Integer three; set in subroutine INIT.

5.1.1 PARAMETER DEFINITIONS (cont'd)

INT4 (GECOM, I1)

Integer four; set in subroutine INIT.

INT5 (GECOM, I1)

Integer five; set in subroutine INIT.

INT6 (GECOM, I1)

Integer six; set in subroutine INIT.

INT7 (GECOM, I1)

Integer seven; set in subroutine INIT.

IORDER(30) (GECOM, I1)

The vector which directs the postprocessor to reorder the elements of a DBFSEG command block in a sequence specified by the IORDER array; see Section 5.6. Set in the Machine Subroutine.

IPARK1 (FXAXCM)

The flag to determine if Head 1 has been parked:

0 = not parked,

1 = parked.

IPARK2 (FXAXCM)

The flag to determine if Head 2 has been parked:

0 = not parked,

1 = parked.

5.1.1 PARAMETER DEFINITIONS (cont'd)

IPAGE (GECOUT, I2)

The flag which designates that the titles have been printed to begin a new page:

0 = print titles,

1 = do not print titles.

IPGCTR (GEOUT, I2)

The page counter and value which is printed for the page number.

IPITCH (GECBAS, I2)

The value at which the decimal point changes from 4 to 5.

IPLANE (GECBAS, I2)

Flag which indicates the plane of the given circle:

0 = XY plane;

1 = ZX plane;

2 = YZ plane.

Set in subroutine SRFCHK.

IPRINT (GECOT3)

The flag which tells GEOUT3 which title to print:

1 = Incremental,

2 = Absolute,

3 = Operator's.

5.1.1 PARAMETER DEFINITIONS (cont'd)

IRETN (GECBAS, I2)

The integer return flag which gives the condition of the program after a record has been read from tape.

ISAFLG (COMBIN)

The flag indicating whether or not the secondary head has been removed from the combined cut:

0 = no,

1 = yes.

ISAFMD (GECBAS, I2)

The flag designating the current SAFETY mode:

0 = OFF,

1 = FACE,

2 = TURN,

3 = BORE,

Set in subroutine SAFETY.

ISFMOD (GECBAS, I2)

The given SFM radius mode set in subroutine SPINDL:

0 = constant radius

1 = X axis radius,

2 = Y axis radius,

3 = Z axis radius.

5.1.1 PARAMETER DEFINITIONS (cont'd)

ISHUFL (GECOUT, I2)

The shuffle flag which indicates whether or not shuffling of registers is necessary:

0 = no shuffling,

1 = shuffle.

See option 59.

ISHVEC(6) (GECOUT, I2)

The shuffle vector which reorders the motion register alphabetic assignment according to options 59 and 60. Set in subroutine DECODE. Standard order is XYZIJK.

ISPDRO (GECBAS, I2)

The row of the spindle speed table SRTAB at which the given spindle speed was selected.

ISPTYP (GECBAS, I2)

The type of spindle used on the NC machine; it is equal to option 19 plus 1; set in subroutine ASSIGN.

ISRNGE (GECBAS)

The number of the range in which the current spindle speed lies; normally set in subroutine SPINDL.

IS21 (COMBIN)

The counter specifying the number of rows for Head 1 to be processed in GEMULT.

IS22 (COMBIN)

(Not presently used).

5.1.1 PARAMETER DEFINITIONS (cont'd)

IS23 (COMBIN)

The present n value for Head 1 from OP/n.

IS24 (COMBIN)

(Not presently used).

IS31 (COMBIN)

The counter specifying the number of rows for Head 2 to be processed in GEMULT.

IS32 (COMBIN)

(Not presently used).

IS33 (COMBIN)

The present n value for Head 2 from OP/n.

IS34 (COMBIN)

(Not presently used).

ITEMP(5) (GECOM, I1)

Temporary storage for integer values; must be used exclusively within a subroutine.

5.1.1 PARAMETER DEFINITIONS (cont'd)

ITHTYP (GECBAS, I2)

The thread type flag:

1 = TURN,

2 = FACE.

Set in subroutine THREAD.

IXSTOR (GECOUT, I2)

The index location of where to store the dwell time after register shuffling has been done. Set in subroutine DECODE.

JAX (FXAXCM)

The index which identifies one of the two non-common axes not shared by both heads; set in subroutine GMINIT.

JCODE (COMBIN)

The integer value of CODE for Head 2.

KAX (FXAXCM)

The index which identifies one of the two non-common axes not shared by both heads; set in subroutine GMINIT.

KEOF (COMBIN)

The end-of-file indicator for Heads 1 and 2.

KTR (GECBAS, I2)

A general counter in core. Used to give the number of rows stored in DBUFER during a circular interpolation sequence; set in subroutine QUADET.

5.1.1 PARAMETER DEFINITIONS (cont'd)

LSTCOL (GECOUT, I2)

The last print column in which data can be printed. Set in CALCPn.

LSTPLN (GECBAS, I2)

The flag indicating the plane in which the last circle lay; see parameter IPLANE.

MAFORK (GECBAS, I2)

The multiaxis fork which calls for the particular set of transforms:

0 = no transforms called for,

1 = calls for the inverse transforms,

2 = calls for the direct transforms.

MAXES (GECBAS, I2)

Flag which specified the machine axes according to option 22:

1 = YZ,

2 = ZX,

3 = XY,

4 = XYZ.

Set in subroutine ASSIGN.

MCHCON (GECBAS, I2)

The flag which indicates the particular type of special function which must be performed in the Machine Subroutine MACFUN; see Section 5.6.1 for the method and list of condition values.

5.1.1. PARAMETER DEFINITIONS (cont'd)

MODFLG (GECBAS, I2)

Flag in subroutine MODE which established the range of the head.

MODPOS (GECBAS, I2)

The positioning mode flag:

0 = OFF,

1 = FINE,

2 = NOBACK,

3 = COARSE,

4 = CORMIL,

Set in subroutine POSITN.

MULTHD (GECOM, I1)

The flag indicating the existence of a multihead processing mode:

0 = single head operation,

1 = multihead operation.

NAXES (GECBAS)

The number of axes considered by the postprocessor:

= 3 for non-multiaxis,

= 5 if MULTAX is given.

Set in subroutine GEBASE.

5.1.1 PARAMETER DEFINITIONS (cont'd)

NCOM (GECBAS, I2)

The number of component items to select from a motion CL record.

NCOM = 3 for non-multiaxis to select xyz values,

= 6 for multiaxis to select xyzijk values.

Set in subroutine GEBASE.

NFP(26) (GECOUT, I2)

The column vector containing the final print positions for each register on the machine. The order is the same as for DABVAL. Set in subroutine CACLPn.

NFP2(20) (GECOT3)

The column vector containing the final print positions for each register on the machine for Head 2. Set in subroutine CALCP3.

NIP(26) (GECOUT)

The column vector containing the initial print positions for each register on the machine. The order is the same as for DABVAL. Set in subroutine CALCPn.

NIP2(20) (GECOT3)

The column vector containing the initial print positions for each register on the machine for head 2. Set in subroutine CALCP3.

NIPA(20) (GECOUT)

The column vector containing the initial print position for each register printed in the Absolute format. Set in subroutine CALCPn.

5.1.1 PARAMETER DEFINITIONS (cont'd)

NIPA2(30) (GECOT3)

The column vector containing the initial print position for each register printed in the Absolute format for Head 2. Set in subroutine CALCP3.

NOCHAR (GECOUT, I2)

The number of BCD characters in a given command block.

NOP (GECBAS, I2)

The n operation value as given by OP/n statement. Set in subroutine OPCODE.

NOPTS (GECBAS, I2)

The number of points (x,y,z) in a CL tape motion record. Set in subroutine GOCIRC.

NOSEG (GECBAS, I2)

The number of segments required to complete a saddle move.

NOW (GECOT3)

The flag which indicates whether or not GEOUT3 should conclude the command block with an EOF code:

0 = conclude with an EOF code,

1 = do not conclude with an EOF code.

NPR(26) (GECOUT)

The column vector containing the number of right decimal places for each register on the machine. The order is the same as for DABVAL. Set in subroutine CALCPn.

5.1.1 PARAMETER DEFINITIONS (cont'd)

NPR2(20) (GECOT3)

The column vector containing the number of right decimal places for each register on the machine for Head 2. Set in subroutine CALCP3.

NPT(26) (GECOUT, I2)

The column vector containing the total number of digits for each register on the machine. The order is the same as for DABVAL. Set in subroutine CALCPn.

NPT2(20) (GECOT3)

The column vector containing the total number of digits for each register on the machine for Head 2. Set in subroutine CALCP3.

NPTA(20) (GECOUT, I2)

The column vector containing the total number of digits for each register printed in the Absolute format. Set in subroutine CALCPn.

NPTA2(20) (GECOT3)

The column vector containing the total number of digits for each register of Head 2 printed in the Absolute format. Set in subroutine CALCP3.

NRNGES (GECBAS, I2)

The number of spindle speed ranges. Given by option 7 and set in subroutine ASSIGN.

NRORNG (GECBAS, I2)

The number of rows per each spindle speed range. Given by option 8 and set in subroutine ASSIGN.

5.1.1 PARAMETER DEFINITIONS (cont'd)

NWPR (GECBAS, I2)

The number of words per CL tape record. Set by the tape reading subroutine.

OPRVAL(20) (GECOUT)

The print vector setup with the Operator Values of the registers; ordered the same as DBFSEG.

OPTAB(250) (GECOM)

The option table which specifies the conditions for the given machine. Initialized in subroutine STDMAC and reset in the Machine Subroutine.

ORGIN(6) (GECOUT, S2)

The array containing the current values of X,Y,Z,A,B,C given in the ORIGIN statement. Set in subroutines ORIGIN and GEPR03.

OUTFLG (GECBAS, S2)

The flag which indicates that the first output has already occurred:

0 = no output yet,

1 = output occurred.

OUTX1 (FXAXCM)

The absolute value of X for the Head 1 parking position.

OUTX2 (FXAXCM)

The absolute value of X for the Head 2 parking position.

5.1.1 PARAMETER DEFINITIONS (cont'd)

OUTY1 (FXAXCM)

The absolute value of Y for the Head 1 parking position.

OUTY2 (FXAXCM0

The absolute value of Y for the Head 2 parking position.

OUTZ1 (FXAXCM)

The absolute value of Z for the Head 1 parking position.

OUTZ2 (FXAXCM)

The absolute value of Z for the Head 2 parking position.

OVCVAL (GECBAS, S2)

The over center value (n) given in the statement OVRCNT/n. Set in subroutine OVRCNT.

PARTID (GECOM, S1)

The parameter which contains the first 6 BCD characters of the first PARTNO statement.

PLUSTL (COMBIN)

The plus tolerance restriction for the secondary head.

POSMAG (GECBAS, S2)

The next position of the tool magazine.

PREVF (GECOUT, S2)

The previous feedrate command.

5.1.1 PARAMETER DEFINITIONS (cont'd)

PREVF1 (GECOT3)

The previous feedrate command of Head 1.

PREVF2 (GECOT3)

The previous feedrate command of Head 2.

PREVG (GECOUT, S2)

The previous G code.

PREVG1 (GECOT3)

The previous G code of Head 1.

PREVG2 (GECOT3)

The previous G code of Head 2.

PREVN (GECBAS, S2)

The previous sequence number value.

PREVS (GECBAS, S2)

The previous spindle speed command.

PREVTL (GECBAS, S2)

The previously used tool number.

PREVS1 (GECOT3)

The previous spindle speed command of Head 1.

5.1.1 PARAMETER DEFINITIONS (cont'd)

PREVS2 (GECOT3)

The previous spindle speed command of Head 2.

PREVX (GECOUT, S2)

The previous value of absolute X.

PREVY (GECOUT, S2)

The previous value of absolute Y.

PREVZ (GECOUT, S2)

The previous value of absolute z.

PRIMHD (COMBIN)

The primary head flag.

PROGK (GECBAS, S2)

The progression constant derived from the spindle speed table and used in the SFM sequence.

PT (GECBAS, S2)

The current point under consideration during a segmentation sequence.

RADGIV (GECBAS, S2)

The parameter which contains the given radius r of the circle as given in a SELECT/RADIUS, r statement.

5.1.1 PARAMETER DEFINITIONS (cont'd)

RADLIN (GECBAS, S2)

The radius r of the tolerance sphere (or cylinder) given in the LINTOL/r statement. Set in subroutine LINTOL.

RAPFED (GECBAS, S2)

The flag which indicates whether or not to issue the feedrate range gear shifting M codes:

0 = yes,

1 = no.

Set in subroutine TSTFLG.

RAPFLG (GECBAS, S2)

The flag which specifies that a RAPID statement has been given:

0 = not given,

1 = given.

Set in subroutines RAPID, RAPIDO, and RAPIDX.

RAPLOW (GECBAS, S2)

The flag which indicates that the low range rapid traverse is in mode:

0 = off,

1 = on.

Set in subroutine RAPIDO.

5.1.1 PARAMETER DEFINITIONS (cont'd)

RAPRNG (GECBAS, S2)

The rapid traverse range:

1 = lowest range,

2 = highest range.

Set in subroutine RAPIDO.

RDPART(75) (GECOUT, S2)

The array which contains the readable PARTNO which is punched into each new tape reel.

REELNO (GECOUT, S2)

The reel number as used during a BREAK sequence. The reel number is printed and punched in readable format.

REFATL (GECBAS, S2)

The turret distance (parallel to the spindle) from the turret center to the centerline of the tool. Given in a TURRET statement and set in subroutine TURRET. These parameters apply only to Head 1 during multihead processing.

REFBTL (GECBAS, S2)

The turret distance (perpendicular to the spindle) from the turret center to the tool tip. Given in a TURRET statement and set in subroutine TURRET. These parameters apply only to Head 1 during multihead processing.

5.1.1 PARAMETER DEFINITIONS (cont'd)

REFSYS (GECBAS, S2)

Flag used in subroutine ROTABI in conjunction with a ROTREF modifier:

0 = the rotations are based upon the true coordinate system;

1 = the rotations are based upon the rotated reference system.

REGFOR(20) (GECOM, S1)

The table of values which specify the decimal format of each register on the machine. Ordered the same as DBFSEG. Initialized in STDMAC and reset in the Machine Subroutine.

REGSTR(20) (GECOM, S1)

The table of BCD register assignments. Initialized in subroutines STDMAC and reset in the Machine Subroutine. Ordered the same as DBFSEG.

RESETF (GECBAS, S2)

The RESET flag which indicates whether or not a RESET mode exists:

0 = no RESET mode,

1 = RESET mode.

Set in subroutine RESET.

RESTCT (COMBIN)

The type of restriction on the secondary head:

0 = none,

1 = RPM,

2 = SFM.

5.1.1 PARAMETER EFINITIONS (cont'd)

RETURN (GECBAS, S2)

The general return flag set by a subroutine which established a program condition.

RFLAG2 (COMBIN)

The read flag for Head 1 (reads from TAPES2):

0 = read next record,

1 = do not read next record.

RFLAG3 (COMBIN)

The read flag for Head 2 (reads from TAPES3):

0 = read next record,

1 = do not read next record.

RHSTEP (GECOM)

The rotary half-step size determined by the value of option 119 * 0.499999; set up in subroutine ASSIGN.

RMS (GECBAS, S2)

The parameter used as a rotary departure calculation tolerance; it is determined by (ROTMAX - OPTAB(119)). Set in subroutine ASSIGN.

RNGDEP(4) (GECOM, S1)

The departure limits which are related to the selcetion of the proper G codes; see Section 3.4.6.

5.1.1 PARAMETER DEFINITIONS (cont'd)

ROTFMN (GECOM, S1)

The rotary feedrate minimum in IPM. Given in RPM in option 113, and set up in subroutine ASSIGN.

ROTFMX (GECOM, S1)

The rotary feedrate maximum in IPM. Given in RPM in option 114, and set up in subroutine ASSIGN.

ROTMAX (GECOM, S1)

The rotary maximum departure in output units. Given in degrees in option 111, and set up in subroutine ASSIGN.

ROTRAP (GECOM, S1)

The rotary rapid traverse maximum feedrate in IPM. Given in RPM in option 115, and set up in subroutine ASSIGN.

ROTUNT (GECOM, S1)

The unit of rotation for the rotary device as indicated by option 118. Set in subroutine ASSIGN.

ROTYPE (GECBAS, S2)

The type of rotary feedrate. Given in option 141.

RPOINT (GECBAS, S2)

The preset rapid point (R). Set in subroutine CYCLE.

SADSFM (GECBAS, S2)

The flag indicating that an SFM mode exists on the saddle:

0 = RPM,

1 = SFM.

5.1.1 PARAMETER DEFINITIONS (cont'd)

SAFHD1(3) (COMBIN)

The direction vector for SAFETY command for Head 1.

SAFHD1(3) (COMBIN)

The direction vector for SAFETY command for Head 2.

SAFLAG (GECBAS, S2)

The flag which specifies the existence of a SAFETY mode:

0 = no,

1 = yes.

Set in subroutine SAFETY.

SAFVEC(80) (FXAXCM)

The saved vectors of both heads retained in several subroutines.

SAVEN (GECBAS, S2)

Parameter which saves the value and condition of the sequence number parameter SEQCTR, as during a SEQNO/OFF mode.

SEFATL (GECBAS, S2)

The turret distance (parallel to the spindle) from the turret center to the centerline of the tool. Given in a TURRET statement and set in subroutine TURRET. These parameters apply only to Head 2 during multihead processing.

SDIV (GECBAS, S2)

The divider value as used in path segmenting sequences.

5.1.1 PARAMETER DEFINITIONS (cont'd)

SEFBTL (GECBAS, S2)

The turret distance (perpendicular to the spindle) from the turret center to the tool tip. Given in a TURRET statement and set in subroutine TURRET. These parameters apply only to Head 2 during multihead processing.

SEGFLG (GECBAS, S2)

The flag which directs GEOUT to print or not to print a line:

0 = print the line,

1 = do not print the line.

Set in subroutine SEGMNT.

SEQ (GECBAS, S2)

SEQCTR (GECBAS, S2)

The current CL tape record number which may be used as a sequence number. (see Option 143.) Set in subroutine GEBASE.

SEQINC (GECBAS, S2)

The increment by which to increase the value of the sequence number.

SEQLIM (GECBAS, S2)

The format size limit of the sequence number DBFSEG(1). Set up in subroutine ASSIGN.

SEQNEW (GECOUT, S2)

The new unit increasing sequence number; set in subroutine GEPRO2.

5.1.1 PARAMETER DEFINITIONS (cont'd)

SEQNH1 (FXAXCM)

The current sequence number for Head 1.

SEQNH2 (FXAXCM)

The current sequence number for Head 2.

SEQOLD (GECOUT, S2)

The old unit increasing sequence number which was formerly SEQNEW. Set up in subroutine GEPR02.

SFHD1W(3) (FXAXCM)

The X,Y,Z departures for withdrawing Head 1 from the work while Head 2 is being parked.

SFHD2W(3) (FXAXCM)

The XYZ departures for withdrawing Head 2 from the work while Head 1 is being parked.

SFHOLD(20) (COMBIN)

The temporary storage of secondary head retraction record when the head is removed from the combined cut.

SFMAXR (GECBAS, S2)

The allowable value of maximum spindle speed in RPM as given by the couplet (MAXRPM,s) in a SPINDL statement.

5.1.1 PARAMETER DEFINITIONS (cont'd)

SFMCIR (GECBAS, S2)

The flag which designates the SFM path condition:

0 = line,

1 = circle.

SFMDES (GECBAS, S2)

The desired value n of SFM as given in a SPINDL/n, SFM statement.

SFMFLG (GECBAS, S2)

The flag which specifies the existence of an SFM mode:

0 = RPM mode,

1 = SFM mode.

Set in subroutine SPINDL.

SFMHD1 (COMBIN)

The SFM value for Head 1.

SFMHD2 (COMBIN)

The SFM value for Head 2.

SFMLOK (GECBAS, S2)

The flag which locks the postprocessor into the current spindle range. It is established during an SFM mode:

0 = unlocked,

1 = locked.

Set in subroutine SFMO.

5.1.1 PARAMETER DEFINITIONS (cont'd)

SFMRAD (GECBAS, S2)

The specified radius to use during an SFM sequence:

-1 = X axis,

-2 = Y axis,

-3 = Z axis.

If value is not negative, it is the given radius value. Set in subroutine SPINDL.

SFMRPM (GECBAS, S2)

The flag which indicates whether an RPM or an SFM mode is called for:

0 = RPM,

1 = SFM.

Set in subroutine SPINDL.

SKPCOD (GECOUT, S2)

The BCD OPSKIP code used when the OPSKIP is on.

SKPFLG (GECBAS, S2)

The flag which indicates whether or not an OPSKIP mode exists:

0 = no,

1 = yes.

Set in subroutine OPSKIP.

5.1.1 PARAMETER DEFINITIONS (cont'd)

SKPLIN (GECBAS, S2)

Flag which specifies whether or not to skip linearity testing:

0 = don't skip,

1 = skip.

SLTOLN (GECBAS, S2)

The length of the selected tool as given in a SELECT/TOOL statement; set in subroutine SELTUL.

SPINON (GECBAS, S2)

The flag which specifies that a spindle-on M code has been selected:

0 = not selected,

1 = selected.

SPNCOM (GECBAS, S2)

The spindle command which is stored into DBFSEG(12).

SPNDIR (GECBAS, S2)

The spindle direction flag:

+1 = CLW,

-1 = CCLW.

SPNMAX (GECBAS, S2)

The maximum spindle speed of the current spindle range.

5.1.1 PARAMET R DEFINITIONS (cont'd)

SPNMIN (GECBAS, S2)

The minimum spindle speed of the current spindle range.

SPNSPD (GECBAS, S2)

The current spindle speed in RPM.

SPNREQ (GECBAS, S2)

Flag which is set in subroutine SPINDL to indicate that a spindle speed had been given.

SRTAB(300) (GECBAS, S1)

The spindle speed table in RPM. Set up in the Machine Subroutine.

STATE(12) (GECBAS, S2)

The vector which contains the status of machine conditions at any given point in time during a part program run.

STEP (GECOM, S1)

The minimum move of the NC machine linear slides. Given in option 14 and set in subroutine ASSIGN.

STOPON (GECBAS, S2)

The flag which designates that a stop condition exists:

0 = no stop,

1 = stop conditions exists.

Used for the automatic reinstatement sequence.

5.1.1 PARAMETER DEFINITIONS (cont'd)

SYSCON (GECBAS, S2)

System of units constant: 12 for English and 1000 for metric system. Set up in subroutine ASSIGN.

TABLEG(120) (GECOM, S1)

The table of preparatory function G codes. Initialized in subroutine STDMAC and reset in the Machine Subroutine.

TABLEM(200) (GECOM, S1)

The table of miscellaneous function M codes. Initialized in subroutine STDMAC and reset in the Machine Subroutine.

TAG(9) (GECBAS, S1)

The array for storing the BCD name of the NC machine which is printed at the top of each page. Set in Machine Subroutine. See Section 5.6.

TEFATL (GECBAS, S2)

The turret distance (parallel to the spindle) from the turret center to the centerline of the tool. Given in a TURRET statement and set in subroutine TURRET. These parameters apply only to Head 3 during multihead processing.

TEFBTL (GECBAS, S2)

The distance (perpendicular to the spindle) from the turret center to the tool tip. Given in a TURRET statement and set in subroutine TURRET. These parameters apply only to Head 3 during multihead processing.

TEMPS2(12) (COMBIN)

Temporary working storage for Head 1.

5.1.1 PARAMETER DEFINITIONS (cont'd)

TEMPS3(12) (COMBIN)

Temporary working storage for Head 2.

THFLAG (GECBAS, S2)

The flag which indicates that a thread had been called for:

0 = no threading,

1 = threading.

THMODE (GECBAS)

The threading mode which specifies the nature of the given pitch:

0 = Constant

+1 = Increasing

-1 = Decreasing

Set in subroutine PITCH.

THLEAD (GECBAS, S2)

The threading lead. It is the reciprocal of the pitch as given by a PITCH statement. Set in subroutine PITCH.

THRATE (GECBAS, S2)

The threading feedrate as given in a PITCH statement. Set in subroutine PITCH.

THRDON (GECBAS, S2)

The thread-on flag which indicates a thread is in mode:

0 = no thread mode,

1 = thread mode.

5.1.1 PARAMETER DEFINITIONS (cont'd)

TIMCUT (GECOM, S2)

The total cutting time in seconds of the machining operations. Set up in subroutine CONTUR.

TIMDWL (GEOUT, S2)

The accumulated dwell time in seconds of the machining operation.

TIME1 (COMBIN)

The time in minutes to cut the Head 1 segment.

TIME2 (COMBIN)

The time in minutes to cut the Head 2 segment.

TLHEAD (GECBAS, S2)

The head flag designating the condition of the selected head(s):

0	=	Head 1,
1	=	Head 2,
2	=	Head 3,
-1	=	Head 1 and 2,
-2	=	Heads 1, 2, and 3.

Set in subroutine SELHED.

TLEN2 (GECBAS, S2)

The length of the tool (TLNOFF) which is ready for unloading.

5.1.1 PARAMETER DEFINITIONS (cont'd)

TLNOFF (GECBAS, S2)

The combined tool number and tool offset number for the tool which is ready for unloading.

TMAX (GECOM, S2)

The tape reader maximum reading time based upon the largest block of possible command data. Given in option 13 and set in subroutine ASSIGN.

TOLCON (GECBAS, S2)

The tool constant flag which directs the CLASSn subroutine to either select or to load the tool constants which are used in the transform relations:

0 = select the constants,

1 = load the constants for use.

TOLDLN (GECBAS, S2)

The length of the tool which is to be loaded into the spindle.

TOLIN (GECOM, S1)

The part programmed cutter inner tolerance INTOL. Set in subroutine GEBASE for a Type 6000 record.

TOLLOD (GECBAS, S2)

The number of the tool which is next to be loaded into the spindle.

TOLOUT (GECOM, S1)

The part programmed cutter outer tolerance OUTTOL. Set in subroutine GEBASE for a Type 6000 record.

5.1.1 PARAMETER DEFINITIONS (cont'd)

TOLSLC (GECBAS, S2)

The tool number of the selected tool as given by a SELECT/TOOL statement.

TOOL (GECBAS, S2)

The tool number given in a TOOLNO or SELECT/TOOL statement. Set in subroutine TOOLNO or SELTUL.

TOOLDN (GECBAS, S2)

The number of the tool which is in the down position.

TOOLEN (GECBAS, S2)

The tool length given in a TOOLNO statement. Set in subroutine TOOLNO or SELECT/TOOL.

TRURAD (GECBAS, S2)

The true radius of the part circle.

TSTLIN (GECBAS, S2)

The flag which indicates whether or not to do linearity testing on one-point CL tape records:

0 = yes,

1 = no.

TURNON (GECBAS, S2)

Flag in subroutine SPINDL which indicates that a spindle-on M code must be made output:

0 = no output,

1 = output.

5.1.1 PARAMETER DEFINITIONS (cont'd)

TUROFF (GECBAS, S2)

The turret offset given in a TURRET statement with the OFSETL modifier. Set in subroutine TURRET.

TURPOS (GECBAS, S2)

The turret position given in a TURRET statement. Set in subroutine TURRET.

UPFLAG (GECBAS, S2)

The flag which indicates the up or down position of the pen or torch:

0 = up,

1 = down.

VALUEM (GECBAS, S2)

The interim storage parameter for carrying the M code for the function just given.

WELDFL (GECBAS, S2)

The flag which indicates the condition of the weld mode:

0 = on,

1 = off.

Set in subroutine WELD.

5.1.2 CUSTOMER COMMON

The array CUST(5), which is in labelled COMMON GECOM/S1, has been set aside for the exclusive use of a customer user of the GECENT III postprocessor. The array will never be used by the postprocessor maintainers.

Many installations may wish to make their own in-house modifications to the postprocessor, and, hence, may require a variable parameter in COMMON for communicating between overlays or subroutines. In order to avoid potential conflict with later GECENT III developments, the user should use the CUST array for all such purposes. Subroutine modifications which use the CUST array become the responsibility of the user, and he may have to update any affected subroutines which are later released by the General Electric Company.

The CUST array is set to zero in GEINIT in subroutine INIT. From this point on, the postprocessor in no way refers to the CUST array again.

5.2 DESCRIPTION OF SUBROUTINES

In the pages which follow, detailed descriptions of each subroutine in the GECENT III postprocessor is given. Each subroutine is identified with its calling name and sequence; the purpose, method, and restrictions are given in brief but complete details. The input and output of each subroutine are identified and explained.

Each subroutine is also identified with the overlay in which it occurs;

for example, FEDRAT
(GECBAS)

This specifies that subroutine FEDRAT is in the GEBASE overlay.

Since a description of one subroutine does not fully explain a programming sequence, e.g., circular interpolation, it is best to first read the relevant chapters of Sections 2, 3, and 4 for a general understanding, and then to refer to the individual subroutine write-up as it occurs in the program flow of the particular sequence. After this, a full understanding of the postprocessor sequence would be obtained if the subroutine listing is consulted for all details.

5.2 DESCRIPTIONS OF SUBROUTINES (cont'd)

Note that some write-ups have restrictions such as DATA statements, APT System COMMON, multiple entries, or machine language coding. These are restrictions only in that they may be computer dependent but not because of any postprocessor requirement; see Section 5.4.1.

5.2.1 SUBROUTINES IN EACH OVERLAY

The subroutines of each major overlay are given under the overlay title. Special subroutines unique to a given installation are not included.

Those subroutines marked with an asterisk (*) may be in APT Section IV on some computers. Multiple entry subroutines are considered to be a separate subroutine. Standard library subroutines are not listed. The Machine Function overlay is not indicated. The complete subroutine descriptions are given in Section 6.0 as they are too voluminous to be included here.

GEMON

COMENT

CONROT

ERDMP1

INPUT

LENGTH

SRAREC

SROREC

WEFREW

GEINIT

ASSIGN

DECODE

GEPRE

IDPART

INIT

Machine Subroutines

REDTAP

STDMAC

5.2.1 SUBROUTINES IN EACH OVERLAY (cont'd)

GEBASE

- AIR
- AUXFUN
- AUXLRY
- BREAK
- CLAMP
- CLRSRF
- COMBIN
- COMTAT
- COOLNT
- CUTCOM
- DELAY
- DSRROW
- DWELL
- EIACOM
- END
- ENTRAP
- FEDOVR
- FEDRAT
- FLOAT
- FROM
- FROM3
- GEBASE
- GEOM
- GEOM3
- GOHOME
- INSERT
- LEADER
- LOAD
- LOCRNG
- MACHIN
- MCHFIN
- MINMOV
- MOTION
- OPCODE
- OPSKIP
- OPSTOP
- ORIGIN
- OUTPUT
- PARTNO
- PICKUP
- PLENTH
- POSITN
- PPRINT
- PREFUN
- PRFSEQ
- RAPID
- RAPIDO
- RAPIDX
- RESET
- RESTAT
- REWIND
- ROTABL
- ROTATE
- SELANG
- SELECT
- SELHED
- SELOFS
- SELRAD
- SELRDR
- SELTAB
- SELTUL
- SEQNO
- SPINDL
- SPTYPE
- SRFCHK
- STOP
- STOREM
- TEST1
- TMARK
- TOOLNO
- TRANS
- TSTEXT
- TSTFLG
- TSTLIM
- UNLOAD
- XOFSET

GETERP

- CHKAX
- CIRINT
- COMPFC
- COMPR
- DEPART
- DETDIR
- DOT
- FEDLIM
- FVARGO
- GOCIRC
- GOLINE
- OFFARC
- PLNSEL
- PROCQD
- QUADET
- QUADNT
- SEGMNT
- SELG
- SELGCR
- TSTFCM

5.2.1 SUBROUTINES IN EACH OVERLAY (cont'd)

GEPOS

CYCLGP

FTYPE2

FTYPE4

FTYPE5

FTYPE6

POSFED

POSMOV

RAPIDP

RETRCT

RFTYPE

ROTABA

ROTIND

ROTMAG

ROTUR

ROTYP1

SET12

TOOLGP

GELATH

COUPLE

CYCLEL

PITCH

SADDLE

SAFEGL

SAFETO

SAFETX

SEGSAD

SFMO

THREAD

THREDO

TOOLL

TSTSAF

TURRET

TURSAD

5.2.1 SUBROUTINES IN EACH OVERLAY (cont'd)

GEMILL

PITCHM
ROTABI
ROTMIN
ROTMOV
ROTOUT
SAFEGM
SELGRO
SELPAL
TABSPD
THEDOM
THREDM
TOOLGM

GEMAXS

ARCTAN
CLASS
CYCLGX
FROM5
GEOM5
LINRTY
LINTOL
NORM
OVRCNT
PIVPLN
SAFEGX
SEGDRC
TRUNC
MODE

5.2.1 SUBROUTINES IN EACH OVERLAY (cont'd)

GEOUT

CHARID

CONBCD*

CONTUR

DOLLAR

GEOUT

PARNEM*

PARNOM*

POSIT

PPUNCH

PUNCHA*

PUNCHB*

PUNIDN*

SETLIN

SHOLZR

SHUFFL

GEOUT1

CALCP1

GEPRN1

GEPRO1

SETUP1

TITLE1

GEOUT2

CALCP2

GEPRN2

GEPRO2

TITLE2

5.2.1 SUBROUTINES IN EACH OVERLAY (cont'd)

GEOUT3

ABSOPR	TITLE3
CALCP3	
GEPRN3	
GEPRO3	
PAGE	
TIMES	

GEOUT4

GEOUT4 uses the same subroutines as GEOUT3.

5.2.1 SUBROUTINES IN EACH OVERLAY (cont'd)

GEMULT

CIRSEG
COMPGC
CONVRT
CREAD
CTCHUP
DRETHD
FEDM
FXMULT
FXPARK
FXTOL
GDWELL
GEMISC
GEMONT
GEMULT
GFDLIM
GMABS
GMCIRL
GMFENC
GMINIT
GMLINE
GMOTIN
GMOUT
GMREAD
FMSTOR
GMWRIT
OUTB
PARK
PERROR
PREPHD
RAPLIM
RAPM
RETHD
RETRET
RETSFY
SAVMCD
SEG
SELGCD
SHFTBK
SPLIT
STOPTS
TEST2
TESTM2

5.2.1 SUBROUTINES IN EACH OVERLAY (cont'd)

GEDRAF

DRAFT

ROTDRF

GEWELD

DRESS

WELD

GESPIN

GESCOM

TYPE0

TYPE01

TYPE02

TYPE03

TYPE04

TYPE05

TYPE06

TYPE07

TYPE08

TYPE09

TYPE10

TYPE11

TYPE12

TYPE13

TYPE14

TYPE15

TYPE16

TYPE17

TYPE18

TYPE19

5.2.1 SUBROUTINES IN EACH OVERLAY (cont'd)

GECLAS

CLASS1

CLASS2

CLASS3

CLASS4

CLASS5

CLASS6

CLASS7

CLASS8

CLASS9

CLAS10

CLAS11

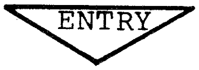
CLAS12

5.3 FLOW CHARTS

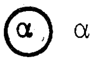
The diagrammed flow charts given in this section illustrate mainly the general flow of the postprocessor and some of its key subroutines. There is little need for flow charts since each subroutine listing is profusely documented with explanatory comments.

The flow charts illustrate only the idea of the subject subroutine and do not give program details. Details can be obtained from the subroutine listing. Flow charts are given only for the major functions of the postprocessor, e.g., linear motion processing or multihead processing.

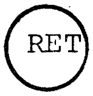
The flow charts use the following defined symbols for operation.



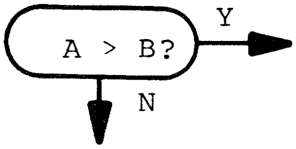
Subroutine entry point.



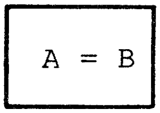
A connector point; α means to continue the flow chart at the point indicated by α. α is the pickup point. Greek letters are used to differentiate from the normal test.



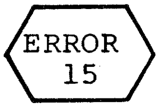
Subroutine exit



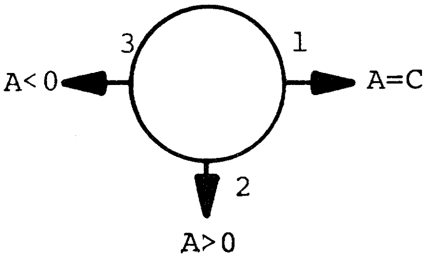
A test condition; in this example: is A greater than B? Y = yes; N = no. In a subroutine this symbol normally refers to a logical IF statement.



Rectangles give program relations and other information.

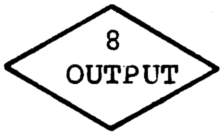


Fatal error; the number is the value assigned to the error; see Section 5.7. The subroutine ERDMP1 is always called for an error condition.



A branch condition; the branch 1, 2, or 3 is taken, depending on the value of A. In a subroutine this symbol refers to an IF statement or to a computed GOTO statement.

5.3 FLOW CHARTS (cont'd)



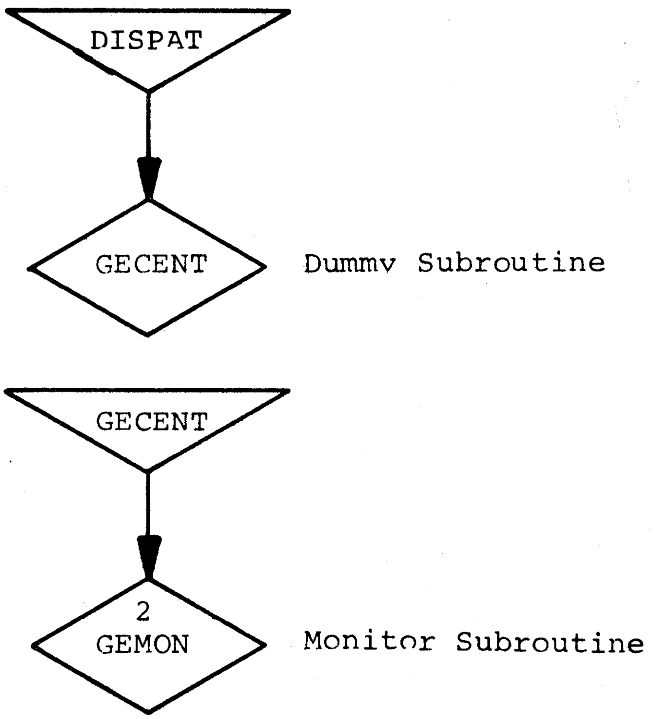
A subroutine call; the subroutine name and flow chart (if any) number is given. In this example, the subroutine called is OUTPUT whose flow chart number is 8.

All closed subroutines use the return symbol; open subroutines do not.

Note that vectors are often set equal to one another, and no indication is given regarding their dimensions. For example, when the flow chart states DPREVP = DPRESP, this means that the vectors are made equal, and that DPREVP(1) = DPRESP(1), DPREVP(2) = DPRESP(2), and so on. Also, when the flow chart states DBUFER = DMBITS, this means the whole DBUFER array is set to DMBITS.

Flow Charts

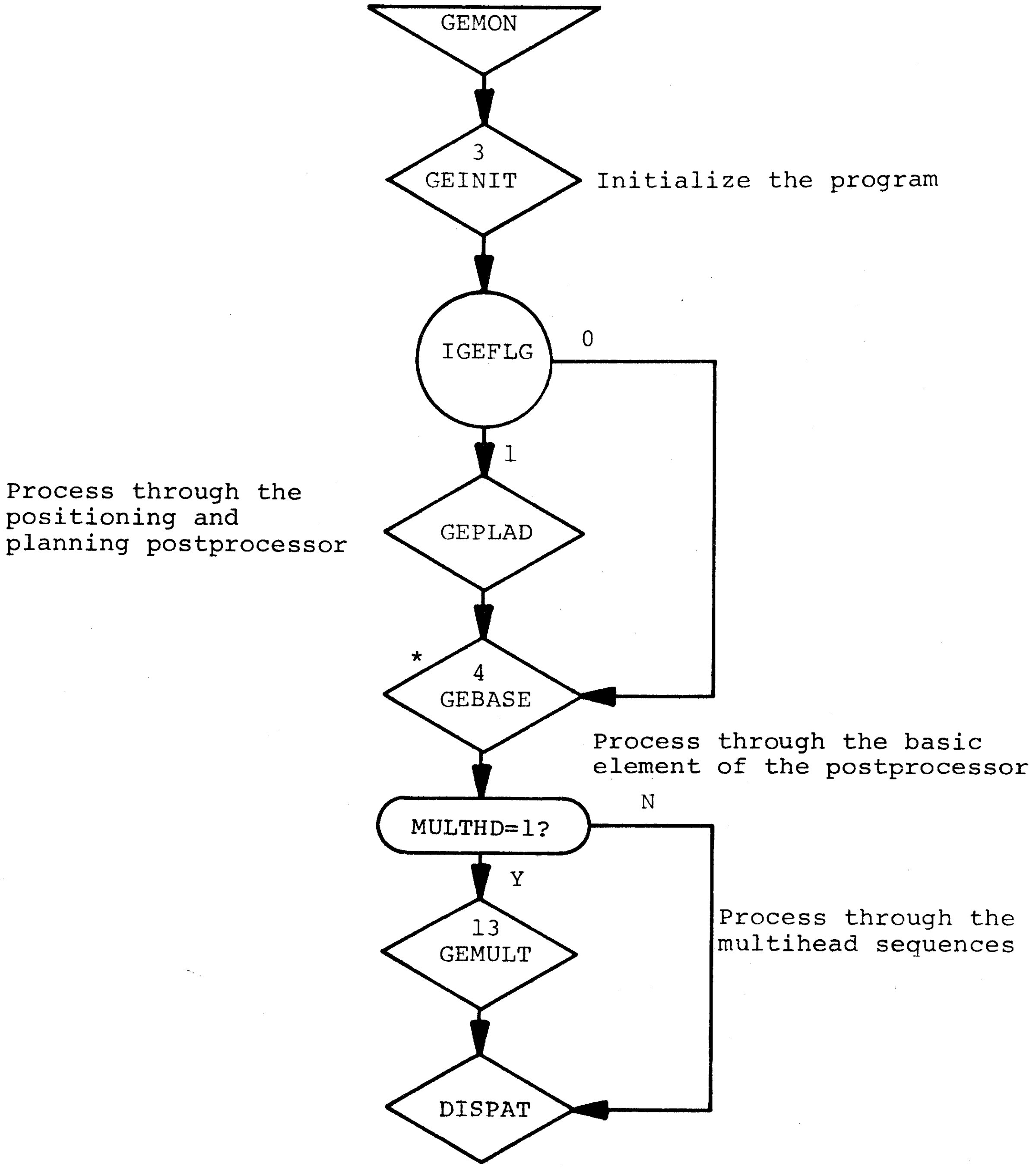
FLOW CHART 1: APT Section IV Selection of GECENT III Post-processor



For all computers, DISPAT calls the dummy subroutine GECENT which then calls subroutine GEMON, the monitor subroutine for the postprocessor. (See Flow Chart 2.)

FLOW CHART 2: Subroutine GEMON (Overlay GEMON)

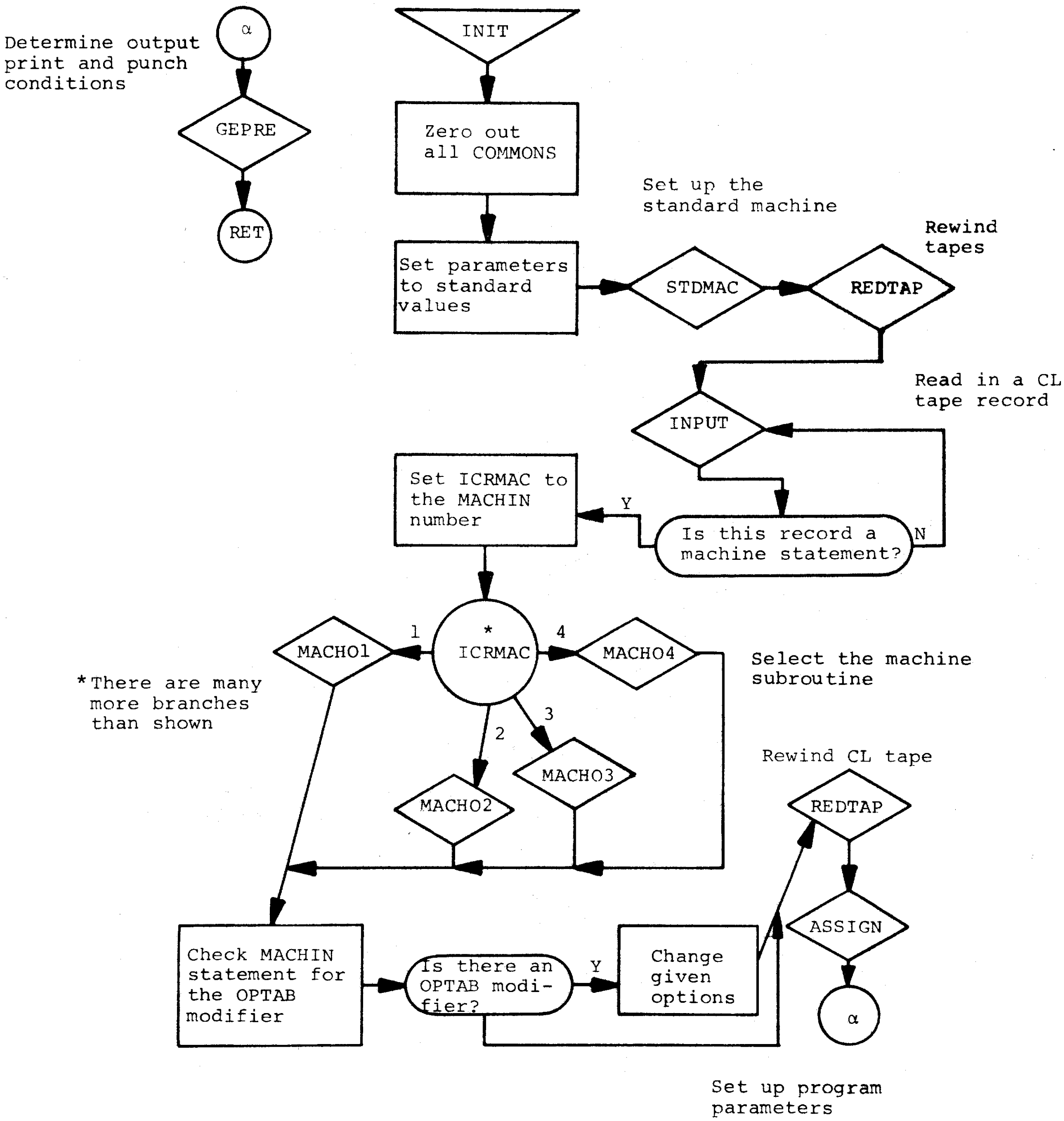
Selection of Postprocessor Overlays


GEMON
3
GEINIT
Initialize the program
IGEFLG
0
1
Process through the positioning and planning postprocessor
GEPLAD
*
4
GEBASE
Process through the basic element of the postprocessor
MULTHD=1?
N
Y
13
GEMULT
Process through the multihead sequences
DISPAT

* NOTE: This chart illustrates the selection sequence for all computers except the GE635; the GE635 subroutine GEMON selects the NC machine type overlay, output overlay, and other overlays through the use of LLINK.

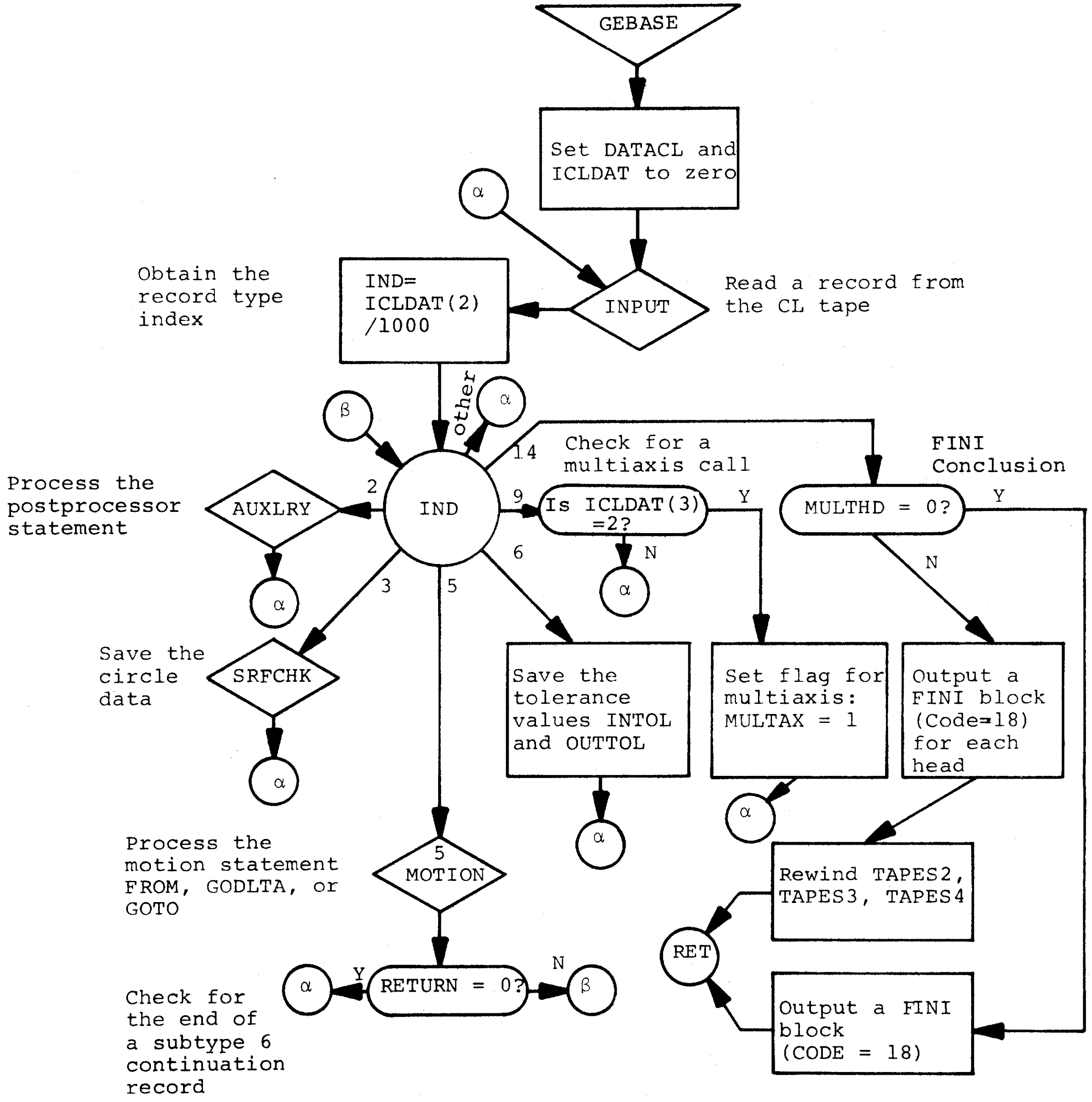
FLOW CHART 3: Subroutine INIT (Overlay GEINIT) Initialization

Initialization of the Postprocessor


Determine output print and punch conditions
α
GEPRE
RET
INIT
Zero out all COMMONS
Set parameters to standard values
Set up the standard machine
STDMAC
Rewind tapes
REDTAP
Read in a CL tape record
INPUT
Set ICRMAC to the MACHIN number
Y
Is this record a machine statement?
N
*
ICRMAC
1
MACHO1
4
MACHO4
2
MACHO2
3
MACHO3
*There are many more branches than shown
Select the machine subroutine
Rewind CL tape
REDTAP
ASSIGN
α
Check MACHIN statement for the OPTAB modifier
Is there an OPTAB modifier?
Y
Change given options
Set up program parameters

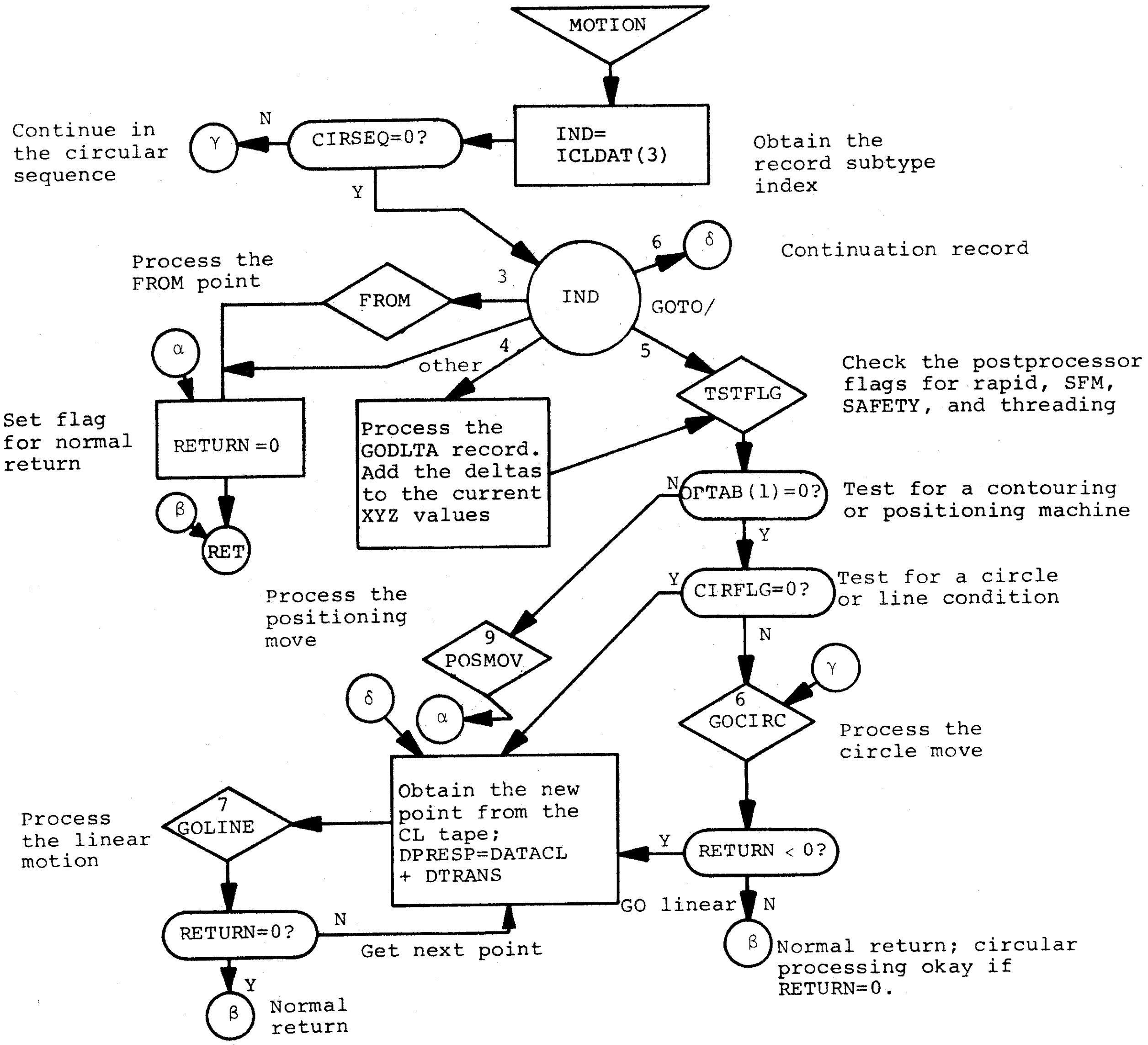
FLOW CHART 4: Subroutine GEBASE (Overlay GEBASE)

Processing Through the Basic Element

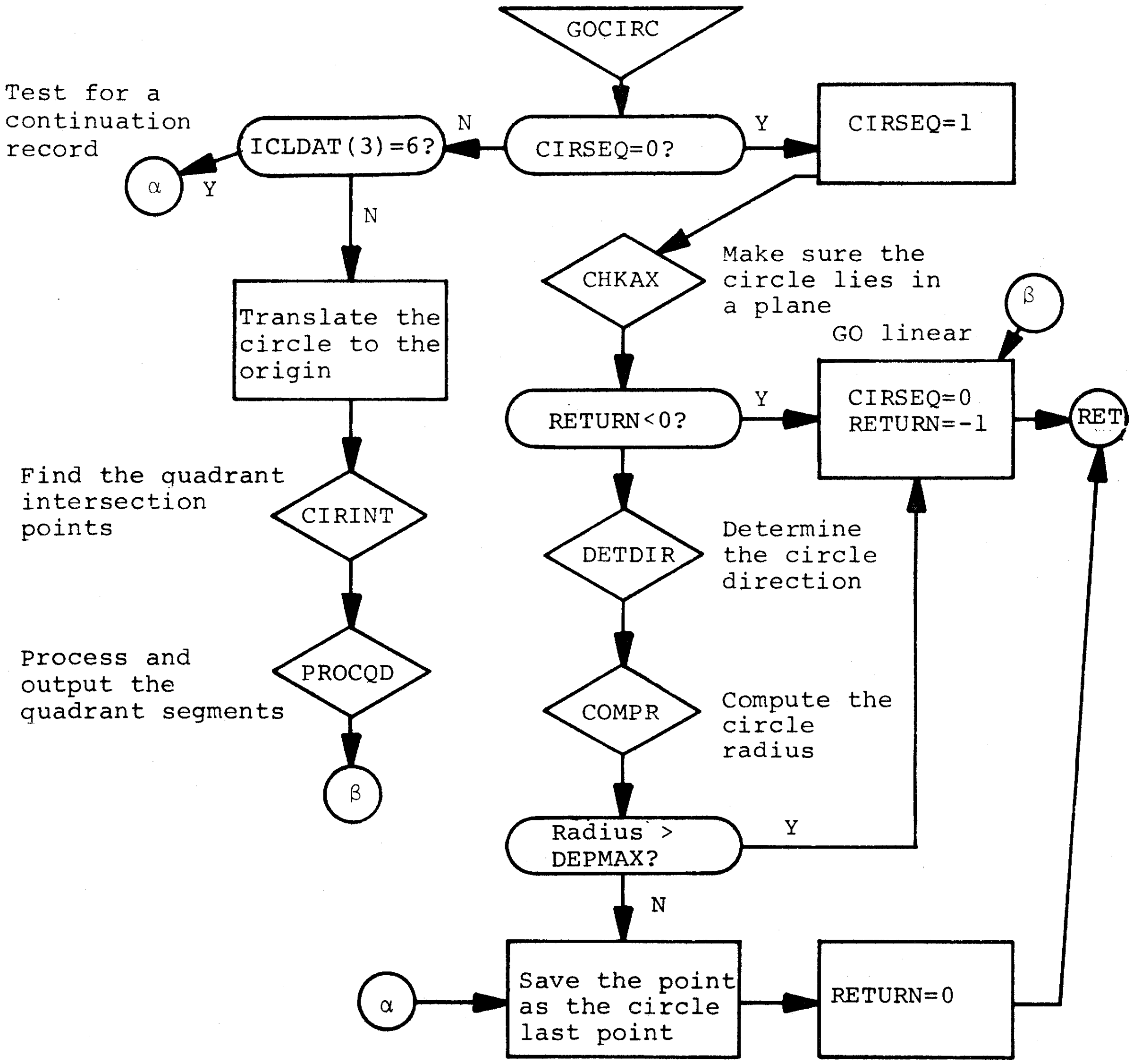

GEBASE
Set DATACL and ICLDAT to zero
α
Obtain the record type index
IND= ICLDAT(2) /1000
INPUT
Read a record from the CL tape
other
β
α
14
Check for a multiaxis call
FINI Conclusion
Process the postprocessor statement
AUXLRY
2
IND
9
Is ICLDAT(3) =2?
Y
MULTHD = 0?
Y
6
N
N
α
α
3
5
Save the circle data
SRFCHK
Save the tolerance values INTOL and OUTTOL
Set flag for multiaxis: MULTAX = 1
Output a FINI block (Code=18) for each head
α
α
α
Process the motion statement FROM, GODLTA, or GOTO
5 MOTION
Rewind TAPES2, TAPES3, TAPES4
RET
N
Y
α
RETURN = 0?
β
Check for the end of a subtype 6 continuation record
Output a FINI block (CODE = 18)

FLOW CHART 5: Subroutine MOTION (Overlay GEBASE)

Processing a Motion Record

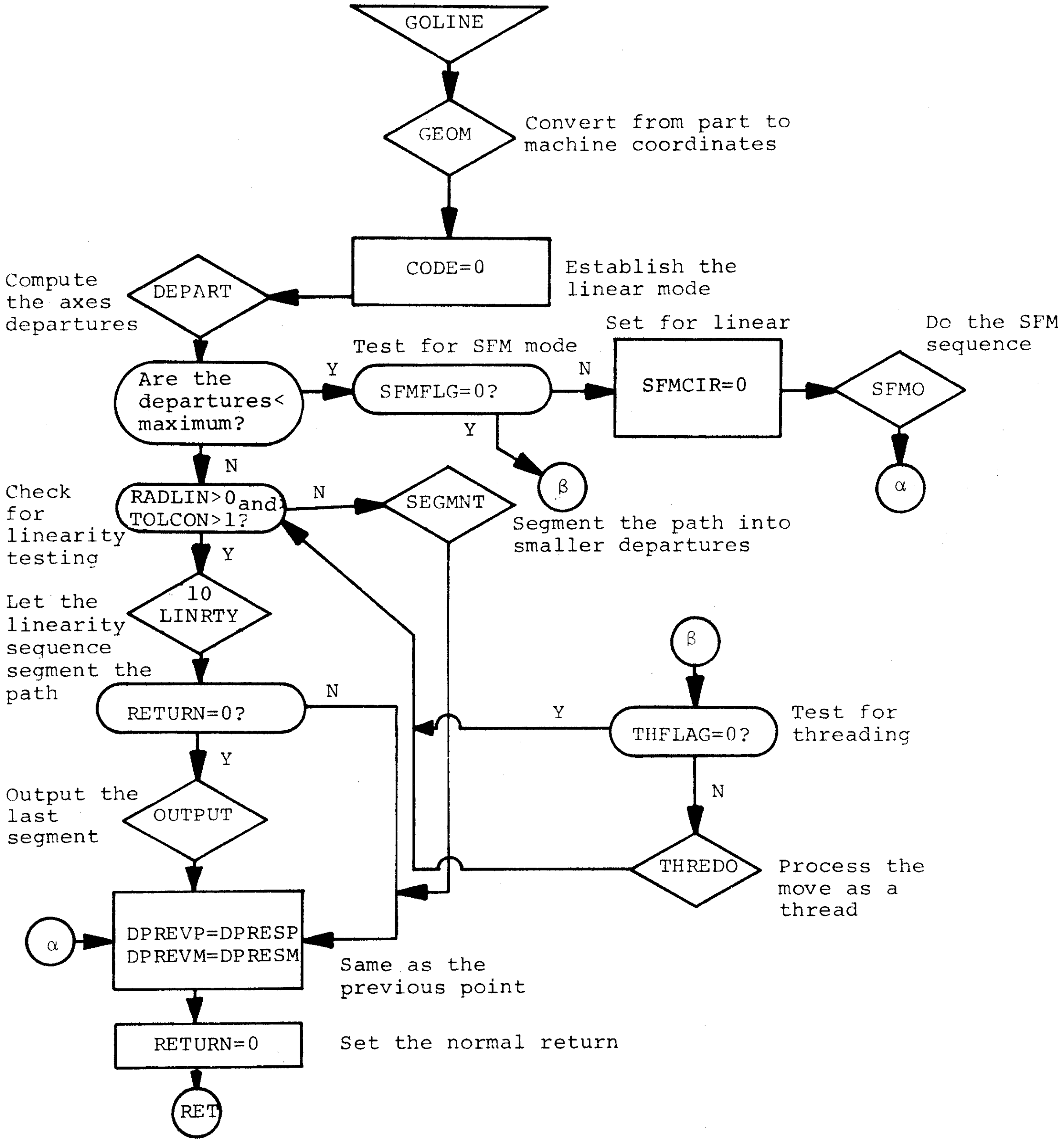

MOTION
IND=
ICLDAT(3)
Obtain the record subtype index
CIRSEQ=0?
N
Y
Continue in the circular sequence
Y
IND
6
δ
Continuation record
3
FROM
Process the FROM point
4
other
5
GOTO/
α
TSTFLG
Check the postprocessor flags for rapid, SFM, SAFETY, and threading
RETURN=0
Set flag for normal return
Process the GODLTA record. Add the deltas to the current XYZ values
β
RET
N
PTAB(1)=0?
Y
Test for a contouring or positioning machine
Y
CIRFLG=0?
Test for a circle or line condition
N
Process the positioning move
9
POSMOV
δ
α
Y
6
GOCIRC
Process the circle move
Obtain the new point from the CL tape;
DPRESP=DATACL
+ DTRANS
7
GOLINE
Process the linear motion
Y
RETURN < 0?
GO linear
N
β
Normal return; circular processing okay if RETURN=0.
RETURN=0?
N
Get next point
Y
β
Normal return

FLOW CHART 6: Subroutine GOCIRC (Overlay GETERP)
Processing a Circular Interpolation Move

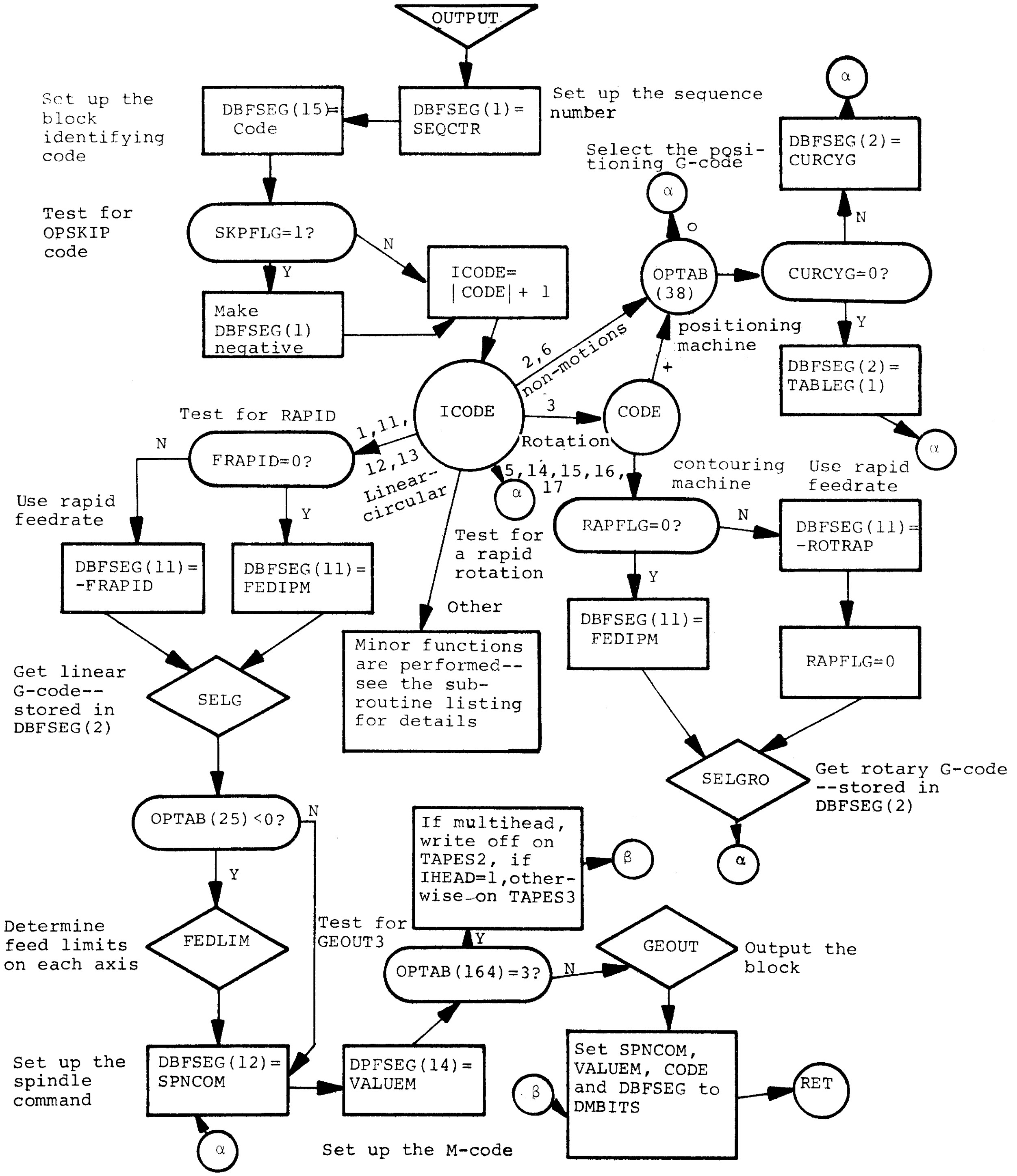

GOCIRC
CIRSEQ=0?
Y
CIRSEQ=1
N
ICLDAT(3)=6?
Test for a continuation record
Y
α
N
Translate the circle to the origin
CHKAX
Make sure the circle lies in a plane
GO linear
β
RETURN<0?
Y
CIRSEQ=0
RETURN=-1
RET
Find the quadrant intersection points
CIRINT
Process and output the quadrant segments
PROCQD
β
DETDIR
Determine the circle direction
COMPR
Compute the circle radius
Radius > DEPMAX?
Y
N
α
Save the point as the circle last point
RETURN=0

FLOW CHART 7: Subroutine GOLINE (Overlay GETERP) Processing

Processing a Linear Interpolation Move

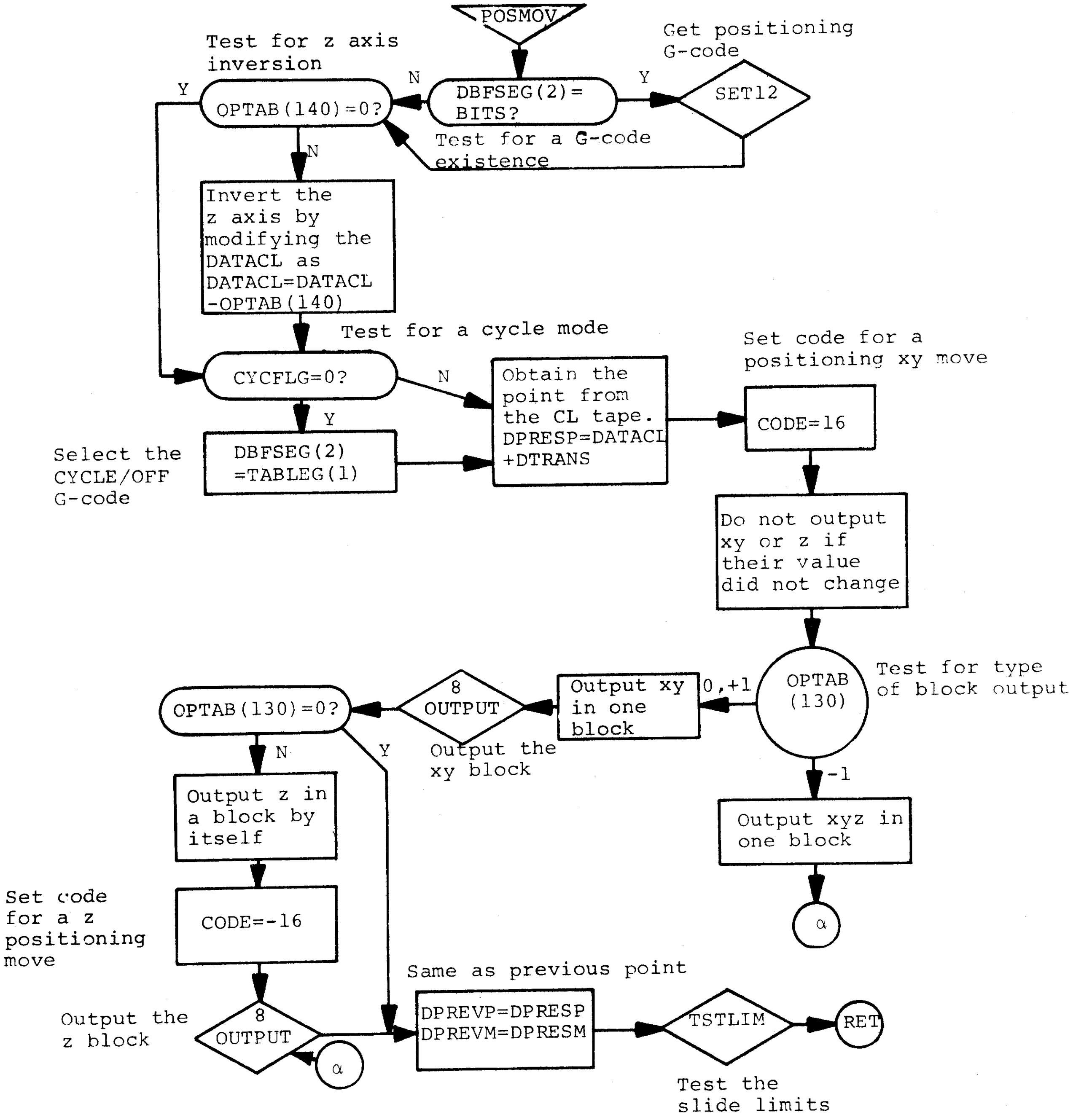

GOLINE
GEOM
Convert from part to machine coordinates
CODE=0
Establish the linear mode
Compute the axes departures
DEPART
Set for linear
Do the SFM sequence
Test for SFM mode
Are the departures< maximum?
Y
SFMFLG=0?
N
SFMCIR=0
SFMO
Y
N
β
α
Check for linearity testing
RADLIN>0 and TOLCON>1?
N
SEGMNT
Segment the path into smaller departures
Y
Let the linearity sequence segment the path
10 LINRTY
β
RETURN=0?
N
Y
THFLAG=0?
Test for threading
Y
N
Output the last segment
OUTPUT
THREDO
Process the move as a thread
α
DPREVP=DPRESP
DPREVM=DPRESM
Same as the previous point
RETURN=0
Set the normal return
RET

FLOW CHART 8: Subroutine OUTPUT (Overlay GEBASE)
Outputting a Command Block

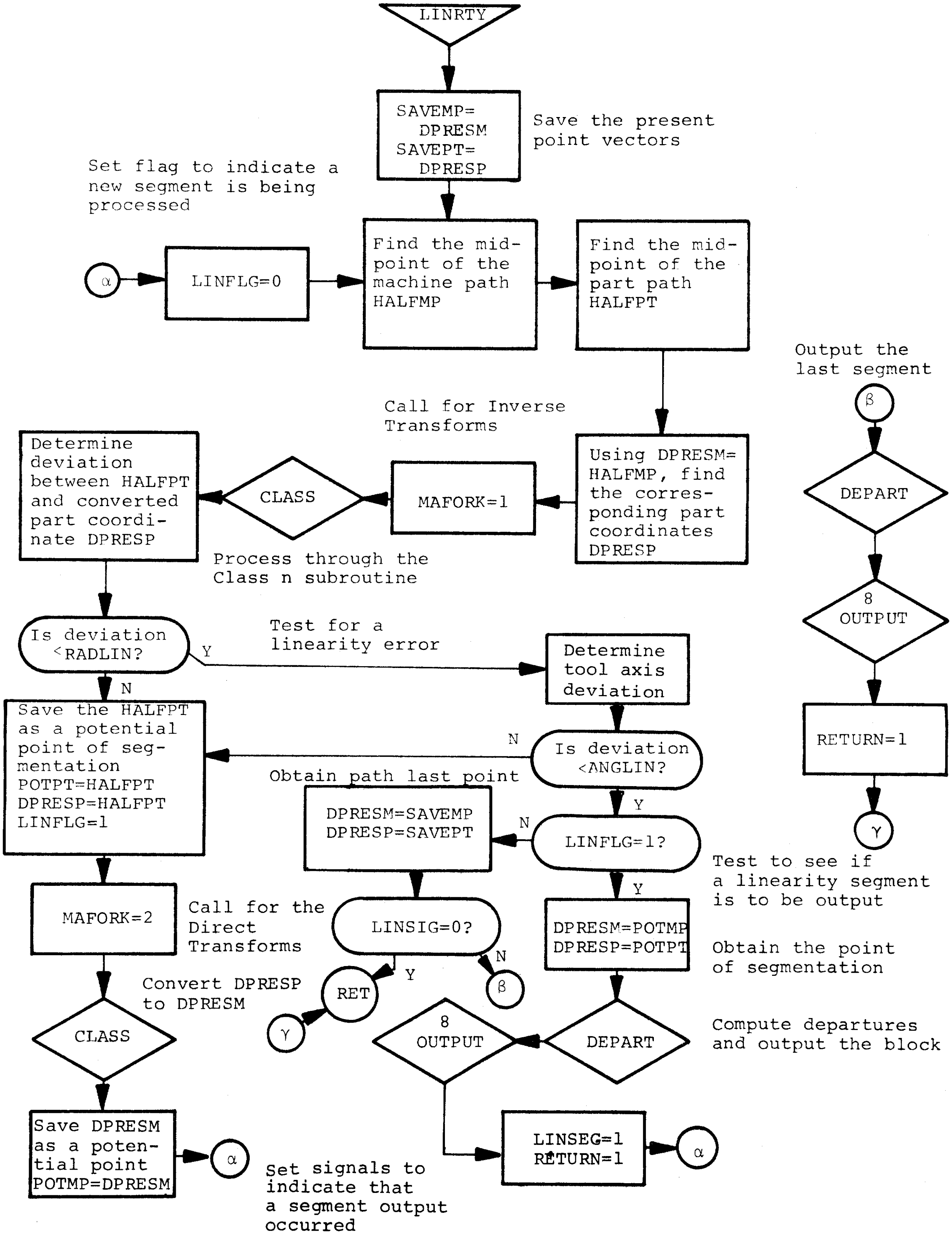


FLOW CHART 9: Subroutine POSMOV (Overlay GEPOS)

Processing a Positioning Move

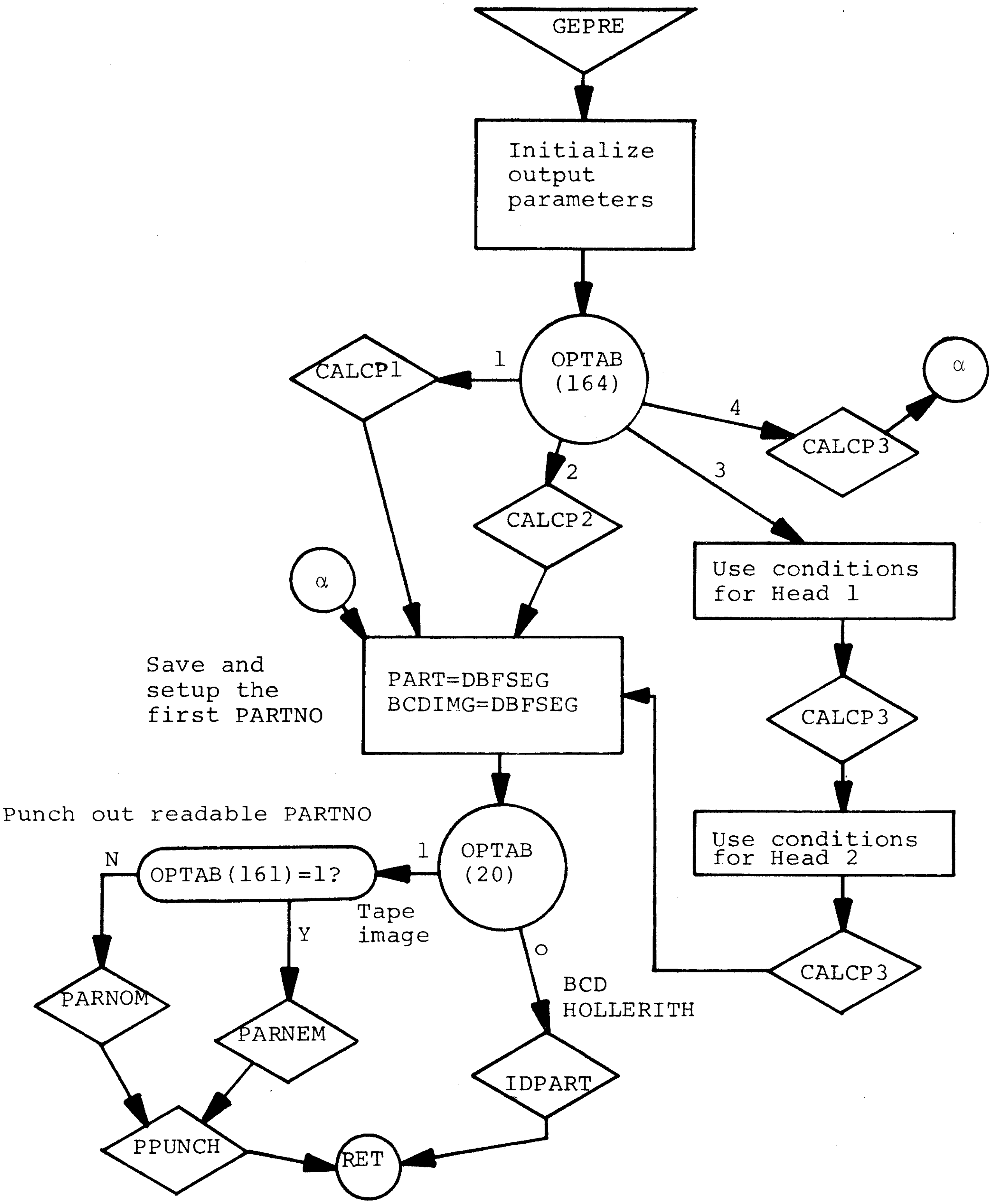

POSMOV
Test for z axis inversion
Get positioning G-code
Y
N
Y
OPTAB(140)=0?
DBFSEG(2)= BITS?
SET12
Test for a G-code existence
N
Invert the z axis by modifying the DATACL as DATACL=DATACL -OPTAB(140)
Test for a cycle mode
Set code for a positioning xy move
CYCFLG=0?
N
Obtain the point from the CL tape. DPRESP=DATACL +DTRANS
CODE=16
Y
Select the CYCLE/OFF G-code
DBFSEG(2) =TABLEG(1)
Do not output xy or z if their value did not change
OPTAB(130)
Test for type of block output
0,+1
Output xy in one block
8 OUTPUT
OPTAB(130)=0?
N
Y
Output the xy block
-1
Output z in a block by itself
Output xyz in one block
Set code for a z positioning move
CODE=-16
α
Same as previous point
Output the z block
8 OUTPUT
α
DPREVP=DPRESP DPREVM=DPRESM
TSTLIM
RET
Test the slide limits

FLOW CHART 10: Subroutine LINRTY (Overlay GEMAXS)
Test and Correction of Linearity Errors

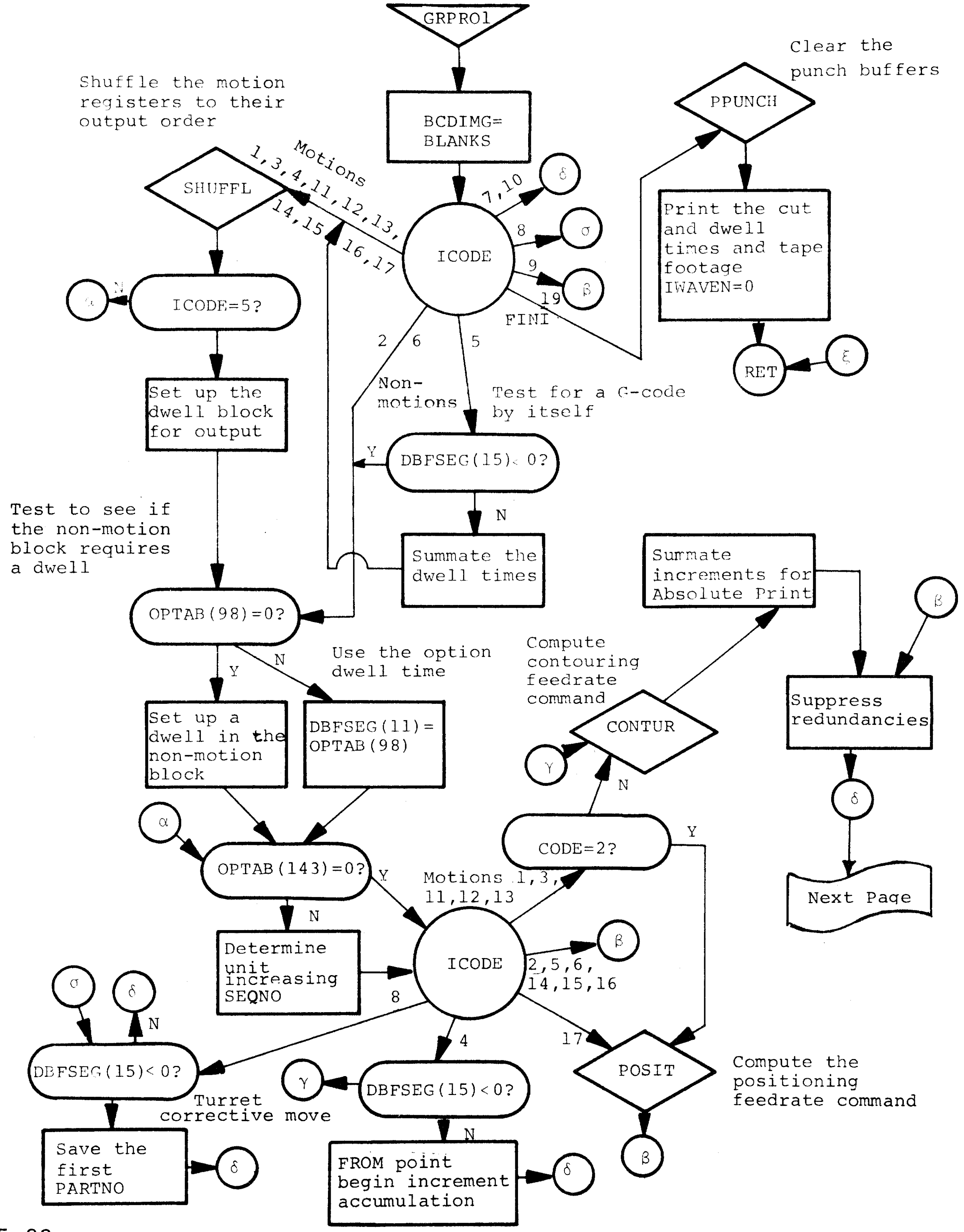

LINRTY
SAVEMP= DPRESM SAVEPT= DPRESP
Save the present point vectors
Set flag to indicate a new segment is being processed
α
LINFLG=0
Find the mid-point of the machine path HALFMP
Find the mid-point of the part path HALFPT
Output the last segment
β
Call for Inverse Transforms
Determine deviation between HALFPT and converted part coordinate DPRESP
CLASS
MAFORK=1
Using DPRESM=HALFMP, find the corresponding part coordinates DPRESP
DEPART
Process through the Class n subroutine
8 OUTPUT
Is deviation <RADLIN?
Y
Test for a linearity error
Determine tool axis deviation
N
RETURN=1
Save the HALFPT as a potential point of segmentation POTPT=HALFPT DPRESP=HALFPT LINFLG=1
N
Is deviation <ANGLIN?
Obtain path last point
Y
DPRESM=SAVEMP DPRESP=SAVEPT
N
LINFLG=1?
γ
Test to see if a linearity segment is to be output
Y
MAFORK=2
Call for the Direct Transforms
LINSIG=0?
DPRESM=POTMP DPRESP=POTPT
Obtain the point of segmentation
N
Y
Convert DPRESP to DPRESM
RET
β
γ
8 OUTPUT
DEPART
Compute departures and output the block
CLASS
Save DPRESM as a potential point POTMP=DPRESM
α
LINSEG=1 RETURN=1
α
Set signals to indicate that a segment output occurred

FLOW CHART 11: Subroutine GEPRE (Overlay GEINIT)

Selection of Proper GEOUT Conditions

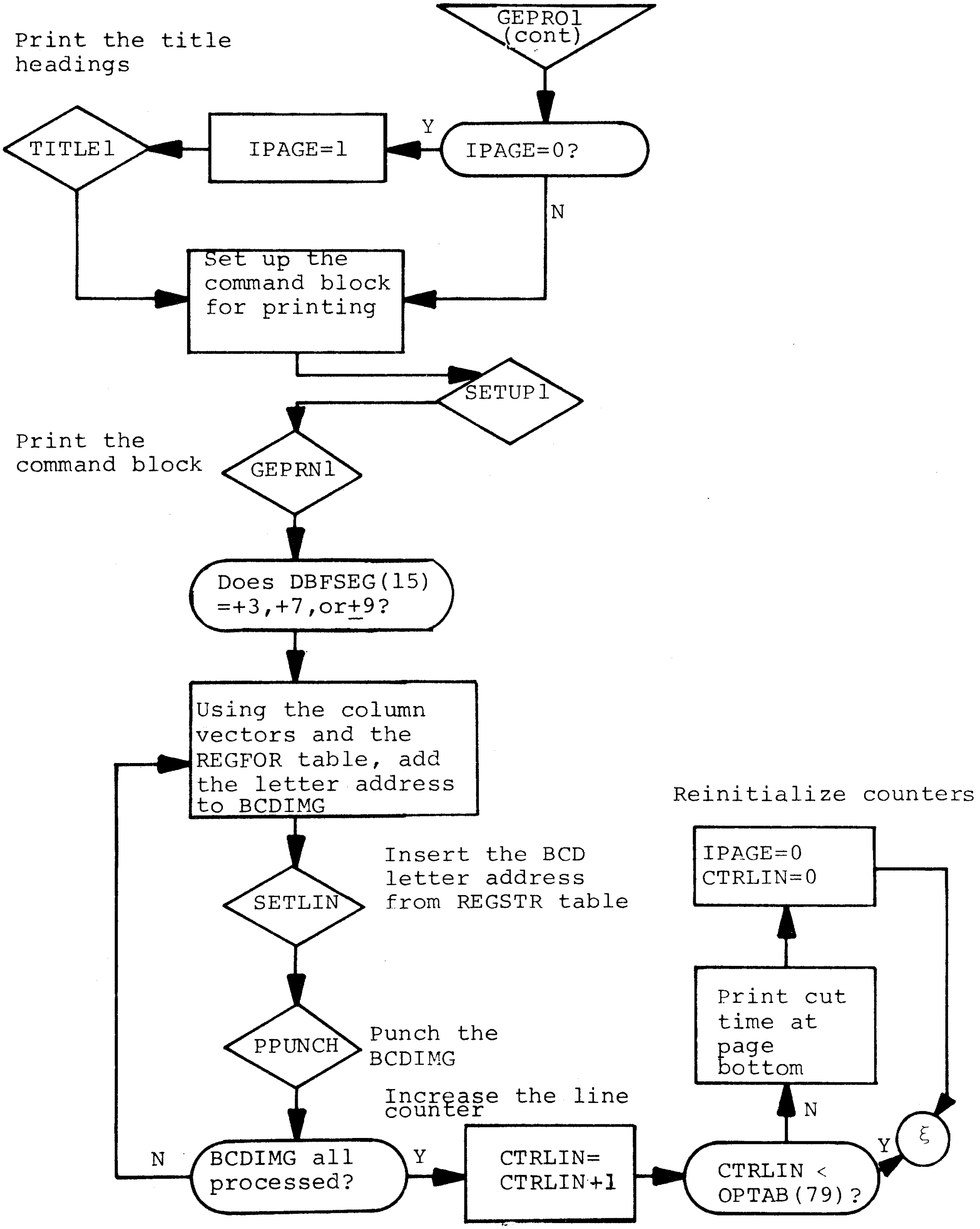


FLOW CHART 12: Subroutine GEPRO1 (Overlay GEOUT)
Printing and Punching an Output Block



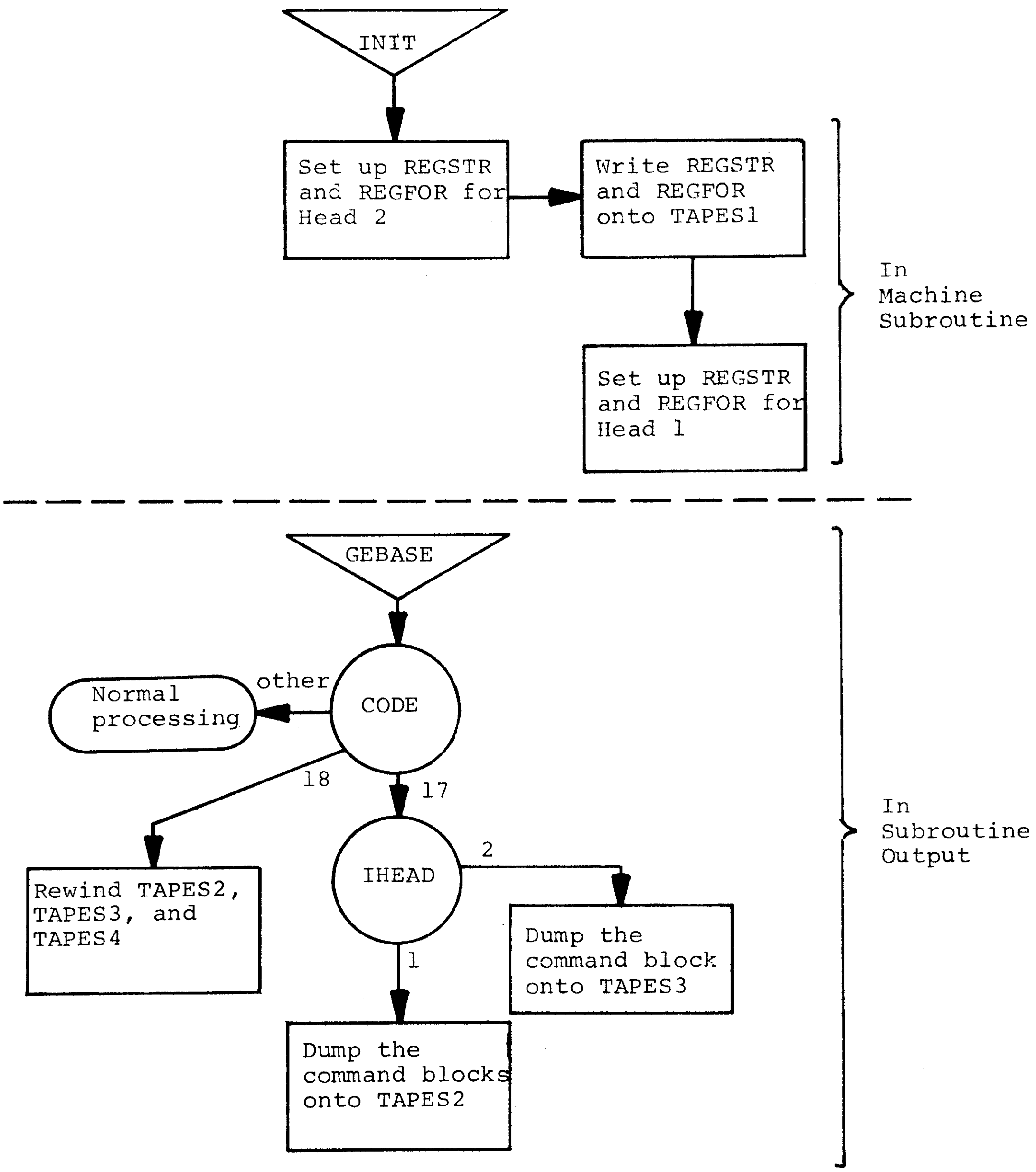
FLOW CHART 12: Subroutine GEPRO1 (Overlay (GEOUT) (continued)

Printing and Punching an Output Block


GEPRO1 (cont)
Print the title headings
TITLE1
IPAGE=1
Y
IPAGE=0?
N
Set up the command block for printing
SETUP1
Print the command block
GEPRN1
Does DBFSEG(15) =+3,+7,or±9?
Using the column vectors and the REGFOR table, add the letter address to BCDIMG
Insert the BCD letter address from REGSTR table
SETLIN
PPUNCH
Punch the BCDIMG
Increase the line counter
N
BCDIMG all processed?
Y
CTRLIN= CTRLIN+1
CTRLIN < OPTAB(79)?
Y
ξ
N
Print cut time at page bottom
Reinitialize counters
IPAGE=0
CTRLIN=0

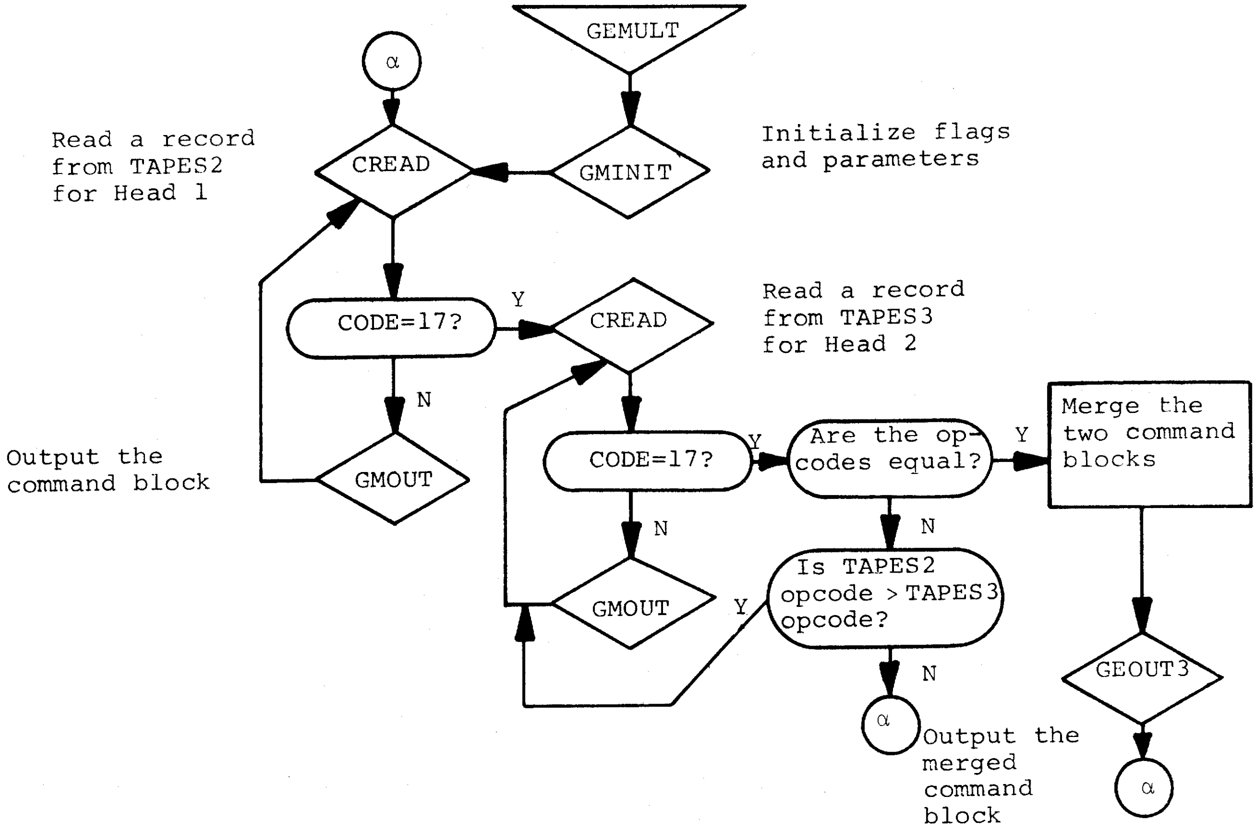
FLOW CHART 13: MULTIHEAD PROCESSING

FIRST PASS

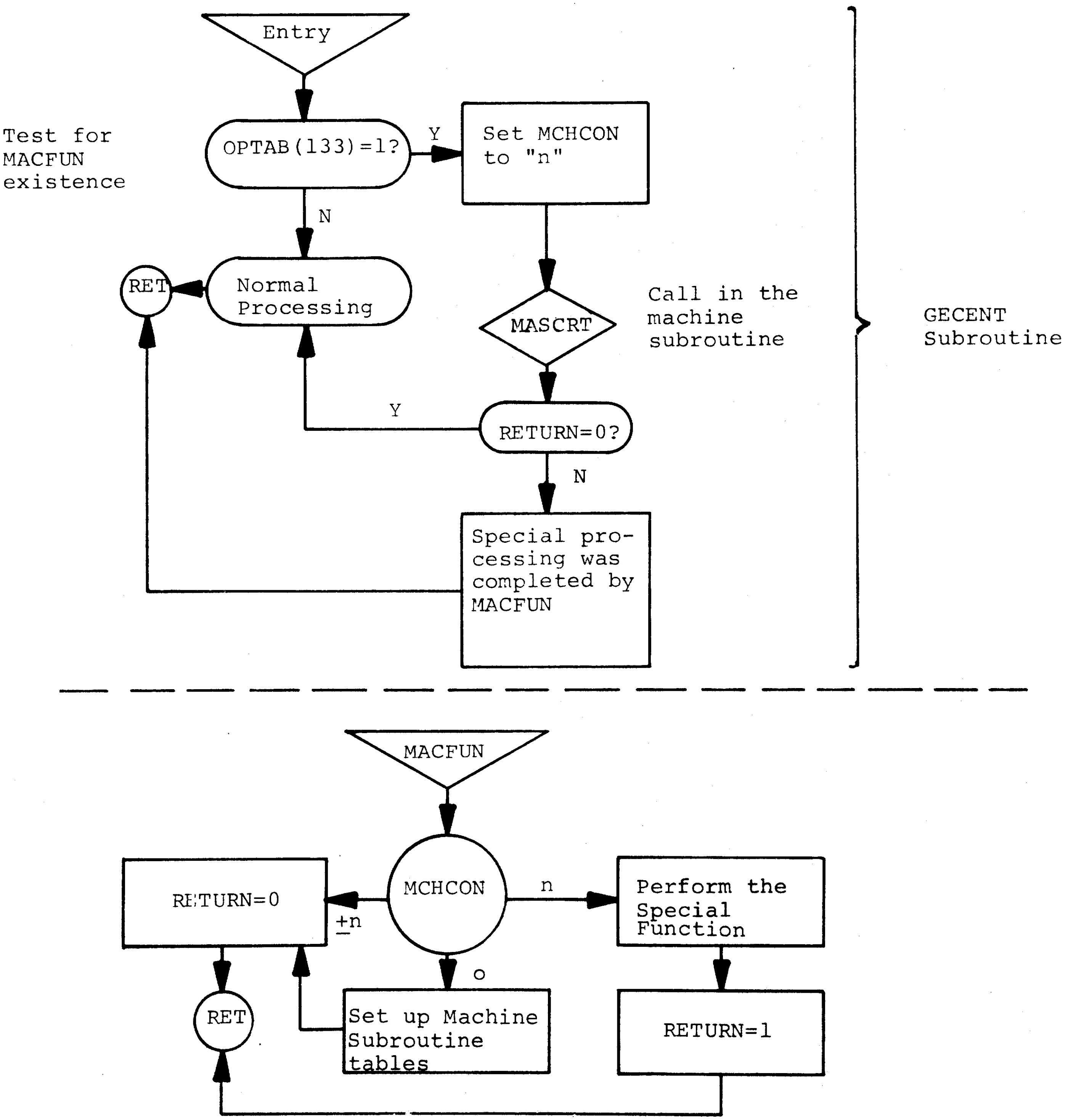

INIT
Set up REGSTR and REGFOR for Head 2
Write REGSTR and REGFOR onto TAPES1
Set up REGSTR and REGFOR for Head 1
In Machine Subroutine
GEBASE
CODE
other
Normal processing
18
17
Rewind TAPES2, TAPES3, and TAPES4
IHEAD
2
Dump the command block onto TAPES3
1
Dump the command blocks onto TAPES2
In Subroutine Output

FLOW CHART 13 (continued)

Second Pass



FLOW CHART 14: Special Machine Functions (MACFUN)


Entry
Test for MACFUN existence
OPTAB(133)=1?
Y
Set MCHCON to "n"
N
RET
Normal Processing
MASCRT
Call in the machine subroutine
GECENT Subroutine
Y
RETURN=0?
N
Special processing was completed by MACFUN
MACFUN
RETURN=0
MCHCON
n
Perform the Special Function
±n
o
RET
Set up Machine Subroutine tables
RETURN=1

5.4 CROSS TRACING OF SUBROUTINES

In order to give an overall view of the GECENT III postprocessor, the following lists of subroutines indicate which subroutines call other subroutines, and conversely, which subroutines are called by other subroutines. Note that even APT Section 0 subroutines are indicated as are the special output subroutines.

No references are made to Machine Subroutines nor to their MACFUNs; all Machine Subroutines are called by subroutine INIT.

The APT Section 0 subroutines referred to are BUFFTP, TAPERD, and TAPEWT; subroutines of similar purpose but different name on some computers apply as well. (For example, the GE635 uses subroutine GETNXR, PUTNXR, REWZ, and so on.)

5.4.1 SUBROUTINES CALLING SUBROUTINES

SUBROUTINE	CALLED
AUXLRY	RETRCT
AUXLRY	REWIND
AUXLRY	ROTATE
AUXLRY	SAFEGL
AUXLRY	SAFEGM
AUXLRY	SAFEGX
AUXLRY	SELECT
AUXLRY	SELHED
AUXLRY	SEQNO
SUXLRY	SPINDL
AUXLRY	STOP
AUXLRY	THREAD
AUXLRY	THREDM
AUXLRY	TMARK
AUXLRY	TOOLNO
AUXLRY	TRANS

5.4.1 SUBROUTINES CALLING SUBROUTINES (cont'd)

SUBROUTINE	CALLED	SUBROUTINE	CALLED
AUXLRY	TURRET	CLASS2	COS
AUXLRY	UNLOAD	CLASS2	SIN
AUXLRY	WELD	COMENT	GMSTOR
AUXLRY	XOFSET	COMENT	OUTPUT
CALCP1	CALCP2	COMENT	STOREM
CALCP1	SETLIN	COMPGC	SELGCD
CALCP2	SETLIN	COMPR	SQRT
CA:CP3	SETLIN	COMTAT	MACSRT
CIRINT	QUADET	CONBCD	DUMP
CIRSEG	CONVRT	CONTUR	EIACOM
CIRSEG	COS	CONTUR	EVARGO
CIRSEG	LENGTH	CONTUR	LENGTH
CIRSEG	SELGCD	CONTUR	SQRT
CIRSEG	SIN	ABSOPR	GEPRN3
CIRSEG	SRAREC	ABSOPR	GMREAD
CIRSEG	STOPTS	ABSOPR	PAGE
CLAMP	COMENT	ABSOPR	TIMES
CLAMP	DWELL	ABSOPR	TITLE3
CLAMP	STOREM	ABSOPR	WEFREW
CLASS1	ARCTAN	AIR	STOREM
CLASS1	COS	ARCTAN	ATAN
CLASS1	NORM	ASSIGN	CONROT
CLASS1	SIN	AUXFUN	OUTPUT
CLASS1	SQRT	AUXFUN	STOREM
CLASS2	ARCTAN	AUXLRY	AIR

5.4.1 SUBROUTINES CALLING SUBROUTINES (cont'd)

SUBROUTINE	CALLED	SUBROUTINE	CALLED
AUXLRY	AUXFUN	AUXLRY	OPCODE
AUXLRY	BREAK	AUXLRY	OPSKIP
AUXLRY	CLAMP	AUXLRY	OPSTOP
AUXLRY	CLRSRF	AUXLRY	ORIGIN
AUXLRY	COMBIN	AUXLR	OVRCNT
AUXLRY	COMENT	AUXLRY	PARTNO
AUXLRY	COOLNT	AUXLRY	PICKUP
AUXLRY	COUPLE	AUXLRY	PITCH
AUXLRY	CUTCOM	AUXLRY	PITCHM
AUXLRY	CYCLEL	AUXLRY	PIVPLN
AUXLRY	CYCLGP	AUXLRY	POSITN
AUXLRY	CYCLGX	AUXLRY	PPRINT
AUXLRY	DELAY	AUXLRY	PPTOL
AUXLRY	DRAFT	AUXLRY	PREFUN
AUXLRY	DRESS	AUXLRY	PRFSEQ
AUXLRY	END	AUXLRY	RAPID
AUXLRY	FEDRAT	AUXLRY	RESET
AUXLRY	FLAME	CONTUR	TSTEXT
AUXLRY	GOHOME	COOLNT	STOREM
AUXLRY	INSERT	COUPLE	COMENT
AUXLRY	LEADER	COUPLE	DWELL
AUXLRY	LOAD	COUPLE	SPTYPE
AUXLRY	MACHIN	COUPLE	STOREM
AUXLRY	MACHTL	CREAD	GETNXR
AUXLRY	MCHFIN	CREAD	IOERR

5.4.1 SUBROUTINES CALLING SUBROUTINES (cont'd)

SUBROUTINE	CALLED	SUBROUTINE	CALLED
CREAD	SAVMCD	DRESS	STOREM
CTCHUP	GMOUT	DRETHD	COMPGC
CUTCOM	OUTPUT	DRETHD	FXTOL
CYCLEL	COMENT	DRETHD	GMOUT
CYCLEL	OUTPUT	DRETHD	PERROR
CYCLEL	RAPIDX	DRETHD	RAPLIM
CYCLEL	STOREM	DRETHD	SRAREC
CYCLGP	COMENT	DSRROW	COMENT
CYCLGP	FLOAT	DUMACH	DISPAT
CYCLGP	FTYPE2	DWELL	OUTPUT
CYCLGP	FTYPE6	EIACOM	COMENT
CYCLGP	MACSRT	END	COMENT
CYCLGP	OUTPUT	END	DWELL
CYCLGP	RAPIDP	END	OUTPUT
CYCLGP	RAPIDX	END	STOREM
CYCLGP	STOREM	ERDMP1	DISPAT
CYCLGP	TSTEXT	ERDMP1	GEDUMP
CYCLGX	COMENT	ERDMP1	LLINK
CYCLGX	MACSRT	ERDMP1	PDUMP
CYCLGX	RAPIDX	FEDLIM	CONVRT
DECODE	ERDMP1	FEDLIM	LENGTH
DELAY	COMENT	FEDOVR	STOREM
DELAY	MACSRT	FEDRAT	COMENT
DELAY	OUTPUT	FEDRAT	DWELL
DELAY	STOREM	FEDRAT	FLOAT
DEPART	ROTMOV	FEDRAT	FTYPE2

5.4.1 SUBROUTINES CALLING SUBROUTINES (cont'd)

SUBROUTINE	CALLED	SUBROUTINE	CALLED
FEDRAT	FTYPE4	FXPARK	DRETHD
FEDRAT	FTYPE6	FXPARK	PARK
FEDRAT	MACSRT	FXPARK	PERROR
FEDRAT	RAPID	FXPARK	RETHD
FEDRAT	RAPIDX	FXTOL	SQRT
FEDRAT	STOREM	GEBASE	AUXLRY
FEDRAT	TSTEXT	GEBASE	COMENT
FROM	FROM3	GEBASE	ERDMP1
FROM	FROM5	GEBASE	INPUT
FROM	OUTPUT	GEBASE	IOERR
FROM	PLNSEL	GEBASE	MOTION
FROM	SRAREC	GEBASE	OUTPUT
FROM	STOREM	GEBASE	REWZ
FROM3	GEOM	GEBASE	SRFCHK
FROM5	ERDMP1	GEBASE	STOREM
FROM5	GEOM	GEBASE	TAPEOP
FROM5	LENGTH	GECPFC	SQRT
FROM5	MACSRT	GEDUMP	FLOAT
FTYPE2	COMENT	GEMON	DISPAT
FUNLNK	ERDMP1	GEMON	FUNLNK
FXMULT	FXPARK	GEMON	GEAD
FXMULT	FXTOL	GEMON	GEBASE
FXMULT	GMOUT	GEMON	GEMULT
FXMULT	PERROR	GEMON	GEPLAD
FXMULT	SEG	GEMON	INIT
FXMULT	SRAREC		

5.4.1 SUBROUTINES CALLING SUBROUTINES (cont'd)

SUBROUTINE	CALLED	SUBROUTINE	CALLED
GEMON	LLINK	GEPRE	CALCP2
GEMONT	GEOUT	GEPRE	CALCP3
GEMULT	COMENT	GEPRE	DECODE
GEMULT	CREAD	GEPRE	IDPART
GEMULT	FXMULT	GEPRE	PARNEM
GEMULT	FXPARK	GEPRE	PARNOM
GEMULT	GMCIRL	GEPRE	PPUNCH
GEMULT	GMFENC	GEPRE	PUNIDN
GEMULT	GMINIT	GEPRO1	CONTUR
GEMULT	GMLINE	GEPRO1	ERDMP1
GEMULT	GMOUT	GEPRO1	GEPRN1
GEMULT	PERROR	GEPRO1	OCMNT1
GEMULT	RETRET	GEPRO1	POSIT
GEOM	GEOM3	GEPRO1	PPUNCH
GEOM	GEOM5	GEPRO1	SETLIN
GEOM	TSTLIM	GEPRO1	SETUP1
GEOM3	SRAREC	GEPRO1	SHUFFL
GEOM5	CLASS	GEPRO1	SRAREC
GEOM5	TRUNC	GEPRO1	SROREC
GEOUT	GEPRE	GEPRO1	TITLE1
GEOUT	GEPRO1	GEPRO2	CONBCD
GEOUT	GEPRO2	GEPRO2	CONROT
GEOUT	GEPRO3	GEPRO2	CONTUR
GEOUT	LLINK	GEPRO2	GEPRN2
GEPRE	CALCP1	GEPRO2	OCMNT1

5.4.1 SUBROUTINES CALLING SUBROUTINES (cont'd)

SUBROUTINE	CALLED	SUBROUTINE	CALLED
GEPRO2	ORIGIN	GETSFC	OUTPUT
GEPRO2	POSIT	GETSFC	SRAREC
GEPRO2	PPUNCH	SETSFO	GECPFC
GEPRO2	SETLIN	GMABS	PDUMP
GEPRO2	SHUFFL	GMCIRL	ATAN2
GEPRO2	SRAREC	GMCIRL	GDWELL
GEPRO2	SROREC	GMCIRL	GMOUT
GEPRO2	TITLE2	GMCIRL	RETSFY
GEPRO3	ABSOPR	GMCIRL	SPLIT
GEPRO3	CONBCD	GMCIRL	SQRT
GEPRO3	CONTUR	GMCIRL	TESTM2
GEPRO3	GEPRN1	GMFENC	EXIT
GEPRO3	GEPRN3	GMFENC	GMSTOR
GEPRO3	GMWRIT	GMFENC	IOERR
GEPRO3	OCMNT1	GMFENC	TAPEOP
GEPRO3	PAGE	GMFENO	DISPAT
GEPRO3	POSIT	GMINIT	BUFFTP
GEPRO3	PPUNCH	GMINIT	IOERR
GEPRO3	SETLIN	GMINIT	PERROR
GEPRO3	SHUFFL	GMINIT	REWZ
GEPRO3	SQRT	GMLINE	GDWELL
GEPRO3	SRAREC	GMLINE	GMOUT
GEPRO3	SROREC	GMLINE	RETSFY
GEPRO3	TIMES	GMLINE	SPLIT
GETSFC	CONVRT	GMLINE	SQRT
		GMLINE	TESTM2

5.4.1 SUBROUTINES CALLING SUBROUTINES (cont'd)

SUBROUTINE	CALLED	SUBROUTINE	CALLED
GMOTIN	EIACOM	GOHOME	DEPART
GMOTIN	SQRT	GOHOME	OUTPUT
GMOUT	ATAN2	GOHOME	SEGMNT
GMOUT	GFDLIM	GOLINE	DEPART
GMOUT	GMABS	GOLINE	GEOM
GMOUT	GMOTIN	GOLINE	LINRTY
GMOUT	GMSTOR	GOLINE	OUTPUT
GMOUT	PERROR	GOLINE	SEGMNT
GMOUT	PREPHD	GOLINE	SFMO
GMOUT	SQRT	GOLINE	THEDOM
GMOUTO	GEMISC	IDPART	CHARID
GMREAD	GETNXR	IDPART	DOLLAR
GMSTOR	ERDMP1	IDPART	ERDMP1
GMSTOR	GEMONT	IDPART	PUNCHB
GMSTOR	GMREAD	INIT	ASSIGN
GMSTOR	LLINK	INIT	ERDMP1
GMSTOR	WEFREW	INIT	INPUT
GMWRIT	ERDMP1	INIT	REDTAP
GMWRIT	TAPEWT	INIT	STDMAC
GOCIRC	CHKAX	INPUT	TAPERD
GOCIRC	CIRINT	LEADER	OUTPUT
GOCIRC	COMENT	LEADER	STOREM
GOCIRC	COMPR	LENGTH	SQRT
GOCIRC	DETDIR	LINRTY	ATAN
GOCIRC	PLNSEL	LINRTY	CLASS
GOCIRC	PROCQD		

5.4.1 SUBROUTINES CALLING SUBROUTINES (cont'd)

SUBROUTINE	CALLED	SUBROUTINE	CALLED
LINRTY	COMENT	OPSKIP	STOREM
LINRTY	DEPART	ORIGIN	OUTPUT
LINRTY	DOT	OUTB	GMOUT
LINRTY	ERDMP1	OUTPUT	ERDMP1
LINRTY	OFFARC	OUTPUT	FEDLIM
LINRTY	OUTPUT	OUTPUT	GEOUT
LINRTY	SEGDRC	OUTPUT	IOERR
LINRTY	SQRT	OUTPUT	PUTNXR
LINRTY	TRUNC	OUTPUT	SELGRO
LOAD	CLASS	OUTPUT	SELG1
LOAD	GOHOME	OUTPUT	TSTEXT
LOAD	MACSRT	PARK	COMENT
LOAD	OUTPUT	PARK	COMPGC
MOTION	COMENT	PARK	FXTOL
MOTION	ERDMP1	PARK	GMOUT
MOTION	FROM	PARK	PERROR
MOTION	GOCIRC	PARK	RAPLIM
MOTION	GOLINE	PARK	SHFTBK
MOTION	POSMOV	PARNOM	WTREC
MOTION	TSTFLG	PERROR	GMFENC
NORM	LENGTH	PERROR	PDUMP
OFFARC	COMENT	PICKUP	OUTPUT
OFFARC	SRAREC	PICKUP	STOREM
OPCODE	COMENT	PLENTH	SQRT
OPCODE	OUTPUT	PLNSEL	OUTPUT
		POSFED	EIACOM

5.4.1 SUBROUTINES CALLING SUBROUTINES (cont'd)

SUBROUTINE	CALLED	SUBROUTINE	CALLED
POSFED	FTYPE2	PROCQD	OFFARC
POSFED	FTYPE4	PROCQD	OUTPUT
POSFED	FTYPE5	PROCQD	SEGDRC
POSFED	FTYPE6	PROCQD	SFMO
POSFED	TSTEXT	PROCQD	SQRT
POSIT	CONROT	PUNCHA	KARKNT
POSIT	POSFED	PUNCHA	TSTCNT
POSIT	RFTYPE	PUNCHA	WTREC
POSITN	OUTPUT	PUNCHB	CARDPN
POSMOV	COMENT	QUADET	QUADNT
POSMOV	MACSRT	RAPID	DWELL
POSMOV	OUTPUT	RAPID	MACSRT
POSMOV	SET12	RAPID	PLENTH
POSMOV	STOREM	RAPID	STOREM
POSMOV	TSTLIM	RAPID	TEST1
PPUNCH	PUNCHA	RAPIDP	OUTPUT
PPUNCH	PUNCHB	RAPIDP	RAPIDO
PREPHD	GMSTOR	RAPIDP	RAPIDX
PRFSEQ	COMENT	RAPIDP	SET12
PRFSEQ	OUTPUT	RAPLIM	FEDM
PROCQD	ATAN	RAPLIM	RAPM
PROCQD	DEPART	RAPLIM	SRAREC
PROCQD	DOT	RAPLIM	TEST2
PROCQD	ERDMP1	RAPM	CTCHUP
PROCQD	GEOM	RAPM	OUTB
		RAPM	PERROR

5.4.1 SUBROUTINES CALLING SUBROUTINES (cont'd)

SUBROUTINE	CALLED	SUBROUTINE	CALLED
REDTAP	BUFFTP	ROTABI	GOLINE
REDTAP	TAPEOP	ROTABI	ROTOUT
REDTAP	WEFREW	ROTABL	FLOAT
REMAIN	OUTPUT	ROTABL	ROTABA
RESTAT	DWELL	ROTABL	ROTABI
RESTAT	STOREM	ROTABL	STOREM
RETHD	COMPGC	ROTABL	TABSPD
RETHD	GMOUT	ROTATE	COMENT
RETHD	RAPLIM	ROTATE	ROTABL
RETRCT	OUTPUT	ROTATE	ROTDRF
RETRCT	RAPIDO	ROTATE	ROTIND
RETRET	GMOUT	ROTATE	ROTMAG
REWIND	COMENT	ROTATE	ROTORC
REWIND	DWELL	ROTATE	ROTUR
REWIND	OUTPUT	ROTDRF	COMENT
REWIND	STOREM	ROTDRF	OUTPUT
RFTYPE	ROTYP1	ROTHED	CLASS
ROTABA	CONROT	ROTHED	COMENT
ROTABA	OUTPUT	ROTHED	CONROT
ROTABA	RFTYPE	ROTHED	FLOAT
ROTABA	SET12	ROTHED	GOLINE
ROTABA	SROREC	ROTHED	REMAIN
ROTABA	STOREM	ROTHED	ROTMIN
ROTABI	CLASS	ROTHED	ROTMOV
ROTABI	COMENT	ROTHED	ROTOUT
ROTABI	CONROT	ROTIND	STOREM

5.4.1 SUBROUTINES CALLING SUBROUTINES (cont'd)

SUBROUTINE	CALLED	SUBROUTINE	CALLED
ROTMAG	MINMOV	SEGMNT	SEGDRC
ROTMAG	OUTPUT	SEGMNT	SFMO
ROTMAG	STOREM	SEGMNT	SRAREC
ROTMOV	STOREM	SEGMNT	SROREC
ROTOUT	OUTPUT	SEGMENT	THEDOM
ROTOUT	SROREC	SEGSAD	SRAREC
ROTUR	STOREM	SELECT	FEDOVR
SADDLE	COMENT	SELECT	SELANG
SADDLE	DWELL	SELECT	SELHED
SADDLE	OUTPUT	SELECT	SELOFS
SADDLE	SEGSAD	SELECT	SELPAL
SADDLE	SFMO	SELECT	SELRDR
SADDLE	SRAREC	SELECT	SELTAB
SADDLE	STOREM	SELGCD	CONVRT
SAFEGL	OUTPUT	SELGCD	PERROR
SAFEGL	STOREM	SELGCD	SQRT
SAFEGM	OUTPUT	SELGCR	ERDMP1
SEG	COMPGC	SELGCO	ERDMP1
SEG	SRAREC	SELGRO	SROREC
SEGDRC	ARCTAN	SELG	ERDMP1
SEGDRC	DOT	SELG	SELGCR
SEGDRC	NORM	SELG	SELGRO
SEGDRC	SIN	SELG	SRAREC
SEGDRC	SQRT	SELHED	OUTPUT
SEGMNT	MAXSRT	SELHED	SADDLE
SEGMNT	OUTPUT		

5.4.1 SUBROUTINES CALLING SUBROUTINES (cont'd)

SUBROUTINE	CALLED	SUBROUTINES	CALLED
SELHED	STOREM	SPINDL	DWELL
SELOFS	OUTPUT	SPINDL	ERDMP1
SELPAL	STOREM	SPINDL	FLOAT
SELRDR	STOREM	SPINDL	FTYPE2
SELTAB	OUTPUT	SPINDL	FTYPE4
SELTUL	CLASS	SPINDL	FTYPE6
SELTUL	COMENT	SPINDL	LOCRNG
SELTUL	OUTPUT	SPINDL	MACSRT
SELTUL	TOOLNO	SPINDL	OUTPUT
SETUP1	CONBCD	SPINDL	SPTYPE
SETUP1	SETLIN	SPINDL	STOREM
SFMO	COMENT	SPINDL	TSTEXT
SFMO	DEPART	SPLIT	CIRSEG
SFMO	DSRROW	SPLIT	SELGCD
SFMO	OFFARC	SPLIT	SRAREC
SFMO	OUTPUT	SPTYPE	DSRROW
SFMO	SPTYPE	SPTYPE	LOCRNG
SFMO	SQRT	SPTYPE	MACSRT
SFMO	TRUNC	SPTYPE	SPNTYP
SFMO	TSTEXT	SRFCHK	SRAREC
SFMO	TSTSAF	STOP	COMENT
SHFTBK	CTCHUP	STOP	ENTRAP
SHFTBK	FEDM	STOP	OUTPUT
SHFTBK	RAPM	STOP	STOREM
SPINDL	COMENT	STOPTS	PERROR
		STOREM	OUTPUT

5.4.1 SUBROUTINES CALLING SUBROUTINES (cont'd)

SUBROUTINE	CALLED	SUBROUTINE	CALLED
TESTM2	COMENT	TRUNC	SRAREC
TESTM2	SQRT	TRUNC	SROREC
THREAD	COMENT	TSTFLG	RAPIDO
THREAD	MACSRT	TSTFLG	RAPIDX
THREAD	SQRT	TSTFLG	RESTAT
THREDM	COMENT	TSTFLG	SAFETO
THREDM	MACSRT	TSTFLG	SAFETX
THREDM	STOREM	TSTLIM	COMENT
TITLE2	GEPRN2	TSTLIM	MACSRT
TITLE3	GEPRN1	TSTSAF	COMENT
TMARK1	PPUNCH	TSTSAF	TSTEXT
TOOLGM	DWELL	TURRET	COMENT
TOOLGM	OUTPUT	TURRET	COMTAT
TOOLGM	STOREM	TURRET	DWELL
TOOLGP	OUTPUT	TURRET	ERDMP1
TOOLL	DWELL	TURRET	MINMOV
TOOLL	OUTPUT	TURRET	OUTPUT
TOOLL	STOREM	TURRET	SRAREC
TOOLNO	COMENT	TURRET	STOREM
TOOLNO	MACSRT	TURRET	TURSAD
TOOLNO	MINMOV	TURSAD	DWELL
TOOLNO	STOREM	TYPE01	EIACOM
TOOLNO	TOOLGM	TYPE02	DWELL
TOOLNO	TOOLGP	TYPE02	GESCOM
TOOLNO	TOOLL	TYPE03	COMENT
		TYPE03	DSRROW

5.4.1 SUBROUTINES CALLING SUBROUTINES

SUBROUTINE	CALLED	SUBROUTINE	CALLED
TYPE03	DWELL	TYPE13	STOREM
TYPE03	STOREM	TYPE14	GESCOM
TYPE04	COMENT	TYPE14	STOREM
TYPE04	DWELL	UNLOAD	COMENT
TYPE04	EIACOM	UNLOAD	MACSRT
TYPE04	STOREM	UNLOAD	OUTPUT
TYPE05	EIACOM	UNLOAD	STOREM
TYPE10	COMENT	WEFREW	BUFFTP
TYPE10	DWELL	WEFREW	ERDMP1
TYPE10	EIACOM	WEFREW	TAPEOP
TYPE10	STOREM	WEFREW	TAPEOP
TYPE12	DWELL	WELD	COMENT
TYPE13	COMENT	WELD	STOREM
TYPE13	DWELL	XOFSET	OUTPUT
TYPE13	EIACOM		

5.4.2 SUBROUTINES CALLED BY SUBROUTINES

SUBROUTINE	CALLED BY	SUBROUTINE	CALLED BY
ABSOPR	GEPR03	BUFFTP	GEBASE
AIR	AUXLRY	BUFFTP	GEFENC
ARCTAN	CLASS1	BUFFTP	GMINIT
ARCTAN	CLASS2	BUFFTP	REDTAP
ARCTAN	CLASS3	CALCP1	GEPRE
ARCTAN	CLASS4	CALCP2	CALCP1
ARCTAN	CLASS6	CALCP3	GEPRE
ARCTAN	CLASS7	CARDPN	PUNCHB
ARCTAN	CLASS8	CHARID	IDPART
ARCTAN	CLASS9	CHKAX	GOCIRC
ARCTAN	CLAS10	CIRINT	GOCIRC
ARCTAN	CLAS11	CIRSEG	SPLIT
ARCTAN	CLAS12	CLAMP	AUXLRY
ARCTAN	SEGDRC	CLASS	GEOM5
ASSIGN	INIT	CLASS	LINRTY
ATAN2	GMCIRL	CLASS	LOAD
ATAN2	GMOUT	CLASS	ROTABI
ATAN	ARCTAN	CLASS	ROTHED
ATAN	LINRTY	CLASS	SELTUL
ATAN	PROCQD	CLRSRF	AUXLRY
AUXFUN	AUXLRY	COMBIN	AUXLRY
AUXLRY	GEBASE	COMPFC	TSTFCM
BREAK	AUXLRY	COMENT	AUXLRY
BUFFTP	WEFREW	COMENT	CLAMP
		COMENT	COUPLE

5.4.2 SUBROUTINE CALLED BY SUBROUTINES (cont'd)

SUBROUTINE	CALLED BY	SUBROUTINE	CALLED BY
COMENT	CYCLEL	COMENT	SELTUL
COMENT	CYCLGP	COMENT	SFMO
COMENT	CYCLGX	COMENT	SPINDL
COMENT	DELAY	COMENT	STOP
COMENT	DSRROW	COMENT	TESTM2
COMENT	EIACOM	COMENT	THREAD
COMENT	EIACOM	COMENT	THREDM
COMENT	END	COMENT	TOOLNO
COMENT	FEDRAT	COMENT	TSTLIM
COMENT	FTYPE2	COMENT	TSTSAF
COMENT	GEBASE	COMENT	TURRET
COMENT	GEMULT	COMENT	TYPE03
COMENT	GOCIRC	COMENT	TYPE04
COMENT	LINRTY	COMENT	TYPE10
COMENT	MOTION	COMENT	TYPE13
COMENT	OFFARC	COMENT	UNLOAD
COMENT	OPCODE	COMENT	WELD
COMENT	PARK	COMPGC	DRETHD
COMENT	POSMOV	COMPGC	PARK
COMENT	PRFSEQ	COMPGC	RETHD
COMENT	REWIND	COMPGC	SEG
COMENT	ROTABI	COMPR	GOCIRC
COMENT	ROTATE	COMTAT	TURRET
COMENT	ROTDRF	CONBCD	GEPR02
COMENT	ROTHED	CONBCD	GEPR03
COMENT	SADDLE		

5.4.2 SUBROUTINE CALLED BY SUBROUTINES (cont'd)

SUBROUTINE	CALLED BY	SUBROUTINE	CALLED BY
CONBCD	SETUP1	COS	CLAS11
CONROT	ASSIGN	COS	CLAS12
CONROT	GEPR02	COUPLE	AUXLRY
CONROT	POSIT	CREAD	GEMULT
CONROT	ROTABA	CTCHUP	SHFTBK
CONROT	ROTABI	CTCHUP	RAPM
CONTUR	ROTHED	CUTCOM	AUXLRY
CONTUR	GEPR01	CYCLEL	AUXLRY
CONTUR	GEPR02	CYCLGP	AUXLRY
CONTUR	GEPR03	CYCLGX	AUXLRY
CONTUR	GEPR05	DECODE	GEPRE
CONVRT	CIRSEG	DELAY	AUXLRY
CONVRT	FEDLIM	DEPART	GOHOME
CONVRT	GETSFC	DEPART	GOLINE
CONVRT	SELGCD	DEPART	LINRTY
COOLNT	AUXLRY	DEPART	PROCQD
COS	CIRSEG	DEPART	SFMO
COS	CLASS1	DETDIR	GOCIRC
COS	CLASS2	DISPAT	GMFENC
COS	CLASS3	DISPAT	ERDMP1
COS	CLASS4	DISPAT	GEMON
COS	CLASS6	DOLLAR	IDPART
COS	CLASS7	DOT	LINRTY
COS	CLASS8	DOT	PROCQD
COS	CLASS9	DOT	SEGDRC
COS	CLAS10		

5.4.2 SUBROUTINE CALLED BY SUBROUTINES (cont'd)

SUBROUTINE	CALLED BY	SUBROUTINE	CALLED BY
DRAFT	AUXLRY	EIACOM	CONTUR
DRESS	AUXLRY	EIACOM	GMOTIN
DRETHD	FXPARK	EIACOM	POSFED
DSRROW	SFMO	EIACOM	TYPE01
DSRROW	SPTYPE	EIACOM	TYPE04
DSRROW	TYPE03	EIACOM	TYPE05
DWELL	CLAMP	EIACOM	TYPE10
DWELL	COUPLE	EIACOM	TYPE13
DWELL	END	ERDMP1	SPINDL
DWELL	FEDRAT	ERDMP1	TURRET
DWELL	RAPID	EXIT	GMFENC
DWELL	RESTAT	FEDLIM	OUTPUT
DWELL	REWIND	FEDM	RAPLIM
DWELL	SADDLE	FEDM	SHFTBK
DWELL	SPINDL	FEDOVR	SELECT
DWELL	TOOLGM	FEDRAT	AUXLRY
DWELL	TOOLL	FLAME	AUXLRY
DWELL	TURRET	FLOAT	CYCLGP
DWELL	TURSAD	FLOAT	FEDRAT
DWELL	TYPE02	FLOAT	GEDUMP
DWELL	TYPE03	FLOAT	ROTABL
DWELL	TYPE04	FLOAT	ROTHED
DWELL	TYPE10	FLOAT	SPINDL
DWELL	TYPE12	FROM3	FROM
DWELL	TYPE13	FROM5	FROM

5.4. SUBROUTINE CALLED BY SUBROUTINES (cont'd)

SUBROUTINE	CALLED BY	SUBROUTINE	CALLED BY
FROM	MOTION	GEMISC	GMOUTO
FTYPE2	CYCLGP	GEMONT	GMSTOR
FTYPE2	FEDRAT	GEMULT	GEMON
FTYPE2	POSFED	GEOM3	GEOM
FTYPE2	SPINDL	GEOM5	GEOM
FTYPE4	FEDRAT	GEOM	FROM3
FTYPE4	POSFED	GEOM	FROM5
FTYPE4	SPINDL	GEOM	GOLINE
FTYPE5	POSFED	GEOM	PROCQD
FTYPE6	CYCLGP	GEOUT	GEMONT
FTYPE6	FEDRAT	GEOUT	OUTPUT
FTYPE6	POSFED	GEPLAD	GEMON
FTYPE6	SPINDL	GEPRE	GEOUT
FVARGO	CONTUR	GEPRN1	GEPR01
FXMULT	GEMULT	GEPRN1	GEPR03
FXPARK	FXMULT	GEPRN1	TITLE3
FXPARK	GEMULT	GEPRN2	GEPR02
FXTOL	DRETHD	GEPRN2	TITLE2
FXTOL	FXMULT	GEPRN3	ABSOPR
FXTOL	PARK	GEPRN3	GEPR03
GDWELL	GMCIRL		
GDWELL	GMLINE	GEPR01	GEOUT
GEAD	GEMON	GEPR02	GEOUT
GEBASE	GEMON	GEPR03	GEOUT
GEDUMP	ERDMP1		

5.4.2 SUBROUTINE CALLED BY SUBROUTINES (cont'd)

SUBROUTINE	CALLED BY	SUBROUTINE	CALLED BY
GESCOM	TYPE02	GMWRIT	GEPR03
GESCOM	TYPE14	GOCIRC	MOTION
GFDLIM	GMOUT	GOHOME	AUXLRY
GMABS	GMOUT	GOHOME	LOAD
GMCIRL	GEMULT	GOLINE	MOTION
GMFENC	GEMULT	GOLINE	ROTABI
GMFENC	PERROR	GOLINE	ROTHED
GMINIT	GEMULT	IDPART	GEPRE
GMLINE	GEMULT	INIT	GEMON
GMOTION	GMOUT	INPUT	GEBASE
GMOUT	CTCHUP	INPUT	INIT
GMOUT	DRETHD	INSERT	AUXLRY
GMOUT	FXMULT	IOERR	CREAD
GMOUT	GEMULT	IOERR	GEBASE
GMOUT	GMCIRL	IOERR	GMFENC
GMOUT	GMLINE	IOERR	GMINIT
GMOUT	OUTB	IOERR	OUTPUT
GMOUT	PARK	LEADER	AUXLRY
GMOUT	RETHD	LENGTH	CIRSEG
GMOUT	RETRET	LENGTH	CONTUR
GMREAD	ABSOPR	LENGTH	FEDLIM
GMREAD	GMSTOR	LENGTH	FROM5
GMSTOR	GMFENC	LENGTH	NORM
GMSTOR	GMOUT	LINRTY	GOLINE
GMSTOR	PREPHD	LINTOL	AUXLRY

5.4.2 SUBROUTINE CALLED BY SUBROUTINES (cont'd)

SUBROUTINE	CALLED BY	SUBROUTINE	CALLED BY
LOAD	AUXLRY	OUTPUT	DWELL
LOCRNG	SPINDL	OUTPUT	END
LOCRNG	SPTYPE	OUTPUT	FROM
MCHFIN	AUXLRY	OUTPUT	GEBASE
MINMOV	ROTMAG	OUTPUT	GOHOME
MINMOV	TOOLNO	OUTPUT	GOLINE
MINMOV	TURRET	OUTPUT	LEADER
MODE	AUXLRY	OUTPUT	LINRTY
MOTION	GEBASE	OUTPUT	LOAD
NORM	CLASS1	OUTPUT	OPCODE
NORM	SEGDRC	OUTPUT	ORIGIN
OFFARC	LINRTY	OUTPUT	PICKUP
OFFARC	PROCQD	OUTPUT	PLNSEL
OFFARC	SFMO	OUTPUT	POSITN
OPCODE	AUXLRY	OUTPUT	POSMOV
OPSKIP	AUXLRY	OUTPUT	PRFSEQ
OPSTOP	AUXLRY	OUTPUT	PROCQD
ORIGIN	AUXLRY	OUTPUT	RAPIDP
ORIGIN	GEPR02	OUTPUT	REMAIN
OUTB	RAPM	OUTPUT	RETRCT
OUTPUT	AUXFUN	OUTPUT	REWIND
OUTPUT	CUTCOM	OUTPUT	ROTABA
OUTPUT	CYCLEL	OUTPUT	ROTDRF
OUTPUT	CYCLGP	OUTPUT	ROTMAG
OUTPUT	DELAY		

5.4.2 SUBROUTINE CALLED BY SUBROUTINES (cont'd)

SUBROUTINE	CALLED BY	SUBROUTINE	CALLED BY
OUTPUT	ROTOUT	PARTNO	AUXLRY
OUTPUT	SADDLE	PDUMP	ERDMP1
OUTPUT	SAFEGL	PDUMP	GMABS
OUTPUT	SAFEGM	PDUMP	PERROR
OUTPUT	SEGMNT	PERROR	DRETHD
OUTPUT	SELHED	PERROR	FXMULT
OUTPUT	SELOFS	PERROR	FXPARK
OUTPUT	SELTAB	PERROR	GEMULT
OUTPUT	SELTUL	PERROR	GMINIT
OUTPUT	SFMQ	PERROR	GMOUT
OUTPUT	SPINDL	PERROR	PARK
OUTPUT	STOP	PERROR	RAPM
OUTPUT	STOREM	PERROR	SELGCD
OUTPUT	TOOLGM	PERROR	STOPTS
OUTPUT	TOOLGP	PICKUP	AUXLRY
OUTPUT	TOOLL	PITCH	AUXLRY
OUTPUT	TURRET	PITCHM	AUXLRY
OUTPUT	UNLOAD	PIVPLN	AUXLRY
OUTPUT	XOFSET	PLENTH	RAPID
OVRCNT	AUXLRY	PLNSEL	FROM
PAGE	ABSOPR	PLNSEL	GOCIRC
PAGE	GEPR03	POSFED	POSIT
PARK	FXPARK	POSIT	GEPR01
PARNEM	GEPRE	POSIT	GEPR02
PARNOM	GEPRE	POSIT	GEPR03
		POSITN	AUXLRY

5.4.2 SUBROUTINE CALLED BY SUBROUTINES (cont'd)

SUBROUTINE	CALLED BY	SUBROUTINE	CALLED BY
POSMOV	MOTION	RAPIDX	CYCLGP
PPRINT	AUXLRY	RAPIDX	CYCLGX
PPUNCH	GEPRE	RAPIDX	FEDRAT
PPUNCH	GEPR01	RAPIDX	RAPIDP
PPUNCH	GEPR02	RAPIDX	TSTFLG
PPUNCH	GEPR03	RAPLIM	DRETHD
PPUNCH	GEPR05	RAPLIM	PARK
PPUNCH	TMARK1	RAPLIM	RETHD
PREFUN	AUXLRY	RAPM	RAPLIM
PREPHD	GMOUT	RAPM	SHFTBK
PRFSEQ	AUXLRY	REDTAP	INIT
PROCQD	GOCIRC	RESET	AUXLRY
PUNCHA	PPUNCH	RESTAT	TSTFLG
PUNCHB	IDPART	RETHD	FXPARK
PUNCHB	PPUNCH	RETRCT	AUXLRY
PUNIDN	GEPRE	RETRET	GEMULT
QUADET	CIRINT	RETSFY	GMLINE
QUADNT	QUADET	REWIND	AUXLRY
RAPID	AUXLRY	RFTYPE	POSIT
RAPID	FEDRAT	RFTYPE	ROTABA
RAPIDO	RAPIDP	ROTABA	ROTABL
RAPIDO	RETRCT	ROTABI	ROTABL
RAPIDO	TSTFLG	ROTABL	ROTATE
RAPIDP	CYCLGP	ROTATE	AUXLRY
RAPIDX	CYCLEL	ROTDRF	ROTATE

5.4.2 SUBROUTINE CALLED BY SUBROUTINES (cont'd)

SUBROUTINE	CALLED BY	SUBROUTINE	CALLED BY
ROTIND	ROTATE	SELECT	AUXLRY
ROTMAG	ROTATE	SELG	OUTPUT
ROTMIN	ROTHED	SELGCD	CIRSEG
ROTMOV	DEPART	SELGCD	COMPGC
ROTMOV	ROTHED	SELGCD	SPLIT
ROTORC	ROTATE	SELGCR	SELG
ROTOUT	ROTABI	SELGRO	OUTPUT
ROTOUT	ROTHED	SELGRO	SELG
ROTUR	ROTATE	SELHED	AUXLRY
ROTYPE1	RFTYPE	SELHED	SELECT
SADDLE	SELHED	SELOFS	SELECT
SAFEGL	AUXLRY	SELPAL	SELECT
SAFEGM	AUXLRY	SELRAD	SELECT
SAFEGX	AUXLRY	SELRDR	SELECT
SAFETO	TSTFLG	SELTAB	SELECT
SAFETX	TSTFLG	SELTUL	SELECT
SAVMCD	CREAD	SEQNO	AUXLRY
SEG	FXMULT	SET12	POSMOV
SEGDRC	LINRTY	SET12	RAPIDP
SEGDRC	PROCQD	SET12	ROTABA
SEGDRC	SEGMNT	SETLIN	CALCP1
SEGMNT	GOHOME	SETLIN	CALCP2
SEGMNT	GOLINE	SETLIN	CALCP3
SEGSAD	SADDLE	SETLIN	CALCP5
SELANG	SELECT	SETLIN	GEPR01

5.4.2 SUBROUTINE CALLED BY SUBROUTINES (cont'd)

SUBROUTINE	CALLED BY	SUBROUTINE	CALLED BY
SETLIN	GEPR02	SIN	CLAS11
SETLIN	GEPR03	SIN	CLAS12
		SIN	SEGDRC
SETLIN	SETUP1	SPINDL	AUXLRY
SETUP1	GEPR01	SPLIT	GMCIRL
SFMO	GOLINE	SPLIT	GMLINE
SFMO	PROCQD	SPNTYP	SPTYPE
SFMO	SADDLE	SPTYPE	COUPLE
SFMO	SEGMNT	SPTYPE	SFMO
SHFTBK	PARK	SPTYPE	SPINDL
SHOLZR	PPUNCH	SQRT	CLASS1
SHUFFL	GEPR01	SQRT	COMPR
SHUFFL	GEPR02	SQRT	CONTUR
SHUFFL	GEPR03	SQRT	FXTOL
		SQRT	COMPFC
SIN	CIRSEG	SQRT	GEPR03
SIN	CLASS1	SQRT	GMCIRL
SIN	CLASS2	SQRT	GMLINE
SIN	CLASS3	SQRT	GMOTIN
SIN	CLASS4	SQRT	GMOUT
SIN	CLASS6	SQRT	LENGTH
SIN	CLASS7	SQRT	LENGTH
SIN	CLASS8	SQRT	LINRTY
SIN	CLASS9	SQRT	PLENTH
SIN	CLAS10	SQRT	PROCQD

5.4.2 SUBROUTINE CALLED BY SUBROUTINES (cont'd)

SUBROUTINE	CALLED BY	SUBROUTINE	CALLED BY
SQRT	SEGDRC	SRAREC	TRUNC
SQRT	SELGCD	SRAREC	TURRET
SQRT	SFMC	SRFCHK	GEBASE
SQRT	TESTM2	SROREC	GEPR01
SQRT	THREAD	SROREC	GEPR02
SRAREC	CIRSEG	SROREC	GEPR03
SRAREC	DRETHD		
SRAREC	FROM	SROREC	ROTABA
SRAREC	FXMULT	SROREC	ROTOUT
SRAREC	GEOM3	SROREC	SEGMNT
SRAREC	GEPRO1	SROREC	SELGRD
SPAREC	GEPRO2	SROREC	TRUNC
SPAREC	GEPRO3	STDMAC	INIT
		STOP	AUXLRY
SRAREC	TSTFCM	STOPTS	CIRSEG
SRAREC	OFFARC	STOREM	AIR
SRAREC	RAPLIM	STOREM	AUXFUN
SRAREC	SADDLE	STOREM	CLAMP
SRAREC	SEG	STOREM	COMENT
SRAREC	SEGMNT	STOREM	COOLNT
SRAREC	SEGSAD	STOREM	COUPLE
SRAREC	SELG1	STOREM	CYCLEL
SRAREC	SPLIT	STOREM	CYCLGP
SRAREC	SRFCHK	STOREM	DELAY
		STOREM	DRESS

5.4.2 SUBROUTINE CALLED BY SUBROUTINES (cont'd)

SUBROUTINE	CALLED BY	SUBROUTINE	CALLED BY
STOREM	END	STOREM	THREDM
STOREM	FEDOVR	STOREM	TOOLGM
STOREM	FEDRAT	STOREM	TOOLL
STOREM	FROM	STOREM	TOOLNO
STOREM	GEBASE	STOREM	TURRET
STOREM	LEADER	STOREM	TYPE03
STOREM	OPSKIP	STOREM	TYPE04
STOREM	PICKUP	STOREM	TYPE10
STOREM	POSMOV	STOREM	TYPE13
STOREM	RAPID	STOREM	TYPE14
STOREM	RESTAT	STOREM	UNLOAD
STOREM	REWIND	STOREM	WELD
STOREM	ROTABA	TABSPD	ROTABL
STOREM	ROTABL	TAPERD	CREAD
STOREM	ROTIND	TAPERD	GMREAD
STOREM	ROTMAG	TAPERD	INPUT
STOREM	ROTMOV	TAPEWT	GMWRIT
STOREM	ROTUR	TAPEWT	GMWRIT
STOREM	SADDLE	TAPEWT	OUTPUT
STOREM	SAFEGL	TEST1	RAPID
STOREM	SELHED	TEST2	RAPLIM
STOREM	SELPAL	TESTM2	GMCIRL
STOREM	SELRDR	TESTM2	GMLINE
STOREM	SPINDL	THEDOM	GOLINE
STOREM	STOP	THEDOM	SEGMNT
		THREAD	AUXLRY

5.4.2 SUBROUTINE CALLED BY SUBROUTINES (cont'd)

SUBROUTINE	CALLED BY	SUBROUTINE	CALLED BY
THREDM	AUXLRY	TSTEXT	TSTSAF
TIMES	ABSOPR	TSTFCM	SELG
TIMES	GEPR03	TSTFLG	MOTION
TITLE1	GEPR01	TSTLIM	GEOM
TITLE2	GEPR02	TSTLIM	POSMOV
TITLE3	ABSOPR	TSTSAF	SFMC
		TURRET	AUXLRY
TMARK	AUXLRY	TURSAD	TURRET
TOOLGM	TOOLNO	UNLOAD	AUXLRY
TOOLGP	TOOLNO	WEFREW	ABSOPR
TOOLL	TOOLNO	WEFREW	GMSTOR
TOOLNO	AUXLRY	WEFREW	REDTAP
TOOLNO	SELTUL	WELD	AUXLRY
TRANS	AUXLRY	XOFSET	AUXLRY
TRUNC	GEOM5		
TRUNC	LINRTY		
TRUNC	SFMO		
TSTEXT	CONTUR		
TSTEXT	CYCLGP		
TSTEXT	FEDRAT		
TSTEXT	OUTPUT		
TSTEXT	POSFED		
TSTEXT	SFMO		
TSTEXT	SPINDL		

5.4.3 COMPUTER DEPENDENT SUBROUTINES

The following list of subroutines have one or more restrictions which can make them computer dependent. These restrictions are defined as:

(1) Requires APT System COMMON (ASC)

(2) Uses DATA statements (DS)

(3) Has multiple entries (ME)

(4) Programmed in Machine Language (ML)

(5) Uses APT Section 0 subroutines (SOS)

(6) The Subroutine is based upon the computer word structure (WS)

(7) The subroutine is very large at the source level (VL)

(8) The subroutine may require an overlay call (OC)

Any one or more of these restrictions may exist on a given computer, thereby requiring a modification to the subroutine to make it compatible to the computer.

The subroutines are listed by overlay grouping. The common multiple entry subroutines are given together.

GEMON

CONROT (DS)

ERDMP1 (ASC,OC)

INPUT (ABC,DS,SOS)

GEMON (DS,OC)

SRAREC (ME-SROREC)

SROREC (ME-SROREC)

WEFREW (ASC, SOS)

GEINIT

DECODE (DS)

GEPRE (DS,OC)

IDPART (DS)

INIT (DS,ASC,OC)

REDTAP (ASC,SOS)

STDMAC (DS)

5.4.3 COMPUTER DEPENDENT SUBROUTINES (cont'd)

GEBASE

BREAK (ME-OPSTOP,STOP)

END (ME-RESET)

ENTRAP (ME-RAPID,RAPIDO,RAPIDX)

FEDRAT (VL)

FLOAT (WS)

GEBASE (ASC)

INSERT (ME-PARTNO,PPRINT)

LEADER (ME-PREFUN,TMARK)

MACHIN (ASC)

OPSTOP (ME-BREAK,STOP)

OUTPUT (ASC)

PARTNO (ME-INSERT,PPRINT)

POSITN (DS)

PPRINT (ME-INSERT,PARTNO)

PREFUN (ME-LEADER,TMARK)

RAPID (ME-ENTRAP,RAPIDO,RAPIDX)

RAPIDO (ME-ENTRAP, RAPID, RAPIDX)

RAPIDX (ME-ENTRAP,RAPID,RAPIDO)

RESET (ME-END)

SELHED (DS)

SPINDL (DS,VL)

STOP (ME-BREAK,OPSTOP)

TMARK (ME-LEADER,PREFUN)

TOOLNO (DS)

TSTLIM (DS,ASC)

5.4.3 COMPUTER DEPENDENT SUBROUTINES (cont'd)

GETERP

COMPR (DS)

DETDIR (DS)

QUADET (DS)

GEPOS

SET12 (DS)

TOOLGP (DS)

SAFEGL (ME-SAFETO,SAGETX)

SAFETO (ME-SAFEGL,SAFETX)

SAFETX (ME-SAFEGL,SAFETO)

SFMO (DS,VL)

THREAD (ME-THREDO)

THREDO (ME-THREAD)

TOOL (DS)

TURRET (DS)

GEMILL

ROTABI (DS,VL)

ROTMOV (DS)

ROTOUT (DS)

SELGRO (DS)

TABSPD (DS)

TOOLGM (DS)

GEMAXS

ARCTAN (DS)

OVRCNT (DS)

SAFEGX (DS)

5.4.3 COMPUTER DEPENDENT SUBROUTINES (cont'd)

GEOUT

CHARID (ML)

*CONBCD (ML)

DOLLAR (ML)

GEOUT (DS,OC)

*PARNEM (ML)

*PARNOM (ML)

*PUNCHA (ML)

*PUNCHB (ML)

*PUNIPN (ML)

SETLIN (ML)

SHOLZR (ML)

*These subroutines may be in APT Section IV on some computers.

GEOUT1

CALCP1 (DS)

GEPRN1 (ASC)

GEPRO1 (DS,ASC,VL)

SETUP1 (DS)

TITLE1 (ASC)

GEOUT2

CALCP2 (DS)

GEPRN2 (ASC)

GEPRO2 (DS,ASC,VL)

TITLE2 (ASC)

GEOUT3

ABSOPR (ASC)

CALCP3 (DS)

GEPRN3 (ASC)

GEPRN3 (DS,ASC,VL)

GEPRO3 (DS,ASC,VL)

PAGE (ASC)

TIMES (ASC)

TITLE3 (ASC)

GEOUT4

GEOUT4 uses the same subroutines as GEOUT3.

5.4.3 COMPUTER DEPENDENT SUBROUTINES (cont'd)

GEMULT

CREAD (ASC,SOS)
DRETHD (ASC)
FEDM (ME-RAPM)
FXPARK (ASC)
GEMULT (ASC,VL)
GMFENC (ASC,SOS)
GMINIT (ASC,SOS)
GMOUT (VL)
GMREAD (ASC,SOS)
GMSTOR (ASC,SOS,OC,DS)
GMWRIT (ASC,SOS)
PARK (ASC)
PERROR (ASC,SOS)
PREPHD (DS)
RAPM (ME-FEDM)

GESPIN

TYPE03 (DS)
TYPE04 (DS)
TYPE09 (DS)
TYPE10 (DS)
TYPE13 (DS)
TYPE14 (DS)

OTHER

GEDUMP (DS,ASC,SOS)
FUNLNK (DS,OC)

5.5 INFORMATION BLOCKS

For non-multihead NC machines, the GECENT III postprocessor reads a record from the CL tape, processes it, and then outputs it, all in one pass. However, multihead NC machines which operate in a combined mode at execution require a two-pass system: the first pass to process the CL tape, and the second pass to merge the generated command blocks; see Section 3.4.8.

Since many postprocessor functions must be performed in the second pass, it is essential to pass along to the second pass those items of information which, though not to be output, are essential to the execution of the postprocessor function. For example, when a multihead circular interpolation command block is set up, the postprocessor must know the radius of the circle and angle of subtended arc. These are immediately available in the one-pass system since all these data concurrently reside in core. But this is not true for the second-pass system.

Therefore, the postprocessor must pass this necessary information to the second pass in a block preceding the functional data, so that the needed information is stored into core to become available when the postprocessor function is executed. Hence, in the above example, when in the second pass the postprocessor prepares to set up a new segment of the circle path (as may occur during a merge sequence), or to compute the feedrate command, the postprocessor will have the circle radius and angle of arc available since these data would have been obtained from the command block immediately preceding the circle move.

Several types of Information Blocks may be required by the second pass, and these are defined below. An Information Command Block is identified by a CODE = -9, and the type of Information Block is identified by the numeric value of DBFSEG(2).

The Information Blocks are generated in the first-pass subroutines; the second-pass subroutine GEMULT interrogates each block CODE, and on a CODE = -9, sets up the appropriate COMMON parameters with the data from the Information Block. These data remain in memory until changed by data from another Information Block.

5.5 INFORMATION BLOCKS (cont'd)

Information Blocks (CODE = -9)

DBFSEG(2) = 1

This gives the tool correction ΔT which makes the summation of the Z-axis motion increments correct for the Absolute Print. ΔT is stored at DBFSEG(3).

DBFSEG(2) = 2

The programmed values of the ORIGIN statement ORIGIN/X,Y,Z,A,B,C are stored in DBFSEG (3 thru 8), respectively. These ORIGIN values are used in producing the Operator Printout.

DBFSEG(3) = 3

The data given in a SEQNO statement are stored beginning in DBFSEG(3) and continuing as needed.

DBFSEG(2) = 4

This block carries either the first or the last cut vector of a circle; it is used in GEAD for determining ϕ , the angle of deflection. DBFSEG(3,4,5) carry X,Y,Z respectively.

DBFSEG(2) = 5

This block gives the increment of X to add to the summation of increments for producing the Absolute Printout when the tool relocates because of a multiaxis head change. The ΔX value is stored in DBFSEG(3).

DBFSEG(2) = 6

(Presently unassigned.)

DBFSEG(2) = 7

This block gives the corrective increments ΔX and ΔY which result when a turret indexes to a new tool and the TURRET statement uses the NEXT modifier. These corrective increments must be added to the incremental summation for the Absolute Print. DBFSEG(3) = ΔX and DBFSEG(4) = ΔY. Note: The corrective increments ΔX and ΔY are shuffled according to the specifications of option 59.

5.5 INFORMATION BLOCKS (cont'd)

DBFSEG(2) = 8

This block is used for a CUTCOM condition which changes the feedrate command maximum when cutter compensation is in mode. DBFSEG(3) contains the feedrate command maximum, and DBFSEG(4) contains the rapid traverse command maximum.

DBFSEG(2) = 9

(Presently unassigned.)

DBFSEG(2) = 10

The data required for multiaxis circular interpolation are stored in this information block.

DBFSEG(3) = part radius of circle

DBFSEG(4) = angle of subtended arc.

5.6 THE MACHINE SUBROUTINE

The generality of the GECENT postprocessor is based on the concept that most NC machines have features common to one another, but when there are differences, these differences can be collected into a table which represents the machine's characteristics. If this collection of differences were compiled for all machines, it would be possible to completely specify any given NC machine simply by selecting those items from the table which were characteristic for that particular machine.

The GECENT postprocessor capitalizes on the collective table concept, but in a restrictive sense, in that only those NC machine differences which occur in the greatest number are included in the table. Incorporating a complete table would not be practical because some differences are unique for one machine, while some differences apply in varying degrees to other machines. In this way the majority of NC machines are represented.

The table of NC machine differences can be further grouped into other smaller groups whose members all share some common feature. Examples of the difference groupings are: (1) items related as preparatory functions; (2) miscellaneous functions; and (3) the formats for output of machine codes. Other groupings are possible but these three are essential for every NC machine.

For the user's convenience, the GECENT postprocessor considers item 3, the output format, as two tables, viz., the word format table and the word address table.

Almost every NC machine has its own set of spindle speeds, and these must be made available to the postprocessor. The postprocessor optimizes storage by using the same table for both speeds and feeds for those NC machines which require a discrete set of feedrates.

The remainder of the major machine differences are collected in a table called OPTAB. All the different groups in the GECENT postprocessor are compiled in the following tables:

(1) Preparatory functions (TABLEG)

(2) Miscellaneous functions (TABLEM)

(3) Register addresses (REGSTR)

(4) Register format descriptions (REGFOR)

(5) Spindle speed table (SRTAB)

(6) The remainder of major machine differences (OPTAB)

5.6 THE MACHINE SUBROUTINE (cont'd)

Actually, OPTAB is a table of options, representing the collected machine differences, each of which can be made a value according to some optional choice. These choices may be requirements stipulated by the part programmer or computer programmer. An example of an option is: Should the postprocessor automatically insert dwells for spindle range changes? Further details concerning table usage are discussed in the paragraphs below.

The GECENT postprocessor extends the generalized concept a bit further by combining all the above-mentioned tables into a representation of a theoretical NC machine, called the Standard Machine. This machine has characteristics that frequently occur in most machines; it is an "average" machine. The postprocessor is developed around the framework of the Standard Machine; all functions, operations, and procedures conform to the Standard Machine requirements.

To use the postprocessor for any given NC machine requires only the specification of the machine's characteristics, i.e., establish tables TABLEG, TABLEM, REGSTR, REGFOR, SRTAB, and OPTAB. Most characteristics are already present as a result of the Standard Machine setup, and the only alterations necessary are those which convert the tables to the needs of the given NC machine. The altering may entail merely deleting a function, adding a function, or changing a function. If the given machine is non-standard, sequences may be added to the Machine Subroutine to handle any specialized functions. These procedures are described in Section 5.6.1.

If a NC machine has the same characteristics as the Standard Machine, the postprocessor can be used without alteration, but whenever changes are required, the changes themselves represent a new machine, viz., the given NC machine. The required alterations to the Standard Machine normally occur in one computer subroutine; every different NC machine will have its own subroutine for postprocessor representation. These subroutines may vary in size depending upon the amount of deviation from EIA-AIA and NAS standards or from the Standard Machine, but in general, most subroutines are very small.

5.6 THE MACHINE SUBROUTINE (cont'd)

The subroutine which represents the particular NC machine is called the Machine Subroutine, and it is named according to the predetermined numeric value assigned to the NC machine. Generally speaking, if the NC machine is given the identification number nm, then the Machine Subroutine MACHnm is selected for GECENT III usage when the postprocessor encounters the statement

MACHIN/GECENT,nm.

The value of nm can be from 01 to 99 depending upon the user's choice and capability of the postprocessor. Subroutine INIT in GEINIT selects the Machine Subroutine through a computed GOTO branch; therefore, the value of nm must be compatible with the number of branches. Before assigning nm, the computer programmer must first check to see if an nm branch is available; if not, he will have to add the branch.

In writing a Machine Subroutine, each of the above-mentioned six tables must be consulted and altered as needed to represent the NC machine. These alterations can be additions, deletions, or changes to the tables. Each of the tables is described according to their use and function; each location of the table has a prescribed assignment and purpose which is unique and invariable. The indicated value assigned to a location constitutes a representation of the Standard Machine. An unassigned value is indicated by the use of the parameter DMBITS. In all cases for TABLEG and TABLEM, the presence of DMBITS at a table location means that that function is not available for the NC machine under consideration.

The next five sections define and describe in detail the five major tables that must be considered in every Machine Subroutine. The technique for programming such a subroutine is described in Section 5.6.6.

5.6.1 TABLE OF PREPARATORY FUNCTIONS (TABLEG)

The table TABLEG is an ordered listing of preparatory functions and their corresponding G codes. Each location of the table is reserved for a particular function, e.g., location 2 in the table is reserved for the G code which represents the preparatory function for straight-line interpolation; location 4 is for the preparatory function for counterclockwise circular interpolation, and so on. Actually, the location assignments are quite arbitrary; the only important factor is that once a table location is assigned, it must never be reassigned or its location redefined. Thus, as long as it is understood that location 4 of TABLEG is the place where the counterclockwise circular interpolation G code is assigned, it will not matter what code number is stored there. Hence, if machine A uses a G03 for counterclockwise circular interpolation, then machine A's TABLEG will have a 3 stored at location 4; if machine B uses a G58 for the same function, its TABLEG will have a 58 stored at location 4. The postprocessor uses whatever code number is stored at location 4 whenever a counterclockwise circular interpolation is processed. This same technique is used for all the other locations of TABLEG, viz., the table location defines the preparatory function, but the number assigned to that location is the operational G code for that function.

Most NC machines do not have all the preparatory functions possible. Hence, in order to indicate to the postprocessor that certain functions are not available, a code number is assigned to the location of those functions. This code number, called DMBITS, is the octal number, -40404040.0.

TABLEG consists of 120 locations. It is important to note that not all preparatory functions are considered by the postprocessor. Only those functions which are indicated in the TABLEG chart are the ones presently considered by the GECENT program. The preparatory functions which are used by the Standard Machine are indicated by their G code number assignment. Thus, for clockwise circular interpolation, the Standard Machine assigns the number 2 to location 3 of TABLEG. The Standard Machine does not use the threading feature, so DMBITS is assigned to the threading preparatory functions. Note, however, that threading is considered by the postprocessor and is available for those machines which can use it.

When setting up TABLEG for a given NC machine, assign that machine's G code numbers to the appropriate preparatory function in TABLEG. If certain preparatory functions are not available on the given NC machine but are given for the Standard Machine, assign those functions as DMBITS.

5.6.1 TABLE OF PREPARATORY FUNCTIONS (TABLEG) (cont'd)

TABLEG(120)

Preparatory Functions

Location	Standard Value	Function
1	DMBITS	Positioning Cycle Off
2	01	Linear Interpolation (Range 0-9.9999 in.)
3	02	Circular Interpolation - Arc CLW (Radius 0-9.9999 inches)
4	03	Circular Interpolation - Arc CCLW (Radius 0-9.9999 inches)

(NOTE: Location 3,4,21,22,23,31,32 and 34 assumes the direction by looking at the machine from above.)

Location	Standard Value	Function
5	04	Dwell
6	DMBITS	Linear Interpolation (Range 0-.00999)
7	DMBITS	Circular Interpolation - Arc CLW (Radius 0-.00999)
8	DMBITS	Circular Interpolation - Arc CCLW (Radius 0-.00999)
9	DMBITS	Circular or linear interpolation for one head (UW); linear interpolation on the other (XZ)
10	DMBITS	Circular or linear interpolation for one head (XZ) linear interpolation on the other (UW)
11	DMBITS	Linear Interpolation (Range 0-99.9999 inches)
12	DMBITS	Linear Interpolation (Range 0-0.9999 inches)
13	DMBITS	Linear Interpolation (Range 0-0.0999 inches)

5.6.1 TABLE OF PREPARATORY FUNCTIONS (TABLEG) (cont'd)

Preparatory Functions

Location	Standard Value	Function
14	DMBITS	Linear Interpolation (Range 0-999.9999 inches)
15	DMBITS	Linear Interpolation (Range 0-9999.9999 inches)
16	DMBITS	Circular Interpolation for One Head (XY)
		Linear Interpolation on the Other (UV)
17	DMBITS	Linear Interpolation for One Head (XY)
		Circular Interpolation on the Other (UV)
18	17	XY Plane Selection
19	18	XZ Plane Selection
20	19	YZ Plane Selection
21	DMBITS	Circular Interpolation - Arc CLW (Radius 0-99.9999 inches)
22	DMBITS	Circular Interpolation - Arc CLW (Radius 0-0.9999 inches)
23	DMBITS	Circular Interpolation - Arc CLW (Radius 0-0.0999 inches)
24	DMBITS	Select Tool (With Small Gripper)
25	DMBITS	Select Tool (With Large Gripper)
NOTE 24-25		Use (24) If Only One Gripper is Available
26	DMBITS	Load Tool (Small Gripper on Head 1)
27	DMBITS	Load Tool (Large Gripper on Head 1)

5.6.1 TABLE OF PREPARATORY FUNCTIONS (TABLEG) (cont'd)

Preparatory Functions

Location	Standard Value	Function
28	DMBITS	Load Tool (Small Gripper on Head 2)
29	DMBITS	Load Tool (Large Gripper on Head 2)
NOTE 26-29		Use (26) If Only One Head or Gripper is Available
30	DMBITS	
31	DMBITS	Circular Interpolation Arc - CCLW (Radius 0-99.9999 inches)
32	DMBITS	Circular Interpolation Arc - CCLW (Radius 0-0.99999 inches)
33	DMBITS	Circular Interpolation Arc - CCLW (Radius 0-0.0999 inches)
34	DMBITS	Constant Lead Thread-Cutting-Normal Lead
35	DMBITS	Increasing Lead Thread-Cutting-Normal
36	DMBITS	Decreasing Lead Thread-Cutting-Normal
37	DMBITS	Constant Lead Thread-Cutting-Extended
38	DMBITS	Increasing Lead Thread-Cutting-Extended
39	DMBITS	Decreasing Lead Thread-Cutting-Extended
40	DMBITS	
41	DMBITS	Cutter Compensation Off
42	DMBITS	Cutter Compensation Left
43	DMBITS	Cutter Compensation Right
44	DMBITS	Circular Interpolation Arc-CLW (Range 100.0-999.9999)

5.6.1 TABLE OF PREPARATORY FUNCTIONS (TABLEG) (cont'd)

Preparatory Functions

Location	Standard Value	Function
45	DMBITS	Circular Interpolation Arc-CCLW (Range 100.0-999.9999)
46	DMBITS	Cutter Compensation Plus
47	DMBITS	Cutter Compensation Minus
48	DMBITS	Dwell Off
49	DMBITS	Circular interpolation - arc CLW (Radius 1000.0 - 9999.9999 inches)
50	DMBITS	Circular interpolation - arc CCLW (Radius 1000.0 - 9999.9999 inches)
51	DMBITS	Dwell (Spindle Not Rotating)
52	DMBITS	Fine Positioning Mode With Backlash Takeup
53	DMBITS	Fine Positioning Mode With No Backlash Takeup
54	DMBITS	Coarse Positioning Mode
55	DMBITS	Fine Positioning Mode With or Without Backlash, Medium Feed
56	DMBITS	Coarse No. 1 Positioning Mode at Medium Feed
57	DMBITS	Positioning Mode at Low Feed or Coarse No. 2 at Feed
58	DMBITS	Traverse to Feed Without a Stop (IPM Mode Assumed)
59	DMBITS	Rotary Feed to Feed Without Stop

5.6.1 TABLE OF PREPARATORY FUNCTIONS (TABLEG) (cont'd)

Preparatory Functions

Location	Standard Value	Function
60	DMBITS	Cancel Corner Milling
61	DMBITS	Corner Milling With The Feedrate In Range 1
62	DMBITS	Corner Milling With The Feedrate In Range 2
63	DMBITS	Traverse To Feed Without A Stop (IPR Mode Assumed)
64	DMBITS	Coarse Position Milling-High Feedrate
65	DMBITS	Coarse Position Milling To Dead Band With Overshoot
66	DMBITS	Fine Position Milling-Low Feedrate
67	DMBITS	Read while positioning X and W plane
68	DMBITS	
69	DMBITS	
70	DMBITS	Tool Holder Index CLW
71	DMBITS	
73	DMBITS	
74	DMBITS	
75	DMBITS	
76	DMBITS	
77	DMBITS	
78	DMBITS	

5.6.1 TABLE OF PREPARATORY FUNCTIONS (TABLEG) (cont'd)

Preparatory Functions

Location	Standard Value	Function
79	DMBITS	
80	DMBITS	
81	DMBITS	Special Cancel Cycle, Used Only For ROTIND
82	DMBITS	Drilling Sequence (Drill to Depth Z) (DRILL)
83	DMBITS	Facing Sequence (Drill to Z and Dwell) (FACE)
84	DMBITS	Deep Hole Drilling (Peck Drilling) (DEEP)
85	DMBITS	Tapping Sequence (TAP)
86	DMBITS	Boring Sequence (BORE)
87	DMBITS	Milling Sequence (MILL)
88	DMBITS	Boring and Stopping Sequence (THRU) Case Drilling
89	DMBITS	Spindle Forward (OUT)
90	DMBITS	Spindle Withdraw (IN)
91	DMBITS	Feed Spindle to Z, Stop Spindle, Rapid to R (BORE,DRAG)
92	DMBITS	Feed Spindle to Z, Stop Spindle (BORE, MANUAL)
93	DMBITS	Feed to Z, Dwell, Stop Spindle, Rapid to R (BORE, DWELL,DRAG)
94	DMBITS	Feed Spindle to Z, Dwell, Feed to R (BORE,DWELL)

5.6.1 TABLE OF PREPARATORY FUNCTIONS (TABLEG) (cont'd)

Preparatory Functions

Location	Standard Value	Function
95	DMBITS	Feed Spindle to Z, Dwell, Stop Spindle (BORE, DWELL, MANUAL)
96	DMBITS	Absolute data input for contouring machine
97	DMBITS	Incremental data input for contouring machine
98	DMBITS	
99	DMBITS	
100	DMBITS	Extended Departure Threading For 100S Control, Lead Max = 1.99998 inches Departure Max = 19.9999 inches
101	DMBITS	Extended Departure Threading For 100S Control, Lead Max = 0.99999 inches Departure Max = 19.9999 inches
102	DMBITS	Extended Departure Threading For 100S Control, Lead Max = 0.099999 inches Departure Max = 19.9999 inches
103	DMBITS	Threading For 100S Control Lead Max = 1.99998 inches (59.998 MM) Departure Max = 9.9999 inches (999.99 MM)
104	DMBITS	Threading For 100S Control Lead Max = 0.99999 inches (29.999 MM) Departure Max = 9.9999 inches (999.99 MM)
105	DMBITS	Threading for 100S Control Lead Max = 0.099999 inches (2.9999 MM) Departure Max = 9.9999 inches (999.99 MM)
106	DMBITS	Threading with no lead. Departure max = 99.9999 inches
107	DMBITS	Threading with no lead. Departure max = 99.9999 inches

5.6.1 TABLE OF PREPARATORY FUNCTIONS (TABLEG) (cont'd)

108	DMBITS
109	DMBITS
110	DMBITS
111	DMBITS
112	DMBITS
113	DMBITS
114	DMBITS
115	DMBITS
116	DMBITS
117	DMBITS
118	DMBITS
119	DMBITS
120	DMBITS

5.6.2 TABLE OF MISCELLANEOUS FUNCTIONS (TABLEM)

The utilization of TABLEM is exactly analogous to that of TABLEG. TABLEM is a table consisting of 200 locations, and it is an ordered listing of miscellaneous functions and their corresponding M codes. The miscellaneous functions considered by the GECENT postprocessor are those which are listed in the TABLEM chart; the miscellaneous functions used by the Standard Machine are those which are indicated by their M code number assignments. DMBITS indicate that the Standard Machine does not consider the miscellaneous function.

The TABLEM location number has no relationship to the M code assigned to the function described at that location. For example, program stop which has the industry-accepted code m00 is always assigned to TABLEM location 1. Even though program stop were to be called by m24, it would have to be assigned to TABLEM location 1 in order to be output whenever the function is called.

NC machines which have multiple feed ranges (not counting rapid traverse) must use TABLEM(113) through (119) for the miscellaneous code assignments which specify the operational feed range. But TABLEM(42) must be assigned the rapid traverse M code, and TABLEM(43) must be assigned the feed range M code to be assumed when no feed range is initially specified by the part programmer. This M code will be one of the M codes assigned in TABLEM(113) through (119).

In some instances a location of TABLEM is assigned to more than one function, e.g., location 85. This is done for economy and only when the function cannot possibly conflict with one of the other assigned functions.

Unless otherwise specified, all assignments are for Head 1.

5.6.2 TABLE OF MISCELLANEOUS FUNCTIONS (TABLEM) (cont'd)

TABLEM(200)

Location	Standard Value	Miscellaneous Function
1	00	Program Stop
2	01	Optional Stop
3	02	End of Program
4	03	Spindle CLW
5	04	Spindle CCLW
6	05	Spindle Off
7	DMBITS	Tool Change
8	DMBITS	Turret Index
9	08	Coolant On
10	09	Coolant Off
11	DMBITS	Saddle Collet Closed, CAXIS
12	DMBITS	Saddle Collet Open, CAXIS
13	DMBITS	Pallet Select Change
14	DMBITS	Switch Reader
15	DMBITS	Thread Forward
16	DMBITS	Thread Reverse
17	DMBITS	Thread Off
18	DMBITS	Dwell (Preset Time)
19	DMBITS	Saddle Turret Coolant On
20	DMBITS	Saddle Turret Coolant Off
21	DMBITS	Punch Off

5.6.2 TABLE OF MISCELLANEOUS FUNCTIONS (TABLEM) (cont'd)

TABLEM(200)

Location	Standard Value	Miscellaneous Function
22	DMBITS	Clamp ON, XAXIS, Table, or Part
23	DMBITS	Unclamp XAXIS, Table, or Part
24	DMBITS	Rapid Traverse on Carriage
25	DMBITS	Rapid Traverse on Saddle
26	DMBITS	All Motions Rapid Traverse
27	DMBITS	All Motions Feed Rate
28	DMBITS	Dress (Welder or Grinder)
29	DMBITS	Front Turret Active
30	DMBITS	Rear Turret Active
31	DM30	End of Tape (Tape Rewind or Transfer)
32	DMBITS	Tool Pickup (Load)
33	DMBITS	Tool Pickup (Unload)
34	DMBITS	Cycle Tap
35	DMBITS	Lead Tap (LEDTAP)
36	DMBITS	
37	DMBITS	Safety Turn
38	DMBITS	Safety Face
39	DMBITS	Safety Bore
40	DMBITS	Safety Off
41	DMBITS	
42	DMBITS	X-Axis Gear Engage Rapid

5.6.2 TABLE OF MISCELLANEOUS FUNCTIONS (TABLEM) (cont'd)

(TABLEM(200)

Location	Standard Value	Miscellaneous Function
43	DMBITS	X-Axis Gear Engage Feed
44	DMBITS	Y-Axis Gear Engage Rapid
45	DMBITS	Y-Axis Gear Engage Feed
46	DMBITS	Z-Axis Gear Engage Rapid
47	DMBITS	Z-Axis Gear Engage Feed

(Note: Use the X-axis locations TABLEM 42 and 43 when only one M code shifts all axes.)

Location	Standard Value	Miscellaneous Function
48	DMBITS	Rotate No. 1 Indexer or Tilt No. 1 Weld
49	DMBITS	Rotate No. 2 Indexer or Tilt No. 2 Weld
50	DMBITS	Rotate Table or Indexer - Off
51	DMBITS	Encoder Engage
52	DMBITS	Encoder Disengage
53	DMBITS	Clamp on YAXIS or Head
54	DMBITS	Unclamp YAXIS or Head
55	DMBITS	Clamp Rail or TUL
56	DMBITS	Unclamp Rail or TUL
57	DMBITS	Engage Bar
58	DMBITS	Disengage Bar
59	DMBITS	
60	DMBITS	Pen Up
61	DMBITS	Pen Down

5.6.2 TABLE OF MISCELLANEOUS FUNCTIONS (TABLEM) (cont'd)

(TABLEM(200)

Location	Standard Value	Miscellaneous Function
62	DMBITS	Dash On
63	DMBITS	Dash Off
64	DMBITS	No. 2 Head (Vertical) Selected for Movement
65	DMBITS	No. 3 Head (Horizontal) Selected for Movement
66	DMBITS	Spindle Speed Change (Including Spindle Start) to Head 1
67	DMBITS	Spindle Speed Change (Including Spindle Start) to Head 2
68	DMBITS	Spindle Orient
69	DMBITS	Spindle Lock
70	DMBITS	Spindle Neutral - See Option 99
71	DMBITS	Spindle Speed Range Shift Range 1 CCLW
72	DMBITS	Spindle Speed Range Shift Range 1 CLW
73	DMBITS	Spindle Speed Range Shift Range 2 CCLW
74	DMBITS	Spindle Speed Range Shift Range 2 CLW
75	DMBITS	Spindle Speed Range Shift Range 3 CCLW
76	DMBITS	Spindle Speed Range Shift Range 4 CCLW
77	DMBITS	Spindle Speed Range Shift Range 4 CCLW
78	DMBITS	Spindle Speed Range Shift Range 4 CLW
79	DMBITS	Spindle Speed Range Shift Range 5 CCLW
80	DMBITS	Spindle Speed Range Shift Range 5 CLW

5.6.2 TABLE OF MISCELLANEOUS FUNCTIONS (TABLEM) (cont'd)

TABLEM(200)

Loaction	Standard Value	Miscellaneous Function
81	DMBITS	Spindle Speed Range Shift Range 6 CCLW
82	DMBITS	Spindle Speed Range Shift Range 6 CLW
83	DMBITS	Rotate Table, Turret, Magazine, CLW or Weld Rock No. 1
84	DMBITS	Rotate Table, Turret, Magazine, CCLW or Weld Rock No. 2

(Note: Locations 83-84 assumes CLW numbering, looking at magazine from top down.)

Loaction	Standard Value	Miscellaneous Function
85	DMBITS	Preheat No. 1 on Master Head No. 1, or Weld Schedule No. 1, BEAM ON
86	DMBITS	Preheat No. 2 on Slave Head No. 2, or Weld Schedule No. 2
87	DMBITS	Preheat No. 1 Off, or Weld Off, BEAM ON
88	DMBITS	Preheat No. 2 Off, or Weld Shift No. 1 BEAM 1
89	DMBITS	Oxygen No. 1 On, or Weld Shift No. 2 BEAM 2
90	DMBITS	Oxygen No. 2 On, or Weld Shift No. 3 BEAM 3
91	DMBITS	Oxygen No. 1 Off, or Weld Shift No. 4 BEAM 4
92	DMBITS	Oxygen No. 2 Off, BEAM 0
93	DMBITS	Height Control No. 1 Down, or Weld Head No. 1 Down
94	DMBITS	Height Control No. 2 Down
95	DMBITS	Height Control No. 1 Up, or Weld Head No. 1 Up

5.6.2 TABLE OF MISCELLANEOUS FUNCTIONS (TABLEM) (cont'd)

TABLEM(200)

Location	Standard Value	Miscellaneous Function
96	DMBITS	Height Control No. 2 Up
97	DMBITS	Offset tool on
98	DMBITS	
99	DMBITS	
100	DMBITS	
101	DMBITS	Coolant, Tap - On
102	DMBITS	Coolant, Mist - On
103	DMBITS	Coolant, Flood - On
104	DMBITS	Coolant, Tap - Off
105	DMBITS	Coolant, Mist - Off
106	DMBITS	Coolant, Flood - Off
107	DMBITS	Coolant, Front Turret - On
108	DMBITS	Coolant, Front Turret - Off
109	DMBITS	Coolant, Rear Turret - On
110	DMBITS	Coolant, Rear Turret - Off
111	DMBITS	Feedrate Override - On
112	DMBITS	Feedrate Override - Off
113	DMBITS	Feed Range 1
114	DMBITS	Feed Range 2
115	DMBITS	Feed Range 3
116	DMBITS	Feed Range 4

5.6.2 TABLE OF MISCELLANEOUS FUNCTIONS (TABLEM) (cont'd)

TABLEM(200)

Location	Standard Value	Miscellaneous Function
117	DMBITS	Feed Range 5
118	DMBITS	Feed Range 6
119	DMBITS	Feed Range 7

(Note: Locations 113-119 assign according to ascending order of ranges. Do not include RAPID TRAVERSE.

Location	Standard Value	Miscellaneous Function
120	DMBITS	Inhibit Creep - On
121	DMBITS	Inhibit Creep - Off
122	DMBITS	Angle Turn No. 1
123	DMBITS	Air - On, circular deflection on
124	DMBITS	Air - Off, circular deflection off
125	DMBITS	Head 1 - On
126	DMBITS	Head 2 - On
127	DMBITS	Head 3 - On
128	DMBITS	Head 1 and 2 - On
129	DMBITS	Angle Turn No. 2
130	DMBITS	Do Not Use This Position; must be DMBITS
131	DMBITS	Collet No. 4 (W-Axis) - Close
132	DMBITS	Collet No. 4 (W-Axis) - Open
133	DMBITS	All Clamps - Close
134	DMBITS	All Clamps - Open
135	DMBITS	
136	DMBITS	

5.6.2 TABLE OF MISCELLANEOUS FUNCTIONS (TABLEM) (cont'd)

TABLEM(200)

Location	Standard Value	Miscellaneous Function
137	DMBITS	
138	DMBITS	
139	DMBITS	
140	DMBITS	
141	DMBITS	
142	DMBITS	X Axis Gear Engage, Rapid, Head 2
143	DMBITS	X Axis Gear Engage, Feed, Head 2
144	DMBITS	Y Axis Gear Engage, Rapid, Head 2
145	DMBITS	Y Axis Gear Engage, Feed, Head 2
146	DMBITS	Z Axis Gear Engage, Rapid, Head 2
147	DMBITS	Z Axis Gear Engage, Feed, Head 2
148	DMBITS	X Axis Gear Engage, Rapid, Head 3
149	DMBITS	X Axis Gear Engage, Feed, Head 3
150	DMBITS	Y Axis Gear Engage, Rapid, Head 3
151	DMBITS	Y Axis Gear Engage, Feed, Head 3
152	DMBITS	Z Axis Gear Engage, Rapid, Head 3
153	DMBITS	Z Axis Gear Engage, Feed, Head 3
154	DMBITS	Automatic SFM, On, Head 1
155	DMBITS	Automatic SFM, Off, Head 1
156	DMBITS	Automatic SFM, On, Head 2
157	DMBITS	Automatic SFM, Off, Head 2

5.6.2 TABLE OF MISCELLANEOUS FUNCTIONS (TABLEM) (cont'd)

TABLEM(200)

Location	Standard Value	Miscellaneous Function
158	DMBITS	Automatic SFM, On, Head 3
159	DMBITS	Automatic SFM, Off, Head 3
160	DMBITS	X Axis Feed Range 1
161	DMBITS	X Axis Feed Range 2
162	DMBITS	Y Axis Feed Range 1
163	DMBITS	Y Axis Feed Range 2
164	DMBITS	Z Axis Feed Range 1
165	DMBITS	Z Axis Feed Range 2
166	DMBITS	Head 2 Turret CLW
167	DMBITS	Head 2 Turret CCLW
168	DMBITS	Rapid X axis
169	DMBITS	Feed X axis
170	DMBITS	
171	DMBITS	
172	DMBITS	Rapid Z axis
173	DMBITS	Feed Z axis

(Note: Locations 168,169, 172 and 173 are used on machines which can move repidly on one axis and feed on the other simultaneously)

Location	Standard Value	Miscellaneous Function
174	DMBITS	
175	DMBITS	
176	DMBITS	
177	DMBITS	

5.6.2 TABLE OF MISCELLANEOUS FUNCTIONS (TABLEM) (cont'd)

TABLEM(200)

Location	Standard Value	Miscellaneous Function
178	DMBITS	
179	DMBITS	
180	DMBITS	
181	DMBITS	
182	DMBITS	
183	DMBITS	
184	DMBITS	
185	DMBITS	
186	DMBITS	
187	DMBITS	
188	DMBITS	
189	DMBITS	
190	DMBITS	
191	DMBITS	
192	DMBITS	
193	DMBITS	
194	DMBITS	
195	DMBITS	
196	DMBITS	
197	DMBITS	
198	DMBITS	
199	DMBITS	
200	DMBITS	

5.6.3 TABLE OF REGISTERS (REGSTR)

The REGSTR table is a table of 30 locations, the first 18 of which are an ordered listing of the identifiers of each standard output register. If output of the postprocessor is Word Address, then each value of REGSTR is the related BCD character which identifies the register; the registers of each block are normally in the order given by REGSTR (see the reference below to the IORDER vector). The normal order and standard register assignments of REGSTR are:

REGSTR

Dimension(30)

Location	Standard Value	Function
1	N	Sequence counter
2	G	Preparatory function
3	X	Linear primary axis, abscissa
4	Y	Linear primary axis, ordinate
5	Z	Linear primary axis, axis of tool
6	A	Rotary axis, head
7	B	Rotary axis, table
8	I	Direction cosine or arc center offset related to abscissa
9	J	Direction cosine or arc center offset related to ordinate
10	K	Direction cosine or arc center offset related to the third primary axis
11	F	Feedrate command
12	S	Spindle speed command
13	T	Tool number
14	M	Miscellaneous function

5.6.3 TABLE OF REGISTERS (REGSTR) (cont'd)

REGSTR

Dimension(30)

Location	Standard Value	Function
15	DO NOT USE	CODE
16	R	Rapid traverse transition point
17	DBLNKS	(Open)
18	C	Third rotary axis

Locations 21 through 30 are available for extra registers. Locations 19 and 20 cannot be used since the postprocessor uses DBFSEG(19) and (20) to store the feedrate in IPM and the spindle speed in RPM.

Registers which are not used can be set to DBLNKS. For example, if the A and B registers are not used, REGSTR(6) = DBLNKS and REGSTR(7) = DBLNKS; however, the related values of the REGFOR table must be set to zero, e.g., REGFOR(6) = 0 and REGFOR(7) = 0.

Any of the assigned registers can be given another letter address, e.g., REGSTR(3) could be called W. The only requirement is that the table location function be observed. Thus, REGSTR(3) could legitimately be called W if the W axis is the linear primary axis.

The locations of REGSTR (and REGFOR) from 21 through 30 can be used for extra registers, if needed.

5.6.4 TABLE OF REGISTER FORMATS (REGFOR)

Different NC machines require different output format specifications. Some NC machines will have an S register, others will not; some machines may require 5 digits for its register, while others may need 6 or more digits. It is the function of the REGFOR table to specify these variations.

The REGFOR table is an ordered table of 30 locations, the first 18 of which are standard settings. Each location represents a machine register, and, the number assigned to each location specifies the required punch format for values of that register. The assigned numbers are two-digit floating point integers; the floating point integer value indicates the number of digits that are required for the register. The first digit of the floating point integer represents the number of digits that are to the left of the decimal point, and the second digit, the number of digits that are to the right of the decimal point. If the floating point integer has a minus value, then the sign of the value also will be punched. Thus, the value "-14.", as found at location 3 in the REGFOR table, means that the third register (normally REGSTR(3) = X) must have 5 digits, one digit to the left of the decimal point, and 4 digits to the right; and, also that the signs must be punched. Similarly, the integer 20 as found at location 2, means two digits to the left and none to the right of the decimal point, and that there whould be no sign.

Certain control systems which have only linear interpolation may have what is called a "floating" or variable format, i.e., a five-digit register with a 14 format in G01 mode, but a 23 format when G10 is used. (See Section 4.8 for a more complete description). For controls of this type, REGFOR is set equal to 14, not 23, but option 4 (maximum departure) is set equal to 99.999. The variable format is taken care of by setting option 41 equal to zero.

The values assigned to the REGFOR table are those used by the Standard Machine. NC machines which require different punch formats need only to substitute their required values into the REGFOR table.

If a machine does not have a particular register, it is essential that a zero be assigned to the REGFOR table location of that register. For example, if the NC machine does not have a spindle speed register, location 12 of the REGFOR table must be specified as a zero.

5.6.4 TABLE OF REGISTER FORMATS (REGFOR) (cont'd)

REGFOR

Dimension(30)

Table Location	Register	Standard Value
1	N	30. (NDP)
2	G	20. (NDP)
3	X	-14.
4	Y	-14.
5	Z	-14.
6	A	0.
7	B	0.
8	I	14.
9	J	14.
10	K	14.
11	F	30.
12	S	30. (NDP)
13	T	20. (NDP)
14	M	20. (NDP)
15	DO NOT USE	---
16	R	0.
17	(Open)	0. (NDP)
18	C	0.

Each REGFOR value is determined by 10L + R,

where L = number of digits to left of decimal point;

R = number of digits to right.

5.6.4 TABLE OF REGISTER FORMATS (REGFOR) (cont'd)

The REGFOR value must be minus if the sign is to be punched.

The required number of digits for the register is L + R. (NDP) indicates that no decimal point is used in the printout for that table location. Thus, REGFOR(1) has integer output only, whereas REGFOR(3) or (16) has decimal output.

Locations 21 through 30 are available for extra registers. Locations 19 and 20 cannot be used since the postprocessor uses DBFSEG(19) and (20) to store the feedrate in IPM and the spindle speed in RPM, respectively.

It should be pointed out that it is not absolutely essential to set the related location of REGSTR to DBLNKS for an unavailable register as long as the related REGFOR location is set to zero. In other words to nullify a register, it is mandatory that the location of REGFOR be set to zero, but not necessarily that REGSTR be set to DBLNKS.

The following two examples illustrate typical possible setups for the REGSTR and REGFOR tables.

5.6.4 TABLE OF REGISTER FORMATS (REGFOR) (cont'd)

Example 1: Registers NGXYZFSTMR are available.

	REGSTR	REGFOR
1	N	30.0
2	G	20.0
3	X	-24.0
4	Y	-24.0
5	Z	-24.0
6	DBLNKS	0.
7	DBLNKS	0.
8	DBLNKS	0.
9	DBLNKS	0.
10	DBLNKS	0.
11	F	21.0
12	S	20.0
13	T	10.0
14	M	20.0
15	(IGNORE)	(IGNORE)
16	R	-22.0
17	DBLNKS	0.
18	DBLNKS	0.

5.6.4 TABLE OF REGISTER FORMATS (REGFOR) (cont'd)

Example 2: Registers NGXZIKFSTMWH are available.

	REGSTR	REGFOR
1	N	30.0
2	G	20.0
3	X	-24.0
4	DBLNKS	0.
5	Z	-24.0
6	DBLNKS	0.
7	DBLNKS	0.
8	I	24.0
9	DBLNKS	0.
10	K	24.0
11	F	31.0
12	S	20.0
13	T	10.0
14	M	20.0
15	(IGNORE)	(IGNORE)
16	W	-14.0
17	H	10.0
18	DBLNKS	0.

5.6.4 TABLE OF REGISTER FORMATS (REGFOR) (cont'd)

Another (and better) way of setting up the tables in Example 2 is to take advantage of the Standard Machine settings. The programmed tables appear then as:

```
REGFOR(3)  = -24.0

REGFOR(4)  =   0.

REGFOR(5)  =  24.0

REGFOR(8)  =  24.0

REGFOR(9)  =   0.

REGFOR(10) =  24.0

REGFOR(11) =  31.0

REGFOR(12) =  20.0

REGFOR(16) = -14.0

REGFOR(17) =  10.0


REGSTR(16) = "W"

REGSTR(17) = "H"
```

5.6.5 TABLE OF OPTIONS (OPTAB)

Machine characteristics which are not handled by the TABLEG, TABLEM, REGSTR, and REGFOR tables, are handled by the OPTAB Table. This table is essentially a further definition of machine characteristics and limitations dictated by a specific hardware configuration. Some of the items in the table can be chosen at the discretion of the part programmer or computer programmer to suit in-house programming practices.

OPTAB considers such items as axis assignments, feedrate and spindle speed determination, positioning vs contouring, departure segmentation, interpolation modes acceleration-deceleration (A/D) machine servo constants, output codes, range specifications, machine limitations, tolerances, assumptions, and so on. A detailed listing of the options and an explanation of each is given below.

Each item in the OPTAB table has a standard value, i.e., the value assumed by the Standard Machine. The method of modifying OPTAB to represent a given NC machine is essentially the same as the method used for the other tables; i.e., simply assign the value to each option that pertains to the characteristics of the given machine. Of course, if the value is already assumed for the Standard Machine, there is no need to reassign it. With OPTAB, <u>never</u> assign any option as DMBITS. Each option in the table must have a value to indicate the choice of the option, and the value is always a floating point number or somtimes a BCD word.

It is possible to change the value of most options in the part program through the MACHIN statement; see the Part Programmer Manual. For example, the statement

```
MACHIN/GECENT, 40, OPTAB, 4,20
```

tells the postprocessor to set option 4 to the value of 20. Such option changes normally can be given legitimately at any point in the part program, but some care must be taken as to which options are changed. Several options cannot be changed because their values are are given in BCD form, and the MACHIN statement values following the OPTAB modifier are floating point numbers.

5.6.5 TABLE OF OPTIONS (OPTAB) (cont'd)

There also are other option values which should not be changed because the postprocessor assumes certain conditions which are related to the option, hence, if the option value changes and the required other modifications are not made, the postprocessor works with insufficient and incorrect data. For example, if option 10 is changed to +1, calling for the "Magic 3" EIA F code, the postprocessor will produce erroneous output unless REFGOR(11) = 30.

For these and similar reasons, the following options cannot be changed through a part program MACHIN statement.

Non Re-assignable Options

1, 7, 8, 10, 13, 14, 19, 22, 33, 34, 35, 69, 111, 112, 113, 114, 115, 118, 119, 138, 165, 174, 191.

Restricte Options

Options 149 and 150, if changed through a MACHIN statement, must be given in Output Units.

When setting up a Machine Subroutine, it is helpful to know from what source to expect information concerning a given option. The following list groups the options according to one of three sources: NC Machine Manufacturer, User Customer, and the General Electric Numerical Equipment Control Department (NECD). Information concerning an option should be obtained from the referenced source since that source is considered the normal one.

NC Machine Manufacturer

1, 3, 4, 6, 7, 8, 9, 10, 12, 13, 14, 16, 18, 19, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 37, 38, 39, 40, 41, 42,43,44, 45, 46, 47, 48, 49, 50, 51, 53, 54, 56, 57, 59, 60, 61, 62, 63, 69, 70, 71, 75, 76, 77, 78, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 81, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127-150, 151-157, 170.

174, 175, 176, 177, 178, 179, 180, 182, 183, 184, 185, 186, 187, 188, 189, 190, 192, 193, 194, 195, 196, 197, 199, 200, 201, 202

Customer User

2, 5, 11, 15, 17, 20, 21, 23, 29, 32, 33, 35, 55, 64, 65, 66, 67, 68, 79, 90, 109, 110, 151, 152, 153, 160, 164, 170, 172, 181, 191, 198,

GE/NECD

69, 70, 71, 72, 73, 73.

OPTION TABLE FOR GECENT III POSTPROCESSOR

Notes on the use of the option table.

1. When setting up a subroutine for a metric system,wherever the word "inch" is used, substitute millimeter. If a value is given in inches, substitute the metric value in millimeters that is to be used in its place.

2. Reference to sections which appear in this table are to the Computer Programmer's Manual unless specifically stated otherwise.

3. A letter may follow the option number indicating that it is applicable only to that type of control system and may be ignored in supplying information for other types of controls. Where no letter follows the option number, that option may have universal application independent of control type. The symbols and there meaning are as follows:

P	- Positioning controls only
C	- Positioning/contouring and contouring controls
MA	- Multiaxis contouring control system
MH	- Multihead control systems

OPTION 1

Type of machine tool:

0 = contouring
1 = positioning
-1 = positioning machine, S codes not merged with T codes
2 = positioning, S code and T code are merged in the following motion block. The part programmer must give the SPINDL and TOOLNO immediately prior to the motion block.
3 = machine has both positioning (absolute) and contouring (incremental) capability (Mark Century 7500 Series)

Standard = 0.0

OPTION TABLE FOR GECENT III POSTPROCESSOR

OPTION 2 C

The postprocessor should be removed from an SFM mode by:

0 = spindle only
1 = FEDRAT/n, IPM, or SPINDL

Standard = 0.0

Zero means the postprocessor will discontinue operating in an SFM mode when a SPINDL statement is given. 1 means the SFM mode operation will discontinue for a SPINDL, or FEDRAT/IPM mode statement. (See Section 4.4)

OPTION 3 C

Maximum percent change of velocity allowed in the acceleration/-deceleration sequence.

Standard = 100.0 percent

This causes the acceleration/deceleration sequence to restrict velocity changes to less than or equal to option value. Used only when option 55 set = 1.0. Not currently tested because option 55 must be set to zero. (See Section 4.3)

OPTION 4 C

Maximum departure in inches for linear moves. See also option 41 for variable format considerations. (See Section 3.4.3.2)

Standard = 9.9999 inches

OPTION TABLE FOR GECENT III POSTPROCESSOR

OPTION 5

Tolerance value in inches to use for a conditional test.

Standard = 0.00005 inches

There are numerous tests in the postprocessor which test one value against another for equality. This option gives an allowable tolerance which specifies how close to absolute equality is required.

Example: A-B=0, + or - tolerance

This tolerance is normally set to one half the minimum step size.

OPTION 6

(Not currently tested.)

OPTION 7

Number of spindle speed ranges.

Standard = 1.0

Spindle considerations are dicussed in detail in Section 4.5.

OPTION 8

Number of spindle speeds in each spindle range.

Standard = 2.0

Each range must have the same number of spindle speeds.

OPTION 9 C

Is circular interpolation available? (See Section 2.4.2.2.2)

1 = YES
0 = NO

Standard = 1.0

OPTION TABLE FOR GECENT III POSTPROCESSOR

OPTION 10 C

Feedrate in IPM is converted to one of the following command formats.

```
   0 = (feedrate in IPM * DIMULT)/L (contouring machine only)
  +1 = a 3 digit EIA number.  REGFOR(11) = 30.0.
  -1 = 1/T, a floating format for F is used.
```

For example:

```
   FCOMAX = 999.9    for a G11 REGFOR(11) = 31.0
   FCOMAX = 099.99   for a G01 REGFOR(11) = 32.0
   FCOMAX = 009.999  for a G10 REGFOR(11) = 33.0

   Set REGFOR(11) = 33.0

  -2 = 1/T.  A floating format for F is used.  For example:

   FCOMAX = 999.     for a G11 REGFOR(11) = 30.0
   FCOMAX = 099.9    for a G01 REGFOR(11) = 31.0
   FCOMAX = 009.99   for a G10 REGFOR(11) = 32.0

   Set REGFOR(11) = 32.0

  -3 = 1/T.  A floating format for F is used.  For example:

   FCOMAX = 999      for a G12 REGFOR(11) = 40.0
   FCOMAX = 0999.9   for a G11 REGFOR(11) = 41.0
   FCOMAX = 0099.99  for a G01 REGFOR(11) = 42.0
   FCOMAX = 0009.999 for a G10 REGFOR(11)= 43.0

   Set REGFOR(11) = 43.0

  -4 = 1/T (7500 series) Variable F format.  For example:

   FCOMAX = 9999.99        for G12 format(11) = 42.0
   FCOMAX = 0999.999       for G11 format(11) = 43.0
   FCOMAX = 0099.9999      for G01 format(11) = 44.0
   FCOMAX = 0009.99999     for G10 format(11) = 45.0
   FCOMAX = 0000.999999    for G23 format(11) = 46.0
   FCOMAX = 0000.0999999   for G26 format(11) = 47.0

   Set REGFOR(11) = 47.0.
```

Standard = 0.0

See Section 4.1 for further information.

OPTION TABLE FOR GECENT III POSTPROCESSOR

OPTION 11

For filler or separator cards, punch out:

+1 = parity punched cards
0 = blank cards
-1 = no filler cards

Standard = +1.0

The filler card, if requested, is produced after an END statement. Parity punched cards produce a punch in channel 5 of the paper tape. The space codes are specified in option 64.

OPTION 12

This option is not currently tested.

OPTION 13

Minimum read time in minutes for the machine tool tape reader to read a maximum length command block, based on 62.5 characters/second.

Standard = 0.012 minutes

The read time relation is: C(MAX)/(60*TRS). C-MAX standard is 45.0. C(MAX) = maximum number of characters possible in a programmed block. TRS = tape reader speed in characters per second. Use the following table as a guide for TRS vs. time in minutes.

TRS		MIN
100	=	0.0075
300	=	0.0025
400	=	0.001875
500	=	0.0015
1200	=	0.000625

Not applicable to non-buffered units, as the Mark Century 100S, NPC controls, and all Mark Century 7540 series controls. For these use 0.001.

OPTION TABLE FOR GECENT III POSTPROCESSOR

OPTION 14

Step size in inches.

Standard = 0.0001 inches

Step size is the increment produced by one pulse of the servo control.

OPTION 15

If continously variable spindle speeds are available, (types 1, 10 and 18 spindles), what percent variation of the spindle speed should be used when in an SFM mode? (See Section 4.4 for SFM details.)

Standard = 0.10 (10 percent)

OPTION 16

System catch-up time before a gear shift into or out of rapid traverse. The catch-up time is issued in a dwell block by itself prior to the block containing the shift code to the new condition. If this option is set to zero, no dwell block is issued. Non-buffered controls and positioning controls should have this option set to zero.

Standard = 0.5 seconds

OPTION 17

For GEOUT3 (option 164=3) and GEOUT4 (option 164=4) what printouts are wanted? (See Section 3.5.)

100	=	incremental
10	=	absolute
1	=	operators
0	=	no printout

Standard = 111.0

200, 20, or 2 gives the first and last points of the cut vector, and the intermediate blocks are deleted when the postprocessor segments a departure.

OPTION TABLE FOR GECENT III POSTPROCESSOR

OPTION 18

Number of feedrate ranges, if more than 1 range is required.

0 = one range
n = number of ranges if greater than one.

Standard = 0.0 (means 1 range)

Check option 91. Also, do not count rapid traverse range as a feedrate range for this option. Maximum and minimum feedrate values per range are stored in table FRTAB. A maximum of seven ranges can be used. See Section 4.1.2.

OPTION 19

Spindle type for this machine.

Standard = 1.0

Spindle types are discussed in Section 4.5. If option(19) = 7.0, then set option(133) = 1.0

OPTION 20

The following output types are available.

0 = Hollerith BCD characters (Punch B)
1 = Paper tape image (Punch A)
-1 = Gapped magnetic tape
-2 = Gapless magnetic tape

Standard = 0.0

Zero will punch out the characters as Fortran hollerith codes. The paper tape image, +1, will punch cards in the image of the programmed paper tape. Each card is 72 columns. See options 52 and 64 thru 68. Also see Section 3.5.1.

OPTION TABLE FOR GECENT III POSTPROCESSOR

OPTION 21 C

Safety should be:

0 = one-shot
1 = modal

Standard = 0.0

Zero applys only to first motion statement after a safety statement. Used for lathes only.

OPTION 22

The machine axes are,

100 = x
10 = y
1 = z

Add the above values to represent the machine coordinate system.

Standard = 111.0

11 = YZ
101 = ZX
111 = XYZ

The axis nomenclature of the machine is defined by its register needs. Example: lathes normally use the ZX reference frame, and the ZX registers. Therefore, option 22 = 101. This option applies only to the three primary linear axes; multihead and multiaxis machines and those with secondary and tertiary axes are handled in a different manner.

OPTION 23

Assumed feedrate in inches per minute, if not specified. Do not set this option equal to zero as postprocessing will cease.

Standard = 5.0 IPM

OPTION TABLE FOR GECENT III POSTPROCESSOR

OPTION 24

Feedrate command maximum. (excluding rapid traverse)

Standard = 500.0

OPTION 25

Maximum feedrate in inches per minute. If each axis has its own separate maximum and minimum feedrate, store the negative value of the highest feedrate as the value of this option. Store the maximum and minimum values for each axis in FRTAB in the same order as given by option 59. Set option 174 equal to twice the number of linear axes.

Standard = 200.0 IPM

OPTION 26 C

Should a feedrate number multiplier constant be used when the calculated feedrate number exceeds the maximum permissible value (option 24.)? To use this option, the machine must have the I, J, and K, registers.

0 = YES
1 = NO

Standard = 0.0

When the feedrate command maximum is exceeded, the use of a multiplier constant permits a wider range of feedrates for short moves. This option can not be used if feedrate uses a 3 digit EIA command form. (Option 10 = 1) See Section 4.1.53., and 4.1.5.1, and option 170 for alternative considerations.

OPTION 27

Assumed spindle speed, if not specified by a spindle statement. Do not set equal to zero as postprocessing will cease.

Standard = 80.0 RPM

OPTION TABLE FOR GECENT III POSTPROCESSOR

OPTION 28 C

Assumed interpolation mode, if not given or previously specified.

0 = LINEAR - linear interpolation will be used for all paths.
1 = LINCIR - will use circular interpolation, if available, for paths that are circular in any of the standard reference planes, and constant in the other axis. All other paths will use linear. Mode of interpolation may be changed through the MACHIN/GECENT statement.

Standard = 1.0

OPTION 29

Should the postprocessor keep track of the rotary table position on a - ROTATE/TABLE,ALPHA - statement?

0 = NO
1 = YES

Standard = 1.0

Absolute systems should set this option to 1. Incremental systems may or may not. If the option = 0, the postprocessor completely ingores the position of the table. This precludes use of the ATANGL and the ROTREF modifiers. It is the responsibility of the part programmer to keep track of the table. If a ROTREF or an ATANGL modifier is called when the option = 0, the postprocessor prints a warning comment, and proceeds as if the option = 1. Option 29 must = 1 if linearity is to be used. 1 in effect gives an automatic ROTREF on all rotary moves. If option 29 = 0, the part programmer has the flexibility to program either with or without a ROTREF, but with no linearity testing.

OPTION 30

Is the optional feature OPSKIP (block delete) available? See Section 4.11.

0 = NO
1 = YES

Standard = 0.0

OPTION TABLE FOR GECENT III POSTPROCESSOR

OPTION 31

For type 2 spindles, what is the speed code increment between ranges?

Standard = 0.0

All ranges must have the same incremental variation between the speed code at the end of a range, and the code at the beginning of the next higher range. See Section 4.5.3.

OPTION 32

(Not currently tested)

Assumed tolerance in inches, if not given by the MCHTOL statement.

Standard = 0.0005 inches

Value given by MCHTOL is used as a dynamic tolerance in A/D testing. Used only when option 55 = 1.0.

OPTION 33

Assumed coolant mode, if none is given or specified.

1 = TAPKUL
2 = MIST
3 = FLOOD
4 = outputs M code stored at TABLEM(9)
5 = SADDLE
6 = FRONT
7 = REAR

Standard = 4.0

If only one coolant mode is available, option 33 = standard value.

OPTION TABLE FOR GECENT III POSTPROCESSOR

OPTION 34

Initially assumed spindle direction, if none is given or specified.

+1 = CLW
-1 = CCLW
 0 = error, and computer processing stops.

Standard = +1.0

OPTION 35

Assumed feedrate mode, if none is specified or given.

0 = IPM
1 = IPR

Standard = 0.0

OPTION 36 C

Should the rapid traverse vector be optimized?

0 = YES
N = NO

where N is the maximum vectorial feedrate in IPM on RAPID move. See Section 4.1.5.4.

Standard = 0.0

OPTION TABLE FOR GECENT III POSTPROCESSOR

OPTION 37 C

Minimum cutter path length for a rapid traverse motion.

Standard = 0.0 inches

For machines that require gear shifting to obtain rapid traverse, it may be advantageous to remain in the feed range and travel at the highest rate in that range. Machines that shift gears for a rapid traverse often require dwells before and after a rapid move. For such moves, the GECENT postprocessor allows an assigned value for the dwell, option 81, for shifting into the rapid traverse range. Similarly an assigned value of dwell, option 82, is used for shifting from rapid traverse back into feed range. To compute the minimum length of path for which it is faster to use the option values of dwell and the rapid traverse range, use the following relationships. Minimum path length equals the greater value of S1, S2, and S3.

$$S_1 = \left[\frac{O_m * O_{39}}{O_m - O_{39}} \right] \left[\frac{O_{81} + O_{82}}{60} \right]$$

$$S_2 = O_{39} O_{13}$$

$$S_3 = O_m \left[O_{13} - \left[\frac{O_{81} + O_{82}}{60} \right] \right.$$

where:

OM = minimum value of option 42, 43, and 44, but not equal to zero.
Option 13 = minimum tape reading time
Option 39 = maximum feedrate
Option 81 = dwell time - shifting to rapid traverse
Option 82 = dwell time - shifting back to feed range
S1 = minimum length of path for which it is better to shift into rapid traverse rather than use the maximum feedrate
S2 = minimum length of path in maximum feedrate before the path is limited by the tape reader
S3 = minimum length of path in rapid traverse before the path is limited by the tape reader

OPTION TABLE FOR GECENT III POSTPROCESSOR

OPTION 38

Must there be a G code in every programmed block?

0 = Redundant G codes are suppressed in punched and printed output.

+1 = No supression of redundant G codes. G code for dwell, TABLEG(5), is used for blocks which do not have a G code already assigned.

-1 = Current cycle G code is used instead of TABLEG(5). No suppression of redundant G codes.

+2 = Suppression of G codes in motion blocks only.

Standard = 0.0

OPTION 39 C

Maximum feedrate in inches per minute obtainable in the feed range, when rapid traverse is prohibited by option 37 limiations.

Standard = 50.0 IPM

This option can be ignored if options 37, 81, and 82 are zero, or if option 25 equals options 42, 43, and 44. Note: There is no range changing on multi-feed range machines.

OPTION 40 P

For positioning machines only should redundant X and Y values be suppressed?

0 = NO
1 = YES except those which appear with a T code

Standard = 1.0

OPTION TABLE FOR GECENT III POSTPROCESSOR

OPTION 41 C

Is there a variable format for X, Y, and Z which changes for a long departure TABLEG(11)?

1 = NO
0 = YES, but use of the long departure is restricted to rapid moves only.
-1 = YES, and applies to both RAPID and FEED, with format changing from 14(normal) to 23(long).
-2 = YES, and applies to both RAPID and FEED, with format changing from 14(normal) to 24(long).

Standard = 1.0

If 0, the postprocessor assumes that G10 blocks have a format of -23.0 and that normal and short blocks have a format of -14.0. Option 4 and TABLEG(11) must be set properly if this option = YES. Ignore this option if the machine has a 6 digit register, or is 7500 series control. See Section 4.8.

OPTION 42

X-axis maximum rapid traverse in inches per minute.

Standard = 200.0 IPM

OPTION 43

Y axis maximum rapid traverse in inches per minute.

Standard = 200.0 IPM

OPTION 44

Z axis maximum rapid traverse in inches per minute.

Standard = 200.0 IPM

OPTION TABLE FOR GECENT III POSTPROCESSOR

OPTION 45

Maximum rapid traverse command number.

Standard = 500.0

If this option is negative, do not output the rapid value of F.

OPTION 46

Are gear shifting M codes required for a rapid traverse?

0 = NO
+1 = YES, indicating that an M code is required for each axis when shifting into or out of rapid traverse.
-1 = YES, indicating that only one M code is required for all axes.

Standard = 0.0

+1 or -1 implies a programmed dwell is required for the shifting of gears. The times are specified in options 81 and 82.

OPTION 47

Numerical increment added to the spindle table row number in order to produce the correct spindle speed command for the minimum speed, first range for type 0, type 2, type 3, type 8 spindles.

Standard = 0.0

See Sections 4.5.1, 4.5.3, 4.5.4, and 4.5.9.

OPTION 48

Feedrate minimum in inches per minute.

Standard = 1.0 IPM

OPTION TABLE FOR GECENT III POSTPROCESSOR

OPTION 49

Feedrate command minimum. (Used also as command minimum for Z axis on positioning feedtype 5 only.)

Standard = 1.0

OPTION 50

(Not currently tested.)

OPTION 51

Should word addresses and values of the REGSTR table, (3 through 10 and 16) which for the standard cases are X,Y,Z,A,B,I,J,K be deleted from the punched output if their values are zero? This applies to punched output only.

0 = delete both the zero value and the word addresses
+1 = delete the zero but keep the word addresses
-1 = output both the word address and the zero value
-2 = delete leading zeros trailing zeros on all departure commands. This can be used only for 100M controls so connected, and for 7500 series controls with variable register length format.

Standard = 0.0

OPTION 52

Should the output cards be packed, (fill each card in columns 1-72) with as many programmed blocks as possible? This pertains only to tapes using Hollerith characters. Tape image always use packed cards.

0 = YES
1 = NO

Standard = 0.0

If option 52 = 0, option 11 must = 0, or +1. See option 20 also. Currently not available on the IBM 360 system.

OPTION TABLE FOR GECENT III POSTPROCESSOR

OPTION 53

(Not currently tested.)

OPTION 54

If the postprocessor is to insert a dwell for spindle speed range changes, what dwell time is used? Also used for a spindle locking dwell time.

Standard = 0.0 seconds

This option should not be used if the spindle must be stopped for changing gears. The part programmer must call for the stop. The spindle must then be restarted by the part programmer. Also, ignore this option if the spindle subroutine automatically inserts dwells for range changes. For example, a type 10 spindle. See related option 89 when a dwell is required for spindle reversal.

OPTION 55 C

Does the postprocessor test the cutter path segments for acceleration and deceleration conditions?

0 = NO
1 = YES

Standard = 0.0

If 1, the postprocessor tests all paths and makes any necessary corrections which allow the cutter to follow the programmed paths within tolerance. Feedrate is reduced when approaching a corner. See Section 4.3.3 for a detailed discussion of A/D. (Not currently tested, therefore the standard value must be used.)

OPTION TABLE FOR GECENT III POSTPROCESSOR

OPTION 56 C

Scale factor for modifying a dwell register so that dwell times are properly punched in the output tape.

Standard = 10.0

Dwell time is given in an axis register, usually the X or Z register. On most machines a dwell times causes a different interpetation of the magnitude of the number given in the axis register. For example, in a motion code block, X42 may equal an X axis motion of 4.2 inches, but in a dwell code block, X42 may mean a dwell of 42 seconds, and not 4.2 seconds. In this example, the correction necessary would be to set option 56 = 10, since dwell time is divided by the option value before it becomes output. For 7500 controls the dwell function has a fixed format of 24 whereas motion may have a format changeable from 14 to 44 by variable connection. Thus a control with a format of 44 must have option 56 set to 0.01 so that a 15 second dwell will be output as X1500 and not as X0015. See Section 4.10. Not appliciable if the feedrate is used for a dwell.

OPTION 57 C

Where should the dwell register be placed? The standard order as assigned by REGSTR is-

N=1	G=2	X=3	Y=4	Z=5	A=6	B=7
I=8	J=9	K=10	F=11	S=12	T=13	M=14

Locations 16-30 are also availalbe.

Standard = 3.0

OPTION 58

(Open for assignment)

OPTION TABLE FOR GECENT III POSTPROCESSOR

OPTION 59

APT operates internally with the axes order +X, +Y, +Z. Output is presented this way, too. How is the output presented for this machine?

Standard = 131415.0 Means(+X+Y+Z)

This option is best explained by an example. If the given machine were a lathe, we normally need output in terms of the X and Z axes. For 2 axes, APT works most efficiently with the axes ordered +X+Y, as indicated in figure 1, but lathe axes are ordered +Z-X, figure 2.

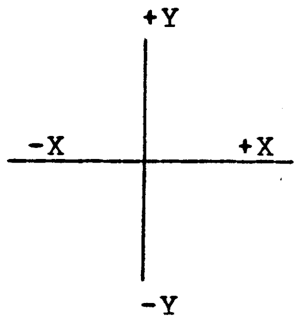


FIGURE 1

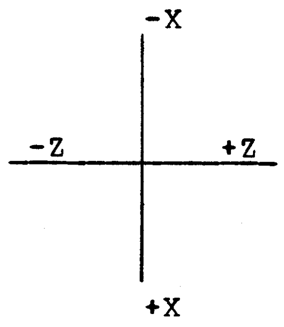


FIGURE 2

Since the output will be in terms of +X, +Y, it is necessary to refer the values to the +Z, -X, reference frame, because the lathe can only operate with these axes. Therefore, option 59 would be given as +Z-X, which means that in the normal output, the word address for +X values should be changed to +Z, the word address for +Y values should be -X. This option requires the use of signs. An unspecified axis is ignored . If the option is given as +Y+Z, no X values will appear in the output. This option must be given in the numerical form

MEANING	BLANK	+	-	X	Y	Z	I	J	K
CODED	0	1	2	3	4	5	6	7	8

See also options 42, 43, and 44.

OPTION TABLE FOR GECENT III POSTPROCESSOR

OPTION 60 C

How should the I J K registers be presented?

Standard = 161718.0 Means(+I+J+K)

This option is completely analogous to option 59. If the option were given as +J+K, then the output word addresses would be changed to J and K, and no I values would appear in the output. Ignore, if the I J K registers are not available on this machine.

OPTION 61

(Open for assignment)

OPTION 62 P

Number of ranges of discrete feedrates. This option applies only to positioning machines which use specific values of feedrate which are selected from a table of values. See Section 4.1.2.

Standard - 1.0

OPTION 63 P

Number of feedrates per range.

Standard = 1.0

All ranges must have the same number of speeds. Use for multiple, discrete values only, as for a table of IPR values.

OPTION 64

Space code. This code is used to produce the leader tape length.

Standard = ((((((Paper tape punch in channel 5.

Also see section 3.5.3.2.

OPTION TABLE FOR GECENT III POSTPROCESSOR

OPTION 65

End-of-record or stop code. Given by the TMARK statement.

Standard = = (equal sign) Paper tape punches in channels 1,2,4

OPTION 66

Tab code.

Standard = * Paper tape punches in channels 2,3,4,5,6. Not currently used.

OPTION 67

End-of-block code.

Standard = $ Paper tape punch in channel 8; magnetic tape uses channels 3,4,5.

OPTION 68

(Open for assignment)

OPTION 69 C

Servo settling time.

Standard = 0.12 seconds

NOTE...options 70-74 should be ignored if option 55 = 0.0

OPTION 70 C

Additional overshoot constant.

Standard = 1.3 (dimensionless)

OPTION TABLE FOR GECENT III POSTPROCESSOR

OPTION 71 C

Overshoot constant.

Standard = 0.0129 seconds

OPTION 72 C

Undershoot constant.

Standard = 0.00905 seconds

OPTION 73 C

Velocity error coefficient (servo gain constant).

Standard = 14.5 cycles per second

OPTION 74 C

Circle error constant.

Standard = 28.0 cycles per second

OPTION 75

Should an M code produced by the AUXFUN statement be made output in a block by itself, or be merged with the next output block?

0 = output a block by itself
1 = merge into the next block

Standard = 0.0

OPTION 76 C

Dwell time in seconds for closing or opening a clamp. Used in conjunction with TABLEM locations 22,23,53,54,55, and 56.

Standard = 0.0 seconds

OPTION TABLE FOR GECENT III POSTPROCESSOR

OPTION 77

In which block should the spindle-on M code be placed with respect to the block in which the spindle speed command appears?

0 = output the M code and S code in the same block
+1 = output the M code in a command block preceding the S code block
-1 = output the M code in a command block following the S code block

Standard = 0.0

OPTION 78 P

Type of feedrate command to be generated with positioning machines. Zero implies no feedrate type. A negative number specifies a type zero command also.

0 = feedrate command = ABS (OPTAB(78)) times feedrate in IPM
1 = feedrate command = feedrate in IPM
2 = function of current spindle speed and range, and a table of discrete IPR feedrate values (see option 144)
3 = feedrate is a 3 digit EIA number
4 = use 2 range table lookup
5 = feedrate table for Z coordinate and table for X and Y also see options 49, 201, 202.
6 = XY (milling feedrate command) = 12 * feedrate in IPM. Z feedrate command = 2 * feedrate in IPM.
7 = the F command is a code number (neither IPM nor IPR) representing manually set feedrate selections.
8 = use FTYPE2, only the feedrate command is an EIA number
9 = similar to FTYPE2, except the feedrate table is given in IPM

Standard = 0.0

Also see section 4.1.2.

OPTION TABLE FOR GECENT III POSTPROCESSOR

OPTION 79

Number of lines of output to be printed per page.

Standard = 45.0 lines

Do not count the 7 lines used for titles. Do not exceed 51 lines.

OPTION 80 MA

If a pallet is used, this height is used for clearance checking which varies as a function of the presence or absence of a pallet.

0 = no pallet
N = the height of the pallet or fixture in inches

Standard value = 0.0 inches

OPTION 81 C

Dwell time for shifting gears into a rapid traverse.

Standard 2.0 seconds

If no gear shifting is required for rapid traverse, ignore options 81 and 82.

OPTION 82 C

Dwell time for shifting gears into the feed range from rapid.

Standard = 1.0 second

OPTION 83 C

Dwell time for turret index or tool change.

Standard = 5.0 seconds

OPTION TABLE FOR GECENT III POSTPROCESSOR

OPTION 84 C

Assumed feedrate for the saddle, if none is specified.

Standard = 5.0 IPM

If the value is negative or zero, an error is assumed.

OPTION 85 C

Assumed feed mode on a saddle, if none is specified.

+1 = IPM
-1 = IPR
0 = error, and computer processing stops.

Standard = +1.0

OPTION 86 P

Automatic tool corrections for positioning machine tool changes. Used to compensate for tool length.

0 = none
+1 = make the tool correction with a LOAD/TOOL orTOOLNO statement
-1 = tool length is compensated in Z and R when the Z axis is inverted (See option 140).
+2 = tool length is subtracted from the Z and R values. This permits programming with the tool base instead of the tool tip
-2 = make the tool correction and insert tool offset (if any) on a ROTATE/TURET statement

Standard = 0.0

This option pertains only to absolute positioning machines. Incremental contouring machines always receive a corrective compensating motion, when needed, whereas positioning machines receive a correction only in the tool length which compensates for the length variation due to a tool change. See description of LOAD/TOOL and TOOLNO and inversion in the Part Programmer's Manual.

OPTION TABLE FOR GECENT III POSTPROCESSOR

OPTION 87 C

Dwell time for a saddle rapid traverse.

Standard = 0.0 seconds

The same dwell time is used in going back to feed.

OPTION 88

Number of tools for a multi-spindle turret, lathe turret, or a tool magazine.

Standard = 0.0

This option intended for tool magazines or changers which rotate (automatically) least distance for next tool. Set equal to zero if direction of rotation is fixed, or if no multispindle or magazine exists. Set value negative if M codes are used for CLW and CCLW rotation. See also option 106.

OPTION 89

Automatic time delay and issuance of the spindle stop M code, when the spindle changes direction. Cannot be used for range change. See option 53 for range change.

0 = NO
Any value = YES

Standard = 0.0 seconds

OPTION 90

If the exact spindle speed is not available from the table, use:

-1 = next lower speed
0 = closest speed
+1 = next higher speed

Standard = -1.0

OPTION TABLE FOR GECENT III POSTPROCESSOR

OPTION 91

Assumed feedrate range, if none is given by initial FEDRAT statement. Used only for multiple feed range machines, excluding rapid traverse.

Standard = 2.0

See option 18

OPTION 92 C

Dwell time to use when coupling or uncoupling the encoder.

Standard = 2.0 seconds

0 = no dwell and no speed reduction. If the spindle speed must be reduced to the lowest speed in its range before coupling, set the value negative. If no dwell is wanted but the speed must be reduced, use a small value; i.e., 0.000001. This action does not occur when uncoupling.

OPTION 93 C

Dwell time used for shifting from one feedrate range to another, not including the rapid traverse range.

Standard = 0.0 seconds

If the option = zero, no dwell block is issued, and the feedrate range M code is merged into a program block. A negative value means a dwell time is given when shifting into and out of only one range. Dwell is output by a special machine function subroutine.

OPTION 94

(Open for assignment)

OPTION 95

(Open for assignment)

OPTION TABLE FOR GECENT III POSTPROCESSOR

OPTION 96

This option is used to determine the relationship between tool (or turret face), tool offset, and turret position when these parameters are part of the format of the T word on contouring systems (usually lathes) as called by the TURRET statement.

It is also used for a particular type of positioning control where the couplet OFSETL, n is used with LOAD/TOOL, SELECT/TOOL, of TOOLNO statement and the output is an M code selecting the tool offset independent of the tool number.

When used with the TURRET statement (contouring), option 96 is the type number of the T word format as given in the table below:

Type	No. of Digits	Format of T word				
No.	in T word	1st Digit	2nd Digit	3rd Digit	4th Digit	5th Digit
0	1	Tool no.				
0	1	Tool no. incl offset				
3	2	Tool no. incl offset	Turret Position			
4	2	Tool no.	Offset			
5	2	Offset	Tool no.			
6	3	Offset	Tool no.	Turret Position		
7	3	Offset	Tool Position			
0	5	Five digit Tool number				
2	5	Tool Offset		Tool no.	Turret Position	
1	5	Tool Offset		Tool number		Turret Position

For positioning controls, a negative value (-K) for option 96 will cause the postprocessor to output an M code whose magnitude is equal to the value of n taken from the couplet OFSETL, n plus the absolute value of K.

See section 4.2.54 of the part programmer's manual.

OPTION TABLE FOR GECENT III POSTPROCESSOR

OPTION 97 C

If available, should the APT statements REWIND and END issue a dwell, in seconds, and a zero T code to count out the tool offset?

0 = NO
any value = YES

Standard = 0.0 seconds

If a value is assigned, the postprocessor issues a dwell block of the given time. The dwell block also carries a zero T code. This dwell precedes the rewind command block.

OPTION 98 C

Do non-motion command blocks require feedrate commands, or an F code?

 0 = no feedrate command
+N = use the value N in all non-motion blocks except dwell blocks.
-N = use the ABS(N) value in all non-motion blocks.

Standard = 0.0

When a feedrate command (N) is given, the postprocessor outputs the G code stored in TABLEG(5) and the block execution time will be determined for this mode. The given feedrate command is also stored in a TMARK block, but the value is not printed. Do not use this option where G04 is used for dwell; see option 148.

OPTION 99

On a SPINDL/NEUTRL, what S code should be used?

Standard = 0.0

If S00 is the command for a SPINDL/NEUTRL, set this option equal to 0.1. If the option equals zero, no spindle speed command is output. The spindle speed command issued for a NEUTRL does not replace the current spindle speed or the spindle range. The value given must be in command form and not in RPM form. Then the value in the absolute printout will also be in command form. This is also used for the S code put into a TMARK block as controlled by option 146.

OPTION TABLE FOR GECENT III POSTPROCESSOR

NOTE...Options 100-105 are reserved for constants which appear in the multi-axis transform equations. They are used for multi-axis machines only. They are not used for 3-axis, and should be set to standard value.

OPTION 100 MA

Distance from head 1 spindle axis to head 2 spindle axis, for head 2.

Standard = 0.0 inches

OPTION 101 MA

Distance from head 1 spindle axis to head 2 spindle axis, for head 1.

Standard = 0.0 inches

OPTION 102 MA

Distance between the rotary head axis and the spindle face, for head 2, small gripper.

Standard = 0.0 inches

OPTION 103 MA

Distance between the rotary head axis and the spindle face, for head 2, large gripper.

Standard = 0.0 inches

OPTION 104 MA

Distance between the rotary head axis and the spindle face, for head 1, small gripper.

Standard = 0.0 inches

OPTION TABLE FOR GECENT III POSTPROCESSOR

OPTION 105 MA

Distance between the rotary head axis and the spindle face, for head 1, large gripper.

Standard = 0.0 inches

OPTION 106

Number of turret positions available.

+N = number of turret positions when numbered CLW facing the turret
-N = number of turret positions when numbered CCLW facing the turret

Standard = 0.0

This is used to determine tool corrective moves for lathe turrets. See also option 88.

OPTION 107

Should redundant F codes be suppressed?

1 = YES
0 = NO
2 = Suppress redundant F codes, except those in a T code block.

Standard = 1.0

OPTION 108

This option is open for assignment

Standard = 0.0

OPTION TABLE FOR GECENT III POSTPROCESSOR

OPTION 109

Mode of all rapid commands.

0 = Rapid is one shot and the initial FEDRAT assumes IPM unless specified otherwise.
+1 = If FEDRAT is greater than the maximum, motion is made rapid
-1 = Rapid is modal and is cancelled only by a FEDRAT statement.
+2 = if FEDRAT is greater than the maximum, the motion is made rapid, and the rapid is modal

Standard = 0.0

OPTION 110

What procedure should be followed regarding axis limit testing?

0 = No testing. If non-zero value given, see options 121-126.
+1 = Print a warning comment when a limit is exceeded, and continue.
-1 = Assume an error and quit.
+2 = Print a warning and continue, but do not produce punched output. This setting is recommended.

Standard = 0.0

OPTION 111 C

Maximum departure in degrees for the rotary axis.

Standard = 35.99964 degrees

OPTION 112 C

Radius, in inches, of the part on the rotary table.

Standard = 6.0 inches

Used to determine the rotary feedrate. Not applicable to positioning machines.

OPTION TABLE FOR GECENT III POSTPROCESSOR

OPTION 113

Minimum feedrate in RPM for the rotary table.

Standard = 0.0012 RPM

OPTION 114

Maximum feedrate in RPM for the rotary table.

Standard = 1.2 RPM

OPTION 115

Rapid traverse speed in RPM for the rotary table.

Standard = 2.0 RPM

OPTION 116 MA

What class geometry transform equations are used? (required only for multiaxis machines)

Standard = 0.0 (no class equation used)

OPTION 117

What direction of rotary move will cause an increase in reading on the rotary axis scale?

+1 = CLW
-1 = CCLW

Standard = +1

NOTE...CLW is defined as advancing a righthand screw along the negative direction of the axis of rotation. (This is opposite to the standard EIA definition.)

Also see option 120.

OPTION TABLE FOR GECENT III POSTPROCESSOR

OPTION 118

Rotary axes are scaled as follows-

0 = parts of a revolution (1 part = 1 revolution = 360 degrees)
+1 = degrees
-1 = 100 parts per revolution (100 parts = 1 revolution = 360 degrees)

Standard = 0.0

Also see option 117

OPTION 119

Rotary axes minimum step size in output units.

Standard = 0.0001

OPTION 120

For incremental machines, the direction of the table rotation for a positive command is:

+1 = CLW
-1 = CCLW
-2 = Outputs a plus sign for CLW and a minus sign for CCLW rotation of absolute positioning machines to show direction of rotation only, even though moving to a less positive position.

Standard = 1.0

See option 117 for definition of CLW and CCLW as used here.

OPTION TABLE FOR GECENT III POSTPROCESSOR

NOTE...Options 121-126 test for departures which exceed axis limits. Slide limit testing is done on the axis designations used by the partprogrammer and before they are "shuffled" by option 59; however the signs given by option 59 are used. Therefore, store the values of options 121-126 in CLTAPE order. For example, an engine lathe with 20 inch longitudinal axis Z, and 10 inch cross-slide X, programmed in the first quadrant (XY) but shuffled by options 59 (output) as +Z-X would have options 121-126 set as follows:

121 = 20   123 = 0   125 = 0
122 = 0   124 = -8   126 = 0

Multiaxes machines do all slide limit testing in the machine function subroutine, but these options may be used for the input. If other limits must be tested, use the machine subroutine.

OPTION 121

Upper or more positive slide limit on the X value taken from the CLTAPE

Standard = 0.0 inches

OPTION 122

Lower or least slide limit on the X value taken from the CLTAPE

Standard = 0.0 inches

OPTION 123

Upper or more positive slide limit on the Y value taken from the CLTAPE

Standard = 0.0 inches

OPTION 124

Lower or least slide limit on the Y value taken from the CLTAPE

Standard = 0.0 inches

OPTION TABLE FOR GECENT III POSTPROCESSOR

OPTION 125

Upper or more positive slide limit on the Z value taken from the CLTAPE

Standard = 0.0 inches

Option 126

Lower or least slide limit on the Z value taken from the CLTAPE

Standard = 0.0 inches

OPTION 127

Spindle is turned off by-

1 = S00
0 = TABLEM(6) whose standard value is M05

Standard = 0.0

OPTION 128 MA

Average head tool swing radius in inches.

Standard = 0.0 inches

Used for feedrate and cut time calculations.

OPTION 129

On a STOP, OPSTOP, BREAK, or SPINDL/OFF, should the postprocessor set the spindle range to 1, or retain the current range.

0 = range 1
1 = current range

Standard = 0.0

A new spindle statement will take precedence.

OPTION TABLE FOR GECENT III POSTPROCESSOR

OPTION 130 P

How should a Z motion be made output on a positioning machine?

 0 = ignore Z
+1 = make it a separate output block
-1 = output X, Y, and Z in one block

Standard = 0.0

OPTION 131

(Open for assignment)

OPTION 132

Type of machine.

0 = lathe
1 = mill
2 = positioning
3 = drafting
4 = flame cutter
5 = welder
6 = filament winder
7 = positioning used as a contouring
8 = positioning machine with contouring option (7500 series)

Standard = 1.0

OPTION 133

Does the machine subroutine have special functions to perform?

0 = NO
1 = YES (GECENT Customer Service will determine this.)

Standard = 0.0

MCHCON is tested to see which particular function is to be performed. For example, if option (19) = 7.0, then set option(133) = 1.0

OPTION TABLE FOR GECENT III POSTPROCESSOR

OPTION 134

Maximum length of punched tape for the control:

0 = ignore the break sequence. See Section 4.7

Standard = 1100.0 feet

OPTION 135

Minimum length or punched tape for the control:

0 = ignore the break sequence

Standard = 900.0 feet

The postprocessor looks for the break statement once the tape footage exceeds the amount given by this option.

OPTION 136

Does the control automatically reset the spindle and coolant to their former conditions when there is a restart after a programmed stop? (M00)

0 = NO
1 = YES

Standard = 0.0

A new spindle or coolant statement will take precedence.

OPTION 137

When changing spindle ranges, is special shifting required? (Not applicable if manual shift is required.)

0 = No special shift required
+N = Reduce speed to nth row of current range. Shift to the nth row of the new range, then the desired speed. Type 2 spindles only.
-N = Reduce speed to /N/ percent of maximum in current range with a 6 second dwell, then shift to the new speed. (Type 10 only)

Standard = 0.0

OPTION TABLE FOR GECENT III POSTPROCESSOR

OPTION 138

Unit system to be used.

0 = inch
1 = metric

Standard = 0.0

OPTION 139

Is an additional feedrate register required?

0 = NO
N = YES, and N gives the location of the register cell (TABLE REGFOR)

Standard = 0.0

If this option is used for a rotary table feedrate, set option 141 also.

OPTION 140 P

For positioning machines, should the Z values of the cutter location data be adjusted for inversion and tool length?

0 = NO
N = YES, where N is the distance from the spindle nose (in the head up position) and the part coordinate zero reference.

Standard = 0.0

If option 86 is set to -1, tool length compensation occurs.

OPTION 141 P

What type of feedrate command does the rotary table have, if any?

0 = No separate feedrate command
1 = FCOM(N) = option(114)/(1.43**(9-N)), where N = 1,2,3,...9

Standard = 0.0

This assumes the table has its own register. See Section 4.1.3.1

OPTION TABLE FOR GECENT III POSTPROCESSOR

OPTION 142

(Open for assignment)

OPTION 143

For SEQNO/ON, should the sequence number be the cutter location record number, or should it be a unit increasing number?

0 = CL tape record number
+1 = Unit increasing number. (In using this, option 17 must not equal 2)
-1 = CL tape record number, with the redundant N codes suppressed.

Standard = 0.0

If no SEQNO statement is used, the standard value is assumed.

OPTION 144 P

For positioning machines only, the increment to add to feed code to produce the correct feedrate command for a type 2 and type 5 feedrate. (see option 78)

Standard = 0.0

OPTION 145

Should the postprocessor reinstate previous machine conditions after a STOP, OPSTOP, or BREAK?

0 = NO
+1 = YES, reinstated conditions are: spindle on, rapid or feed, tool or turret code, and coolant.
-1 = YES, and the postprocessor will always out the rapid M codes before reinstating the conditions.
-2 = Reinstate rapid M codes only.

OPTION TABLE FOR GECENT III POSTPROCESSOR

OPTION 146

Should TMARK blocks have the following? (TMARK is End of Record)

0 = nothing, but a TMARK and any reguired F code.
+1 = a G code, value from TABLEG(5)
-1 = G code, value from TABLEG(5), + an S code, value from option 99.

Standard = 0.0

OPTION 147 C

Should the G01 and G11 be suppressed in the punched output?

0 = NO
1 = YES

Standard = 0.0

OPTION 148 C

Should a dwell be produced for a non-motion block?

0 = NO
N = value in seconds
-N = N second dwell in a STOP or OPSTOP block only

Standard = 0.0 seconds

Do not use this option if option 98 is non-zero. Used for the G04 type dwells only.

OPTION 149

Lower limit for head rotation in degrees.

Standard = 0.0 degrees

OPTION TABLE FOR GECENT III POSTPROCESSOR

OPTION 150

Upper limit for head rotation in degrees.

Standard = 0.0 degrees

OPTION 151 MH

Percent of tolerence band for which segmentation will not take place. (Used only with equal time method of segmentation.) See Section 3.4.8.

Standard = 0.0 percent

OPTION 152 MH

Is a second set of preparatory codes available? (Assigned a letter other than G.)

0 = NO
1 = YES

Standard = 0.0

OPTION 153 MH

Is a second or third S code available?

0 = NO
2 = 2 spindle tables
3 = 3 spindle tables

Standard = 0.0

OPTION 154 MH

Is a second set of miscellaneous codes available?

0 = NO
1 = YES

Standard = 0.0

OPTION TABLE FOR GECENT III POSTPROCESSOR

OPTION 155 MH

In multi-head, do the heads share a common axis?

0 = NO
1 = YES, X-axis
2 = YES, Y-axis
3 = YES, Z-axis

Standard = 0.0

OPTION 156 MH

Interpolation modes available.

0 = line-line
1 = circle-line
2 = circle-circle
3 = circle-circle and line-circle

Standard = 0.0

OPTION 157 MH

Should common axis component feedrates on each head be within a given tolerance?

0 = NO
N = YES, and gives the tolerance in percent.

Standard = 0.0 percent

OPTION 158 MH

Clearance between heads on the second axis by CLTAPE order.

0 = NONE
N = YES, and gives the value in inches.

Standard = 0.0 inches

OPTION TABLE FOR GECENT III POSTPROCESSOR

OPTION 159 MH

Clearance between heads on the first axis by CLTAPE order.

0 = NONE
N = YES, and gives the value in inches.

Standard = 0.0 inches

OPTION 160

Number of print positions to be used in the calculation and spacing of the printed output.

Standard = 120.0 columns

OPTION 161

What routine should produce the readable PARTNO when using tape image type of output, PUNCHA? (option 20 = 1.0)

0 = odd parity (PARNOM)
1 = even parity (PARNEM)

Standard = 0.0

OPTION 162

(Open for assignment)

OPTION 163

Is the readable PARTNO to be made output when using BCD type output, PUNCHB? (option 20 = 0.0)

0 = YES
1 = NO

Standard = 0.0

OPTION TABLE FOR GECENT III POSTPROCESSOR

OPTION 164

Select the printed output form desired.

1 = GEOUT1 Summary printout.
2 = GEOUT2 Combined printout. Used also for multi-axes.
3 = GEOUT3 Multi-head printout. (see option 17)
4 = GEOUT4 Printout for non-multi-head machines

Standard = 1.0 See Section 3.5.

OPTION 165

1 = 100S
0 = not a 100 S control

Standard = 0.0

OPTION 166

(Open for assignment)

Standard = 0.0

OPTION 167

This option is open for assignment

Standard = 0.0

OPTION 168

Should leader be produced after an END statement?

0 = NO
N = YES, and gives the length in inches

Standard = 0.0

OPTION TABLE FOR GECENT III POSTPROCESSOR

OPTION 169

Should the non-readable PARTNO image be punched in the tape?

0 = YES
1 = NO

Standard = 0.0

This does not pertain to the readable PARTNO.

OPTION 170 C

Should the postprocessor segment linear paths as a function of the available G codes?

0 = Do not segment
1 = Attempt to segment feedrate command limited blocks

Standard = 0.0 See Section 4.1.5.1

This segmentation sequence will attempt to segment linear cut paths when the desired feedrate produces a feedrate command exceeding the feedrate command maximum. (option 24)
Do not use if-

1. positioning machine or multi-axes machine
2. circular interpolation is available
3. only one linear G code is available
4. absolute input used with contouring controls (7500 series)

OPTION 171

Should redundant S words be output?

0 = YES
1 = NO

Standard = 0.0

OPTION TABLE FOR GECENT III POSTPROCESSOR

OPTION 172 C

Should GEOUT1 produce the ABSOLUTE data or OPERATOR data.

0 = ABSOLUTE
1 = OPERATOR

Standard = 0.0

Positioning control systems use ABSOLUTE data.

OPTION 173

Punched output control.

1 = print listing and punch tape
0 = printed output only, no punched tape output.

Standard = 0.0

OPTION 174

If the feedrate table (FRTAB) is required, what size must it be? (See option 18.)

Standard = 0.0

The size of FRTAB + SRTAB = 300.0

OPTION 175 C

Maximum spindle speed permitted when threading encoder is coupled.

Standard = 600.0 RPM

OPTION 176

(Open for assignment)

Standard = 0.0

OPTION TABLE FOR GECENT III POSTPROCESSOR

OPTION 177

Computer manufacturer

0 = IBM 360
1 = RCA
2 = GE 635
3 = CDC
4 = UNIVAC

Standard = 0.0

OPTION 178

(Open for assignment)

OPTION 179 C

Number of rotary table speed ranges.

Standard = 1.0

OPTION 180 C

Number of table speeds per range.

Standard = 2.0

OPTION 181 C

Assumed table speed in RPM, if none is given.

Standard = 1.0 RPM

OPTION 182 C

Dwell time when the rotary table changes direction.

Standard = 0.0 seconds

OPTION 183 C

Dwell time for table startup.

Standard = 0.0 seconds

OPTION TABLE FOR GECENT III POSTPROCESSOR

NOTE...Options 184-189 are by CLTAPE order, not by option 59 order.

OPTION 184 C

Home position in X (Machine coordinate)

Standard = 0.0

OPTION 185 C

Home position in Y (Machine coordinate)

Standard = 0.0

OPTION 186 C

Home position is Z (Machine coordinate)

Standard = 0.0

OPTION 187

Home position for the rotary head

Standard = 0.0

OPTION 188

Home position for the rotary table

Standard = 0.0

OPTION 189

Home position for the third rotary axis

Standard = 0.0

OPTION TABLE FOR GECENT III POSTPROCESSOR

OPTION 190

Home position for the secondary axis

Standard = 0.0

OPTION 191 MH

Should the multi-head flag (MULTHD) be set in the postprocessor without giving a COMBIN/ statement in the part program?

0 = NO
1 = YES

Standard = 0.0

OPTION 192

Maximum feedrate in RPM for the rotary head.

Standard = 0.0 RPM

OPTION 193

Dwell time in seconds for changing spindle speeds.

Standard = 0.0 seconds

OPTION 194

Should a spindle stop code be output before a range change?

0 = NO
1 = YES

Standard = 0.0

OPTION TABLE FOR GECENT III POSTPROCESSOR

OPTION 195 MH

In multi-head, do the heads share a common dwell register?

0 = NO
Letter= desired dwell register name (Alphabetic character)

Standard = 0.0

OPTION 196

For a type 12 spindle (option 19 = 12.0) what is the cross-over spindle speed at which a dwell should be issued?

Standard = 0.0

OPTION 197 MH

In multi-head, are there more than 2 heads, and does more than one head share the same set of registers and format?

0 = NO
1 = YES, heads 1 and 2 share the same registers and format
+2 = YES, heads 2 and 3 share the same registers and format

Standard = 0.0

OPTION 198 MA

Does the part reference system move with the table and part when ROTREF is not given?

0 = YES
1 = NO

Standard = 0.0

OPTION 199

Lower slide limit for the second rotary head.

Standard = 0.0

OPTION TABLE FOR GECENT III POSTPROCESSOR

OPTION 200

Upper slide limit for the second rotary head.

Standard = 0.0

OPTION 201 P

Number of feedrate ranges for Z. (Positioning feedtype 5 only)

Standard = 0.0

OPTION 202 P

Minimum feedrate command for X and Y (Positioning feedtype 5 only)

Standard = 0.0

OPTION 203

Are there special functions to be performed in the output section?

0 = NO
1 = YES The machine function subroutine is called MSRTGO not MACSRT.

Standard = 0 (GECENT Customer Service will determine this.)

OPTION 204

(Open for assignment)

OPTION 205 C

Type of system used for threading

0 = Three range system
1 = Two range system

Standard = 0

OPTION TABLE FOR GECENT III POSTPROCESSOR

OPTION 206

A ROTATE/TABLE, α , CLW statement causes the table to move-

+1 = CLW
-1 = CCLW

Standard = +1

(See options 117 and 120)

OPTIONS 207-212

These options are not tested by the standard GECENT III postprocessor. Functions performed are used only in special machine function subroutines and are independent of the machine/control hardware.

Standard = 0

OPTION 213

Is this a contouring machine with a postioning control?

0 = NO
1 = YES

Standard = 0.0

OPTION 214

What parity is used with PUNCHB output?

0 = no parity
1 = odd
2 = even

Standard = 0.0

This option is used only with PUNCHB on the GE635

OPTION TABLE FOR GECENT III POSTPROCESSOR

OPTION 215 P

For a Type 5 feedrate what is the feed code increment between ranges?

Standard = 0

Any value is the increment

OPTION 216

(Open for assignment)

OPTION 217

(Open for assignment)

OPTION 218

Should MACHIN/OFF print out total cut and dwell times and slew a page?

0 = NO
1 = YES

Standard = 0

OPTION 129 C

Is the feedrate range M code required to be output in a block before the dwell time for changing ranges?

0 = NO
1 = YES

Standard = 0

OPTION 220 C

Lower limit for table rotation in degrees.

Standard = 0.0

OPTION TABLE FOR GECENT III POSTPROCESSOR

OPTION 221 C

Upper limit for table rotation in degrees.

Standard = 0.0

OPTION 222 C

On a threading record should I, J, K be modified to give a more accurate feedrate command?

0 = NO
1 = YES

Standard = 0.0

OPTION 223 C

Is the maximum departure for circular interpolation different from the maximum departure for linear interpolation?

0 = NO
N = value of max. departure for circular interpolation

Standard = 0.0

OPTION 224 C

Is it necessary to use I, J, K in calculating a feedrate command for linear moves with a departure greater than a specified minimum. (Applicable only to 100S control in the metric system. Option 26 must equal 1).

0 = NO
N = YES where N equals the minimum value

Standard = 0.0

Options are dimensioned thru 250. All standard values are presently = 0.0

The options 246 - 250 are reserved for customer use only.

5.6.5.1 CUSTOMER OPTIONS

Options 246 through 250 are reserved for customer usage only and will never be touched by the GECENT III postprocessor as part of its standard operation.

Installations which have need of optional assignments for such items as computer operation, method of off-line processing, NC machine special sequences, and so on, may make use of these options. The references and postprocessor modifications made for the customer options are the responsibility of the user customer; subsequent developments of the GECENT III postprocessor which affect these modified areas of the postprocessor require the user customer to make the necessary changes which retains the programmed integrity of the customer option functions.

An example of how the customer options can be used effectively is the following case wherein one Machine Subroutine is made to apply to several NC machines whose only difference is the number and value of spindle speeds. The Machine Subroutine is written such that, dependent upon the settings of, say, options 246 and 247, SRTAB is set up with the correct spindle speeds for the particular NC machine.

The customer options permit the user to choose one of the several spindle tables by defining the options as follows:

OPTION 246

The number of speeds in the spindle table; there may be either 20 or 100. As a standard, set the option to 20 in the Machine Subroutine.

OPTION 247

A parameter for choosing the machine size. As a standard, set the option equal to 4 in the Machine Subroutine.

A possible list of values for option 247 might be as follows:

Option 247	Machine Size
1	16"
2	18"
3	32"
4	40"
5	64"
6	70"
7	90"
8	100"

For example, if the 100 speed table for the 40" machine is desired, OPTAB(246) = 100 and OPTAB(247) = 4 would be set in the MACHIN statement. The Machine Subroutine then sets up SRTAB accordingly.

5.6.6 WRITING A MACHINE SUBROUTINE

As was mentioned above in Section 5.6, when setting up a Machine Subroutine the only items of the tables which must be specified are those which deviate from the Standard Machine. The tables show the standard values they assume and so each table must be separately consulted when writing the Machine Subroutine.

For convenience, the Standrad Machine is summarized here. It reperesents a three-axis milling machine with a tape controlled Type 1 spindle and a tool changer; it has a contouring, incremental system which uses both linear and circular interpolation.

Standard Machine

TABLEG	TABLEM
(2) = 1	(1) = 0
(3) = 2	(2) = 1
(4) = 3	(3) = 2
(5) = 4	(4) = 3
(18) = 17	(5) = 4
(19) = 18	(6) = 5
(20) = 19	(9) = 8
	(10) = 9
	(31) = 30

5.6.6 WRITING A MACHINE SUBROUTINE (cont'd)

Standard Machine

REGSTR	REGFOR
(1) = N	(1) = 30
(2) = G	(2) = 20
(3) = X	(3) =-14
(4) = Y	(4) =-14
(5) = Z	(5) =-14
(6) = A	(6) = 0
(7) = B	(7) = 0
(8) = I	(8) = 14
(9) = J	(9) = 14
(10) = K	(10) = 14
(11) = F	(11) = 30
(12) = S	(12) = 30
(13) = T	(13) = 20
(14) = M	(14) = 20
(15) (IGNORE)	(15) (IGNORE)
(16) = R	(16) = 0
(17) = (OPEN)	(17) = 0
(18) = C	(18) = 0

SRTAB(1) = 1

SRTAB(2) = 300

The Standard Machine assumes the standard settings of the Option Table.

In general, setting up a Machine Subroutine is very simple since all that is normally necessary is to set certain known values in their proper table location. In addition to setting up TABLEG, TABLEM, REGSTR, REGFOR, and SRTAB, there are some other special items which may have to be considered.

5.6.6.1 TAG ARRAY

The array TAG (dimensioned at 6) is a BCD storage array for the NC machine identification title which is printed at the top of each page. For example, the title

"MACHINE 40 BRACK LATHE"

is set up in BCD form (as through a DATA statement) and stored into TAG.

```
TAG(1) = MACHIN

TAG(2) = E 1  20  1  1

TAG(3) = BRACK 1

TAG(4) = LATHE 1
```

The TAG array is initialized to BLANKS.

It is essential to set up the TAG array; if it is not set up, a blank line is printed. But it is a very convenient device for identifying printouts and its usage is strongly advised. The words stored in TAG in this example assume six characters to a word.

5.6.6.2 IORDER VECTOR*

Some NC machines require a certain order of input of the word address registers, thereby requiring the postprocessor to output the registers in that order. Normally, the postprocessor outputs the registers and their associated data in the order given in REGSTR, i.e.,

N G X Y Z A B I J K F S T M

But if a different order is required, say,

N G X I Y J Z K F S A B T M

then the postprocessor must be instructed as to the proper sequence. This is the function of the IORDER (dimensioned at 30) vector.

The IODER vector is ordered according to the REGSTR standard order. Thus, if no reordering of the registers is required, the IORDER vector is not even used. However, if a reordering of registers is needed, the elements of the IORDER vector indicate the relocation spot relative to the fixed order of REGSTR.

* Not yet available.

5.6.6.2 IORDER VECTOR (cont'd)

In the above example, the IORDER vector would be set up as:

IORDER(1)	=	1	(N remains unchanged)
(2)	=	2	(G remains unchanged)
(3)	=	3	(X remains unchanged)
(4)	=	5	(Y goes to cell 5)
(5)	=	7	(Z goes to cell 7)
(6)	=	11	(A goes to cell 11)
(7)	=	12	(B goes to cell 12)
(8)	=	4	(I goes to cell 4)
(9)	=	6	(J goes to call 6)
(10)	=	8	(K goes to cell 8)
(11)	=	9	(S goes to cell 9)
(12)	=	10	(S goes to cell 10)
(13)	=	13	(T remains unchanged)
(14)	=	14	(M remains unchanged)

5.6.6.3 FRTAB TABLE

When an NC machine has more than one feedrate range (not counting rapid traverse as a feedrate range), the postprocessor requires an additional table to specify the maximum and minimum values of each range. Normally for one range, only options 48 (feedrate minimum) and 25 (feedrate maximum) need be set; but for multiple feedrate ranges, another device must be used, and this is the feedrate table FRTAB.

Actually, the FRTAB does not in itself exist since it derives from being made equivalent to SRTAB. This technique is used in order to minimize core loads. Therefore, the maximum size of FRTAB must be such that

$$\text{FRTAB} + \text{SRTAB} \leq 300.$$

5.6.6.3 FRTAB TABLE (cont'd)

The spindle speeds are stored in SRTAB beginning at SRTAB(1) and stored consecutively therefrom. The feedrate range values are stored at the opposite end of SRTAB such that the last feedrate value is stored in SRTAB(300).

For example, assume an NC machine has 4 feedrate ranges. It is necessary to set up the minimum and maximum values of each range.

FRTAB(293) = Range 1 Minimum

FRTAB(294) = Range 1 Maximum

FRTAB(295) = Range 2 Minimum

FRTAB(296) = Range 2 Maximum

FRTAB(297) = Range 3 Minimum

FRTAB(298) = Range 3 Maximum

FRTAB(299) = Range 4 Minimum

FRTAB(300) = Range 4 Maximum

Note that the minimum value must be first given, then followed by the maximum value of that range; the ranges must be given in increasing numeric order as illustrated.

Two important options related to the use of FRTAB are:

Option 18 - the number of feedrate ranges;

Option 174- the size of the FRTAB.

For the above example, option 18 = 4, and option 174 = 8, i.e., eight rows are used. Subroutine ASSIGN computes the FRTAB index INDFR by

INDFR = 300 - option 174.

For our example then, INDFR = 292. The index INDFR is used in DO loops beginning from I = 1; thus, INDFR + I gives the correct FRTAB table location as needed.

If M codes are required in order to shift from one feedrate range to another, the codes must be assigned in TABLEM(113) through (119).

Other pertinent options pertaining to multiple feedrate ranges are options 91 and 93.

5.6.6.4 MACHINE SUBROUTINE CHECKLIST

The summary checklist given here provides the step-by-step precedure which should be followed in setting up a Machine Subroutine.

(1) Assign a number (not more than two digits) to the NC machine; this number also identifies the related subroutine.

(2) Check the NC machine's preparatory and miscellaneous functions versus the functions listed in TABLEG and TABLEM to ascertain that the machine functions are available in the postprocessor.

(3) Delete Standard Machine G and M functions which are not desired; deletion is accomplished by setting the G code or M code to DMBITS.

(4) Assign the NC machine's G codes or M codes, where needed, to their respective table locations.

(5) Change the REGSTR table values, where needed, to correspond with the output requirements of the NC machine.

(6) Change the REGFOR table values, where needed, to correspond with the output requirements of the NC machine. Delete with a zero any registers not desired.

(7) Setup the Tag array with the identifying BCD title.

(8) List the spindle speeds (if any) sequentially, and group them into ranges for storage into SRTAB.

(9) Store the feedrate minimum and maximum values (if any) into FRTAB.

(10) Set up the IORDER vector, if needed.

(11) Setup the OPTAB table as required. Be sure to reverify the following key options:

a. Options 7 and 8; their multiple must equal the number of spindle speeds stored into SRTAB.
b. Option 19 specifies the spindle type.
c. Options 18 and 174 must correlate with the number of feedrate ranges stored into FRTAB.
d. Option 20 specifies the type of punched output and option 164 the type of printed output.
e. Options 59 and 60 specify the register shuffling.
f. If the NC machine is a multiaxis machine, option 116 specifies the geometry class.
g. Option 132 specifies the type of NC machine.
h. Option 133 must be set properly if the Machine Subroutine performs functions; see Section 5.6.7.

(12) The compiled subroutine goes into overlay GEINIT. Be sure to remove the dummy subroutine of the same name if one exists.

5.6.6.4 MACHINE SUBROUTINE CHECKLIST (cont'd)

The following example illustrates a possible Machine Subroutine and the related key procedures which are followed when developing the subroutine. In Section 5.6.8 are given examples which illustrate Machine Subroutines for a positioning machine, a lathe, a 3-axis mill, a multiaxis mill, and a multihead machine.

Sample Machine Subroutine

```
   SUBROUTINE MACH40

   (GECOM and GECBAS COMMON)

   DIMENSION TNAME(4), TABA(6), REG(1)

   DATA TNAME/8HMACHINE ①, 8H40 ① BRACK, 8HLATHE ② /*

   DATA TABA/10.,20.,30.,20.,40.,60./

   DATA REG/8HH ⑦ /

   DO 1 I=1,4

 1 TAG(I)=TNAME(I)    (Save the identification title)

   DO 10 I=1,6

10 SRTAB(I) = TABA(I)    (Store the spindle speeds)

   TABLEG(3) = DMBITS   (No circular interpolation available.)

   TABLEG(4) = DMBITS   (Delete all related functions.)

   TABLEG(18) = DMBITS

   TABLEG(19) = DMBITS

   TABLEG(20) = DMBITS

   TABLEM(2) = DMBITS   (No optional stop is available)

   TABLEM(31) = DMBITS  (No Rewind is available)
```

* The DATA statement given here assumes an eight character BCD word. Note also that the method of writing a DATA statement may differ considerably from one computer to another.

SAMPLE MACHINE SUBROUTINE (cont'd)

```
TABLEM(37)  = 70.0    (SAFETY features are available)

TABLEM(38)  = 71.0

TABLEM(39)  = 72.0

TABLEM(40)  = 80.0

TABLEM(113)  = 10.0

TABLEM(114)  = 20.0

REGSTR(8)  = DBLNKS

REGSTR(9)  = DBLNKS

REGSTR(10)  = DBLNKS

REGSTR(12)  = REG(1) (The spindle uses an H register
                     instead of an S register.)

REGFOR(3)  = -23      (X must have 2 digits to the left and 3
                     digits to the right of the decimal point.)

REGFOR(8)  = 0        (No IJK registers are available.)

REGFOR(9)  = 0

REGFOR(10)  = 0

REGFOR(11)  = 40      (F is to have 4 digits.)

OPTAB(7)  = 2         (Number of spindle ranges)

OPTAB(8)  = 3         (Number of speeds per range.)

OPTAB(9)  = 0         (No circular interpolation)

OPTAB(18)  = 2        (Two feedrate ranges available)

OPTAB(19)  = 0        (Spindle Type 0)

OPTAB(26)  = 1        (Cannot use multiplification factor since
                     IJK registers are not available.)

OPTAB(28)  = 0        (Assume Linear)
```

SAMPLE MACHINE SUBROUTINE (cont'd)

```
    OPTAB(37)  = 1.17     (Minimum cut length)

    OPTAB(59)  = 132415   (Inverted Y axis)

    OPTAB(132) = 1        (NC machine is a mill)*

    OPTAB(138) = 0        (ENGLISH system of units)*

    OPTAB(164) = 2        (GEOUT2 is requested)

    OPTAB(174) = 4        (Size of FRTAB)

    OPTAB(177) = 2        (Type of computer)

    FRTAB(297) = 0.1      Feedrate Range 1 Minimum

    FRTAB(298) = 10       Feedrate Range 1 Maximum

    FRTAB(299) = 7        Feedrate Range 2 Minimum

    FRTAB(300) = 60       Feedrate Range 2 Maximum

    RETURN

    END
```

* These option values are already standard values and would not have to be reset, but they are listed for completeness.

5.6.7 MACHINE SUBROUTINE FUNCTIONS (MACFUN)

Despite the generality of the GECENT III postprocessor there are functions which an NC machine may possess that are not handled by the postprocessor. In many instances the machine function is so unique as not to warrant it's inclusion as a general feature of the GECENT III postprocessor. Rather than modify the postprocessor, and create a non-standard postprocessor, a better and simpler technique is used which isolates the special NC machine function to the Machine Subroutine, so that the function is executed in the postprocessor only when the particular Machine Subroutine is in core; this is the so-called MACFUN portion of the Machine Subroutine.

5.6.7 MACHINE SUBROUTINE FUNCTIONS (MACFON) (cont'd)

This technique requires that the special machine functions be programmed completely within the Machine Subroutine. A flag MCHCON is set by the postprocessor which, when interrogated by the Machine Subroutine, branches to the proper sequence within the subroutine to execute the required function; return is to the source subroutine for continued processing.

An example will best illustrate the method. Suppose an NC machine requires a special tool loading sequence which is not a standard operation of the postprocessor, and suppose further that this function is to be initiated by a LOAD/TOOL statement. To obtain the desired machine function, the normal program flow of the postprocessor in subroutine LOAD must be temporarily interrupted so that the Machine Subroutine can perform the function. This is accomplished by option 133 which, if non-zero, indicates that the postprocessor must branch to the Machine Subroutine for some special operations. If option 133 is zero (the standard value), then program flow is normal.

Thus, in the example, since option 133 has been set to 1, subroutine LOAD calls subroutine MACSRT which branches to the Machine Subroutine.

Before branching to the Machine Subroutine, subroutine LOAD sets the flag MCHCON (Machine Constant) to 6 which indicates that the source subroutine is subroutine LOAD. The first statement in the Machine Subroutine tests MCHCON and branches according to it's value; for example:

MCHCON = 0; no special function is called for, therefore, set up the standard tables (TABLEG,TABLEM,etc), and return.

MCHCON = 6; a LOAD/TOOL statement was given; branch to the special loading sequence programmed within the Machine Subroutine, execute the special function, set the return flag RETURN to 1, and return.

MCHCON ≠ 0 or 6; the source subroutine is not one recognized by this Machine Subroutine, therefore, set the return flag RETURN to 0, and return.

5.6.7 MACHINE SUBROUTINE FUNCTIONS (MACFON) (cont'd)

Upon returning to the source subroutine (subroutine LOAD in the example), the flag RETURN is immediately tested. If RETURN is zero, then the normal program flow is followed, but if RETURN is non-zero, then the subroutine bypasses the remainder of its subroutine and branches to the subroutine exit since the LOAD/TOOL function was alreadv performed in the Machine Subroutine. (see Flow Chart 14, Section 5.3 for a diagram of the sequence flow.)

This technique can be used for all such special functions. The only requisite is that the affected source subroutine must test option 133 and branch accordingly as described above. At present there are many source subroutines which permit a great variety of special functions through this technique. The following list gives the presently defined MCHCONs.

MCHCON

0 = Set up the tables in the Machine Subroutine

1 = Tool change sequence in subroutine TOOLNO - (Positioning Machines only)

= Also special dwell for a range change; subroutine FEDRAT (lathes)

2 = Slide limit testing subroutine TSTLIM

3 = Used for a geometry class subroutine

4 = Tab sequential special printout

5 = Spindle neutral effect - subroutine SPINDL

6 = Tool change sequence specified in subroutine LOAD

7 = Special cycle sequences - CYCLGP, CYCLGX - from subroutine CYCLE.

8 = Test the FROM point - subroutine FROM5

9 = subroutine DELAY

10 = T code re-arrangement - subroutine COMTAT

11 =

12 = Special threading sequences - subroutine THREDO

13 = Subroutine CYCLE - after setup of R + Z terms

14 =

15 =

16 = Feedrate and rapid subroutines: subroutines FEDRAT, RAPIDO, RAPIDX

17 =

18 = Special sequence for unloading the tool - subroutine UNLOAD

19 = Special motion record - subroutine POSMOV

20 = Subroutine CUTCOM

21 = Subroutine SELECT

5.6.7 MACHINE SUBROUTINE FUNCTIONS (MACFON) (cont'd)

22	=	Subroutine TURRET
23	=	Special sequence for rapid and feed M codes: subroutines RAPIDO, RAPIDX
24	=	Modify spindle table - subroutine MACHIN
25	=	Extended lead threading - subroutine SEGMNT
26	=	
27	=	Special M codes for feedrate range change - subroutine FEDRAT
28	=	Bypass standard ROTATE/TABLE sequence - subroutine ROTABL
29	=	Subroutines GOLINE, ROTOUT
30	=	Subroutine SPINDL
31	=	Positioning feedrate type - Subroutines POSFED, FEDRAT, CYCLE, SPINDL
32	=	
33	=	Subroutine SET12
34	=	Subroutine CONTUR
35	=	Subroutine MOTION
36	=	Spindle off - Subroutine SPINDL
37	=	Switch tape readers - Subroutine SELRDR
38	=	Subroutine MODE
39	=	Subroutine POSITN
40	=	
41	=	
42	=	

There is no effective limit to MCHCON since it is easy to adopt to any need. If a new MCHCON assignment is required by a customer user, he should request the MCHCON number and related additions to the source subroutine from the GECENT III maintenance group of the General Electric Company. See the foreward of this manual for the appropriate address.

On some computers the MACFUN portion of the Machine Subroutine may have to be split off from the rest of the subroutine. Once the standard tables (TABLEG, TABLEM,etc.) are loaded into COMMON, there are no further need for them and this part of the Machine Subroutine can be overlayed; but the MACFUN part must always be in core. Whichever method is used (viz., keeping the entire Machine Subroutine in core or keeping only the MACFUN part in core) is strictly up to the user and computer. If core becomes a premium factor, then only MACFUN should be kept in core.

See the sample Machine Subroutine for a multiaxis mill in Section 5.6.8 for a typical use of multiple MCHCON settings.

5.6.8 SAMPLE MACHINE SUBROUTINES

The examples which follow are actual Machine Subroutines for different types of NC machines. For simplicity, the DATA statements infer a six character BCD word size and are programmed in a manner which illustrate the intent of usage. In actual application the DATA statements must be writtem according to the format required by the computer in use.

Also, the labelled COMMONs GECOM and GECBAS are only referred to, but not included.

5.6.8.1 POSITIONING MACHINE

```
          SUBROUTINE MACH24
(SAMPLE MACHINE SUBROUTINE FOR A POSITIONING MACHINE)
*GECOM*  LABELED COMMON FOR OVERLAY *GECOM*
*GECBAS*  LABELED COMMON FOR OVERLAY *GEBASE*.
          DIMENSION SPNTAB(24), TNAME(5)
          DATA TNAME/6HSAMPLE,6H ① POSIT,6HIONING,6H ① MACHI,6HNE ④ /
          DATA WRDS16/6HR ⑤ /
          DATA(SPNTAB(I),I=1,24)/
         170.,100.,140.,200.,355.,530.,710.,1060.,
         2100.,150.,200.,300.,530.,800.,1060.,1600.,
         3150.,225.,300.,450.,800.,1200.,1600.,2400./
          DO 1 I=1,5
          TAG(I) = TNAME(I)
          TABLEG(1) = 80.0
          TABLEG(82) = 81.0
          TABLEG(83) = 82.0
          TABLEG(84) = 83.0
          TABLEG(85) = 84.0
          TABLEG(86) = 85.0
          TABLEG(87) = 86.0
          TABLEG(88) = 87.0
          TABLEM(2) = DMBITS
          TABLEM(3) = DMBITS
          TABLEM(4) = DMBITS
          TABLEM(5) = 2.0
          TABLEM(6) = DMBITS
          TABLEM(31) = 30.0
          TABLEM(102) = 6.0
          TABLEM(105) = 7.0
          OPTAB(48) = 0.5
          OPTAB(50) = 1.0
          (Option 59 is:  +X-Y+Z)
```

5.6.8.1 POSITIONING MACHINE (cont'd)

```
      OPTAB(59) = 132415.0
      OPTAB(60) = 0.
      OPTAB(83) = 3.0
      OPTAB(86) = 1.0
      OPTAB(132) = 2.0
      DO 100 I=1,24
  100 SRTAB(I) = SPNTAB(I)
      RETURN
      END
      TABLEM(123) = 4.0
      TABLEM(124) = 5.0
      REGFOR(3) = -24.0
      REGFOR(4) = -24.0
      REGFOR(5) = -24.0
      REGFOR(8) = 0.
      REGFOR(9) = 0.
      REGFOR(10) = 0.
      REGFOR(11) = 20.0
      REGFOR(12) = 20.0
      REGFOR(13) = 20.0
      REGFOR(16) = -22.0
      REGFOR(16) = WRDS16
      OPTAB(1) = 1.0
      OPTAB(4) = 98.0
      OPTAB(7) = 3.0
      OPTAB(8) = 8.0
      OPTAB(9) = 0.
      OPTAB(12) = 24.0
      OPTAB(17) = 101.0
      OPTAB(19) = 8.0
      OPTAB(23) = 0.
      OPTAB(24) = 99.0
      OPTAB(25) = 49.5
      OPTAB(26) = 1.0
      OPTAB(27) = -1.0
      OPTAB(45) = 99.0
```

5.6.8.2 LATHE

```
        SUBROUTINE MACH04
(Sample Machine Subroutine For A Lathe)
*GECOM* LABELED COMMON FOR OVERLAY *GECOM*
*GECBAS* LABELED COMMON FOR OVERLAY *GEBASE*
        DIMENSION TNAME(2)
        DATA TNAME/6HSAMPLE,6H ① LATHE/
        DO 1 I=1,2
      1 TAG(I)=TNAME(I)
        TABLEG(3)  = 3.0
        TABLEG(4)  = 2.0
        TABLEG(12)  = 11.0
        TABLEG(18)  = DMBITS
        TABLEG(19)  = DMBITS
        TABLEG(20)  = DMBITS
        TABLEG(22)  = 31.0
        TABLEG(32)  = 21.0
        TABLEG(34)  = 33.0
        TABLEG(35)  = 34.0
        TABLEG(36)  = 35.0
        TABLEM(4)  = DMBITS
        TABLEM(5)  = DMBITS
        TABLEM(7)  = 6.0
        TABLEM(9)  = 8.0
        TABLEM(10)  = 9.0
        TABLEM(37)  = 36.0
        TABLEM(38)  = 38.0
        TABLEM(39)  = 37.0
        TABLEM(40)  = 39.0
        TABLEM(42)  = 40.0
        TABLEM(43)  = 41.0
        TABLEM(51)  = 50.0
        TABLEM(52)  = 51.0
        TABLEM(71)  = 61.0
        TABLEM(72)  = 60.0
        TABLEM(73)  = 63.0
        TABLEM(74)  = 62.0
        TABLEM(113)  = 42.0
        TABLEM(114)  = 41.0
        REGFOR(4)  = 0.
        REGFOR(9)  = 0.
        REGFOR(11)  = 32.0
        REGFOR(13)  = 50.0
        FRTAB(297)  = 0.01
        FRTAB(298)  = 2.0
        FRTAB(299)  = 0.1
        FRTAB(300)  = 20.0
```

5.6.8.2 LATHE (cont'd)

```
OPTAB(4)  = 9.9999
OPTAB(5)  = 0.00005
OPTAB(7)  = 2.0
OPTAB(8)  = 2.0
OPTAB(10)  = 0.
OPTAB(12)  = 48.0
OPTAB(13)  = 0.00133
OPTAB(15)  = 0.1
OPTAB(16)  = 0.
OPTAB(18)  = 2.0
OPTAB(19)  = 10.0
OPTAB(21)  = 1.0
OPTAB(22)  = 101.0
OPTAB(24)  = 750.0
OPTAB(25)  = 20.0
OPTAB(27)  = 40.0
OPTAB(37)  = 0.18
OPTAB(38)  = 1.0
OPTAB(39)  = 20.0
OPTAB(42)  = 100.0
OPTAB(43)  = 0.
OPTAB(44)  = 100.0
OPTAB(45)  = 750.0
OPTAB(46)  = -1.0
OPTAB(48)  = 0.01
OPTAB(49)  = 0.01
OPTAB(53)  = 0.
OPTAB(54)  = 2.0
OPTAB(55)  = 0.
OPTAB(56)  = 10.0
OPTAB(57)  = 5.0
OPTAB(59)  = 152300.0
OPTAB(60)  = 181600.0
OPTAB(69)  = 0.01
OPTAB(73)  = 16.7
OPTAB(81)  = 0.
OPTAB(82)  = 0.
OPTAB(83)  = 2.0
OPTAB(89)  = 6.0
OPTAB(90)  = 0.
OPTAB(92)  = 1.0
OPTAB(95)  = 400.0
OPTAB(96)  = 2.0
OPTAB(97)  = 2.0
OPTAB(110)  = 1.0
OPTAB(121)  = 62.0
```

5.6.8.2 LATHE (cont'd)

```
      OPTAB(122)  = 0.
      OPTAB(123)  = 26.0
      OPTAB(124)  = -9.0
      OPTAB(125)  = 0.
      OPTAB(126)  = 0.
      OPTAB(129)  = 1.0
      OPTAB(132)  = 0.
      OPTAB(137)  = -0.25
      OPTAB(145)  = -2.0
      OPTAB(164)  = 1.0
      OPTAB(172)  = 1.0
      OPTAB(174)  = 4.0
      SRTAB(1)  = 4.0
      SRTAB(2)  = 130.0
      SRTAB(3)  = 18.0
      SRTAB(4)  = 600.0
      RETURN
      END
```

5.6.8.3 THREE-AXIS MILL

```
           SUBROUTINE MACH23
    (SAMPLE MACHINE SUBROUTINE FOR A MILL)
    *GECOM* LABELED COMMON FOR OVERLAY *GEMON*
    *GECBAS* LABELED COMMON FOR OVERLAY *GEBASE*
           DIMENSION TNAME(2)
           DATA TNAME/6HSAMPLE,6H①MILL①/
           DO 1 I=1,2
         1 TAG(I)=TNAME(I)
           OPTAB(4)  = 99.9999
           OPTAB(11)  = 0.0
           OPTAB(13)  = 0.0025
           OPTAB(17)  = 112.0
           OPTAB(25)  = 60.0
           OPTAB(26)  = 1.0
           OPTAB(36)  = 0.
           OPTAB(38)  = 1.0
           OPTAB(39)  = 60.0
           OPTAB(42)  = 60.0
           OPTAB(43)  = 60.0
           OPTAB(44)  = 60.0
           OPTAB(48)  = 0.0
           OPTAB(69)  = 0.15
           OPTAB(70)  = 1.20
           OPTAB(71)  = 0.00286
           OPTAB(72)  = 0.00615
           OPTAB(73)  = 28.0
           OPTAB(74)  = 145.0
           OPTAB(132)  = 1.0
           OPTAB(164)  = 1.0
           TABLEG(9)  = 8.0
           TABLEG(10)  = 9.0
           TABLEG(11)  = 10.0
           TABLEG(12)  = 11.0
           TABLEG(21)  = 20.0
           TABLEG(22)  = 21.0
           TABLEG(31)  = 30.0
           TABLEG(32)  = 31.0
           TABLEM(3)  = DMBITS
           REGFOR(1)  = 30.0
           REGFOR(2)  = 20.0
           REGFOR(3)  = -24.0
           REGFOR(4)  = -24.0
           REGFOR(5)  = -24.0
           REGFOR(6)  = -6.0
           REGFOR(8)  = 24.0
           REGFOR(9)  = 24.0
```

5.6.8.3 THREE-AXIS MILL (cont'd)

```
REGFOR(10)  = 24.0
REGFOR(12)  = 0.0
REGFOR(13)  = 0.0
RETURN
END
```

5.6.8.4 MULTIAXIS MILL

```
   SUBROUTINE MACH40
(Sample Machine Subroutine For A Multiaxis Mill)
*GECOM*   LABELED COMMON FOR OVERLAY *GEMON*
*GECBAS*  LABELED COMMON FOR OVERLAY *GEBASE*
   DIMENSION TNAME(4), TABA(131)

   DATA TNAME/6HSAMPLE,6H ① MULTI,6HAXIS ① M,6HILL ③ /
   DATA WA/6HA ⑤ /
   DATA WC/6HC ⑤ /
   DATA WH/6HH ⑤ /

  1    /10.0,350.0,28.3,991.0,40.0,1400.0,113.0,
  23968.,101.,1.,12.8333333,4.,11.333333,10.,12.,14.,
  316.,18.,20.,22.,24.,26.,28.,30.,32.,34.,36.,38.,
  440.,42.,44.,46.,48.,50.,52.,54.,56.,58.,60.,
  562.,64.,66.,68.,70.,72.,74.,76.,78.,80.,
  682.,84.,86.,88.,90.,92.,94.,96.,98.,100.,
  7104.,108.,112.,116.,120.,124.,128.,132.,136.,140.,
  8144.,148.,152.,156.,160.,164.,168.,172.,176.,180.,
  9184.,188.,192.,196.,200.,204.,208.,212.,216.,220.,
  1224.,228.,232.,236.,240.,244.,248.,252.,256.,260.,
  2264.,268.,272.,276.,280.,284.,288.,292.,296.,300.,
  3310.,320.,330.,340.,350./


   TABLEG(5) = DMBITS
   TABLEG(12)  = 11.0
   TABLEG(24)  = 23.0
   TABLEG(25)  = 24.0
   TABLEG(26)  = 25.0
   TABLEG(27)  = 26.0
   TABLEG(28)  = 27.0
   TABLEG(29)  = 28.0
   TABLEG(89)  = 50.0
   TABLEG(90)  = 59.0
   TABLEM(4)  = DMBITS
   TABLEM(5)  = DMBITS
   TABLEM(72)  = 3.0
   TABLEM(71)  = 4.0
   TABLEM(102)  = 7.0
   TABLEM(103)  = 8.0
   TABLEM(22)  = 10.0
   TABLEM(23)  = 11.0
   TABLEM(74)  = 17.0
   TABLEM(73)  = 18.0
   TABLEM(69)  = 19.0
```

5.6.8.4 MULTIAXIS MILL (cont'd)

```
TABLEM(13) = 36.0
TABLEM(14) = 37.0
REGFOR(6) = -14.0
REGFOR(7) = -14.0
REGFOR(8) = 0.
REGFOR(9) = 0.
REGFOR(10) = 0.
REGFOR(11) = 31.0
REGFOR(13) = 50.0
REGFOR(17) = 10.0
REGSTR(6) = WA
REGSTR(7) = WC
OPTAB(7) = 2.0
OPTAB(9) = 0.
OPTAB(10) = -1.0
OPTAB(13) = .001875
OPTAB(17) = 12.0
OPTAB(19) = 13.0
OPTAB(24) = 999.9
OPTAB(26) = 1.0
OPTAB(27) = 10.0
OPTAB(28) = 0.
OPTAB(30) = 1.0
OPTAB(33) = 3.0
OPTAB(45) = 999.9
OPTAB(48) = .0001
OPTAB(49) = 0.1
OPTAB(57) = 11.0
OPTAB(69) = 0.1
OPTAB(80) = 20.25
OPTAB(90) = 0.
OPTAB(95) = 400.0
OPTAB(98) = 200.0
OPTAB(100) = 10.0
OPTAB(101) = 0.
OPTAB(102) = 3.0
OPTAB(103) = 8.5
OPTAB(104) = 4.0
OPTAB(105) = 9.5
OPTAB(110) = 2.0
OPTAB(112) = 16.0
OPTAB(113) = 0.0001
OPTAB(114) = 2.0
OPTAB(116) = 1.0
OPTAB(118) = -1.0
OPTAB(120) = 1.0
```

5.6.8.4 MULTIAXIS MILL (cont'd)

```
   OPTAB(128)  =  12.0
   OPTAB(129)  =  1.0
   OPTAB(132)  =  1.0
   OPTAB(133)  =  1.0
   OPTAB(145)  =  0.
   OPTAB(164)  =  2.0
     DO 6 I=1,114
   6 SRTAB(I)=TABA(I)
     SRTAB(180)  =  9.0
   RETURN
   END
```

The MACFUN portion of the Machine Subroutine is shown here as a separate split-off subroutine.

```
   SUBROUTINE MACF40
(Sample MACFUN For A Multiaxis Mill)
*GECOM*LABELED COMMON FOR OVERLAY *GEMON*
*GECAS* LABELED COMMON FOR OVERLAY *GEBASE*
       DATA FORK/0.0/
       DATA(ASFTAB(I) ,I=1,10)
      1/1.0,1.40,1.95,2.75,3.85,5.40,7.50,10.6,14.8,20.7/
       IF(MCHCON.GT.0 and.MCHCON.LT.9) GO TO 30
  8    RETURN = 0.
  9    MCHCON = 0.
 10    RETURN
       (TEST FOR SPECIAL MACHINE FUNCTIONS)
 30    GO TO(100,200,8,8,8,600,700,800) ,MCHCON
       (TOOLNO SEQUENCE)
100    IF ICLDAT(6).EQ.7)GO TO 140
130    GRIP = 0.
       GO TO 150
140    GRIP = 1.0
150    IF(FORK.EQ.0.) GO TO 170
160    TOLSLC = TOOL
       GRPSLC = GRIP
       SLTOLN = TOOLEN
       GO TO 171
170    TOLLOD = TOOL
       GRPLOD = GRIP
       TOLDLN = TOOLEN
       FORK = 1.0
171    DBFSEG(13)  = DMBITS
172    RETURN = 1.0
       GO TO 9
       (LIMIT TESTING)
200    RETURN = 0.
       IF(OPTAB(110).EQ.0.)GO TO 8
```

5.6.8.4 MULTIAXIS MILL (cont'd)

```
201     IF(DBFSEG(15).EQ.2.0) GO TO 331
205     IF(TLHEAD.NE.0.) GO TO 220
210     XLOW = -19.0
        XHIGH = 29.0
        GO TO 230
220     XLOW = -29.0
        XHIGH = 19.0
230     IF(DPRESM(1).LT.XLOW) GO TO 235
231     IF(DPRESM(1).LE.XHIGH) GO TO 260
235     RETURN = 1.0
        IF(OPTAB(110).GT.0.) GO TO 260
240     ERROR = 101.0
        CALL ERDMP1
260     IF(DPRESM(2).GE.0.) GO TO 270
         (OVERCENTER CUTTING)
261     DPRESM(2)  = -DPRESM(2)
        DPRESM(1)  = -DPRESM(1)
        DPRESM(4)  = -DPRESM(4)
        DPRESM(5)  = DPRESM(5)  - 50.0
        IF(DPRESM(5).LT.0.) DPRESM(5)  = DPRESM(5)  + 100.0
263     IF(DPRESM(1).LT.XLOW) GO TO 265
264     IF(DPRESM(1).LE.XHIGH) GO TO 266
265     RETURN =  1.0
        IF(OPTAB(110).GT.0.) GO TO 320
        GO TO 240
266     RADLIN = -1.0
        ANGLIN = 0.
270     IF(DPRESM(2).GE.6.0) GO TO 290
280     IF(DPRESM(3).LT.19.0) GO TO 335
        GO TO 330
290     IF(DPRESM(2).GE.30.0) GO TO 310
300     IF(DPRESM(3).LT.11.75) GO TO 335
        GO TO 330
310     IF(DPRESM(2).GT.48.0) GO TO 265
320     IF DPRESM(3).LT.5.625) GO TO 336
330     IF(DPRESM(3).LE.53.625) GO TO 331
336     RETURN = 1.0
        IF(OPTAB(110).GT.0.) GO TO 9
        GO TO 240
331     IF(DPRESM(4))332,9,337
332     IF(DPRESM(4).GT.(-91.666666))GO TO 9
334     IF  (DPRESM(4).LT.(-33.333333))GO TO 336
        GO TO 9
337     IF(DPRESM(4).LE.8.3333333)GO TO 9
338     IF(DPRESM(4).GE.66.666666)GO TO 9
        (LOAD SEQUENCE-RETURN TOOL TO CHANGE POINT)
```

5.6.8.4 MULTIAXIS MILL (cont'd)

```
600     DPRESM(2)  =  48.0
        DPRESM(3)  =  48.75
        DPRESM(4)  =  -25.0
        IF(TLHEAD.EQ.0.) GO TO 620
610     DPRESM(1)  =  -29.0
        GO TO 8
620     DPRESM(1)  =  -19.0
        GO TO 8
        (CYCLE SEQUENCE)
700     IF(DATACL(7).NE.0.)CYFED = DATACL(7)
720     IF(ICYTYP)8,730,740
730     TEMP1 = 51.0
        GO TO 760
740     IF(ICYTYP-8)741,765,8
741     GO TO  (8,8,750,8,8,8,765) ,.ICYTYP
750     TEMP1 = 55.0
760     DBFSEG(2)  =  TEMP1 + DATACL(6)/2.0
765     IF(CYFED.GT.ASFTAB(1))  GO TO 780
770     ROW = 1.0
        GO TO 790
780     DO 785 I=2,10
        ROW = I - 1
        IF(CYFED-ASFTAB(I))790,791,785
785     CONTINUE
        ROW = 10.0
790     DBFSEG(17)  = ROW-1.0
        CODE = -4.0
        CALL OUTPUT
        RETURN =  1.0
        GO TO 9
791     ROW = I
        GO TO 790
        (FROM POINT TESTING)
800     IF  (TLHEAD.NE.0.)  GO TO 820
810     IF  (ABS(            + 19.0).GT.EPSLON)GO TO 870
        GO TO 830
820     IF(ABS(DPRESM(1)  + 29.0).GT.EPSLON)  GO TO 8
830     IF(ABS(DPRESM(2)  - 48.0).GT.EPSLON)  GO TO 8
840     IF(ABS(DPRESM(3)  - 31.75).GT.EPSLON)  GO TO 8
850     IF(ABS(DPRESM(4)  + 25.0).GT.EPSLON)  GO TO 8
860     IF  (ABS(DPRESM(4)  - 25.0).LE.EPSLON)  GO TO 8
        (Print Comment that *FROM* point is not the home position)
        GO TO 8
        END
```

5.6.8.5 MULTIHEAD MACHINE (LATHE)

```
 SUBROUTINE MACH51
 (Sample Machine Subroutine For A Multihead Machine)
 (APT SYSTEM COMMON)
 *GECOM LABELED COMMON FOR OVERLAY *GEMON*
 *GECbAS LABELED COMMON FOR OVERLAY *GEBASE*
 DIMENSION TNAME(4)
 DIMENSION TABA(100),HED1(20),HED2(20)
DATA TNAME/6HSAMPLE,6H ① MULTI,6HHEAD ① M,6HACHINE/
1DATA HED1/6HN      ,6HG      ,6HX      ,6H       ,6HZ      ,6H       ,
26H        ,6HI     ,6H       ,6HK      ,6HF      ,6HS      ,6HT      ,
36HM       ,6H      ,6H       ,6H       ,6H       ,6H       ,6H       /
 DATA HED2/6H       ,6HG      ,6HU      ,6H       ,6HW      ,6H       ,
 16H       ,6HH     ,6H       ,6HJ      ,6HE      ,6HS      ,6HT      ,
 26HM      ,6H      ,6H       ,6H       ,6H       ,6H       ,6H       ,
 30.0,20.0,-33.0,0.,-33.0,0.,0.,33.,0.,33.,40.,20.0,40.,20.,6*0.0/
 DATA TABA/1.0,2.0,3.0,4.0,5.0,6.0,7.0,8.0,9.0,10.0,
112.0,14.0,16.0,18.0,20.0,22.0,24.0,26.0,28.0,30.0,
234.0,38.0,42.0,46.0,50.0,54.0,58.0,62.0,66.0,70.0,
376.0,82.0,88.0,96.0,102.0,108.0,114.0,120.0,126.0,132.0,
4140.0,148.0,156.0,164.0,172.0,180.0,188.0,196.0,204.0,212.0,
5220.0,228.0,236.0,244.0,252.0,260.0,268.0,276.0,284.0,292.0,
6300.0,310.0,320.0,330.0,340.0,350.0,360.0,370.0,380.0,390.0,
7400.0,410.0,420.0,430.0,440.0,450.0,460.0,470.0,480.0,490.0,
8500.0,515.0,530.0,545.0,560.0,575.0,590.0,605.0,620.0,635.0,
9650.0,665.0,680.0,695.0,720.0,735.0,750.0,765.0,780.0,795.0,
    DO 4 I=1,4
   4    TAG(I) = TNAME(I)
    TABLEG(11) = 10.0
    TABLEG(12) = 11.0
    TABLEG(16) = 18.0
    TABLEG(17) = 15.0
    TABLEG(21) = 20.0
    TABLEG(22) = 21.0
    TABLEG(31) = 30.0
    TABLEG(11) = 10.0
    TABLEG(12) = 11.0
    TABLEG(16) = 18.0
    TABLEG(17) = 15.0
    TABLEG(21) = 20.0
    TABLEG(22) = 21.0
    TABLEG(31) = 30.0
    TABLEG(32) = 31.0
    TABLEM(42) = 40.0
    TABLEM(43) = 41.0
    TABLEM(142)  = 44.0
    TABLEM(143)  = 45.0
```

5.6.8.5 MULTIHEAD MACHINE (LATHE) (cont'd)

```
   TABLEM(111)  = 32.0
   TABLEM(112)  = 33.0
   REGFOR(3)  = -33.0
   REGFOR(4)  = 0.
   REGFOR(5)  = -33.0
   REGFOR(8)  = 33.0
   REFOR(9)  = 0.
   REGFOR(10)  = 33.0
   REGFOR(11)  = 40.0
   REGFOR(12)  = 20.0
   REGFOR(13)  = 40.0
   DO 470 KM=1,17
470    REGSTR(KM)  = HED1(KM)
   OPTAB(4)  = 999.9999
   OPTAB(7)  =  1.0
   OPTAB(8)  = 100.0
   OPTAB(16)  = 0.0
   OPTAB(19)  = 2.0
   OPTAB(22)  =  101.0
   OPTAB(31)  = 1.0
   OPTAB(37)  = 35.0
   OPTAB(38)  = 1.0
   OPTAB(39)  = 152.0
   OPTAB(42)  = 1520.0
   OPTAB(46)  = -1.0
(Option 59 is +X +Z)
   OPTAB(59)  - 131500.0
(Option 60 is +I +K)
   OPTAB(60)  = 161800.0
   OPTAB(81)  = 3.0
   OPTAB(82)  = 0.5
   OPTAB(83)  = 0.0
   OPTAB(95)  = 400.0
   OPTAB(132)  = 0.
   OPTAB(139)  = 1.0
   OPTAB(142)  =100.0
   OPTAB(151)  = 0.1
   OPTAB(152)  = 0.0
   OPTAB(156)  =  1.0
   OPTAB(164)  = 3.0
   DO 475 KM = 1,100
475    SRTAB(KM)  = TABA(KM)
(Write the Head 2 REGSTR and REGFOR tables onto TAPES1)
   CALL WEFREW (TAPES1,IND,1)
   CALL GMWRIT (TAPES1,IND,INT1,INT1,HED2(1),20)
   CALL GMWRIT (TAPES1,IND,INT2,INT1,HED2(21),20)
   RETURN
   END
```

5.7 ERROR DIAGNOSTICS AND WARNING COMMENTS

In order to facilitate computer usage, the GECENT postprocessor has very few error diagnostics which result in program cessation, otherwise known as a "fatal error or abort". More commonly the postprocessor prints an error comment which flags the error condition and proceeds with the rest of the program. In this way a part programmer can usually detect all of the existing errors in a program in one computer run.

Generally speaking, a fatal error results when a condition arises which prevents further processing, e.g., a failure to read the CL tape. Other fatal errors are when the postprocessor detects conditions to be so incompatible that further processing is pointless, e.g., looping or convergence failure in linearity testing.

When a non-fatal error is encountered, the postprocessor prints a warning comment to this effect and continues processing. For example, if the initial SPINDL statement does not give a spindle speed the postprocessor assumes one (option 27) and then prints the comment, OPTION VALUE IS ASSUMED FOR THE SPINDLE SPEED.

The warning comments are set up and issued from the source subroutine; in the above example, subroutine SPINDL issues the comment. The technique involves setting up a warning comment in BCD form through a labeled data statement in the source subroutine. The label or address of the DATA statement is given in the calling sequence to subroutine COMENT which outputs the BCD statement in a row of DBFSEG with a CODE = 9. Because of the DBFSEG limitation, a comment must not have more than 14 words.See subroutines SPINDLE and COMENT for examples of this technique.

When a fatal error is encountered, the postprocessor stores the related error number into the parameter ERROR (in GECOM) and calls subroutine ERDMP1. This subroutine prints the error number (see Section 5.7.1), and then calls the Section 0 subroutine PDUMP to obtain a core dump for debugging. The overlay GEDUMP is then called into memory to print a comment which better defines the fatal error, and to print the input arrays ICLDAT and DATACL in interger and floating point forms, respectively. The parameters of labelled common are then printed by a NAMELIST dump.

A suggested procedure for error debugging is:

5.7 ERROR DIAGNOSTICS AND WARNING COMMENTS (cont'd)

(A) Determine the error and its subroutine location; this is done by finding the given error number in the error list.

(B) Check the parameter SEQCTR to determine which CL tape record is being processed.

(C) Check the COMMON storages DPREVP, DPRESP, ICLDAT and DATACL to determine the program condition when the error occurred; compare values with CL tape listing.

(D) Depending upon which subroutine is involved, check all relevant parameters used in the subroutine.

(E) The part program listing should be checked for erroneous programming.

(F) If the trouble is yet not evident, the computer programmer should follow the postprocessor flow from the point where the CL tape is read, all the way to the point where the error occurred. The various parameters can be checked for accuracy by comparing their values according to the parameter definition (Section 5.1.1). Section 5.3 and 5.4 may prove to be of benefit as well.

5.7.1 FATAL ERRORS

The following list represents the fatal errors that might occur in the GECENT III postprocessor.

Error Number	Subroutine	Reason
1	GEBASE	Cannot read the CL tape
7	DECODE	Option 59 or 60 is set up incorrectly.
14	MOTION	RETURN flag from subroutine GOLINE is incorrect
19	FROM5	$\left\|\sqrt{I^2+J^2+K^2}-1\right\| > \varepsilon$
24	SPINDL	Option 27 says to assume an error when no initial spindle speed is given.
25	LINRTY	RETURN flag from subroutine DEPART is incorrect.
39	SELGRO	Rotary departure is too large; it should have been segmented.
41	TURRET	The turret corrective move is much too large; the turret tool offsets are probably wrong.

5.7.1 FATAL ERRORS (cont'd)

54	IDPART	L indicates an illegal BCD character was found.
138	SELG	No G code is available for the segment length.
139	SELG	Segment length is too long; it should have been broken up. A reason can be that because of some error, DPRESM will equal DPREVM, thus giving a zero movement in subroutine DEPART; subroutine DEPART will then give the signal for a zero motion, i.e., RETURN = -1, and hence, DBFSEG(3-5) will remain as DMBITS. Another possibility is that the cutter may be going across the part center line when in an SFM mode.
142	SELGCR	Circle radius is too large; it must be < DEPMAX.
144	SELGCR	No G code is available for the circle radius size.
151	PROCQD	DBUFER is DMBITS which indicates that subroutine QUADET did not set it up properly.
999	INIT	The requested MACHIN number is larger than the computed GOTO list permits.
1020	GEMULT	New n value for code 17 is less than previous n value.

5.7.1 FATAL ERRORS (cont'd)

1021	GEMULT	Feedrate for code 17 is not acceptable.
1022	GEMULT	Feedrate for code 17 is not acceptable.
1023	GEMULT	ICODE is not acceptable for Head 1.
1024	GEMULT	n value or feedrate of code 17 is not acceptable on Head 1.
1025	GEMULT	ICODE is not acceptable for Head 2.
1026	GEMULT	n value or feedrate of code 17 is not acceptable on Head 2.
1027	GEMULT	ICODE or JCODE not acceptable.
1028	GEMULT	Illegal output.
3000	GEPR01	CODE = 17; GEOUT1 cannot be used for a multihead printout.
7000	GMSTOR	Tape read error.
7001	WEFREW	Error opening tape for reading.
7002	WEFREW	Error opening tape for writing.
7003	GEMULT	Only one head was selected.
8000	FUNLNK	Option 132 calls for an overlay which is not available.
8500	OUTPUT	CODE>18.
9000	INIT	CL tape read error or EOF; if EOF is encountered, no MACHIN statement was given.
9001	OUTPUT	IHEAD flag was not set up properly.

6.0 SUBROUTINE DESCRIPTIONS

This section contains the complete prose description of each subroutine used within the postprocessor. In the previous edition of the postprocessor copious flow charts were drawn for the purpose. Experience has shown that it was not practical to keep abreast of the many changes with revised charts. Consequently, in their stead the prose description replaces the flow chart as the best means of describing the characteristics of each subroutine. These are more readily replaced with updated material.

You will find each subroutine description listed alphabetically in this section. Under each name is the name of the overlay in which the subroutine is used. The following elements are discussed for each subroutine: purpose, input, output, method, diagnostics, requirements, and restrictions.

Section 5.3 contains those few flow charts which are a part of the documentation. They cover the broader aspects of the program, rather than the fine details as was attempted previously. In addition, Section 5.4 contains two very useful indices. The first gives a alphabetical list of subroutines against which are given the names of the subroutines which are called by the listed subroutine. The second is a list of subroutine names paired with the name of the subroutine which calls it. These two indices can be very helpful in following the flow of information through the postprocessor.

In The Subroutine Descriptions, the following nomenclature is utilized:

1. The title is the GE635, UNIVAC 1108, and CDC 6600 name.
2. The title in parenthesis is the IBM 360 name.
3. The subroutine names in the body of the manual are the GE635, UNIVAC 1108, and CDC 6600 names.
4. The parmeter names in the body of the manual are IBM 360 names.

ABSOPR (ABSOPR)
(GEOUT)

PURPOSE

To print the absolute and the operator manuscripts for the GEOUT3 printout.

INPUT

CALL ABSOPR (PRINTH)

where:
PRINTH contains the BCD representation of the word HEAD.

OUTPUT

The absolute printout is on TAPES1 and the operator printout is on TAPES4.

The absolute and operator manuscripts are printed.

METHOD

The absolute printout is printed first, if it is desired. The title and page number are printed at the top of each page. When the number of lines on a page is exceeded, the page number is printed at the bottom of the page and the page is restored. The cutting time and tape footage are printed at the bottom of the last page. If the operator printout is desired, it is printed in the same manner.

DIAGNOSTICS

None

REQUIREMENTS

Called by subroutine GEPRO3

Calls subroutines WEFREW, TITLE3, GEPRN3, GMREAD, TIMES and PAGE

RESTRICTIONS

The APT COMMON is used

AIR (AIRGB)
(GEBASE)

PURPOSE

To establish the M code for an AIR/ON or OFF statement

INPUT

CALL AIR

The input arrays ICLDAT and DATACL are used.

OUTPUT

The proper M code is output in a block by itself.

METHOD

For ON, TABLEM(123) is used.

For OFF, TABLEM(124) is used.

DIAGNOSTICS

None

REQUIREMENTS

Calls subroutine STOREM

Called by subroutine AUXLRY

RESTRICTIONS

None

ARCTAN (ATANGX)
(GEMILL)

PURPOSE

To compute the angle (radians) whose tangent has the X and Y coordinates as specified in the calling sequence

INPUT

CALL ARCTAN (IR,X,Y,ANGLE)

where:

IR = Range number:

= 1 when range is $(-\frac{3\pi}{2}$ to $\frac{\pi}{2})$

= 2 when range is $(-\pi$ to $\pi)$

= 3 when range is $(-\frac{\pi}{2}$ to $\frac{3\pi}{2})$

= 4 when range is $(0$ to $2\pi)$

X = X coordinate of given point

Y = Y coordinate of given point

ANGLE = computed angle

OUTPUT

The output is the computed angle (ANGLE) in radians.

METHOD

The system subroutine ATAN is used to compute the angle. A special check is made to ensure that the angle is in the desired range.

REQUIREMENTS

Called by subroutines CLASS1, CLASS2 and SEGDRC

Calls subroutine ATAN

RESTRICTIONS

Subroutine uses a DATA Statement

ASSIGN (ASGNGE)
(GEMON)

PURPOSE

To assign values to COMMON variables based upon option values

INPUT

CALL ASSIGN

The following parameters and arrays in COMMON are used: OPTAB, TABLEG.

OUTPUT

Flags used: REGFOR, BIGDEP, ROTMAX, OPTAB, STEP, NRINGES, MAXES, LSTPLN, FCOMAX, FRMAX, FRMIN, HSTEP, RHSTEP, BMS, RMS, XYZDEC, EPSLON, ISPTYP, RNGDEP, SYSCON, TLEAD, FACDEP, CMULT, ICLMOD, TMAX, ROTRAP, ROTFMN, ROTFMX, INDFR, ROTUNT, SPNDIR, FRMOD, SEQLIM, MULTHD, IADRET

METHOD

Option values are tested and interpreted to determine machine limits, defaults, initial values, and standards.

DIAGNOSTICS

None

REQUIREMENTS

Calls subroutine CONROT

Called by subroutine INIT

RESTRICTIONS

None

AUXFUN (AUXFGB)
(GEBASE)

PURPOSE

To process the M code for the auxilary function statement AUXFUN

INPUT

CALL AUXFUN

The input array ICLDAT and DATACL are used.

OUTPUT

The M code is output either in a block by itself or is merged with the next block.

See METHOD below.

METHOD

Any pending M code is forced out from VALUEM.

DATACL(4) is used as the next M code.

If option 75 is zero, the M code is forced out in a block by itself. Otherwise, it is merged into the next command block.

DIAGNOSTICS

None

REQUIREMENTS

Calls subroutines STOREM, OUTPUT

Called by subroutine AUXLRY

RESTRICTIONS

None

AUXLRY (AUXLGB)
(GEBASE)

PURPOSE

To select the appropriate subroutine for processing the given APT 2000 type statement

INPUT

CALL AUXLRY

OUTPUT

The flags used: ENDFLG, AXMULT

The result is a call to a subroutine such as subroutine FEDRAT, COOLNT or SPINDL as determined by a computed GOTO statement.

METHOD

A branch is made to determine which subroutine to call for processing the Cutter Location File (CLFILE) record subtype depending on the contents of ICLDAT(3).

DIAGNOSTICS

None

REQUIREMENTS

Called by subroutine GEBASE

Calls subroutines AIR, END, SET, LOAD, STOP, WELD, BREAK, CLAMP, DELAY, DRAFT, DRESS, FLAME, PITCH, PPTOL, RAPID, RESET, SEQNO, TMARK, TRANS, AUXFUN, CLRSEF, COMBIN, COMENT, COOLNT, COUPLE, CUTCOM, CYCLEL, CYCLGP, CYCLGX, FEDRAT, GOHOME, INSERT, LEADER, MACHIN, MACHTL, MCHFIN, OPCODE, OPSKIP, OPSTOP, ORIGIN, OVRCNT, PARTNO, PICKUP, PITCHM, PIVPLN, POSITN, PPRINT, PREFVN, PREFSEQ, RETRCT, REWIND, ROTATE, SAFEGL, SAFEGM, SAFEGX, SELECT, SELHED, SPINDL, THREAD, THREDM, TOOLNO, TURRET, UNLOAD, and XOFSET

RESTRICTIONS

None

BREAK (BRAKGB)
(GEBASE)

PURPOSE

To break the punched tape at a specified length.

INPUT

CALL BREAK
This subroutine is a multiple entry in STOP.

OUTPUT

Flags used: STOPON, CURNGE, ISRNGE, FLONKL, FLONSP, and IENTRY.

The STOP M code is output in a block by itself.

DIAGNOSTICS

None

REQUIREMENTS

Called by subroutine AUXLRY.
Calls subroutines COMENT, ENTRAP, OUTPUT and STOREM.

RESTRICTIONS

None

CALCP1 (CALCGO)
(GEOUT1)

PURPOSE

To format the spacing for printed output of the various registers for GEOUT1

INPUT

CALL CALCP1

The following parameters and arrays in common are used: OPTAB, REGFOR and REGSTR.

OUTPUT

The following print arrays are set for the NC machine printout: LASTFP, NIP, NFP, NPR, NPT, BCDREG

METHOD

Based on the formats of each register and the total number of columns available, the initial and final print positions, the number of decimal places as well as the total number of positions required, are calculated for each register.

DIAGNOSTICS

Comment: GEOUT1 CANNOT BE USED.
(GEOUT2 is used in this case)

REQUIREMENTS

Calls subroutines CALCP2, SETLIN

Called by subroutine GEPRE.

RESTRICTIONS

The subroutine contains a Data Statement.

CALCP2 (CALCP2)
(GEOUT)

PURPOSE

To set up the printing and spacing format for GEOUT2.

INPUT

CALL CALCP2

The following parameters and arrays in COMMON are used: REGFOR, OBTAB, REGSTR.

OUTPUT

Flags used: REGFOR, NIPA, NPTA, NIP, NFP, NPR, NPT, BCDREG

The print vectors are set up with the column data needed for printing a line.

METHOD

By examining the formats (REGFOR) for the various registers, this subroutine calculates spacing and the initial and final print positions for each register, the number of digits in the register, and the number of places to the right of the decimal point.

DIAGNOSTICS

None

REQUIREMENTS

Calls subroutine SETLIN

Called by subroutines CALCP1, GEPRE

RESTRICTIONS

Uses a Data Statement

CALCP3 (CALCP3) (GEOUT3)

PURPOSE

To set up the spacing for GEOUT3 and to compute the initial and final print positions

INPUT

CALL CALCP3 (NIPAT, NPTAT, NIPT, NFPT, NPRT, NPTT, FORMT, WORDST, BCDRG)

where:

NIPAT—initial print position for Absolute Printout

NPTAT—final print position for Absolute Printout

NIPT—initial print position for Incremental Printout

NFPT—final print position for Incremental Printout

NPRT—number of places to right of decimal

NPTT—total number of digits in register

FORMT—the REGFOR values

WORDST—the REGSTR values

BCDRG—the array for setting up the BCD output image

OUTPUT

The print vectors are set up for printing the output in columnar form in accordance with the REGFOR table.

DIAGNOSTICS

None

REQUIREMENTS

Called by subroutine GEPRE

Calls subroutine SETLIN

RESTRICTIONS

None

CHARID (CHARGO) (GEOUT)

PURPOSE

To return with the BCD value of the leftmost character of WORD stored in L

INPUT

CALL CHARID (WORD,L)

where:

WORD is the BCD representation.

L is a code.

OUTPUT

L contains the BCD, code number of the left most character of WORD. Prior to returning from CHARID, WORD is shifted one character to the left.

DIAGNOSTICS

None

REQUIREMENTS

Called by subroutine IDPART

Calls no subroutines

RESTRICTION

This is a machine language coded subroutine.

CHKAX (CHKXGT) (GETERP)

PURPOSE

To determine if a circle segment lies in a given plane by checking the non-planar axis for consistency

INPUT

CALL CHKAX

The following parameters and array in COMMON are used: IPLANE, NCOM, NOPTS, DATACL, EPSLON.

OUTPUT

Flags used:

RETURN =	1:	Circle lies in given plane
	−1:	Circle does not lie in given plane

METHOD

If points corresponding to the constant axis differ by more than EPSLON, the circle does not lie in the plane.

DIAGNOSTICS

None

REQUIREMENTS

Calls no subroutines

Called by subroutine GOCIRC

RESTRICTIONS

None

CIRINT (CRINGT) (GETERP)

PURPOSE

To communicate to subroutine QUADET the plane of circular interpolation by decoding parameter IPLANE

INPUT

CALL CIRINT

The following parameters and arrays in COMMON are used: IPLANE

OUTPUT

Calls subroutine QUADET indicating in which plane the circular interpolation is.

METHOD

IPLANE:	PLANE:
> 1 ——>	XZ
= 1 ——>	ZX
< 1 ——>	XY

DIAGNOSTICS

None

REQUIREMENTS

Calls subroutine QUADET

Called by subroutine GOCIRC

RESTRICTIONS

None

CIRSEG (CIRSEG) (GEMULT)

PURPOSE

To segment a circle into two parts and to compute the two G codes for the two new segments.

INPUT

CALL CIRSEG (RATIO, AA, KCODE, RADIUS, ARCLEN, ABL)

where RATIO = the percent value defining how circle will be segmented.

AA = buffer where input data is stored, and where the segmented circle will be stored.

KCODE = the code to identify the block. The values 10, 11, or 12 indicates a circle.

ARCLEN = arc length of the circle before segmentation.

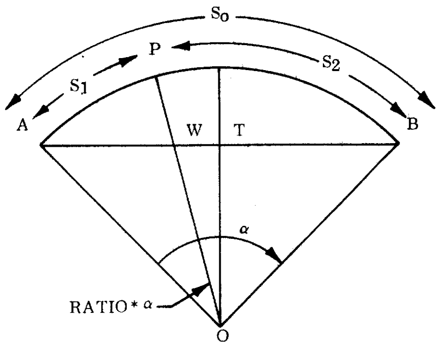
ABL = chord length of circle before segmentation.

OUTPUT

The output, buffer AA, consists of a circle segmented into two parts with the calculated g code for each segment.

METHOD

The following illustration and explanation shows the method for segmenting the circle.



Given: Vectors $\overline{AB}$(ABL), $\overline{AO}$(RADIUS), length S_0 (ARCLEN), and the percent value (RATIO $= S_1/S_0$).

Find: Vectors $\overline{AP}$, $\overline{PB}$ and $\overline{PO}$.

Solution:

$$|AO| = R = \text{circle radius}$$
$$= S^0/R$$

$$|AT| = R \sin \frac{\propto}{2}$$

$$|OT| = R \cos \frac{\propto}{2}$$

$$|WT| = |OT| \tan \left(\frac{\propto}{2} - \text{RATIO}\cdot\propto\right)$$

$$|AW| = |AT| - |WT|$$
$$AW = \frac{|AW|\cdot\overline{AB}}{|\overline{AB}|}$$
$$OW = \overline{AW} - \overline{AO}$$

$$OP = \frac{R}{|OW|}\cdot\overline{OW}$$

$$AP = \overline{AO} + \overline{OP}$$
$$PB = \overline{AB} - \overline{AP}$$
$$PO = -\overline{OP}$$

DIAGNOSTICS

None

REQUIREMENTS

Calls CONVRT, LENGTH, SELGCD, SRAREC, and STOPTS.

Called by SPLIT.

RESTRICTIONS

None

CLAMP (CLMPGB) (GEBASE)

PURPOSE

To process the sequence for the statement CLAMP

INPUT

CALL CLAMP

OUTPUT

The flags used are: CLMPEX and CLMPFL. Output is an M code for the specified clamp condition.

DIAGNOSTICS

Comment: REQUESTED MISCELLANEOUS FUNCTION CODE IS NOT AVAILABLE ON THIS MACHINE.

REQUIREMENTS

Called by AUXLRY

Calls subroutines COMENT, STOREM

RESTRICTIONS

None

CLASS (IBM 360 only) (CLASS)

PURPOSE

To call in the desired class equations.

INPUT

CALL CLASS

The common array OPTAB is used.

OUTPUT

Calls in the desired class equations.

DIAGNOSTICS

Comment: ILLEGAL CLASS EQUATION CALLED FROM SUBROUTINE GECLASS—SUBROUTINE CLASON WAS CALLED IN ERROR.

REQUIREMENTS

Calls subroutines CLAS01, CLAS02, CLAS03, CLAS04, CLAS05, CLAS06, CLAS07, CLAS08, CLAS09.

Called by subroutines GEOM5, LINRTY, LOAD, ROTABI, and SELTUL.

RESTRICTIONS

Used on the IBM 360 only.

CLAS0n (CLAS0n) (CLASS)

PURPOSE

To compute the direct and inverse transform equations for the Classes of NC machine axis configurations, where n currently ranges from 1 to 9.

INPUT

CALL CLAS0n

OUTPUT

The only flag used is MAFORK.

The output is the current position of the tool tip in machine coordinates and part coordinates.

METHOD

See Sections 4.2.1 through 4.2.9 for the Class equations for each multiaxis machine configuration.

The inverse transforms (x,y,z,i,j,k) are computed from the present machine point coordinates.

The direct transforms (x,y,z,a,b,c) are computed from the present part coordinates.

DIAGNOSTICS

None

REQUIREMENTS

Called by subroutine CLASS

Calls subroutine ARCTAN

RESTRICTIONS

These equations are restricted to a machine of Class n configuration.

CLRSRF (CLRSGP) (GEBASE)

PURPOSE

To establish a clearance surface retract plane as given by the APT statement CLRSRF

INPUT

Call CLRSRF

The input arrays ICLDAT and DATACL are used.

OUTPUT

The flag CLERP contains the plane value.

DIAGNOSTICS

None

REQUIREMENTS

Calls no subroutines

Called by subroutine AUXLRY

RESTRICTIONS

None

COMBIN (COMBIN) (GEBASE)

PURPOSE

To process the APT statement COMBIN

INPUT

CALL COMBIN

The key flag MULTHD is also used.

OUTPUT

The flag MULTHD is set equal to the value given in the COMBIN statement.

DIAGNOSTICS

None

REQUIREMENTS

Calls no subroutines

Called by subroutine AUXLRY

RESTRICTIONS

None

COMENT (CMNTGB) (GEBASE)

PURPOSE

To output the specified warning comment

INPUT

CALL COMENT (FN)

where:

FN = number which designates the comment to be printed.

OUTPUT

Postprocessor comment or warning is printed.

REQUIREMENTS

Called by subroutines AUXLRY, CLAMP, COUPLE, CYCLEL, CYCLGP, CYCLGX, DELAY, DSRROW, EIACOM, END, FEDRAT, FTYPE2, GEBASE, GEMULT, GOCIRC, LINRTY, MOTION, OFFARC, OPCODE, PARK, POSMOV, PRFSEQ, REWIND, ROTABI, ROTATE, ROTDRF, ROTHED, SADDLE, SELTOL, SFMO, SPINDL, STOP, TESTM2, THREAD, THREDM, TOOLNO, TSTLIM, TSTSAF, TURRET, TYPE3, TYPE4, TYPE10, TYPE13, UNLOAD, and WELD.
Calls subroutines OUTPUT and STOREM.
Also calls subroutine GMSTOR on the IBM 360

RESTRICTIONS

This subroutine has multiple entry points for subroutines PPRINT, INSERT, and PARTNO.

COMENT (CMNTGB)
(GEMON)

PURPOSE

To output the postprocessor comments.

INPUT

CALL COMENT (DN)
where DN is the BCD comment.

OUTPUT

The postprocessor comment is output with CODE = 9.0.

METHOD

CODE, VALUEM, SPNCOM and DBFSEG are saved; the comment is stored in DBFSEG and is output; CODE, VALUEM, SPNCOM and DBFSEG are reinstated.

DIAGNOSTICS

None

REQUIREMENTS

Calls subroutines GMSTOR and OUTPUT
Called by subroutines AUXLRY, CLAMP, COUPLE, CYCLEL, CYCLGP, CYCLGX, DELAY, DSRROW, EIACOM, END, FEDRAT, FTYPE2, GEBASE, GEMULT, GOCIRC, LINRTY, MOTION, OFFARC, OPCODE, PARK, POSMOV, PRFSEQ, REWIND, ROTABI, ROTATE, ROTDRF, SADDLE, SELTUL, SFMO, SPINDL, STOP, TESTM2, THREAD, THREDM, TOOLNO, TSTLIM, TSTSAF, TURRET, TYPE03, TYPE04, TYPE10, TYPE13, UNLOAD, WELD.

RESTRICTIONS

Computer dependent for the IBM 360 only.

COMPFC (CMPFGB)
(GETERP)

PURPOSE

To calculate the feedrate command code for a conturing machine

INPUT

CALL COMPFC (A, DIST, FCOM)

where: A = array ordered as X,Y,Z
DIST = linear distance
FCOM = feed command

OUTPUT

The computed feed command is stored in FCOM; also, the linear distance is stored in DIST.

METHOD

The feed command FCOM is determined by:

$$F_{COM} = \frac{GDIMUL \cdot FEDIPM,}{DIST}$$

where:

$DIST = \sqrt{A(1)^2 + A(2)^2 + A(3)^2}$
GDIMUL = dimension multiplier
FEDIPM = current feedrate in IPM

DIAGNOSTICS

None

REQUIREMENTS

Called by subroutine TSTFCM

Calls no subroutines

RESTRICTIONS

OPTAB(170) = 1

COMPGC (COMPGC) (GEMULT)

PURPOSE

To set up the required parameters for subroutine SELGCD

INPUT

CALL COMPGC (BUF, AAA, II)

where:

BUF = the input array, BUFPRE, for subroutines SELGCD

AAA = the array containing incremental data for a particular move

II = the row of AAA to be used

OUTPUT

The computed G code is stored in AAA (2, II).

DIAGNOSTICS

None

REQUIREMENTS

Called by subroutines DRETHD, PARK, RETHD, and SEG

Calls subroutine SELGCD

RESTRICTIONS

None

COMPR (CMPRGT) (GETERP)

PURPOSE

To compute the radius of a circle

INPUT

Call COMPR (CIRPTS)

where:

CIRPTS contains the X,Y,Z values of the circle.

OUTPUT

The parameter CIRRAD contains the circle radius.

METHOD

(Cartesian Metric):

$$R = \sqrt{\Delta_1^2 + \Delta_2^2}$$

where Δ_1 and Δ_2 are axial components of the radius vector R measured in the plane of the circle. The radius is measured from the tool control point to the circle center.

DIAGNOSTICS

None

REQUIREMENTS

Calls no subroutines

Called by subroutine GOCIRC

RESTRICTIONS

Uses a Data Statement

COMTAT (TCODGB) (GEBASE)

PURPOSE

To combine the tool number and turret position into a command code

INPUT

CALL COMTAT

The following parameters and arrays in COMMON are used: OPTAB, RETURN, REGFOR, TOOL, TURPOS, STOPON, TUROFF

OUTPUT

The combined T code is stored in DBFSEG(13).

DIAGNOSTICS

None

REQUIREMENTS

Calls subroutine MACSRT

Called by subroutine TURRET

RESTRICTIONS

None

CONROT (CROTGE) (GEMON)

PURPOSE

To convert from (to) degrees to (from) output units

INPUT

Call CONROT (VALUE, IV)

where:

VALUE = the rotary value to be converted

IV = +1, convert from degrees to output units

IV = −1, convert from output units to degrees

OUTPUT

The parameter VALUE contains the converted value.

METHOD

Nothing is done if the output units are degrees.

DIAGNOSTICS

None

REQUIREMENT

Calls no subroutines

Called by ASSIGN, GEPRO2, POSIT, ROTABA, ROTABI, ROTHED

RESTRICTIONS

Uses a data statement

CONTUR (CONTGO) (GETERP)

PURPOSE

To determine the feedrate command for an incremental motion

INPUT

CALL CONTUR

The following parameters and arrays in COMMON are used: DPATH, DBFSEG, AXMULT, OPTAB, SRTAB, GDIMUL

OUTPUT

Flags set:

DPATH, FCOMIN, DABVAL, FCOM.

METHOD

The feedrate command is computed and stored into DBFSEG(11).

DIAGNOSTICS

None

REQUIREMENTS

Calls subroutines LENGTH, EIACOM, FVARGO

Called by subroutines GEPRO1, GEPRO2, GEPRO3

RESTRICTIONS

None

CONVRT (CVRTGB) (GEBASE)

PURPOSE

To set to zero the elements of a given vector which contain DMBITS.

INPUT

CALL CONVRT (A,B)

where:

A and B are dimensioned at 3. A is the vector that contains the elements to be tested.

OUTPUT

B contains the vector A but has its elements equal to zero when the corresponding elements in A are DMBITS.

DIAGNOSTICS

None

REQUIREMENTS

Calls no subroutines

Called by subroutines CIRSEG, FEDLIM, GETSFC, and SELGCD

RESTRICTIONS

None

COOLNT (COOLGB) (GEBASE)

PURPOSE

To process the APT statement COOLNT/ and set up the appropriate M code for the given coolant ON/OFF condition.

INPUT

CALL COOLNT

The input arrays ICLDAT and DATACL are used.

OUTPUT

The following flags are used: FLONKL, ICIMOD, and STOPON

The proper M code is stored in parameter VALUEM.

DIAGNOSTICS

None

REQUIREMENTS

Called by subroutine AUXLRY

Calls subroutine STOREM(M), where M is the M code value set-up in accordance with the modifiers of the COOLNT/ statement

COUPLE (CUPLGL) (GELATH)

PURPOSE

To engage or disengage the encoder

INPUT

CALL COUPLE

The following parameters and arrays in COMMON are used: SPNSPD, OPTAB, ICLDAT, SRTAB, DATACL

OUTPUT

The coupling M code is output in a block by itself.

METHOD

If the spindle speed is greater than the maximum allowed and the encoder is coupled, the spindle speed is reduced to the range minimum. In such a case, a dwell block is output with the encoder M code.

DIAGNOSTICS

Comment: SPINDLE SPEED COULD NOT BE FOUND.

REQUIREMENTS

Calls subroutines STOREM, SPTYPE, DWELL

Called by subroutine AUXLRY

RESTRICTIONS

None

CREAD . (CREAD) (GEMULT)

PURPOSE

To read a record from either TAPES2 or TAPES3

INPUT

CALL CREAD (IHEAD, IEOF)

where:

IHEAD = 1 means read a record from TAPES2

IHEAD = 2 means read a record from TAPES3

IEOF = 0 signifies a good data record

IEOF = 1 signifies an end of file on given tape

OUTPUT

A record from TAPES2 is stored in AS2 and a record from TAPES3 is stored in AS3. The storage arrays AS2 and AS3 are analogous to DBFSEG. The parameter FDMHD contains the M code (if any) for either the feed or rapid range.

DIAGNOSTICS

Tape read errors are output. See the APT tape read routine for the meaning of the errors.

REQUIREMENTS

Called by subroutine GEMULT

Calls subroutines GMREAD and SAVMCS

RESTRICTIONS

Needs APT COMMON

CTCHUP (CTCHUP) (GEMULT)

PURPOSE

To output the system catch-up time when changing rapid-feed ranges

INPUT

The calling sequence is:

CALL CTCHUP (AAA,IH)

where:

AAA = an array containing the incremental move

IH = 1 if head 1 is being processed

IH = 2 if head 2 is being processed

OUTPUT

A system catch-up dwell block is output by calling subroutine GMOUT.

REQUIREMENTS

Called by subroutines RAPM and SHFTBK

Calls subroutine GMOUT

RESTRICTIONS

None

CUTCOM (CTCMGB)
(GEBASE)

PURPOSE

To process the APT statement CUTCOM.

INPUT

CALL CUTCOM

OUTPUT

The related preparatory function G code is output in a block by itself.

DIAGNOSTICS

None

REQUIREMENTS

Called by subroutine AUXLRY

Calls subroutine OUTPUT

RESTRICTIONS

None

CYCLEL (CYCLEL)
(GELATH)

PURPOSE

To output a command block according to the specifications given in the CYCLE statement for a lathe.

INPUT

CALL CYCLEL

The input arrays ICLDAT and DATACL are used.

OUTPUT

Flags used: CYCFLG, ICYTYP, CURCYG, DBFSEG, DATACL, CODE.

A command block for the specified CYCLE modifier is output.

METHOD

A G code is selected for the type of canned cycle requested, (e.g. BORE, TAP, DRILL, etc.); CODE is set to -16.

DIAGNOSTICS

Comment: IMPROPER FORMAT, STATEMENT SKIPPED.

REQUIREMENTS

Calls subroutines STOREM, RAPIDX, OUTPUT

Called by subroutine AUXLRY

RESTRICTIONS

None

CYCLGP (CYCLGP) (GEPOS)

PURPOSE

To set up the command block as specified by the given CYCLE statement

INPUT

CALL CYCLGP

The following COMMON parameters are used: ICLDAT, DATACL, TABLEG

The CYCLE statement is a 2000 type record. The code numbers for the minor words used in the CYCLE statement are:

Minor Word	Code Number
DRILL	163
FACE	81
BORE	82
TAP	168
THRU	152
DEEP	153
MILL	151
OUT	49
IN	48
INHIBT	279
IPM	73
IPR	74

OUTPUT

The command block is set up in DBFSEG, and the flag ICYTYP is set where:

ICYTYP	=	1	for	FACE
ICYTYP	=	2	for	BORE
ICYTYP	=	3	for	TAP
ICYTYP	=	4	for	THRU
ICYTYP	=	5	for	DEEP
ICYTYP	=	6	for	MILL
ICYTYP	=	7	for	OUT
ICYTYP	=	8	for	IN
ICYTYP	=	9	for	DRILL

METHOD

The CYCLE block is set up thus:

DBFSEG(2)	=	G code
DBFSEG(5)	=	Z (the amount of plunge)
DBFSEG(16)	=	R (distance the tool moves at rapid traverse)

If the R register is not available, the routine simulates the canned CYCLE by outputting the distance R at RAPID in a block by itself.

DIAGNOSTICS

The recoverable error comment 6 is printed when the CYCLE statement is written incorrectly.

Comment: IMPROPER FORMAT, STATEMENT SKIPPED.

REQUIREMENTS

Called by subroutine AUXLRY

Calls Subroutines FLOAT, COMENT, FTYPE2, FTYPE6, MACSRT, OUTPUT, RAPIDP, RAPIDX, STOREM, and TSTEXT

RESTRICTIONS

None

CYCLGX (CYCLGX)
(GMAXES)

PURPOSE

To output a command block according to the specifications given in the CYCLE statement

INPUT

CALL CYCLGX

COMMON parameters used: ICYTYP, TAPSTO and TAPSAV.

OUTPUT

Flags used: FLRPON, CYCFLG, and CURCYG and MCHCON.

METHOD

A command block is issued for the programmed canned cycle.

DIAGNOSTICS

Comment: IMPROPER FORMAT STATEMENT SKIPPED.

REQUIREMENTS

Called by subroutine AUXLRY

Calls subroutines COMENT, MACSRT, and RAPIDX

RESTRICTIONS

None

DECODE (DCODGO)
(GEOUT)

PURPOSE

To set up the shuffle vector for rearranging registers for output according to options 59 and 60

INPUT

CALL DECODE

OUTPUT

Flags used: ISHVEC, ERROR, ISHUFL, IXSTOR.

The words of DBFSEG (3,4,5,8,9,10) have their contents shuffled to the register arrangement specified by options 59 and 60.

METHOD

Common standard axes configurations are first tested; and if not present, X Y Z and I J K are unpacked from options 59 and 60.

The values from the ORIGIN statement are also shuffled similarly.

DIAGNOSTICS

Error 7: Improper setting of option 59

REQUIREMENTS

Calls subroutine ERDMP1

Called by subroutine GEPRE

RESTRICTIONS

The subroutine has a DATA Statement.

DELAY (DLAYGB)
(GEBASE)

PURPOSE

To process the delay statement and output a dwell block.

INPUT

CALL DELAY

Common parameters used are DATACL, SPNSPD, TABLEG and OPTAB.

OUTPUT

A dwell block (CODE = 4) is output if TABLEG(5) has a value. If TABLEG(5) = DMBITS, the preset dwell M code (TABLEM(18)) is output.

METHOD

If the modifier REV is given in the DELAY statement, the dwell time is computed from:

$$\text{TIME} = \frac{\text{DATACL(4)} \cdot 60.0}{\text{SPNSPD}}$$

If REV is not given and option 57 = 11,

$$\text{TIME} = \frac{60.0}{\text{DATACL(4)}}$$

If REV is not given and option 57 ≠ 11,

$$\text{TIME} = \text{DATACL(4)}$$

If TIME = 0, CODE is set to −1.0 and subroutine OUTPUT is called.

If TABLEG(5) = DMBITS, the preset dwell M code (TABLEM(18)) is output in a block by itself.

If TABLEG(5) ≠ DMBITS, DBFSEG(2) is set to TABLEG(5), DBFSEG(3) is set to the dwell time, CODE = 4.0 and subroutine OUTPUT is called.

The DELAY statement may be interrogated in the machine function routine; then the above sequence is bypassed.

DIAGNOSTICS

Comment: DELAY TIME FOR REV NOT CORRECT SINCE NO SPEED IS GIVEN.

REQUIREMENTS

Calls subroutines: COMENT, MACSRT, OUTPUT, and STOREM.

Called by subroutine AUXLRY.

RESTRICTIONS

None

DEPART (DPRTGT)
(GETERP)

PURPOSE

To calculate and output the incremental departures for linear motions and the rotary departures for rotary motions

INPUT

CALL DEPART

OUTPUT

The incremental departures are stored in the COMMON parameters DEPX, DEPY, DEPZ, DEPA, DEPB and into the output buffer (DBFSEG(3-7)) depending on the number of axes available on the machine tool.

METHOD

The departures are determined from the machine points DPRESM(1-3) and DPREVM(1-3) to minimize the rounding error. The number of axes (MAXES) and the multiaxis flag (AXMULT) are tested to determine the number of motions available on the machine tool. If a departure is less than half the step size of the machine (HSTEP), the departure is set to zero for that axis of motion. The flag RETURN is set to +1 for a motion or to −1 if there is no motion.

DIAGNOSTICS

None

REQUIREMENTS

Called by subroutines GOHOME, GOLINE, LINRTY, PROCQD, SFMO

Calls subroutine ROTMOV

RESTRICTIONS

Used for all contouring machines: i.e., OPTAB(1) = 0.0

DETDIR (DDIRGT)
(GETERP)

PURPOSE

To determine the direction of motion (CLW or CCLW) along a circular path

INPUT

Call DETDIR

The following parameters and arrays in COMMON are used: IPLANE, DATACL, CIRDAT.

OUTPUT

The flag CIRDIR is set to:

CIRDIR = 0 for CLW

= 1 for CCLW

METHOD

The circle is translated to the origin. The cross product of the position vector to the first point with the position vector of the second point is calculated. If the cross product is positive, CLW is recognized. If it is negative, CCLW is recognized. If it is zero (first and second point are same), the cross product with the position vector to the next CL point on the circle is examined.

REQUIREMENTS

Calls no subroutines

Called by subroutine GOCIRC

RESTRICTIONS

Uses a Data Statement

DOLLAR (DOLRGO) (GEOUT)

PURPOSE

To insert end-of-block characters at the beginning and end of the readable identification

INPUT

CALL DOLLAR (S,E,EOB)

S = Starting Record

E = Ending Record

EOB = End of block character

OUTPUT

The character in EOB (OPTAB(67)) is placed as the first character of S, and E is set to 77777$.

DIAGNOSTICS

None

REQUIREMENTS

Called by subroutine IDPART

Calls no subroutines

RESTRICTIONS

This is a machine language coded subroutine.

DOTPRO (DOTGM) (GMAXES)

PURPOSE

To calculate the scalar product of two vectors

INPUT

Call DOTPRO ($\bar{A}$,$\bar{B}$,C)

where A and B are the two vectors and C is the returned scaler product.

OUTPUT

$C = \bar{A} \cdot \bar{B}$

C contains the scalar product

METHOD

$C = A_xB_x + A_yB_y + A_zB_z$

DIAGNOSTICS

None

REQUIREMENTS

Calls no subroutines

Called by subroutines LINRTY, PROCQD, SEGDRC

RESTRICTIONS

None

DRAFT (DRFTGD) (GEDFRT)

PURPOSE

To process and output a command block for the operations as designated in the APT statement, DRAFT

INPUT

CALL DRAFT

The input arrays ICLDAT and DATACL are used.

OUTPUT

The proper operation is obtained by setting the related D code in DBFSEG(16) and outputting the command block.

METHOD

Test ICLDAT(4): If equal to (ON), set the draft flag UPFLAG equal to 1. Output a 1 in the D register. If OFF is programmed, turn off the draft flag by setting UPFLAG = 0, and output a 2 in the D register, DBFSEG(16). If DASH is programmed with OFF, output a 5 in DBFSEG(16). If SOLID is programmed with an ON, branch to output a 1 in DBFSEG(16). If DASH is programmed with an ON, test the UPFLAG, and if zero, output a 1 in DBFSEG(16) to lower the turret, then output a 4 in DBFSEG(16) to establish the code.

DIAGNOSTICS

None

REQUIREMENTS

Called by subroutine AUXLRY

Calls subroutine OUTPUT

RESTRICTIONS

OPTAB(132) must equal 3.

DRESS (DRESGP) (GEPOS)

PURPOSE

To set up a block to output an M code for the APT statement DRESS

INPUT

CALL DRESS

OUTPUT

TABLEM(28) is stored into DBFSEG(14) which is then output in a block by itself.

METHOD

Store the M code in TABLEM(28) and output it in a block by itself by calling STOREM(130) to clear the buffer.

DIAGNOSTICS

None

REQUIREMENTS

Called by subroutine AUXLRY

Calls subroutine STOREM

RESTRICTIONS

None

DRETHD (DRETHD)
(GEMULT)

PURPOSE

To return a parked head to the work when the alternate head is already in the work

INPUT

CALL DRETHD (AAA, BBB, SEQNA, SAFVA, SAFWB, ABSA, OUTX, OUTY, OUTZ, IHA, IHB, SAFWA)

where:

AAA	=	delta move for head which is parked
BBB	=	delta move for head in the work
SEQNA	=	sequence number for parked head blocks
SAFVA	=	deltas used to park the head and return it
SAFWB	=	deltas used to withdraw alternate head
ABSA	=	absolute coordinate system for parked head
OUTX	=	absolute X value at which head was parked
OUTY	=	absolute Y value at which head was parked
OUTZ	=	absolute Z value at which head was parked
IHA	=	number of the parked head
IHB	=	number of the alternate head
SAFWA	=	deltas used to withdraw the parked head

OUTPUT

Both heads are positioned in the work ready to move.

METHOD

(1) The incremental moves in AAA and BBB are saved.

(2) The alternate head is withdrawn from the work.

(3) The absolute coordinate system for the parked head is reset to the point at which it was parked.

(4) The parked head is returned to the withdrawal position.

(5) The coolant is turned back on.

(6) A combined motion returns both heads back to the work.

(7) The incremental motions are restored in AAA and BBB.

(8) The array BUFPRE is reset to DMBITS.

DIAGNOSTICS

None

REQUIREMENTS

Called by subroutine FXPARK

Calls subroutines FXTOL, GMOT, COMPGC, PERROR, RAPLIM, and SRAREC

RESTRICTIONS

None

DSRROW (SROWGB) (GEBASE)

PURPOSE

To determine the row of the requested spindle speed by scanning the SRTAB

INPUT

CALL DSRROW

OUTPUT

The spindle speed is stored in the parameter SPNSPD.

METHOD

Calculate the starting index for the lowest speed in the requested range INDX1. Compare the requested spindle speed SPNSPD against those speeds SRTAB(K) available on the machine. If a speed cannot be found, output comment (63). If the requested spindle speed is not exactly available, test OPTAB(90) to determine if the lower, higher, or closest speed is desired. Based upon this OPTAB, set a value for the row index for the range ISPDRO. Test the spindle type flag ISPTYP (type spindle +1) to determine if the type routine stores the SRTAB(K) value into SPNSPD. If not, store this value in SPNSPD before returning. If the lowest speed in a range is used, output a comment (80).

DIAGNOSTICS

Comment: SPINDLE SPEED COULD NOT BE FOUND.

Comment: LOWEST SPEED IN RANGE IS OUTPUT.

REQUIREMENTS

Calls COMENT

Called by subroutines SFMO, SPTYPE, TYPE03

RESTRICTIONS

Machine tool must have an S code register.

DUMACH (DUMACH)

PURPOSE

To provide entries for dummy machines and call DISPAT when a machine not in the overlay is referenced

INPUT

CALL DUMACH

The input arrays ICLDAT and DATACL are used.

OUTPUT

The APT system COMMON variable IOUTAP is used to print comment 24.

DIAGNOSTICS

Comment: THE REQUESTED MACHINE IS NOT AVAILABLE.

(Followed by call to DISPAT)

REQUIREMENTS

Calls DISPAT

Called by no subroutines (only entries are referenced)

RESTRICTIONS

Multiple Entry

Uses APT System COMMON

Computer Dependent

DWELL (DWELGB) (GEBASE)

PURPOSE

To output a dwell block

INPUT

Call DWELL(N)

where N is the OPTAB location of the dwell time.

OUTPUT

A dwell command block (code = 4) is output.

METHOD

If N $\neq$ 0, a block is output with the G code from TABLEG(5) and dwell time given by OPTAB(N). CODE is set to 4.0 and subroutine OUTPUT is called.

If N = 0, CODE is set to 1.0 and subroutine OUTPUT is called.

DIAGNOSTICS

None

REQUIREMENTS

Calls subroutine OUTPUT.

Called by subroutines CLAMP, COUPLE, END, FEDRAT, RAPID, RESTAT, REWIND, SADDLE, SPINDL, TOOLGM, TOOLL, TURRET, TURSAD, TYPE02, TYPE03, TYPE04, TYPE10, TYPE12, TYPE13.

RESTRICTIONS

None

EIACOM (EIACGB) (GEBASE)

PURPOSE

To convert any number within the range from 0.0001 to 9999 to the corresponding EIA three digit command form

INPUT

CALL EIACOM (GIVALU)

where:

GIVALU is the number to be converted.

OUTPUT

The converted 3 digit EIA number is restored in GIVALU.

METHOD

The sign of the value is saved in DTEMP(1). If the GIVALU equals 1,10,100, pick up the converted value and return. Based on the value of GIVALU, add a factor to the value to take care of a rounding problem. Determine the value of the exponent 10•(KY) for values greater than 1. The value DTEMP(2) becomes the exponent on the first digit of the code. The value DTEMP(3) becomes the factor used to convert the programmed value to a number of 2 digits accuracy for the last 2 digits of the code.

DIAGNOSTICS

Comment: THE NUMBER TO BE CONVERTED TO THE MAGIC THREE FORM IS TOO LARGE OR TOO SMALL.

REQUIREMENTS

Called by subroutines CONTUR, GMOTIN, POSFED, TYPE01, TYPE04, TYPE05, TYPE10, TYPE13

Calls subroutine COMENT

RESTRICTIONS

None

END (ENDGB)
(GEBASE)

PURPOSE

To process an END block and reset flags.

INPUT

CALL END

Subroutine RESET is a multiple entry in subroutine END.

OUTPUT

If option 97 has a positive value, this value is output as a dwell time in a block with a zero T code. The End of Program M code is output in a block by itself.

Flags and parameters used: IENTRY, ENDFLG, RESETF, CURNG, REFATL, REFBTL, SEFBTL, FLRPON, FIRST, FLONSP, PREVS, FLSFON, STOPON, SFMFLG, DTRANS, and STATE.

DIAGNOSTICS

Comment 22: MISCELLANEOUS FUNCTION CODE NOT AVAILABLE ON THIS MACHINE.

REQUIREMENTS

Called by subroutine AUXLRY
Calls subroutines DWELL, COMENT, OUTPUT, and STOREM.

RESTRICTIONS

None

ENTRAP (RAPEGB)
(GEBASE)

PURPOSE

To output a rapid block in all cases after a STOP, OPSTOP, or BREAK statement

INPUT

Call ENTRAP

OUTPUT

Flags used: STATE, FRAPID, RAPRNG, FLRPON

A command block is output which places the machine in the rapid mode.

METHOD

This occurs only when automatic reinstatement is used.

DIAGNOSTICS

None

REQUIREMENTS

Calls subroutines DWELL, STOREM

Called by subroutine STOP

RESTRICTIONS

This is an entry to subroutine RESTAT.

ERDMPI (DUMPGE) (GEMON)

PURPOSE

To output the error number and give a dump of core when an unrecoverable error is detected.

INPUT

CALL ERDMPI

OUTPUT

The error number (ERROR) is printed and a dump of core is given.

REQUIREMENTS

Calls GEDUMP and DISPAT.
Called by GEBASE, DECODE, MOTION, FROM5, SPINDL, LINRTY, SELGRO, TURRET, IDPART, SELG, SELGCR, PROCQD, INIT, GEPRO1, WEFREW, FUNLNK, OUTPUT, and INIT.

RESTRICTIONS

Uses CALL LLINK statement on the GE 635.

REQUIREMENTS

Calls MACSRT

Called by POSMOV and POSFED.

RESTRICTIONS

OPTAB(78) = 5.0
OPTAB(1) = 1 or 2

FEDLIM (FDLMGB) (GETERP)

PURPOSE

To determine the axes component feedrate values, check for the number of feedrate ranges, test for rapid traverse, check for tape reader limitation, and to make sure the component feedrate is within specified limits

INPUT

CALL FEDLIM

OUTPUT

Flags used: FORK1 and IFDRNG

The tested feedrate is stored in DBFSEG(11).

METHOD

See Section 4.1.5 of the manual

DIAGNOSTICS

None

REQUIREMENTS

Called by subroutine OUTPUT

Calls subroutines CONVRT and LENGTH

RESTRICTION

None

FEDOVR (FEDOVR)
(GEBASE)

PURPOSE

To process M codes for the SELECT/FEDOVR,
ON
OFF Statement.

(Feedrate override)

INPUT

Call FEDOVR

The input arrays ICLDAT and DATACL are used.

OUTPUT

The M code is output in a block by itself.

METHOD

If ON is given, TABLEM(112) is output.

If OFF is given, TABLEM(111) is output.

DIAGNOSTICS

None

REQUIREMENTS

Calls subroutine STOREM

Called by subroutine SELECT

RESTRICTIONS

None

FEDRAT (FEDRGB) (GEBASE)

PURPOSE

To produce the proper feedrate and condition flags as specified by the APT source statement FEDRAT.

INPUT

CALL FEDRAT

The input arrays ICLDAT and DATACL are used.

OUTPUT

The given feedrate (not feedrate command number) is held in the parameter FEDIPM ready to be inserted in the programmed block. The feedrate mode and traverse flags will be set according to the conditions indicated.

METHOD

Initialize the MAXIPM flag IMAXFL, the RANGE flag IRANFL, and the feed-given flag IFEDFL to zero. First check to see if DATACL(4) is a floating value number. Call subroutine FLOAT and if ANS = 0, the value is a floating point number. Store the value in FDHOLD and set the feed-given flag IFEDFL equal to 1. If not, test to see if ICLDAT(4) is an IPM or IPR. If so, DATACL(5) is assumed to be a floating point number and is stored in FDHOLD. Set the feed mode flag FRMOD to 0 for IPM or 1 for IPR. If the mode flag is set to 0, test OPTAB(2) to see if SFM is to be cancelled by an IPM statement. If so, set the SFM flag SFMFLG and the range lock flag SFMLOK to zero. Continue the scan of the feedrate statement. If RANGE is found, look for modifiers MEDIUM, HIGH, LOW, and if none is found, the value following is assumed to be a floating point range number.

In any case set the range flag IRANGE to the proper range number. If MAXIPM is found, store the value in SFMAXI and set the MAXIPM flag IMAXFL to 1. At this point the feedrate statement has been completely scanned. Test the feedrate mode flag FRMOD: if set for IPM, store FDHOLD into the feedrate IPM parameter FEDIPM. If not, store FDHOLD into the feedrate parameter FEDIPR and calculate the IPM using the known spindle speed. If the mode is IPR, the feedrate in IPM is computed from the relation:

Feedrate in IPM = Feedrate in IPR•Spindle speed in RPM.

Test the rapid mode option (OPTAB(109); depending on the setting, go into RAPID or get out of RAPID. Assuming the standard setting, test OPTAB(18) to determine the number of feed ranges. If greater than 1, set the no-down shift flag RAPFED = 1. If a range was given and is different from the previous range, calculate the index N for selecting the proper TABLEM value. Store the M code for output by calling STOREM(N). Store the plus value in STATE(6), for possible reinstatement or a minus value in STATE(6) if currently in a Stop Condition (STOPON = 1). If a range change dwell is required (OPTAB(93)), a dwell is made output. Depending on the range and how OPTAB(25) is set, determine the minimum and maximum feedrates allowed, and store in FRMIN and FRMAX. If a MAXIPM is programmed, (IMAXFL = 1), test to see if SFMAXI is greater than FRMAX. If so, set SFMAXI equal to FRMAX. Call subroutine TSTEXT to set FEDIPM to the limit exceeded, if one is exceeded. Test to see if the machine is a positioning machine (OPTAB(1) = 0), and if so call the appropriate feedrate type; otherwise return.

DIAGNOSTICS

Comment: NO SPINDLE SPEED GIVEN FOR IPR MODE.

OPTION FEEDRATE RANGE ASSUMED.

REQUIREMENTS

Called by subroutine AUXLRY

Calls subroutines FLOAT, RAPID, RAPIDX, STOREM, DWELL, MACSRT, TSTEXT, FTYPE4, FTYPE8, FTYPE6

RESTRICTIONS

None

FLAME (FLAMGF) (GEFLAM)

PURPOSE

To output the auxiliary function M codes to establish the operating condition for a flame cutter.

INPUT

CALL FLAME

The input arrays ICLDAT and DATACL are used.

OUTPUT

The M Code for the designated operation is output in a block by itself.

METHOD

If the rapid flag (FLRPON = 0) is on, call subroutine RAPIDX to get out of rapid. If not, begin the scan of the statement. The head parameter IHEAD is set to −1 if BOTH is the modifier; if SLAVE, set to 1; if MASTER, set to 0. If UP or DOWN is programmed, select and output an M code based upon the value of IHEAD. If OXYGEN is programmed, select and output an M code based upon ON or OFF and IHEAD. If PREHET is programmed, select and output an M code based upon ON, OFF, and IHEAD. If the statement is incorrect, output comment 42.

DIAGNOSTIC

Comment: THE VOCABULARY IS NOT ACCEPTABLE IN THE FLAME STATEMENT.

REQUIREMENTS

Called by subroutine AUXLRY

Calls subroutines STOREM, COMENT, RAPIDX

RESTRICTION

OPTAB(132) = 4

FLOAT (GE/635) (FLFXGE) (GEMON)

PURPOSE

To determine if a value is a floating point number

INPUT

CALL FLOAT (T1, T2)

T1 is the number given for testing

OUTPUT

Flag T2 = 0 if T1 is a floating point number

T2 = 1 if T1 is not a floating point number

METHOD

Boolean Mask:

T1 is ANDed with actal constant 0777777000000.

If the result is zero, then T1 was not a floating point number.

DIAGNOSTICS

None

REQUIREMENTS

Called by subroutines CYCLGP, FEDRAT, GEDUMP, ROTABL, ROTHED, SPINDL

Calls no subroutines

RESTRICTIONS

Computer dependent—(GE635)

The subroutine uses a data statement.

FLOAT (IBM/360)
(GEMON)

PURPOSE

To determine if a value is a floating point number

INPUT

CALL FLOAT(IVAL, T2)

IVAL is the number given for testing

OUTPUT

T2 = 0 if the number (IVAL) is a floating point

T2 = 1 if the number (IVAL) is not a floating point number

METHOD

The address communicated through the first argument is double precision. Within this subroutine the first argument is a single precision array having two elements. The first element IVAL(1) is the left half of the DP word, i.e., it is the first 4 bytes. If IVAL(1) is all zero, then the DP word is an integer.

DIAGNOSTICS

None

REQUIREMENTS

Called by subroutines CYCLGP, FEDRAT, GEDUMP, ROTABL, ROTHED, SPINDL

Calls no subroutines

RESTRICTIONS

Computer dependent - (IBM360)

First argument must be double precision

FROM (FROMGE) (GEBASE)

PURPOSE

To setup and output a FROM point

INPUT

CALL FROM

The following parameters and arrays in COMMON are used: CLDATA, ICLDAT, TRANSL, AXMULT, DPRESM, DBFSEG

OUTPUT

CURNTZ, ENDFLG, DPRESP, DPREVP, DPREVM, CODE

The FROM point is output

METHOD

Any pending M code is forced out. If present, a plane G code is also output. The FROM point is selected from the CLTAPE, CODE set to 3, and subroutine OUTPUT called after the previous part and machine point vectors are set equal to present values.

DIAGNOSTICS

None

REQUIREMENTS

Calls subroutines STOREM, PLNSEL, FROM3, FROM5, SRAREC, OUTPUT

Called by subroutine MOTION

RESTRICTIONS

None

FROM3 (FRM3GB) (GEBASE)

PURPOSE

To set up DBFSEG for a non-multiaxis machine

INPUT

CALL FROM3

The following parameters and arrays in COMMON are used: DPRESM, ICLDAT, and CLDATA

OUTPUT

The FROM point is set up in DBFSEG(3,4,5).

DIAGNOSTICS

None

REQUIREMENTS

Calls subroutine GEOM

Called by subroutine FROM

RESTRICTIONS

None

FROM5 (FRM5GX) (GEMAXS)

PURPOSE

To check values, set up DBFSEG, and call one of the class subroutines to determine the transforms for the FROM point for a multiaxis machine

INPUT

Call FROM5

The following parameters and arrays in COMMON are used: DPRESP, EPSLON, DPRESM, OPTAB, ROTUNT, ICLDAT, and CLDATA.

OUTPUT

The multiaxis FROM point is set up in DBFSEG(3,4,5,6,7).

METHOD

The direction cosine accuracy of the FROM point is tested and the part data is converted to machine coordinates.

Rotary moves are modified to be within 360 degrees.

DIAGNOSTICS

(Direction cosine inaccuracy)

Error 19: Direction cosines error since

$$\sqrt{I^2 + J^2 + K^2} \neq 1$$

REQUIREMENTS

Calls subroutines LENGTH, ERDMP1, GEOM, MACSRT

Called by subroutine FROM

RESTRICTIONS

None

FTYPE2 (FTY26P)
(GEPOS)

PURPOSE

To determine the feedrate command and select the feedrate in IPM from the feedrate table stored in SRTAB

INPUT

CALL FTYPE2(F,K,IFEDFL)

```
where:  F = returned feedrate
        K = 1 for feedrate command
          = -1 for feedrate in IPM
   IFEDFL = 1 feedrate value given
          = 0 no feedrate value given
```

COMMON parameters used: FDHOLD, ISRNGE, SPNSPD, and SRTAB

Flags used: FRMOD and IFDRNG

OUTPUT

The appropriate or desired feedrate in IPM or feedrate command is output.

DIAGNOSTICS

Comment: OPTION VALUE IS ASSUMED FOR THE SPINDLE SPEED.

REQUIREMENTS

Called by subroutines CYCLGP, FEDRAT, POSFED and SPINDL

Calls subroutines COMENT and EIACOM

RESTRICTIONS

None

FTYPE4 (FTYPE4)
(GEPOS)

PURPOSE

To determine a feedrate command for a type 4 positioning machine.

Range 1: Feed command is based on the feedrate in IPR.

Range 2: Feed command is based on the feedrate in IPR*5.

Input

CALL FTYPE4 (F,K)

where F is the returned feedrate command or the returned feedrate in IPM;

and, K = 1 for the feedrate command,

or, K = 0 for the feedrate in IPM.

OUTPUT

The appropriate or desired feedrate in IPM or feedrate command is output.

Flags set; (1) FEDRTR
(2) FCOMM

DIAGNOSTICS

None called.

REQUIREMENTS

Calling routines; (1) FEDRAT
(2) CYCLGP
(3) SPINDL
(4) POSFED

RESTRICTIONS

Option (78) = 4.0

FTYPE6 (FTYPE6)
(GEPOS)

PURPOSE

To determine a feedrate for a Type 6 positioning feed command

Range 1: $F_{COM} = 12*F_{IPM}$

Range 2: $F_{COM} = 2*F_{IPM}$

INPUT

CALL FTYPE6(F,K):

where K = 0, F = Programmed feedrate in IPM

where K = 1, F = Actual feedrate in IPM

OUTPUT

Parameter FEDIPM contains:

when K = 0, actual feedrate in IPM available on the machine tool

when K = 1, feedrate command

METHOD

The feedrates available on the machine tool are stored in the feedrate table (the upper portion of SRTAB) by the Machine Subroutine. Based upon the number of feedrates per range (OPTAB(63)) and the size of the feedrate table (OPTAB (174)), the beginning and ending indices J2 and J3 are calculated. The programmed feedrate is compared against the feedrates available to determine the selected feedrate in IPM when K = 0. At output time this routine is again called when K = 1 to calculate the feedrate command. During the CYCLE/MILL mode all feedrates are assumed to be in range 2.

FTYPE6 (cont'd)

DIAGNOSTICS

None

REQUIREMENTS

Called by subroutines:

CYCLGP for K = 0

FEDRAT for K = 0

POSFED for K = 1

SPINDL for K = 0

Calls no subroutines.

RESTRICTIONS

OPTAB(78) = 6

OPTAB(1) = 1 or 2

FUNLNK (GEFNLK)
(FUNLNK)

PURPOSE

To interrogate the tables of the Machine Subroutine and set up a BCD list of function overlays for GEMON to pull in

INPUT

CALL FUNLNK(LINKOV)

OUTPUT

Flag used: ISPTYP, LINKOV, ERROR

The 5 elements of the array LINKOV are set to the BCD names of the links to be loaded.

DIAGNOSTICS

Error 8000: Option 132 is set incorrectly.

REQUIREMENTS

Calls subroutine ERDMP1

Called by subroutine GEMON

RESTRICTIONS

Data statement is used

Computer Dependent for the GE635 only

FVARGO (FVARGO) GETERP)

PURPOSE

To modify the F register in accordance with the G CODE (Variable F Format)

INPUT

CALL FVARGO(FCOMIN)

where FCOMIN is the feed command minimum

COMMON parameters used: NPR, NPT, NPTA, NFP, and NIP

OUTPUT

A series of command blocks derived from path segmentation is output.

METHOD

See Section 4.1.5.1 of the manual

DIAGNOSTICS

None

REQUIREMENTS

Called by subroutine CONTUR

Calls subroutine COMPFC

RESTRICTIONS

None

FXMULT (FXMULT) (GEMULT)

PURPOSE

To combine the motion blocks for both heads when the heads share a common axis

INPUT

CALL FXMULT

The following COMMON parameters are used:

AS2 the current record being processed for head 1

ABS2 the absolute coordinate system for head 1

AS3 the current record being processed for head 1

ABS3 the absolute coordinate system for head 2

ABCF1	= 0	incremental moves have not been added to the absolute coordinate system for head 1
	= 1	incremental moves have been added to the absolute coordinate system for head 1
ABCF2	= 0	incremental moves have not been added to the absolute coordinate system for head 2
	= 1	incremental moves have been added to the absolute coordinate system for head 2

FXMULT (cont'd)

OUTPUT

The portion of the cuts which can be combined are output and any cuts which cannot be output are cut separately. The following flags are used in this subroutine.

RFLAG2 is the read flag for head 1:

0 = read next record

1 = do not read next record

RFLAG3 is the read flag for head 2:

0 = read next record

1 = do not read next record

H1DIR indicates the direction of head 1, either plus or minus.

H2DIR indicates the direction of head 2, either plus or minus.

DJR gives the major direction of motion, either plus or minus.

H1FLG is the flag which shows if any motion is left for head 1:

0 = no motion on head 1

1 = motion remains on head 1

H2FLG is the flag which shows if any motion remains for head 2:

0 = no motion on head 2

1 = motion remains on head 2

FXMULT (cont'd)

METHOD

(1) Add new delta moves to absolute coordinate system (unless it is the remainder from a previous combine attempt).

(2) If either of the blocks is not a motion block, output it and return to get another.

(3) Determine if the two cuts share a common axis.

(4) If not, output the segment which best allows the system to "catch up" so combining can be done. Go to 6.

(5) If cuts share the common axis, determine if the component feedrates on the common axis are within the tolerance specified in OPTAB(157). If they satisy the tolerance conditions, output as a combined block. If not, output separately.

(6) If a portion of either vector remains, set it up as a new incremental move, and go read a record for the alternate head.

DIAGNOSTICS

Certain impossible branches from IF tests produce the following comment: AN ERROR IN FXMULT - TYPE UNRECOVERABLE. Subroutine PERROR is then called.

REQUIREMENTS

Called by subroutine GEMULT.

Calls subroutine SEG, FXTOL, GMOUT, FXPARK, PERROR, and SRAREC.

RESTRICTIONS

None

FXPARK (FXPARK)
(GEMULT)

PURPOSE

To determine if the specified head is parked and to park it if it is not parked. If the alternate head is parked, it is returned to the work piece.

INPUT

CALL FXPARK(1HEAD)

where:

1HEAD = 1 HEAD 1 is to be parked;

1HEAD = 2 HEAD 2 is to be parked;

1HEAD = 3 neither head is to be parked.

The following COMMON parameters are used:

AS2 the current record being processed for head 1;

ABS2 the absolute coordinate system for head 1;

AS3 the current record being processed for head 2;

ABS3 the absolute coordinate system for head 2;

IPARK1 flag to determine if head 1 has been parked;

0 = not parked;
1 = parked.

IPARK2 flag to determine iř head 2 has been parked.

0 - not parked;
1 = parked.

SAFHD1 the X,Y,Z departures from the SAFETY command for head 1;

FXPARK (cont'd)

INPUT (cont'd)

SAFHD2	the X,Y,Z departures from the SAFETY command for head 2;
SEQNH1	the current sequence number for head 1;
SEQNH2	the current sequence number for head 2;
SFHD1W	the X,Y,Z departures for withdrawing head 1 from the work while had 2 is being parked;
SFHD2W	the X,Y,Z departures for withdrawing head 2 from the work while head 1 is be parked.

OUTPUT

The specified head will be parked and the alternate head postioned in the work, or if 1HEAD = 3, both hands will be positioned in the work.

METHOD

Subroutine PARK is called to park a head. Surbroutine RETHD is called to return a head to the work when the alternate head is parked. Subroutine DRETHD is called to return a head to the work when the alternate head is in the work.

DIAGNOSTICS

If 1HEAD is not 1, 2, or 3, the following comment is printed: ERROR IN FXPARK ROUTINE IN GEMULT LINK XXX. XXX is the value of 1HEAD: subroutine ERROR is then called.

REQUIREMENTS

Called by subroutine GEMULT.

Calls subroutine PARK, RETHD, DRETHD, PERROR.

RESTRICTIONS

One head must always be in the work when this subroutine is called.

FXTOL (FXTOL) (GEMULT)

PURPOSE

To determine if the two cut vectors have a common axis component feedrate within the tolerance specified in OPTAB(157)

INPUT

CALL FXTOL(IND)

where:

IND is a flag to be set (see subroutine OUTPUT)

IND = 0 if tolerance condition is met

IND = 1 if tolerance condition is not met

The following COMMON parameters are also used:

AS2 the current DBFSEG record for head 1

AS3 the current DBFSEG record for head 2

METHOD

(1) If there is no common axis motion, return to calling subroutine.

(2) Compute the common axis component feedrates and the tolerance.

(3) If the tolerance is met, return to calling subroutine.

(4) If the tolerance is not met, determine if either move is in rapid.

(5) If the move is not in rapid, set IND = 1 and return.

(6) If the move is in rapid, use the smaller feedrate to compute a new feedrate for the other motion block. The new feedrate is within the tolerance.

FXTOL (cont'd)

DIAGNOSTICS

None

REQUIREMENTS

Calls no subroutines.

Called by subroutines DRETHD, EXMULT, and PARK.

RESTRICTIONS

None

GDWELL (GDWELL)
(GEMULT)

PURPOSE

To set up a data record when either or both of the heads have a G04

INPUT

CALL GDWELL

The COMMON parameters used are:

AS2	the current DBFSEG record for head 1
AS3	the current DBFSEG record for head 2
ICODE	the CODE for head 1
JCODE	the CODE for head 2

OUTPUT

AS2 and AS3 are set as described under METHOD.

METHOD

If only one head has a G04, the dwell time and feedrate for that head is set to DMBITS since the cut motion of the other head automatically takes care of the dwell on the first head. If both heads have a G04, then the dwell time is identical. Since only one value is necessary, the dwell time and feedrate for one head are set to DMBITS.

DIAGNOSTICS

None

REQUIREMENTS

Called by subroutines GMCIRL and GMLINE

RESTRICTIONS

None

GEBASE (GEBASE)
(GEBASE)

PURPOSE

To function as the control subroutine for overlay GEBASE

INPUT

CALL GEBASE

The input arrays ICLDAT and CLDATA are used.

OUTPUT

The following flags are used:

ENDFLG, CIRSEQ, RETURN, AXMULT, and MULTHD.

DIAGNOSTICS

TAPE READ ERROR (ERROR = 1.0)

Comment: NO END STATEMENT HAS BEEN GIVEN BEFORE THE FINI STATEMENT.

REQUIREMENTS

Calls subroutine AUXLRY, COMMENT, ERDMP1, MOTION, OUTPUT,

SRFCHK, STOREM, REWZ, WEFW, INPUT, IOERR

Called by subroutine GEMON

RESTRICTIONS

APT COMMON must be used.

GEMISC (GEMISC)
(GEMULT)

PURPOSE

To set up the output buffer GMHBUF with spindle speeds, M codes, and T codes

INPUT

CALL GEMISC(IH)

where:

IH = 1 means only head 1 has data to output

= 2 only head 2 has data to output

= 3 both heads 1 and 2 have data to output with head 1 being the primary head

= 4 both heads 1 and 2 have data to output with head 2 beign the primary head

The COMMON arrays AS2 (the current DBFSEG record for head 1) and AS3 (the current DBFSEG record for head 2) are also used.

OUTPUT

GMBUF is output as described below under METHOD.

GEMISC (cont'd)

METHOD

When both heads have their own S, T and M registers, GMHBUF will be set up as:

HEAD 1 Buffer

GMHBUF(12) = spindle code

GMHBUF(13) = T code

GMHBUF(14) = M code

GMHBUF(20) = spindle speed

HEAD 2 Buffer

GMHBUF(32) = spindle code

GMHBUF(33) = T code

GMHBUF(34) = M code

GMHBUF(40) = spindle speed

If the two heads share a register, the head 2 value for that register is stored in the head 1 region of GMHBUF.

DIAGNOSTICS

None

REQUIREMENTS

Called by subroutine GMOUT

RESTRICTIONS

None

GEMON (635) (GEMON)
(GEMON)

PURPOSE

To establish the memory overlay structure necessary for the given NC machine type

INPUT

CALL GEMON

The following paramerters and arrays in COMMON are used: LINKOV, OPTAB, IGEFLG, MULTHD, ICLDAT and CLDATA

OUTPUT

The array LINKOV contains the needed overlays. The overlay structure is then loaded into core.

METHOD

CALL LINK statements are used to load various modules into memory without transferring control to them. A DATA statement establishes the Hollerith names of various links. Array LINKOV is setup with BCD names of overlays.

DIAGNOSTICS

None

REQUIREMENTS

Calls subroutine LLINKs with arguments LKINIT, MACSAV, IPLAD,
KBASE, IFNLNK, IGOUT, IGEAD, and
IGMUIT

Calls subroutines INIT, GEPLAD, FUNLNK, GEBASE, GEAD,
GEMULT, DISPAT

Called by subroutine AUXLRY

GEMON (cont'd)

RESTRICTIONS

Uses a Data statement

Uses CALL LLINK statement

Computer dependent subroutine

GEMONT (GEMONT) (GEMON)

PURPOSE

To call subroutine GEOUT from the GEMULT link

INPUT

CALL GEMONT

OUTPUT

Subroutine GEOUT is called

METHOD

On some computers when an overlay is overlayed, the transfer vectors to routines in overlays further down in the structure are destroyed. In order to call a routine further down in the structure from an overlayed overlay, a routine in the initial overlay of the structure must be called, and then this routine calls in the desired overlay. A call could not be made directly to GEOUT from the overlay GEMULT, so GEMONT is called from GEMULT; and GEMONT then calls GEOUT.

DIAGNOSTICS

None

REQUIREMENTS

Called by subroutine GMSTOR

Calls subroutine GEOUT

RESTRICTIONS

None

GEMULT (GEMULT)
(GEMULT)

PURPOSE

To direct the processing of the multihead overlay for the rerging and output of command blocks

INPUT

CALL GEMULT

Data for head 1 are on TAPES2 and data for head 2 are on TAPES3.

OUTPUT

The processed data are stored in GMHBUF and passed on to GEOUT3 for final processing.

The following flags are used:

ICIRLN CIRCLE - line flag:

-1 = head 1 is linear and head 2 is circular

+1 = head 1 is circular and head 2 is linear

0 = both heads are linear

2 = both heads are circular

IEOF Read flag:

0 = a good data record

1 = end of file

IHOP flag:

1 = head 1 OP number less than head 2 OP number

2 = head 2 OP number less than head 1 OP number

GEMULT (cont'd)

OUTPUT (cont'd)

IS2 OP flag for head 1:

0 = no words are given in the OP statement except for the OP number

2 = additional words are given in the OP statement

IS3 OP flag same as IS2 except for head 2.

IS23 Present n value of head 1 for OP/n

IS33 Same as IS23 except for head 2

ISAFLG Flag indicating whether secondary head has been removed from combined cut:

0 = no
1 = yes

ICODE CODE for head 1

JCODE CODE for head 2

RFLAG2 read flag for head 1:

0 = read next record

1 = do not read next record

RFLAG3 same as RFLAG2 except for head 2

METHOD

Subroutine GEMULT reads records from TAPES2 (head 1) and TAPES3 (head 2) and determines whether the data from each head should be made output as separate data records or should be merged into a combined data record. If the data is to be merged into a combined cut, it also determines whether the data is Line-Line, Line-Circle, Circle-Line, or Circle-Circle type data. All the major subroutines are called from this routine, and the logical program flow is initiated and controlled by subroutine GEMULT.

GEMULT (cont'd)

DIAGNOSTICS

The following recoverable warning comments are printed:

Comment:	DATA CANNOT BE MERGED BECAUSE SECONDARY HEAD HAS BEEN REMOVED FROM COMBINED CUT.
Comment:	SECONDARY HEAD REMOVED FROM PART.

The following unrecoverable errors are printed:

ERROR 1020:	New n value for CODE 17 is less than previous n value.
ERROR 1021:	Word 11 of CODE 17 is not acceptable.
ERROR 1022:	Word 11 of CODE 17 is not acceptable.
ERROR 1023:	ICODE is not acceptable for head 1.
ERROR 1024:	n value or word 11 of CODE 17 is not acceptable on head 1.
ERROR 1025:	ICODE is not acceptable for head 2.
ERROR 1026:	n value or word 11 of CODE 17 is not acceptable on head 2.
ERROR 1027:	ICODE or JCODE not acceptable.
ERROR 7003:	Tape read error

REQUIREMENTS

Called by subroutine GEMON

Calls Subroutines CREAD, GMOUT, COMENT, FXMULT, FXPARK,

GMCIRL, GMFENC, GMINIT, GMLINE, PERROR, RETRET, and RETSFY

RESTRICTIONS

At present only two heads can be merged.

GEOM (GEOMGB)
(GEBASE)

PURPOSE

To convert the part coordinate vector DPRESP to the machine coordinate vector DPRESM

INPUT

CALL GEOM

The following parameters and arrays in COMMON are used: AXMULT

OUTPUT

DPRESM contains the machine coordinate data.

METHOD

Subroutine GEOM3 is called if non-multiaxis.

Subroutine GEOM5 is called if multiaxis.

Subroutine TSTLIM is called in either case.

DIAGNOSTICS

None

REQUIREMENTS

Calls subroutine GEOM3, TSTLIM, GEOM5

Called by subroutine MOTION

RESTRICTIONS

None

GEOM3 (GEO3GB)
(GEBASE)

PURPOSE

To convert the part coordinate point to the machine coordinate point for non-multiaxis machines

INPUT

CALL GEOM3

The vector DPRESP contains the CL data point with the TRANS value added. DPRESP is dimensioned at six and ordered as (x, y, z, i, j, k).

OUTPUT

The vector DPRESM contains the rounded and truncated values of DPRESP.

DPRESM is dimensioned at six and ordered as (x, y, z, a, b, c).

METHOD

DPRESP is stored in DPRESM after the values of DPRESP are rounded to the STEP size by subroutine SRAREC.
The STEP size is given in option 14.

DIAGNOSTICS

None

REQUIREMENTS

Calls subroutine SRAREC

Called by subroutine GEOM

RESTRICTIONS

None

GEOM5 (GEO5GX) (GEMAXS)

PURPOSE

To convert the part coordinate point to the machine coordinate point for a multiaxis machine

INPUT

CALL GEOM5

The vector DPRESP contains the CL data point with the TRANS value added. DPRESP is dimensioned at six and ordered as (x, y, z, i, j, k).

OUTPUT

The vector DPRESM contains the converted, rounded, and truncated values of the machine point corresponding to the part point DPRESP. DPRESM is dimensioned at six and ordered as (x, y, z, a, b, c).

METHOD

The particular class equations are called to transform the part coordinates to the machine coordinates.

DIAGNOSTICS

None

REQUIREMENTS

Calls subroutines CLASS, TRUNC

Called by subroutine GEOM

RESTRICTIONS

None

GEOUT (GEOUT)
(GEOUT)

PURPOSE

To pull in the proper punch package

This subroutine also selects the correct type of output sequence to be used.

INPUT

CALL GEOUT

OUTPUT

Flags used: FORK and NOW

METHOD

Option 164 is tested to see which output sequence is to be used, then the appropriate link and subroutines are called.

DIAGNOSTICS

None

REQUIREMENTS

Called by subroutine GEMON and OUTPUT

Calls subroutine GEPRE, LLINK, GEPR01, GEPR02, and GEPR03

RESTRICTIONS

This subroutine is computer dependant.

Uses Data Statement

GEPRE (GEPRE)
(GEOUT)

PURPOSE

To initialize control flags and forks used in GEOUT1 to test for a PARTNO record, and to set up a punch card identification.

INPUT

CALL GEPRE

COMMON parameters used: DABVAL, OPRVAL, DPRTNO, PREVS, PREVX and PREVY

OUTPUT

Flags and Forks used: IBLNK, IPGCTR, NEWSEQ, REELNO, IDLINE, TIMCUT, TIMDWL, CTRLIN, IPAGE, SKPCOD, and NOCHAR

DIAGNOSTICS

None

REQUIREMENTS

Called by subroutine GEOUT

Calls subroutine CALCP1, CALCP3, DECODE, IDPART, PARNEM, PARNOM, PPUNCH and PUNIDN

RESTRICTIONS

This subroutine uses Data Statements.

GEPRN1 (GEPRN1)
(GEOUT)

PURPOSE

To print out a line in GEOUT1

INPUT

CALL GEPRN1

The following parameters and arrays in COMMON are used: SEQCTR, BCDIMG, OPTAB

OUTPUT

The array BCDIMG is printed on IOUTAP.

120 positions are printed*

METHOD

*If option 143 ≠ 0.0, then the CL record number is output after print position 120.

DIAGNOSTICS

None

REQUIREMENTS

Calls no subroutines

Called by subroutines GEPR01, GEPR03, TITLE3

RESTRICTIONS

None

GEPRN2 (GEPRN2) (GEOUT)

PURPOSE

To write a 126 character print line on the output tape for GEOUT2

INPUT

CALL GEPRN2

The BCDIMG array is set up in BCD form for printing.

OUTPUT

The line is printed.

METHOD

BCDIMG is written according to a FORTRAN FORMAT statement.

DIAGNOSTICS

None

REQUIREMENTS

Calls no subroutines.

Called by subroutine GEPRO2, TITLE2.

RESTRICTIONS

Uses APT System COMMON.

GEPRN3 (GEPRN3)
(GEOUT)

PURPOSE

To print the BCDIMG array as used in GEOUT3

INPUT

CALL GEPRN3(INTX)

where: INTX is the number of BCD words to print

OUTPUT

The line of data is printed.

DIAGNOSTICS

None

REQUIREMENTS

Called by subroutines ABSOPR, and GEPRO3

Calls no subroutines

RESTRICTIONS

APT System COMMON is used.

GEPRO1 (GEPRO1)
(GEOUT)

PURPOSE

To complete processing and output a block for GEOUT1

INPUT

CALL GEPRO1

The following parameters and arrays in COMMON are used: FORK, REGFOR, ICODE, ISHUFL, OPTAB, DBFSEG, IXSTOR, TIMDWL, SEQNEW, ORGIN, FIRST, SPNSPD, DABVAL, TABLEM, PREVS, PREVF, IPAGE, NOCHAR, CTRLIN, TIMCUT, IPGCTR

OUTPUT

Flags used: FORK, BCDIMG, ERROR, TIMDWL, SEQNEW, DABVAL, FIRST, DPRTNO, PREVS, PREVG, PREVF, IPAGE, ORGIN, NOCHAR, CTRLIN, TIMCUT, IPGCTR, ITEMP, IWAVEN

A command block is printed and punched.

METHOD

ICODE indicates the type of block for output. Output registers are shuffled, dwell times are accumulated, as are cut times (total and per page). Sequence number and page counter as well as the line counter are calculated.

DIAGNOSTICS

Error 3000: Improper CODE number

REQUIREMENTS

Calls subroutines ERDMP1, SHUFFL, POSIT, CONTUR, SRAREC, SROREC, TITLE1, OCMNT1, SETUP1, SETLIN, PPUNCH

Called by subroutine GEOUT

RESTRICTIONS

Uses a Data Statement and APT System COMMON

GEPRO2 (GEPRO2)
(GEOUT)

PURPOSE

To complete the processing and output a block for GEOUT2

INPUT

CALL GEPRO2

The following parameters and array in common are used:

ICODE, NIPA(12), NPTA(12), OPTAB, DBFSEG,

TIMDWL, NPR, ISHUFFL, ORGIN(16), DPRESM,

TABLEM, TABLEG, IPAGE, ITHTYP, NFP, CTRLIN,

NIP, NPT.

OUTPUT

Flags used: FLAG, BCDIMG, TIMDWL, DBFSEG, DABVAL, OPRVAL,

PREVF, PREVF, PREVS, PREVG, IOUTAP, IPAGE,

ORGIN, SAVE, CTRLIN, ITEMP.

A command block is printed and punched.

METHOD

Parameter ICODE carries a value characterizing the block to be output. When all the needed register values are set, they are converted to BCD image and printed. Absolute and Operator lines are printed after the incremental line for each motion block. According to option specification, redundant codes are suppressed.

DIAGNOSTICS

None

GEPRO2 (cont'd)

REQUIREMENTS

Called by subroutine GEOUT

Calls subroutines SHUFFL, POSIT, CONTUR, SRAREC, SROREC, CONBCD, SETLIN, GEPRN2, CONROT, PPUNCH, TITLE2

RESTRICTIONS

Uses Data Statement

APT System common

Computer dependent

GESCOM (GESCOM)
(GEBASE)

PURPOSE

To determine the spindle command SPNCOM for the general Type 2 form of the spindle

INPUT

CALL GESCOM(IRNG, IROW)

where:
IRNG = new range number

IROW = is the row of the range where the speed is stored in SRTAB.

OUTPUT

Store the spindle command in SPNCOM

METHOD

Calculate the spindle command SPNCOM by adding to the row of the range IROW, the incremental adder OPTAB(47) minus 1 plus (range requested IRNG - 1) * (number of rows in the range NRORNG + the increment between ranges OPTAB(31) - 1)

DIAGNOSTICS

None

REQUIREMENTS

Called by subroutines TYPE02, TYPE14 spindle types

Calls no subroutines

RESTRICTIONS

OPTAB(19) = 2 or 14 only

GFDLIM (CFDLIM)
(GEMULT)

PURPOSE

To recalculate the feedrate command using the I,J,K registers, when the feedrate command exceeds the feedrate command maximum

INPUT

CALL GFDLIM(ITY, ABC, FRN)

where:

ITY = head number

ABC = current BUFSEG record being processed

FRN = true feedrate command which exceeds the command maximum

OUTPUT

The I,J,K values are calculated and the feedrate number is recalculated and stored in A,B,C.

METHOD

FRN is divided by integer multiples until it is less than FCOMAX. FRN and the I,J,K values are then calculated. The I,J,K values are the X,Y,Z values multiplied by the integer used to find the revised FRN.

GFDLIM (cont'd)

For example:

$$f = 500, \ FCOMAX = 500, \Delta X = 3, \Delta Y = 4.$$

$$FRN = \frac{10 \ * \ 500}{\sqrt{3^2 + 4^2}} = 1000.$$

Largest integer division which makes the FRN less than the FCOMAX is 3. Therefore,

$$FRN\,(new) = \frac{FRN}{3} = \frac{1000}{3} = 333.3$$

$$I = \Delta X*3 = 3*3 = 9$$

$$J = \Delta Y*3 = 4*3 = 12$$

See Section 4.1.5 for details on this technique.

REQUIREMENTS

Called by subroutine GMOUT

Calls no subroutines

RESTRICTIONS

The NC control system must have I, J, K registers

GMABS (GMABS) (GEMULT)

PURPOSE

To calculate the absolute coordinate values for each head

INPUT

CALL GMABS(IHH)

where:

IHH = 1 Head 1 data only;

IHH = 2 Head 2 data only;

IHH = 3 Data for both heads.

OUTPUT

The following common parameters are also used:

AS2 The current DBFSEG record for head 1.

AS3 The current DBFSEG record for head 2.

The absolute coordinate values for heads 1 and 2 are stored in ABS2 and ABS3, respectively.

METHOD

CODE is tested to see if the record is a motion record or a FROM point. Motion data is in incremental form. The incremental data is added to the absolute data to establish the new coordinate points. The FROM data is stored without alteration as an absolute coordinate.

DIAGNOSTICS

Subroutine PDUMP is called if |CODE| + 1 is greater than 18 or less than 1. |CODE| cannot be any valve>17.

GMABS (cont'd)

REQUIREMENTS

Called by subroutine GMOUT

Calls subroutine PDUMP

RESTRICTIONS

None

GMCIRL (GMCIRL)
(GEMULT)

PURPOSE

To set up a record of multihead merged data when the condition Circular-Linear; Linear-Circular, or Circular-Circular exists for the two mergeable command blocks.

INPUT

CALL GMCIRL

The following COMMON parameters are used:

AS2 the current DBFSEG record for head 1

AS3 the current DBFSEG record for head 2

IMAX maximum time which restricts feedrate

ICODE CODE for head 1

JCODE CODE for head 2

OUTPUT

If no restrictions are found, the combined motion record is output by calling subroutine GMOUT.

METHOD

The routine attempts to set up a record of data in the Circle-Line, Line-Circle, or Circle-Circle mode. There are three ways this can be done:

(1) Set up head 1 and head 2 data as is, with no changes.

(2) Set up head 1 as is and segment head 2 into two parts, such that the cut times of both heads are equal

(3) Set up head 2 as is, and segment head 1 into two parts, such that the cut times of both heads are equal

GMCIRL (cont'd)

METHOD (cont'd)

The cut times of both heads are compared to determine which of the three methods are to be used. Segmentation will not occur if any of the following conditions are found to be true:

(1) Delta time is less than TMAX.

(2) Delta time divided by cut time of either head is less than OPTAB(15).

After setting up a record of data, subroutine TESTM2 is called to see if any special restrictions will inhibt the data record from being sent to subroutine GMOUT. If no restrictions are found, the data record is sent to subroutine GMOUT. If restrictions are found, subroutine RETSFY is called to set flags so the data is output in a non-combined mode.

DIAGNOSTICS

None

REQUIREMENTS

Called by subroutine GEMULT

Calls subroutines GMOUT, SPLIT, GDWELL, RETSFY, and TESTM2

RESTRICTIONS

None

GMFENC (GMFENC) (GEMULT)

PURPOSE

To rewind TAPES2 and TAPES3 and terminate the GEMULT link.

INPUT

CALL GMFENC

TAPES2 the scratch file used for head 1 data.

TAPES3 the scratch file used for head 2 data.

OUTPUT

Not applicable

DIAGNOSTICS

See APT writeup of rewind subroutine for error comments.

REQUIREMENTS

Called by subroutines GEMULT and PERROR.

Calls subroutines GMSTOR and DISPAT.

RESTRICTIONS

Needs APT COMMON.

GMINIT (GMINIT) (GEMULT)

PURPOSE

To set all parameters in the block COMMON, COMBIN and FXAXCM to their initial values, and to open TAPES2 and TAPES3 for the read mode.

INPUT

CALL GMINIT

OUTPUT

Parameters in the block COMMON, COMBIN and FXAXCM are initialized. Scratch devices TAPES2 and TAPES3 are opened for reading.

DIAGNOSTICS

See the APT I/O routines for error comments. The comment ERROR in GMINIT-GEMULT LINK is printed if OPTAB(155) is greater than 3.

REQUIREMENTS

Called by subroutine GEMULT.

Calls subroutine WEFREW and PERROR.

RESTRICTIONS

Requires APT COMMON.

GMLINE (GMLINE)
(GEMULT)

PURPOSE

To set up a record of multihead merged data when the condition Linear-Linear exists for the two mergable command blocks.

INPUT

CALL GMLINE.

The following COMMON parameters are used:

AS2 the current DBFSEG record for head 1;

AS3 the current DBFSEG record for head 2;

IMAX the maximum time which restricts feedrate;

ICODE CODE for head 1;

JCODE CODE for head 2.

If no restrictions are found, the combined motion record is output by calling subroutine GMOUT.

METHOD

The routine attempts to set up a record of data in the Line-Line mode. There are three ways this can be done:

(1) Set up head 1 and head 2 data as is, with no changes.

(2) Set up head 1 as is, and segment head 2 into two parts, such that the cut times of both heads are equal.

METHOD (cont'd)

GMLINE (cont'd)

(3) Set up head 2 as is, and segment head 1 into two parts, such that the cut times of both heads are equal. The cut times of both heads are composed to determine which of the three methods are to be used. Segmentation will not occur if any of the following conditions are found to be true.

a. Delta time is less than TMAX;

b. Delta time divided by cut time of either head is less than OPTAB(151).

After setting up a record of data, subroutine TESTM2 is called to see if any special restrictions will inhibit the data record from being sent to subroutine GMOUT. If any restrictions are found, subroutine RETSFY is called to set flags so the data is output in a non-combined mode.

DIAGNOSTICS

None

REQUIREMENTS

Called by subroutine GEMULT.

Calls subroutine GMOUT, SPLIT, GDWELL, RETSFY, and TESTM2.

RESTRICTIONS

None

GMOTIN (GMOTIN)
(GEMULT)

PURPOSE

To compute the feedrate command for the current motion record.

INPUT

CALL GMOTIN(BUFCOM, FRN)

where:

BUFCOM = the current motion record. The feedrate value is stored in BUFCOM(11).

FRN = true feed command. This value may exceed feed command maximum.

The following COMMON parameters are used:

TABLEG, TMAX(the maximum time which restricts feedrate).

OUTPUT

The feed command is stored in BUFCOM(11).

METHOD

The dimension multiplier GDIMUL is selected by checking the G-code number against TABLEG. The path length D is computed from $\sqrt{X^2 + Y^2 + Z^2}$, and the feedrate command is computed from the formula:

$$FRN = \frac{f * GDIMUL}{D} .$$

The command value is checked against the command maximum. If the command is greater than the maximum, the maximum replaces the original number in BUFCOM(11). A check is also made to see if the command is tape reader limited. See Section 4.1.1.1.

DIAGNOSTICS

None

GMOTIN (cont'd)

REQUIREMENTS

Called by subroutine GMOUT

Calls subroutine EIACOM

RESTRICTIONS

None

GMOUT (GMOUT) (GEMULT)

PURPOSE

To set up a record of data for transmittal to GEOUT3.

INPUT

CALL GMOUT(IH)

where:

IH = 1 only head 1 has data for transmittal

= 2 only head 2 has data for transmittal

= 3 both heads 1 and 2 have data for transmittal with head 1 being the primary head

= 4 both heads 1 and 2 have data for transmittal with head 2 being the primary head

The COMMON parameters used are:

AS2	the current DBFSEG record for head 1.
AS3	the current DBFSEG record for head 2.

OUTPUT

The sixty work buffer GMHBUF is set up for transmittal to GEOUT3. Subroutine GMSTOR is called to perform the actual transmission.

GMOUT (cont'd)

METHOD

GMOUT recognizes the 6 following types of data output:

(1) Head 1 data only

(2) Head 2 data only

(3) Data from both heads in the Line-Line mode

(4) Data from both heads in the Circle-Line mode

(5) Data from both heads in the Line-Circle mode

(6) Data from both heads in the Circle-Circle mode

Feedrate command (FRN) and G codes are calculated based on one of the above six types of data. When it is necessary, the I,J,K registers are modified for the FRN calculation. GMOUT keeps track of the absolute coordinates for internal use.

DIAGNOSTICS

The unrecoverable ERROR 1028 is printed when the CODE does not exist for this output.

REQUIREMENTS

Called by subroutines CTCHUP, DRETHD, FXMULT, GEMULT, GMCIRL, GMLINE, OUTB, PARK, RETHD, and RETRET.

Calls subroutines GMABS, GEMISC, GFDLIM, GMOTIN, GMSTOR, PERROR, and PREPHD.

RESTRICTIONS

None

GMREAD (GMREAD)
(GEMON)

PURPOSE

To read a record from the designated file.

INPUT

CALL GMREAD(NF,IND,NREC,INTI,DBFSEG,NWPR)

where: NF is the file code;

IND is the error indicator;

NREC is the record number;

INTI is the number of blocks of data;

DBFSEG is the data block region;

NWPR is the number of words of data.

The record is stored in the buffer BUFSEG.

METHOD

This subroutine calls the APT subroutine which reads a record. The APT subroutine will vary with the computer.

DIAGNOSTICS

None

REQUIREMENTS

Called by subroutines ABSOPR and CREAD.

Calls the APT subroutine to read a record.

RESTRICTIONS

The APT subroutine for reading a record will be different on different computers.

GMSTOR (GMSTOR) (GEMULT)

PURPOSE

To initially read GMWORD (the register table for head 2) and GMFORM (the format table for head 2), and to rewind and open TAPES1 and TAPES4 for writing; to transmit the output buffer GMHBUF to GEOUT3.

INPUT

CALL GMSTOR

The arrays GMHBUF, GMWORD, and GMFORM are used.

OUTPUT

TAPES1 and TAPES4 are rewound and opened for writing. GMHBUF is reset to DMBITS after each call to GEOUT3.

METHOD

Not applicable.

DIAGNOSTICS

The unrecoverable error 7000 is printed when an error is encountered while reading TAPES1. The APT I/0 routine may also give errors.

REQUIREMENTS

Called by subroutines COMENT, GMFENC, GMOUT, and PREPHD.

Calls subroutines ERDMP1, GEMONT, GMREAD, and WEFREW.

RESTRICTIONS

Needs APT-COMMON. The subroutine has a DATA statement.

GMWRIT (GMWRIT)
(GEMON)

PURPOSE

To write a record on a file.

INPUT

CALL GMWRIT(NF,IND,NREC,INTI,DBFSEG,NWPR)

where: NF is the file number;

IND is the error indicator;

NREC is the record number;

INTI is the number of blocks of data;

DBFSEG is the data block region;

NWPR is the number of words of data.

DBFSEG is written on the file NF.

METHOD

This subroutine calls the APT subroutine which writes a record.

The APT subroutine will vary with the computer.

DIAGNOSTICS

The unrecoverable error 7003 is printed when a write error is encountered.

REQUIREMENTS

Called by subroutines GEPRO3 and OUTPUT.

Calls the APT routine to write a record.

RESTRICTIONS

The APT subroutine for writing a record will be different on different computers.

GOCIRC (CIRCGT) (GETERP)

PURPOSE

To test the CL data circle record to see if circular interpolation is permissible; if so, the first and last points of the circle are found and retained

INPUT

CALL GOCIRC

The following parameters and arrays are used: ITHFLG, CIRSEQ, RETURN, DPREVP, TRANSL, SEQCTR, CIRRAD, CLDATA, TOLIN, TOLOUT, EPSLON, IPLANE

OUTPUT

The following flags are set for the conditions of the circle: CIRSEQ, RETURN, CIRFLG, CIRRAD, CIRDIR, and CRCODE

The arrays DCRPT1 and DCRPT2 contain the first and last points of the circle; the array CIRDAT contains the circle center.

METHOD

A check is made to ensure that the circle is in a plane. The circle direction and radius are determined. If the radius is greater than the maximum departure, or if the move is for a thread, flags are set so as to process using linear interpolation.

For a type 5000 record, subtype 6, the non-planar axis is checked for continuity. Then the circle center is translated, and the circle quadrants are determined in the event the circle passes through more than one quadrant.

DIAGNOSTICS

Comment: WARNING - CIRCLE DOES NOT LIE IN A PLANE

GOCIRC (cont'd)

REQUIREMENTS

Calls subroutines CHKAX, DETDIR, COMPR, COMENT, CIRINT,
PLNSEL, PROCQD

Called by subroutine MOTION

RESTRICTIONS

None

GOLINE (LINEGT) (GETERP)

PURPOSE

To process a linear motion for output

INPUT

CALL GOLINE

The following parameters and arrays in COMMON are used:

RETURN, BIGDEP, ROTMAX, BMS, RMS, FLRPON, FRAPID,

DBFSEG, SFMFLG, THFLAG, AXMULT, RADLIN, TOLCON,

DPRESP, DPRESM

OUTPUT

The linear motion is set up in the DBFSEG command block and is made output.

METHOD

A path may be segmented for an SFM condition or because it exceeds the maximum departure.

DIAGNOSTICS

None

REQUIREMENTS

Calls subroutines STOREM, GEOM, DEPART, LINRTY, OUTPUT,

SEGMNT, THREDO, THEDOM, SFMO

Called by subroutines MOTION, ROTHED

RESTRICTIONS

None

IDPART (IDPTGO) (GEOUT)

PURPOSE

To process and punch a readable identification of a PARTNO for BCD Hollerith (PUNCHB) tapes

INPUT

CALL IDPART

The input arrays ICLDAT and CLDATA are used.

OUTPUT

The readable PARTNO is punched out.

METHOD

Character codes are selected from a Hollerith table so that the holes punched in the tape form the characters of the PARTNO statement.

DIAGNOSTICS

Error 54: Improper branch on an IF test

REQUIREMENTS

Calls subroutines CHARID, DOLLAR, PUNCHB, ERDMP1

Called by subroutine GEPRE

RESTRICTIONS

(GEOUT1 only)

Uses a Data Statement

Computer dependent

INIT (GEINIT)
(GEINIT)

PURPOSE

To clear COMMON areas, define constants, initialize standard values, search the CL tape for a MACHIN statement, and load special machine function subroutines if any

INPUT

CALL INIT

SYSTEM COMMONS: GECOM, GECBAS, GECOUT, GEMULT, GECOT3

OUTPUT

The postprocessor is set for standard processing.

METHOD

After initializing the COMMON areas and obtaining the MACHIN Statement, the machine subroutine is called. The MACHIN Statement is further searched for OPTAB, LINEAR, and LINCIR modifiers and the options are appropriately adjusted.

DIAGNOSTICS

ERROR 999: Given Machine Number is < 1 or > 99.

REQUIREMENTS

Calls subroutines STDMAC, REDTAP, INPUT, MACHxx, ASSIGN

Called by subroutine GEMON

RESTRICTIONS

APT System COMMON

Computer Dependent

Data Statement

INPUT (INPTGE)
(GEMON)

PURPOSE

To read a record from the CL tape and store it into CLDATA

INPUT

CALL INPUT

The input arrays ICLDAT and CLDATA are used.

OUTPUT

Flags used: IRETN

The arrays ICLDAT and CLDATA contain the data record read from the cutter location tape.

DIAGNOSTICS

None

REQUIREMENTS

Called by subroutines GEMON, GEBASE, INIT

Calls the Computer system tape-read subroutine

RESTRICTIONS

Computer dependent

APT System COMMON

LEADER (LEDRGB) (GEBASE)

PURPOSE

To output the leader length specified by the APT statements LEADER

INPUT

CALL LEADER

OUTPUT

The requested leader length is stored in DBFSEG (3) which is output with a CODE = -8.

DIAGNOSTICS

None

REQUIREMENTS

Called by subroutine AUXLRY

Calls subroutine STOREM and OUTPUT

RESTRICTIONS

This subroutine uses multiple entry points.

LENGTH (LNTHGB) (GEBASE)

PURPOSE

To compute a vector length from given component departures

INPUT

CALL LENGTH (DELTA, XL)

where DELTA contains the departure values as:

$$\text{DELTA}(1) = \Delta X, \quad \text{DELTA}(2) = \Delta Y, \quad \text{DELTA}(3) = \Delta Z$$

OUTPUT

The parameter XL contains the determined length.

METHOD

Cartesian metric:

$$XL = \left[\sum_{1}^{3} (\text{DELTA})_i^2 \right]^{\frac{1}{2}}$$

DIAGNOSTICS

None

REQUIREMENTS

Called by subroutines CIRSEG, CONTUR, FEDLIM, FROM5, NORM

Calls no subroutines

RESTRICTIONS

None

LINRTY (LNRTGX) (GEMAXS)

PURPOSE

To check and compensate for the nonlinearity in the motion of the tool tip during a multiaxis motion

INPUT

CALL LINRTY

The following parameters and arrays in COMMON are used:

CODE, STEP, OPTAB, FRAPID, DPRESM, ROTUNT, DPREVM, RADLIN, CRCODE

OUTPUT

Flags used: RETURN, SKPLIN, TSTLIN, NWPR, DPREVM, DPRESM, LINFLG, HALFPT, PT, MAFORK, CODE, ERROR

A series of motion command blocks are output which correct the linearity errors.

METHOD

The deviation of the tool tip from a linear path is tested against tolerance. If tolerances are not satisfied, the midpoint of the line between the part points is calculated and the transform equations applied to yield a potential intermediate machine point. Then linearity is tested along this segment, and if no linearity error is found, the path is segmented at that point and made output;otherwise, testing continues along the path until no error is found.

LINRTY (cont'd)

DIAGNOSTICS

Comment: WARNING - LINEARITY SEGMENTATION CANNOT BE CONTINUED ON THIS PATH.

Comment: WARNING - LINEARITY TOOL AXIS CORRECTIONS CANNOT BE CONTINUED ON THIS PATH.

Comment: NO LINEARITY TESTING THIS PATH ** SPECIAL GEOMETRY CONDITION.

Error 25: No motion from subroutine LINRTY

REQUIREMENTS

Calls subroutines SEGDRC, CLASS, OFFARC, DEPART, OUTPUT, TRUNC, COMMENT, ERDMP1

Called by subroutine MOTION

RESTRICTIONS

Linearity testing is done only on linear motions.

LINTOL (LINTOL)
(GEMAXS)

PURPOSE

To reinstate or cancel linearity testing, or to establish the linearity tolerance value

INPUT

CALL LINTOL

The following parameters and arrays in COMMON are used: RADLIN, ICLDAT, CLDATA

OUTPUT

The following parameters are set: TSTLIN, RADLIN, ANGLIN

METHOD

If RADLIN < 0, Nothing is done. If ON is given, the TSTLIN flag is set to 1.0. If OFF is given, the flag is set to 0.0. If values N and γ are given, RADLIN is set to N, and ANGLIN is set to γ .

DIAGNOSTICS

None

REQUIREMENTS

Calls no subroutines

Called by subroutine AUXLRY

RESTRICTIONS

None

LOAD (LOADGM)
(GEMAXS)

PURPOSE

To load a tool while at the home position

INPUT

CALL LOAD

The input arrays ICLDAT and CLDATA are used.

OUTPUT

Command blocks are output which returns the tool to the home position and loads the selected tool.

METHOD

Cancel linearity testing for the load sequence by setting flag TOLCON to 1. Call subroutine GOHOME to move the slides of the machine to the specified home position. Based on the gripper selected (GRPLOD) and the selected head (TLHEAD), determine the TABLEG (26-29) to output. Store the tool number TOLLOD in DBFSEG(13) and (26-29) to output. Store the tool number TOLLOD in DBFSEG(13) and output as a CODE = -4 block. Store the selected tool number TOLSLC into the loaded tool TOLLOD; selected gripper GRPSLC into the gripper loaded GRPLOD; selected tool length SLTOLN into the tool loaded length TOLDLN. Store the present machine point DPRESM into loaded length TOLDLN. Store the present machine point DPRESM into the previous machine point DPREVM. Set the multiaxis fork MAFORK to zero and obtain the tool and gripper constants from the CLASS equations.

REQUIREMENTS

Called from subroutine AUXLRY.

Calls subroutines MACRST, GOHOME, OUTPUT, CLASS

RESTRICTIONS

OPTAB(116) must be set to a value.

A FROM must be given before the LOAD statement.

LOCRNG (SRNGGB)
(GEBASE)

PURPOSE

To find the lowest range a spindle speed is in when no range was specified

INPUT

CALL LOCRNG(GIVALU)

where:

GIVALU = requested spindle speed

OUTPUT

The lowest range containing the speed requested is stored in ISRNGE.

METHOD

If the SFM range lock flag SFMLOK is not equal to zero, return; a range change cannot take place while the SFM mode is in effect. Initialize the range ISRNGE to 1. Depending on the spindle type OPTAB(19), calculate the storage index K of the highest speed in each range for all ranges available and store the value in parameter NRNGES. When the lowest range speed available is found, return. ISRNGE will contain that range number.

DIAGNOSTICS

None

REQUIREMENTS

Called by subroutine SPINDL

Calls no subroutine

RESTRICTIONS

None

MACHIN (MACHGB)
(GEBASE)

PURPOSE

To establish the program conditions and interpolation mode for a particular numerical machine tool as specified according to the APT source statement MACHIN

INPUT

CALL MACHIN

The input arrays ICLDAT and CLDATA are used

OUTPUT

The program conditions are set for the specified machine tool by resetting the parameters affected by the options changed in the MACHIN statement. The interpolation modifiers LINCIR and LINEAR are checked, as is the OFF modifier.

DIAGNOSTICS

None

REQUIREMENTS

Called by subroutine AUXLRY

Calls subroutines OUTPUT and DISPAT (if OFF modifier present)

RESTRICTIONS

This routine requires the APT system COMMON, and is Computer Dependant.

MINMOV (MNMVGP)
(GEBASE)

PURPOSE

To determine the minumum direction for turret indexing when rotation can take place in either direction

INPUT

CALL MINMOV(TURDIR)

OUTPUT

Flag TURDIR is set for direction of minimum rotation.

METHOD

In the initial entry the number of positions available in 180 is determined and stored in BIGMOV. The distance in position increments to be moved is calculated using the old position FFRSTL and the requested new position POSMAG. The distance to be moved DISMOV is compared with the 180° maximum motion. The direction of minimum rotation TURDIR is set:

TURDIR = 1 for CLW rotation

TURDIR = -1 for CCLW rotation

DIAGNOSTICS

None

REQUIREMENTS

Called by subroutines ROTMAG, TOOLNO, TURRET

Calls no subroutines

RESTRICTIONS

OPTAB(88) must be set ≠ 0.

MOTION (MOTNGB) (GEBASE)

PURPOSE

To Process a Type 5000 (Motion) record.

INPUT

CALL MOTION

The following parameters and arrays in COMMON ar e used: CLDATA, TRANSL, ICLDAT

OUTPUT

Flags used: MCHCON, ENDFLG, CIRSEQ, RETURN, AND CIRFLG

Subroutine calls are made to various subroutines depending on whether a FROM point, GODLTA, or GOTO move is processed.

DIAGNOSTICS

Comment 47: END OR RESET HAS NOT BEEN FOLLOWED BY A FROM POINT.

Comment: OPTION FEEDRATE ASSUMED.

Error in subroutine GOLINE (error = 14.0)

REQUIREMENTS

Called by subroutines GEBASE.

Calls subroutines FROM, COMENT, ERDMP1, GOCIRC, GOLINE, MACSRT, POSMOV, and TSTFLG

NORM (NORMGX) (GEMAXS)

PURPOSE

To normalize direction cosines

INPUT

CALL NORM(DIRCOS)

where:

DIRCOS contains the direction cosines as (i, j, k).

OUTPUT

The direction cosines are normalized and stored in DIRCOS.

METHOD

The components of the vector are divided by the vector magnitude.

DIAGNOSTICS

None

REQUIREMENTS

Calls subroutines LENGTH

Called by subroutine CLASS, SEGDRC

RESTRICTIONS

None

OFFARC (OFARGT) (GETERP)

PURPOSE

To compute the arc center offsets for a circle segment

INPUT

CALL OFFARC

The following parameters and arrays in COMMON are used: CIRDAT, DPREVM, REGFOR, IPLANE, AXMULT, DEPA, DEPB, ARCANG

OUTPUT

DBFSEG(8), (9), and (10), contain the arc center offsets

For multiaxis machines the rotary center offsets are stored in DBFSEG(16,17).

METHOD

The arc center offset is computed by taking the axial difference between the circle center to the beginning point of the circle; that is,

$$I=|X_c-X_1| \quad , \quad J=|Y_c-Y_1| \quad , \quad K=|Z_c-Z_1|$$

DIAGNOSTICS

None

REQUIREMENTS

Calls subroutines SRAREC, COMENT

Called by subroutines LINRTY, PROCQD, SFMO

RESTRICTIONS

None

OPCODE (OPCODE)
(GEBASE)

PURPOSE

To process the part programming statement

```
     NONE
OP/n,RPM , t , t
     SFM    1   2
```

INPUT

CALL OPCODE

OUTPUT

A command block (CODE = 17) is set up as:

DBFSEG(2) = n (from OP/n)

DBFSEG(7) = 0 if SFM, = 1 if RPM

DBFSEG(8) = head number

DBFSEG(9) = t_1

DBFSEG(10) = t_2

DBFSEG(11) = 0 for restrictions
= 2 for none

DBFSEG(15) = 17(CODE)

DIAGNOSTICS

None

REQUIREMENTS

Called by subroutine AUXLRY

Calls subroutines COMENT and OUTPUT

RESTRICTIONS

None

OPSKIP (OSKPGB) (GEBASE)

PURPOSE

To set a signal so that each programmed block will be given the skip code when processed for output

INPUT

CALL OPSKIP

SKPFLG, the flag for OPSKIP which appears in COMMON

OUTPUT

The skip flag is set to 0 (OFF) or 1 (ON) according to the specifications as given by the APT source statement OPSKIP

DIAGNOSTICS

None

REQUIREMENTS

Calls no subroutines

Called by subroutine AUXLRY

RESTRICTIONS

None

ORIGIN (ORGNGB) (GEBASE)

PURPOSE

To save the given ORIGIN values, and to communicate through an information block to the output routine, the origin values needed to produce the machine system printout.

INPUT

CALL ORIGIN

The input arrays ICLDAT and CLDATA are used.

OUTPUT

The X,Y,Z,A,B,C values are stored in DBFSEG(3 through 7), DBFSEG(2) is set to 2, and CODE is set to -9 to output the block as an information block.

METHOD

If option 164 is 2 or option 172 is 0, the values from the CL tape are stored in DBFSEG. The rotary values are converted to output units. The information block is set up with CODE = -9 and DBFSEG(2) =2.

DIAGNOSTICS

None

REQUIREMENTS

Calls subroutine OUTPUT

Called by subroutines AUXLRY, GEPRO2

RESTRICTIONS

None

OUTB (OUTB)
(GEMULT)

PURPOSE

To set up a block for shifting in to or out of rapid traverse

INPUT

CALL OUTB(XMCOD, AAA, IH, IF)

where:

XMCOD is the M code for shifting

AAA is the array used for outputting the block

IH is the head number

IF is shift flag:

= 1 for shifting from feed into rapid

= 2 for shifting from rapid into feed

OUTPUT

The block is output in the array AAA.

METHOD

The present value of AAA is saved. Then AAA is set up thus:

AAA(1, 1) = sequence number

AAA(14, 1) = M code

AAA(15, 1) = -1.0

Subroutine GMOUT is called to output the block and then the original value of AAA is restored.

DIAGNOSTICS

None

OUTB (cont'd)

REQUIREMENTS

Called by subroutine RAPM

Calls subroutine GMOUT

RESTRICTIONS

None

OVRCNT (OVRCNT) (GEMAXS)

PURPOSE

To process the OVRCNT/n statement for overcenter cutting

INPUT

CALL OVRCNT

The input arrays ICLDAT and CLDATA are used.

OUTPUT

The given overcenter value n is saved in OVCVAL, or OVCVAL is set as a flag according to the given minor modifier.

METHOD

When the overcenter value is given, the value is saved in OVCVAL. If OFF is given, OVCVAL is set to zero. If ON is given, the most recently specified value is reinstated.

DIAGNOSTICS

None

REQUIREMENTS

Calls no subroutines

Called by subroutine AUXLRY

RESTRICTIONS

None

PAGE (PAGE)
(GEOUT)

PURPOSE

To print the page number on each page

INPUT

CALL PAGE

COMMON parameters used: CTRLIN, IPAGE and IPGCTR

OUTPUT

The current page number is printed at the top of each page.

DIAGNOSTICS

None

REQUIREMENTS

Called by subroutines ABSOPR and GEPRO3

Calls no subroutines

RESTRICTIONS

APT System COMMON is used

PARK (PARK) (GEMULT)

PURPOSE

To park the specified head

INPUT

CALL PARK(IHEAD, LALT, AAA, BBB, SEQMA, SEQNB, SAFWB, SFPRKA, OUTX, OUTY, OUTZ, ABSA LALTPK)

where:

IHEAD = head to be parked

LALT = alternate head number

AAA = incremental move for head to be parked

BBB = incremental move for alternate head

SEQNA = sequence number for head to be parked

SEQNB = sequence number for alternate head

SAFWB = withdrawal distance for secondary head

SFPRKA = delta move used for parking head

OUTX = absolute X for head being parked

OUTY = absolute Y for head being parked

OUTZ = absolute Z for head being parked

ABSA = absolute coordinate system for head to be parked

LALTPK = park flag for alternate head

The following COMMON arrays are also used:

OPTAB, TABLEG, TABLEM

PARK (cont'd)

OUTPUT

The specified head will be parked and the alternate head, if in the work, will be properly positioned.

METHOD

(1) Save incremental moves in AAA and BBB.

(2) Withdraw the alternate head if it is in the work.

(3) Turn off coolant for the head being parked.

(4) Park the specified head. This will be combined with the alternate head withdrawal, if possible.

(5) Return the alternate head from the withdrawal position (if necessary).

(6) RESTORE the incremental moves in AAA and BBB.

(7) Save the absolute coordinate point at which the head was parked.

DIAGNOSTICS

Comment: HEAD 1 HAS BEEN PARKED

Comment: HEAD 2 HAS BEEN PARKED

If the tolerance test on the common axis feedrate fails while in a combined move, a comment to this effect is printed and subroutine PERROR is called.

REQUIREMENTS

Called by subroutine FXPARK

Calls subroutines FXTOL, GMOUT, COMENT, COMPGC, PERROR,
RAPLIM, and SHFTBK

RESTRICTIONS

None

PERROR (PERROR) (GEMULT)

PURPOSE

To print the error number and produce a memory dump

INPUT

CALL PERROR

COMMON parameter used: ERROR.

OUTPUT

The ERROR number followed by a dump is printed.

DIAGNOSTICS

The error comment ERROR IN GEMULT, ERROR NO. IS____

REQUIREMENTS

Called by subroutines DRETHD, FXMULT, FXPARK, GEMULT, GMINIT, GMOUT, PARK, RAPM, SELGCD, and STOPTS

Calls subroutine PDUMP

RESTRICTIONS

Uses APT COMMON

PICKUP (PKUPGP) (GEPOS)

PURPOSE

To set up a block to output an M code or a G code to load the tool when this function is not performed as a part of the T-code output.

INPUT

CALL PICKUP

The input arrays ICLDAT and CLDATA are used.

OUTPUT

The M code from TABLEM(32) or a G code from TABLEG(24) is stored into DBFSEG and output.

METHOD

If TABLEG(24) is equal to bits, an M code is assumed. The current angle is turned off by (TABLEG(1)), M code TABLEM(32) is stored for output, the tool number of the tool removed is saved in TLNOFF and its length is saved in TLEN2. If TABLEM(32) equals DMBITS, a TABLEG(24) is output as a code =-4 block (a G code in a block by itself).

DIAGNOSTICS

None

REQUIREMENTS

Called by subroutine AUXLRY

Calls subroutines STOREM and OUTPUT

RESTRICTIONS

TABLEM(32) <u>or</u> TABLEG(24) must have a value.

PITCH (PTCHGL)
(GELATH)

PURPOSE

TO process the PITCH statement, calculate the lead, and set the threading flags

INPUT

CALL PITCH

The input arrays ICLDAT and CLDATA are used

OUTPUT

Flags used: THLEAD, THRATE, THMODE

The flags are set in accordance with the PITCH statement.

METHOD

$$\text{LEAD} = \frac{1}{\text{PITCH}}$$

For increasing or decreasing pitch, the threading feedrate is given by:

$$\text{THRATE} = \frac{(\text{final lead})^2 - (\text{initial lead})^2}{2 * \text{thread length}}$$

DIAGNOSTICS

None

REQUIREMENTS

Calls no subroutines

Called by subroutine AUXLRY

RESTRICTIONS

None

PITCHM (PITCHM)
(GEMILL)

PURPOSE

To process the APT statement PITCH for multihead machines

INPUT

CALL PITCHM

COMMON parameters used: NWPR, CLDATA, and THLEAD

OUTPUT

The parameters THMODE and possibly THRATE are setup with the programmed values.

DIAGNOSTICS

None

REQUIREMENTS

Called by subroutine AUXLRY

Calls no subroutines

RESTRICTIONS

None

PLENTH (PATHGB)
(GEBASE)

PURPOSE

To compute the path length for a move

INPUT

CALL PLENTH

The following parameters and arrays in COMMON are used: NWPR, CLDATA, NCOM, TRANSL, DPREVP

The input arrays ICLDAT and CLDATA are used

OUTPUT

The COMMON parameter PL contains the path length.

METHOD

The coordinate data from the CLDATA array are reduced to increments in X, Y, Z and the length found by

$$PL = \sum_{i=1}^{n} \sqrt{\Delta x_i^2 + \Delta y_i^2 + \Delta z_i^2}$$

where n is the number of cut vectors making up the move.

DIAGNOSTICS

None

REQUIREMENTS

Called by subroutine RAPID

Calls no routines

RESTRICTIONS

None

PLNSEL (PLNSGB)
(GEBASE)

PURPOSE

To select the G code for the circular interpolation plane

INPUT

CALL PLNSEL

The flag MAXES specifies the plane to select.

OUTPUT

A DBFSEG block with the G code by itself is made output.

CODE = -4

METHOD

If circular interpolation is available and there are at least three axes, a G code is entered in DBFSEG according to the following scheme:

TABLEG LOCATION	PLANE SELECTED
(18)	XY
(19)	ZX
(20)	YZ

DIAGNOSTICS

None

REQUIREMENTS

Calls subroutine OUTPUT

Called by subroutines FROM, GOCIRC

RESTRICTIONS

None

POSFED (PFEDGO)
(GEPOS)

PURPOSE

To convert the feedrate in IPM to its command value depending on the positioning Feed Type

INPUT

CALL POSFED

OUTPUT

Current feedrate in IPM is stored in DABVAL(11) and the feedrate in command form is stored in DBFSEG(11).

METHOD

If the current feedrate in IPM is not equal to DMBITS (DBFSEG(11)=bits) store this feedrate in DABVAL(11).

Test OPTAB(28) and call the appropiate FTYPE subroutine to calculate the feed command.

Test this command in subroutine TSTEXT for a valid value.

If the value calculated is beyond the limits, subroutine TSTEXT will set DBFSEG(11) to limit exceeded.

DIAGNOSTICS

None

REQUIREMENTS

Called by subroutine POSIT

Calls subroutines FTYPE2, FTYPE4, FTYPE5, FTYPE6, depending on OPTAB(78) = type number and TSTEXT

RESTRICTIONS

OPTAB(1) must equal 1 or 2.

POSIT (POSIGO)
(GEPOS)

PURPOSE

To suppress redundant X and Y values under optional control (OPTAB(40)) and to call subroutine RFTYPE or POSFED to calculate the feedrate

INPUT

CALL POSIT

OUTPUT

The redundant X and Y values are suppressed in DBFSEG and the calls to the feedrate command calculation contains are made.

METHOD

If redundant X and Y are to be suppressed (OPTAP(40) =1), test DBFSEG (3-4) against PREVX and PREVY. If identical, set DBFSEG (3-4) to BITS. If code equals 2 (an absolute rotary table move block), call subroutine RFTYPE to compute the rotary feed command. Otherwise call subroutine POSFED to calculate the linear feed command. In all cases the absolute table location is saved in DABVAL(7) and converted to degrees through a call to subroutine CONROT (DABVAL(7), - 1).

DIAGNOSTICS

None

REQUIREMENTS

Called by subroutines GEPRO1, GEPRO2, GEPRO3

Calls subroutines RFTYPE, POSFED, CONROT

RESTRICTIONS

OPTAB(1) must equal 1 or 2

POSITN(PSTNGP)
(GEPOS)

PURPOSE

To set the positioning mode flag MODPOS according to the positioning mode specified in the POSITN statement

INPUT

CALL POSITN

The input arrays ICLDAT and CLDATA are used.

OUTPUT

For mode OFF,MODPOS =0

For mode FINE,MODPOS = 1

For mode NOBACK,MODPOS = 2

For mode COARSE,MODPOS = 3

For mode CORMIL,MODPOS = 4

For mode TRAV,MODPOS = - |MODPOS|

METHOD

ICLDAT(4) is scanned to determine the mode specified. MODPOS is set to a value shown above. In the OFF mode, a block is output with a TABLEG(60) and CODE = -4 to turn off the positioning mode. The MODPOS flag is used by subroutine SET12 to determine the proper G code.

DIAGNOSTICS

None

POSITN (cont'd)

REQUIREMENTS

Called by subroutine AUXLRY

Calls subroutine OUTPUT

RESTRICTIONS

OPTAB(1) must equal 1 or 2

TABLEG(52-62) ≠ DMBITS

POSMOV (PSMVGP)
(GEPOS)

PURPOSE

To process the motion records for a positioning type machine

INPUT

CALL POSMOV

The input arrary ICLDAT and CLDATA are used.

OUTPUT

The X, Y, Z motions are output in a DBFSEG command block.

METHOD

If the weld-off flag is on (IGEFLG = 1), output a weldcycle-off M code (TABLEM(87)). If DBSFEG(2), the G Code, does not have a value and the spindle dwell flag is not set, (IDWLFL = 0), call subroutine SET12 to select the positioning mode. Set the IDWLFL flag to 0, and if initial entry (POSFRK ≠ 360), set the current height CURNTZ to DMBITS. If the Z axis is inverted (OPTAB(140) ≠ 0), modify CLDATA(8) by the inversion modifier OPTAB(140). If tool length modification is to take place (OPTAB(86) is negative), adjust CLDATA(8) by the tool length TOOLEN. If CLDATA(8) is less than 0, output comment 66. If not under the influence of a position move, DBFSEG(2) = BITS and no cycle was specified, (CYCFLG = 0), turn off all cycles by setting DBFSEG(2) = TABLEG(1). Store the coordinates in the PRESPT vector from CLDATA plus the translation value TRANSL. Save the untranslated Z height in ZHOLD. Assume a CODE = 16 block for X,Y motions only. Test to determine how the Z motion is to be output according to OPTAB(130). If zero, ignore Z; if +1, output Z in a separate block, or if -1, output X,Y,Z, in one block. Store the feed and either the rapid feedrate, FRAPID, rate or the feedrate in IPM, FEDIPM, into DBFSEG(11). Store the present point into previous point. Call subroutine TSTLIM to test for slide limits, and if record is complete, save the current Z height in CURNTZ and return.

POSMDV (cont'd)

DIAGNOSTICS

None

REQUIREMENTS

Called by subroutine MOTION

Calls subroutines STOREM, SET12, MACSRT, OUTPUT, TSTLIM

RESTRICTIONS

OPTAB(1) ≠ 0

PREPHD (PREPHD)
(GEMULT)

PURPOSE

To output the preparatory function G code for mixed simultaneous Linear-Circular interpolation on the two heads, i.e., one head uses linear, and the other uses circular interpolation.

INPUT

CALL PREPHD(ICAB)

where:

ICAB = 1 Head 1 is in circular mode

= 2 Head 2 is in circular mode

TABLEG, the preparatory code table is also input

OUTPUT

A special G code is output in a record by itself.

METHOD

This special G code is either:

TABLEG(16) Circular interpolation for head 1 and linear interpolation for head 2

or TABLEG(17) Linear interpolation for head 1 and circular interpolation for head 2

The G code is modal and is output as a record only when the other G code is in effect.

DIAGNOSTICS

None

PREHD (cont'd)

REQUIREMENTS

Called by subroutine GMOUT

Calls subroutine GMSTOR

RESTRICTIONS

Contains a DATA statement

PROCQD (PCQDGT)
(GETERP)

PURPOSE

To output the quadrant segments for a circular interpolation move.

INPUT

CALL PROCQD

The following parameters and arrays in COMMON are used:

AXMULT, DPREVP, DATCIR, DBUFER, PROD, CIRRAD, RETURN, IPLANE, DEPX, DEPY, EPSLON, FRAPID, SFMFLG

OUTPUT

Flags used: V1 V2, DIV, PT, ERROR, DPRESP, DBFSEG, CODE, SFMCIR, DBUFER

Each quadrant segment is output as a command block.

METHOD

Each quadrant segment is computed from the data stored in DBUFER.

The departures and arc center offsets are computed and stored in DBFSEG.

See Section 3.4.4 for a complete description

REQUIREMENTS

Calls subroutines DOT, SEGDRC, ERDMP1, OUTPUT, GEOM, DEPART

Called by subroutine GOCIRC

RESTRICTIONS

None

QUADET (QDETGT)
(GETERP)

PURPOSE

To compute the quadrants in which the first and last points of a given circle lie, and depending on the circle direction, compute the coordinates of the points on this circle as it passes through quadrants.

INPUT

CALL QUADET (IC, JC, KC)

If circle lies in XY plane, then: IC =1, JC =2, KC =3.

If circle lies in YZ plane, then: IC =2, JC =3, KC =1.

If circle lies in ZX plane, then: IC =1, JC =3, KC =2.

OUTPUT

DBUFER contains the circle quadrant change points and the end point. KTR contains the number of points in DBUFER.

METHOD

The given circle is translated to the origin. Subroutine QUADNT is called for the first and last points to determine which quadrant these points fall in. Depending on the direction of motion, the number of quadrants are calculated and the coordinates of the quadrant change points are also calculated. The points are then translated back to the original position of the circle and stored into DBUFER .

DIAGNOSTICS

None

REQUIREMENTS

Calls subroutine QUADNT

Called by subroutine CIRINT

RESTRICTIONS

The subroutine contains a DATA Statement.

QUADNT (QDNTGT)
(GETERP)

PURPOSE

To determine the quadrant in which a given point lies

INPUT

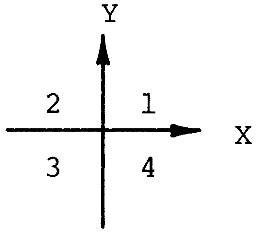
CALL QUADNT(ICD, PX, PY, Q)

where:

ICD indicates either the first (1) or last (2) point of the circle; and PX and PY are the plane coordinates of the circle. The COMMON flag CIRDIR gives the circle direction.

OUTPUT

Q = quadrant number

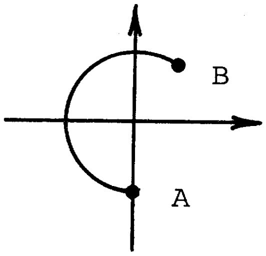


METHOD

The signs of PX and PY are tested to determine the quadrant.

Note: this routine is called twice in succession with two points on an arc.

If either of these points is on an axis, the point is analyzed as if it were displaced along the arc between the two points. For example; A is taken to be in quadrant 3.



DIAGNOSTICS

None

QUADNT (cont'd)

REQUIREMENTS

Calls no subroutines

Called by subroutine QUADET

RESTRICTIONS

The subroutine contains a DATA Statement.

RAPID (RAPGB)
(GEBASE)

PURPOSE

To set the flag to indicate a rapid move

INPUT

CALL RAPID

OUTPUT

The flag RAPFLG is set for a rapid traverse condition.

DIAGNOSTICS

None

REQUIREMENTS

Calls no subroutines

Called by subroutines AUXLRY, FEDRAT

RESTRICTIONS

Multiple entries RAPIDX, RAPIDO, ENTRAP

RAPIDO (RAPOGB)
(GEBASE)

PURPOSE

To set the flags so that the path motion will be at rapid traverse

INPUT

CALL RAPIDO

Flags used: OPTAB, FLRPON, DPATH, MULTHD, IHEAD, TABLEM,

IFDRNG, MCHCON.

OUTPUT

Flags used: RAPLOW, RAPFLG, FLRPON, FRAPID, RAPRNG

A command block is output with the M code which puts the machine into the rapid traverse range.

METHOD

Shift and dwell blocks are set up if necessary.

DIAGNOSTICS

None

REQUIREMENTS

Calls subroutines PLENTH, TEST1, DWELL, MACSRT

Called subroutines by RAPIDP, RETRCT, TSTFLG

RESTRICTIONS

This subroutine is a multiple entry to RAPID.

RAPIDP (RAPPGP)
(GEPOS)

PURPOSE

To process the simulation of an R register on a positioning machine

INPUT

CALL RAPIDP(Z,R)

where:

Z is the programmed depth

R is the requested rapid traverse distance

OUTPUT

Flags used: FRKRAP, DBFSEG(2,5,11), CYCFLG, CODE, FLRPON.

The requested value of R is output as a Z move.

METHOD

The R distance is output as Z using a rapid traverse. The remaining part of the motion is then output in Z at the programmed feedrate.

DIAGNOSTICS

None

REQUIREMENTS

Calls subroutines OUTPUT, RAPIDX, RAPIDO, SET12

Called by subroutine CYCLGP

RESTRICTIONS

None

RAPIDX (RAPXGB)
(GEBASE)

PURPOSE

To turn off the rapid flag and output M codes for gear shifting, if necessary.

INPUT

CALL RAPIDX

Flags used: OPTAB, FLRPON, MULTD, IHEAD, TABLEM,
IFDRNG

OUTPUT

Flags used: FLRPON, RAPRNG, FRAPID, MCHCON

A command block is output with an M code which causes the machine to shift into the feedrate range. An M code calling for a gear shift would not be output if the machine's shifting mechanism were not under tape control.

DIAGNOSTICS

None

REQUIREMENTS

Calls subroutines DWELL, STOREM, MACSRT

Called by subroutines CYCLEL, CYCLGP, CYCLGX, FEDRAT,
RAPIDP, TSTFLG

RESTRICTIONS

This subroutine is a multiple entry in subroutine RAPID.

RAPLIM (RAPLIM)
(GEMULT)

PURPOSE

To test the motion over the path to see if it can be made at rapid traverse

INPUT

CALL RAPLIM(AAA,IHD)

where:

AAA is the array containing the incremental move

IHD is the head number

OUTPUT

If the motion cannot be at rapid, the dwells and M codes for shifting to the feedrate gear are output (if needed). If the motion can be at rapid, the dwells and M codes for shifting into rapid are output (if needed).

METHOD

The dwells and M codes for shifting to the feedrate gear are output if:

(1) The path is shorter than that specified by option 37

(2) The path is so short as to be tape reader limited

If either of these two conditions exist, the motion will be at feedrate.

DIAGNOSTICS

None

RAPLIM (cont'd)

REQUIREMENTS

Called by subroutines DRETHD, PARK, and RETHD

Calls subroutines FEDM, RAPM, TEST2 and SRAREC

RESTRICTIONS

None

RAPM (RAPM) (GEMULT)

PURPOSE

To output the necessary M codes for a rapid traverse move

INPUT

CALL RAPM(AAA, IH, INTA, INTD)

where:

AAA is the array containing incremental moves

IH is the head number

INTA is the index to use for values in TABLEM

INTD is a shifting flag: 1 = shift into feed
2 = shift into rapid

The miscellaneous code table, TABLEM is also used.

OUTPUT

The necessary M codes for shifting into rapid or shifting into feed are output.

METHOD

The number of records of M codes are optimized by determining which axes have motion and which axes have a common M code.

DIAGNOSTICS

If there is no motion, subroutine PERROR is called.

REQUIREMENTS

Called by subroutines RAPLIM and SHFTBK

Calls subroutines OUTB, CTCHUP, and PERROR

RESTRICTIONS

Subroutine FEDM is an entry point in RAPM.

REDTAP (REDYGI)
(GEINIT)

PURPOSE

To ready the tape by rewinding and buffering

INPUT

CALL REDTAP

COMMON parameters used: CLTAPE, TAPES1, TAPES2, TAPES3, TAPES4

OUTPUT

The tapes are rewound and buffered.

DIAGNOSTICS

None

REQUIREMENTS

Called by subroutine INIT

Calls subroutines REWZ, BUFFTP and WEFREW

RESTRICTIONS

This subroutine is computer oriented.

Requires APT system COMMON

RESTAT (REINGB)
(GEBASE)

PURPOSE

To reinstate upon a start-up the part program conditions which are automatically turned off at a STOP, OPSTOP, BREAK, or END statement

INPUT

CALL RESTAT

The following parameters and arrays in COMMON are used: OPTAB, STATE, TABLEM.

The STATE table contains the condition state of the program at the time prior to the STOP, OPSTOP, BREAK, or END statement.

OUTPUT

A DBFSEG command block is output for each "turned back on" condition. This can include a reinstatement of the coolant, spindle direction and speed, SFM value, feedrate or rapid range, and the turret or tool code.

METHOD

The STATE vector has an ordered structure as follows:

STATE(1) = T code

STATE(2) = SPNCOM

STATE(3) = Spindle Conditions (CLW,CCLW,OFF)

STATE(4) = Spindle Range

STATE(5) = SFM value

STATE(6) = M code for feed or rapid range

STATE(7) = M code for feed or rapid range if two codes are required

STATE(8) = Coolant #1

STATE(9) = Coolant #2

RESTAT (cont'd)

METHOD (cont'd)

In the related subroutine as each condition is established, e.g., the spindle speed and direction in subroutine SPINDL, the established condition is saved in the appropriate location of the STATE vector. If a "stop" condition exists (i.e., a STOP, OPSTOP, BREAK, or END had been given), the newly established condition is made negative before storing it into the STATE vector.

In subroutine RESTAT, the STATE vector is scanned; and for every non-negative condition, a reinstatement output block is issued.

DIAGNOSTICS

None

REQUIREMENTS

Calls subroutines STOREM and DWELL

Called by subroutine TSTFLG

RESTRICTIONS

None

RETHD (RETHD)
(GEMULT)

PURPOSE

To return the specified head to the work when the alternate head is parked

INPUT

```
CALL RETHD(AAA, ABS, BBS, SEQN, OUTX, OUTY, OUTZ, SAFEV,
          IHD, SAFEW)
```

where:

AAA = array containing incremental move for head being returned

ABS = absolute coordinate system for head being returned

BBS = absolute coordinate system for alternate head

SEQN = sequence number for head being returned

OUTX = absolute X coordinate where head was parked

OUTY = absolute Y coordinate where head was parked

OUTZ = absolute Z coordinate where head was parked

SAFEV = deltas used to park the head and return it

IHD = head number

SAFEW = deltas used to withdraw the head concerned

The miscellaneous code table, TABLEM, is also used.

OUTPUT

The specified head is returned to the work and the coolant turned back on.

RETHD (cont'd)

METHOD

(1) Save the incremental move in AAA.

(2) Position the head at the same X value as when it was parked, if it is not already there.

(3) Restore the absolute coordinate system for the head being returned.

(4) Return the parked head to the withdrawal position, turn on the coolant, and then move the head from the withdrawal position back to the workpiece.

DIAGNOSTICS

None

REQUIREMENTS

Called by subroutine FXPARK

Calls subroutines GMOUT, COMPGC, RAPLIM and SRAREC

RESTRICTIONS

None

RETRCT (RTRCGP) (GEPOS)

PURPOSE

To retract the tool at rapid traverse to the height specified in the CLRSRF statement

INPUT

CALL RETRCT

The input arrays ICLDAT and CLDATA are used.

OUTPUT

A block to raise the spindle to the CLRSRF height is output.

METHOD

Determine the Z height (DBFSEG(5)) by adding the programmed CLRSRF height to the TRANS value TRANSL(3). Determine if the motion can take place at RAPID; if so, set feedrate (DBFSEG(11)) equal to -OPTAB(44) (the rapid rate); set the rapid flag RAPFLG = 1 and call subroutine RAPIDO. If the motion cannot take place at the rapid rate, set the feedrate DBFSEG(11) to the high feedrate OPTAB(39). Store the now current Z height in DPREVP(3) and save the programmed CLRSRF in CURNTZ. Output the above block as a CODE = -16 (a Z motion only block).

DIAGNOSTICS

None

REQUIREMENTS

Called by subroutine AUXLRY

Calls subroutine RAPIDO

RESTRICTIONS

A CLRSRF statement must be programmed prior to a RETRCT statement.

Machine must have a programmable Z register.

RETRET (RETRET)
(GEMULT)

PURPOSE

To return the secondary head to the cutter location from which it was withdrawn when a restriction had been found which prevented operation in the combined mode.

INPUT

CALL RETRET

The following COMMON parameters are input:

AS2	the present record being processed for head 1
AS3	the present record being processed for head 2
PRIMHD	= 1 for head 1 the primary head
	= 2 for head 2 the primary head
SFHOLD	secondary head retraction record when head is removed from combined cutting

OUTPUT

A record is output to return the secondary head to the cutter location.

METHOD

When the secondary head was withdrawn from the combined mode, the incremental motion for withdrawal was left stored in either AS2 or AS3. After withdrawal, subroutine GMOUT reverses the sign of the coordinate axes. Subroutine RETRET now calls subroutine GMOUT to return the secondary head to its cutter position. The next cutter motion statement had previously been stored in the buffer SFHOLD. This motion statement is now restored into either AS2 or AS3.

DIAGNOSTICS

None

RETRET (cont'd)

REQUIREMENTS

Called by subroutine GEMULT

RESTRICTIONS

None

RETSFY (RETSFY)
(GEMULT)

PURPOSE

To set up a data record to withdraw the secondary head from its cutting sequence in the combined mode

INPUT

CALL RETSFY

The following parameters are used:

AS2	The current DBFSEG record for head 1
AS3	The current DBFSEG record for head 2

OUTPUT

A data record is set up to withdraw the secondary head from the combined cutting sequence. The secondary head retraction record is stored in SFHOLD.

METHOD

The flags IDAB(1) and ISAFLG are set to indicate that combined cutting has been restricted. If the movements of either head have been segmented into two records of data, the movements are stored back to their original values. Then a data record is set up to withdraw the secondary head from the cutting sequence. The withdrawal position is found by multiplying the values in the SAFETY statement by 0.05.

DIAGNOSTICS

None

REQUIREMENTS

A SAFETY statement must be given in order to withdraw the secondary head.

Called by subroutines GMCIRL and GMLINE

RESTRICTIONS

None

REWIND (REWDGB)
(GEBASE)

PURPOSE

To output a rewind code and to set the machine flags to the "off" condition

INPUT

CALL REWIND

The input arrays ICLDAT and CLDATA are used.

OUTPUT

The rewind code from TABLEM(31) is output in a block by itself.

METHOD

Call subroutine STOREM(130) to output any pending M codes. Test OPTAB(97) = 1 to determine if a T00 block is to be output. If so, set T code (DBFSEG(13)) to 0 and call subroutine DWELL(92), then STOREM(31) for rewind code. If STOREM(31) is not available, output comment. If code STOREM(31) is available, set the current spindle range CURNG = 1, turn off the spindle FLONSP = 0, and turn off the coolant FLONKL = 0. Output the M code = -1 block (a stop function in a block by itself).

DIAGNOSTICS

Comment: REQUESTED MISCELLANEOUS FUNCTION CODE IS NOT AVAILABLE ON THIS MACHINE.

REQUIREMENTS

Called by subroutine AUXLRY

Calls subroutines STOREM, DWELL, OUTPUT and COMENT

RESTRICTIONS

TABLEM(31) ≠ DMBITS

RFTYPE (RFTYPE)
(GEPOS)

PURPOSE

To determine the feedrate for the rotary table or to calculate its command form

INPUT

CALL RFTYPE (ROTFED,K)

where:
ROTFED = given feedrate

K = 0 when called from subroutine POSIT

= 1 when called from subroutine ROTABA

OUTPUT

ROTFED contains the feedrate or feed command depending on the value of K.

METHOD

If the machine has a separate feedrate register (OPTAB(141) = N), call the related feedtype subroutine.

DIAGNOSTICS

None

REQUIREMENTS

Called by subroutines POSIT, ROTABA

Calls subroutine ROTYP1

RESTRICTIONS

OPTAB(1) must equal 1 or 2.

ROTABA (RTBAGP)
(GEPOS)

PURPOSE

To process a rotary motion (table or head) and set up values for output for an Absolute System machine.

INPUT

CALL ROTABA(ROTDIR, INCABS, K, ROTFED)

where:

ROTDIR = direction of rotation

INCABS = 1 for absolute system

K = storing index

ROTFED = rotary feedrate

Flags used: FORKT, ROTRAD, DBFSEG, CODE, RAPFLG, ROTDIR

METHOD

In subroutine ROTABA, the shorter rotation distance is chosen and used since this is the distance which will be selected by the NC control system. Therefore, if the programmed direction is the long way, the proper move must be forced by segmentation. Subroutine ROTABA does its own segmentation.

DIAGNOSTICS

None

REQUIREMENTS

Calls subroutines STOREM, CONROT, SROREC, RFDTYP, OUTPUT, SET12

Called by subroutine ROTABL

RESTRICTIONS

None

ROTABI (RTBIGM)
(GEMILL)

PURPOSE

To process a rotary move (head or table) and set up values for output for an Incremental System Machine

INPUT

CALL ROTABI (ROTREF, ROTDIR, INCABS, K, NEXTFL)

where:

ROTREF	=	0 Do not rotate the reference system
	=	1 Rotate the reference system
ROTDIR	=	+1 Clockwise direction
	=	-1 Counterclockwise direction
INCABS	=	0 incremental motion
	=	1 absolute motion
K	=	the location in machine coordinate vector containing the rotary value.
NEXTFL	=	0 output rotary move now
	=	1 output rotary move in the next block

The following parameters in COMMON are used:

AXMULT, OPTAB, PREVMP, CLDATA, MAFORK, DPREVP, PRVPOS, OPTAB

OUTPUT

Flags used: AXMULT, PRVPOS, ROTRAD, ROTDIR, DPRESP, PRVPOS, DPRESM, XTEMP, ROTPOS, DPREVP

The incremental rotary motion is output as a command block.

ROTABI (cont'd)

METHOD

The control for an incremental machine moves CLW or CCLW according to the sign of the output increment. (This is specified in option 120). This routine does its own segmentation.

DIAGNOSTICS

Comment: ROTREF AND ATANGL CANNOT BE USED

REQUIREMENTS

Calls subroutines CONROT, COMENT, CLASS, GOLINE, ROTOUT

Called by subroutine ROTABL

RESTRICTIONS

Uses a DATA statement.

If the absolute position is not retained, ROTREF and ATANGL cannot be used.

ROTABL (RTBLGB)
(GEBASE)

PURPOSE

To process the ROTATE/TABLE or HED statement

INPUT

CALL ROTABL

The following parameters and arrays in COMMON are used: AXMULT, CLDATA, OPTAB, ROTFED, DPRESM, DPRESP

OUTPUT

Flags used: ROTDIR, ROTREF, INCABS, TABSPD, ITABSD, K,
ROTFED, FEDIPM, DPREVM, PREVPT, AXMULT

The rotary motion is translated into a command block which is output.

METHOD

The ROTATE/ statement is first scanned for modifiers. Then the parameters K (head or table), INCABS (ATANGL or INCR), ROTDIR (CLW or CCLW), ROTREF, and ITABSD are set accordingly. In a positioning - absolute system the present point is stored as the previous point after subroutine ROTABA is called.

DIAGNOSTICS

None

REQUIREMENTS

Calls subroutines TABSPD, FLOAT, ROTABI, ROTABA

Called by subroutine ROTATE

RESTRICTIONS

None

ROTATE (RTATGB)
(GEBASE)

PURPOSE

To determine the module (head, table, torch, magazine, turret, or indexer) given in the ROTATE statement, and to call the respective subroutine

INPUT

CALL ROTATE

The input arrays ICLDAT and CLDATA are used.

OUTPUT

The appropriate subroutine as designated by the minor word modifier is called.

METHOD

CLDATA(4) is checked to determine the module referenced. If the turret is referenced, there is a check to see if it is on a drafting machine.

DIAGNOSTICS

Comment: OPTION FEEDRATE IS ASSUMED.

REQUIREMENTS

Calls subroutines ROTORC, ROTABL, ROTMAG, ROTUR, ROTDRF,
COMENT, ROTIND

Called by subroutine AUXLRY

RESTRICTIONS

None

ROTDRF (RDRFGO) (GEPOS)

PURPOSE

To rotate a turret on a drafting machine

INPUT

CALL ROTDRF

OUTPUT

A command block which rotates the turret is output.

DIAGNOSTICS

Comment: WARNING -- TURRET INDEXED WHILE PEN IS DOWN -- WARNING.

REQUIREMENTS

Calls subroutines COMENT, OUTPUT

Called by subroutine ROTATE

RESTRICTIONS

None

ROTIND (RINDGP)
(GEPOS)

PURPOSE

To output a sequence of blocks which rotate an indexer right or left to some requested point

INPUT

CALL ROTIND

OUTPUT

One or more M code blocks are output.

METHOD

If the direction of rotation is not specified, right is assumed.

Output n number of M code blocks as specified by CLDATA(5), by calling STOREM(48) for right, or by calling STOREM(49) for left. Then call STOREM(130) to output the past requested M code.

DIAGNOSTICS

None

REQUIREMENTS

Called by subroutine ROTATE

Calls subroutine STOREM

RESTRICTIONS

TABLEM(48,49) ≠ DMBITS

OPTAB(1) must be equal to 1 or 2

ROTMAG (RMAGGP)
(GEPOS)

PURPOSE

To select the M code for rotation direction of the tool changer magazine

INPUT

CALL ROTMAG

OUTPUT

A block with the direction M code and T code for magazine selection is output.

METHOD

If no tool (magazine position number) is given assume the tool in the up position. Store tool selected in HOLD and determine if a direction of rotation was specified. If not, call subroutine MINMOV to determine the closest direction. Store TABLEM(83) for CLW or TABLEM(84) for CCLW. The tool number is stored in DBFSEG(13) and the block is output as a code = -1.

DIAGNOSTICS

None

REQUIREMENTS

Called by subroutine ROTATE

Calls subroutines STOREM, MINMOV, OUTPUT

RESTRICTIONS

TABLEM(83,84) ≠ BITS

OPTAB(1) must equal 1 or 2

ROTMOV (RMOVGM)
(GEMILL)

PURPOSE

To determine and process the rotary departure for milling multiaxis machines

INPUT

CALL ROTMOV

COMMON parameters used: DPREVM(4), DPREVM(5), DPRESM(4),
DPRESM(5), ROTUNT, DEPA and DEPB

OUTPUT

Flags used: CLMPFL, RETURN and LOCK

The parameters DEPA and DEPB contain the rotary departures which are also stored into DBFSEG(6) and (7).

DEPA = DPRESM(4) - DPREVM(4)

DEPB = DPRESM(5) - DPREVM(5)

DIAGNOSTICS

None

REQUIREMENTS

Called by subroutines DEPART and ROTHED

Calls subroutine STOREM

RESTRICTIONS

This subroutine uses DATA statements.

ROTORC (RTORGF)
(GEFLAM)

PURPOSE

To process the ROTATE/TORCH statement

INPUT

CALL ROTORC

The following parameters and arrays in COMMON are used:

UPFLAG, CLDATA, CODE, ICLDAT

OUTPUT

A command block which rotates the torch is output.

METHOD

If necessary, this routine segments the rotation. The output value is rounded and truncated. CODE is set to +2.

DIAGNOSTICS

Comment: WARNING -- TORCH ROTATION HAS EXCEEDED 360 DEGREES -- WARNING.

Comment: TORCH ROTATION IS TOO LARGE TO BE PUT IN A BLOCK WITHOUT SEGMENTATION.

REQUIREMENTS

Calls subroutines COMENT, SRAREC, OUTPUT

Called by subroutine ROTATE

RESTRICTIONS

None

ROTOUT (ROUTGM)
(GEMILL)

PURPOSE

To output a rotary move command block.

INPUT

CALL ROTOUT (XTEMP, N)

where:
XTEMP = rotary incremental move

N = DBFSEG storage index

OUTPUT

Flags used: FLAG, DBFSEG, CODE, XTEMP

The rotary move command block is output.

METHOD

The A and B registers are set to zero; then DBFSEG(N) is set to the output value XTEMP. Losses caused by rounding procedures are accumulated and checked here.

DIAGNOSTICS

None

REQUIREMENTS

Calls subroutines SROREC, OUTPUT

Called by subroutine ROTABI, ROTHED

RESTRICTIONS

Uses a DATA statement

ROTUR (RTURGP)
(GEPOS)

PURPOSE

To rotate the spindle turret into position for tool pickup or tool unload, and to move the spindle into machining position

INPUT

CALL ROTUR

Flags used: TABLEG, TOOLDN, TLNOFF, DNLEN, TLEN2, OPTAB

OUTPUT

DBFSEG, TOOLDN, DNLEN, TLEN, TRANSL, VALUEM

A command block which rotates the turret into position is output.

METHOD

If called for, the tool length compensation is computed, and a tool offset M code is calculated.

DIAGNOSTICS

None

REQUIREMENTS

Calls subroutine STOREM

Called by subroutine ROTATE

RESTRICTIONS

None

ROTYP1 (ROTYP1)
(GEPOS)

PURPOSE

To locate the closest feedrate available on the machine or to find that feedrate command value for a type 1 rotary feedrate

INPUT

CALL ROTYP1 (ROTFED, K)

where: ROTFED = programmed feedrate in IPM

K = 0 obtain feedrate command

K = 1 find closest feedrate to ROTFED

METHOD

Initially, KI is set equal to the rotary feedrate register position in DBFSEG, (OPTAB(139)). Compute the linear feedrate maximum CIRCUM in inches per minute for some specified radius using the maximum feedrate in RPM (OPTAB(114)) and the part radius OPTAB(112). Compute a ten speed table where FROT(I) = CIRCUM/(1.43 ** (10-I)). When K = 1, the subroutine is entered to find the closest feedrate available. This value from FROT is returned as ROTFED. Conversely, when K = 0, the subroutine is entered to calculate the feed command for the input ROTFED value. The converted command value is returned as ROTFED.

DIAGNOSTICS

None

REQUIREMENTS

Called by subroutine RFTYPE

Calls no subroutines

RESTRICTIONS

OPTAB(1) must equal 1 or 2.

OPTAB(14) must equal 1.

SADDLE (SADLGL) (GELATH)

PURPOSE

To set up the command blocks for a saddle motion according to the specifications given by the statement

INPUT

CALL SADDLE

The following COMMON variables are used: ICLDAT, CLDATA, VALUEM, TABLEM, TABLEG, OPTAB, SFMFLG and SFMRAD. The modifier code for SELECT is 1074. The minor words in the statement have these code numbers: HED(238), SADDLE(150), TRAV(154), IPM(73) and IPR(74).

OUTPUT

The saddle motion output blocks are stored in DBFSEG.

The following flags are set:

SADMOD = -1	for feedrate in IPR
SADMOD = 1	for feedrate in IPM
SADSFM = 1	for saddle SFM
SADSFM = 0	after SFM is called

METHOD

The saddle move is stored in DBFSEG(16). If the saddle move exceeds the departure maximum the move is segmented by calling subroutine SEGSAD. The modifier TRAV specifies that the saddle moves at rapid traverse. If a value is given for feedrate, then IPM or IPR specifies the mode.

DIAGNOSTICS

The comment OPTION FEEDRATE MODE ASSUMED is printed when the feedrate mode is not given. This is a recoverable warning and does not terminate processing.

SADDLE (cont'd).

REQUIREMENTS

Called by subroutine SELHED.

Calls subroutines SFMO, DWELL, COMENT, OUTPUT, SEGSAD,
SRAREC, and STOREM

RESTRICTIONS

None

SAFEGM (SAFEGM)
(GEMILL)

PURPOSE

To process the SAFETY statement for multihead milling machines.

INPUT

CALL SAFEGM

The input arrays ICLDAT and CLDATA are used.

OUTPUT

The DBFSEG array is output with:

```
          DBFSEG(2)  = NOP

         DBFSEG(11)  = 7.0

               CODE  = 17.
```

DIAGNOSTICS

None

REQUIREMENTS

Calls subroutine SAFEGM

Calls subroutine OUTPUT

RESTRICTION

None

SAFEGX (SAFEGX)
(GEMAXS)

PURPOSE

To establish the R register value for a SAFETY statement for NC machines possessing this feature

INPUT

CALL SAFEGX

The input arrays ICLDAT and CLDATA are used.

OUTPUT

A DUFSEG command block containing the R register value is output.

METHOD

If OFF is given, DBFSEG(16) is set to zero and nothing else is done. The R value is determined by a three digit combination derived from the following table:

	X	Y	Z
R111	+.01 inches	+.01 inches	+.01 inches
R222	-.01 inches	-.01 inches	-.01 inches
R333	+.10 inches	+.10 inches	+.10 inches
R444	-.10 inches	-.10 inches	-.10 inches
R555	+1.0 inches	+1.0 inches	+1.0 inches
R666	-1.0 inches	-1.0 inches	-1.0 inches

Any one or any combination of axes may be programmed by proper arrangement of the R address. For example, address R304 would indicate that the X axis is to move .1 inch in the plus direction and the Z is to move .1 inch in the minus direction. The programmer is restricted to calling for only one of the three values given (.01, .1, 1.0).

DIAGNOSTICS

None

SAFEGX (cont'd)

REQUIREMENTS

Calls subroutine OUTPUT

Called by subroutine AUXLRY

RESTRICTIONS

The subroutine contains a DATA statement.

SAVMCD (SAVMCD)
(GEMULT)

PURPOSE

To save the M code for feed or rapid

INPUT

CALL SAVMCD(AAA,INDX)

where:

AAA = current DBFSEG record being processed

INDX = head number

The miscellaneous function table TABLEM is also used.

OUTPUT

The M code for the feed or rapid range is stored in FDMHD.

METHOD

The M code in AAA is compared against the feed and rapid M codes. If AAA is found to have a feed or rapid M code, it is stored in FDMHD.

DIAGNOSTICS

None

REQUIREMENTS

Called by subroutine CREAD

RESTRICTIONS

None

SEG (SEG)
(GEMULT)

PURPOSE

To segment a cut vector

INPUT

CALL SEG(AAA,XB,XA)

where:

AAA = array containing the cut vector to be segmented

XB = absolute x coordinate at termination of alternate cut vector

XA = absolute x coordinate of the cut vector to be segmented

OUTPUT

The segmented vector is stored into AAA, the first portion in AAA(I,1) and the remainder in AAA(I,2).

METHOD

The cut vector in AAA is segmented at XB. The remainder of the cut vector is set up as a separate motion statement in the second column of AAA. All new delta moves are rounded to the step size.

DIAGNOSTICS

None

REQUIREMENTS

Called by subroutine FXMULT

Calls subroutines COMPGC and SRAREC

RESTRICTIONS

None

SEGDRC (SDRCGX)
(GEMAXS)

PURPOSE

To "segment" the direction cosines in accordance with the segmentation of the X,Y,Z coordinate values.

INPUT

CALL SEGDRC(PREV, PRES, ANS)

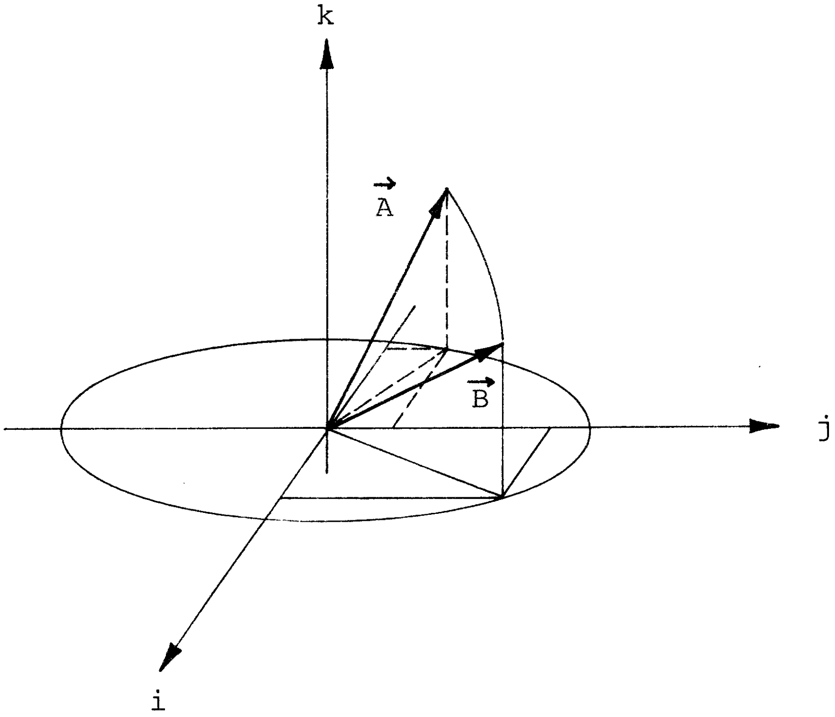
where:

PREV = first point of the path

PRES = last point of the path

ANS = the resultant direction cosine values

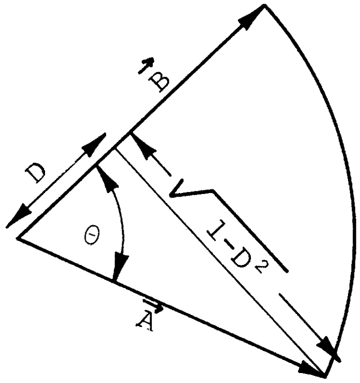
Let $\vec{A}$ and $\vec{B}$ be the previous and present tool axis vectors respectively.



SEGDRC (cont'd)

INPUT (cont'd)

Consider the plane of A,B below:



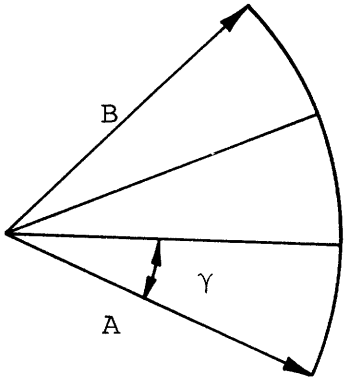
Angle Θ is obtained by

$$\Theta = \tan^{-1}\left[\sqrt{\frac{|1 - D^2|}{D}}\right]$$

where: $D = \vec{A} \cdot \vec{B}$ →

Note: $|\vec{A}| = |\vec{B}| = 1$

The arc of the tool tip is divided into equal segments thus:



$\gamma = \frac{PT * \Theta}{NDIV}$; PT = 1

SEGDRC (cont'd)

INPUT (cont'd)

The segmented direction cosines are given by a linear proportion between the cosines of $\vec{B}$ and $\vec{A}$:

$$i_S = \alpha i_A + \beta i_B$$

$$j_S = \alpha j_A + \beta j_B$$

$$k_S = \alpha k_A + \beta k_B$$

where:

$$\alpha = \frac{\sin\ (\Theta - \gamma)}{\sin\ \Theta} \quad \text{and} \quad \beta = \frac{\sin\ \gamma}{\sin\ \Theta}$$

DIAGNOSTICS

None

REQUIREMENTS

Calls subroutines DOT, ARCTAN, NORM

Called by subroutines LINRTY, PROCGD, SEGMNT

RESTRICTIONS

None

SEGMNT (SGMTGT)
(GETERP)

PURPOSE

To segment a move which has a component departure larger than the maximum allowed

INPUT

CALL SEGMNT

Flags used: AXMULT, PRESP, PREVP, DPRESM, DPREVM, FRAPID,
SFMFLG, THFLAG, OPTAB

OUTPUT

Flags used: DPRESM, DPRESP, SFMCIR, DPREVM, DPREVP,
TOLCON, DEPX, DEPY, DEPZ, DEPA, DEPB

The original path is segmented and output.

METHOD

The largest linear and (if present) the largest rotary departure are determined. The number of linear segments and the number of rotary segments required are calculated. The larger of these two is used. Rounding is done in the buffer while exact values are kept for the deltas and the machine and part vectors.

DIAGNOSTICS

None

REQUIREMENTS

Calls subroutines MACSRT, SROREC, SEGDRC, SRAREC, OUTPUT,
THREDO, THREDM

Called by subroutines GOLINE, LOAD

RESTRICTIONS

None

SELHED (SLHDGB)
(GEBASE)

PURPOSE

To select the designated head by outputting the appropriate M code

INPUT

CALL SELHED

The input arrays ICLDAT and CLDATA are used.

OUTPUT

Flags used: RAPLOW, FRAPID, RAPFLG, RAPRNG, FLRPON, TLHEAD, MULTHD.

The appropriate M code is output in a block by itself.

METHOD

The subroutine also processes the following statements:

```
           1
SELECT/HED,2
           3

SELECT/HED,BAR

SELECT/HED,SADDLE
```

DIAGNOSTICS

None

REQUIREMENTS

Called by subroutine AUXLRY and SELECT

Calls subroutines OUTPUT, SADDLE and STOREM

RESTRICTIONS

This subroutine uses DATA statements.

SELECT (SLCTGB)
(GEBASE)

PURPOSE

To process the generic APT statement SELECT

INPUT

CALL SELECT

OUTPUT

The subroutine specified by the minor modifier of the SELECT statement is called.

DIAGNOSTICS

None

REQUIREMENTS

Called by subroutine AUXLRY

Calls subroutines SELOFS, SELTAB, SELPAL, SELRDR, SELHED, SELTUL, SELANG, FEDOVR

RESTRICTIONS

None

SELG (SELG)
(GETERP)

PURPOSE

To select the motion preparatory G code

INPUT

CALL SELG

The following parameters and arrays in COMMON are used: CODE, DBFSEG, RNGDEP, TABLEG, STEP, AXMULT, OPTAB, STEP

OUTPUT

The proper dimensional G code is stored in DBFSEG(2). The parameter GDIMUL contains the related dimensional constant.

METHOD

The departure is tested against the departure limits to determine the dimension multiplier and the G code to be used. A check is made for a variable X,Y,Z,A,B format.

DIAGNOSTICS

Error #138:	No G code is available for the segment length.
Error #139:	Segment length is too long - it should have been segmented.

REQUIREMENTS

Calls subroutines SELGCR, ERDMP1, SELGRO, SRAREC

Called by subroutine OUTPUT

RESTRICTIONS

None

SELGCD (SELGCD)
(GEMULT)

PURPOSE

To select the appropriate preparatory G code as a function of the path length, path condition (linear or circular), and G code availability

INPUT

CALL SELGCD(JMN)

where:

JMN = 1 linear motion;

= 2 circular motion.

The following COMMON variables are also used:

BUFPRE	the current buffer being processed
RNGDEP	the departure limits for G codes
TABLEG	table of preparatory functions
OPTAB	the option table

OUTPUT

The appropriate G code is stored in BUFPRE(2).

SELGCD (cont'd)

METHOD

The G codes are selected according to the following lengths:

English System

0	Path Length	.0999
.1	Path Length	.9999
1.0	Path Length	9.9999
10.0	Path Length	99.9999
100.0	Path Length	999.9999

Metric System

0	Path Length	.999
1.0	Path Length	9.999
10.0	Path Length	99.999
100.0	Path Length	999.999

DIAGNOSTICS

The following unrecoverable errors are printed:

ERROR 138: no G code is available for the segment length; turret corrective motion may be too large.

ERROR 139: segment length is too long - it should have been broken up.

ERROR 141: ICODE has a forbidden value.

ERROR 142: circle radius is too large.

ERROR 143: linear code number for circle

ERROR 144: no G code is available for this radius size.

SELOCD (cont'd)

REQUIREMENTS

Called by subroutines CIRSEG, COMPGC, and SPLIT

Calls subroutine CONVRT and PERROR

RESTRICTIONS

None

SELGCR (SLGRGT)
(GETERP)

PURPOSE

To select the preparatory function G code for a rotary motion.

INPUT

CALL SELGCR

The following parameters and arrays in COMMON are used:
CIRRAD, RNGDER, OPTAB, TABLEG, DBFSEG

OUTPUT

The selected G code is stored in DBFSEG(2); the related dimension multiplier value is stored in the parameter DIMULT.

METHOD

The G code selected depends upon the size of the radius and the circle direction given by CODE which is stored in DBFSEG(15)). The dimension multiplier is set when the G code is selected.

DIAGNOSTICS

ERROR 142: Circle radius is greater than maximum departure.

ERROR 144: No G code is available for the radius size.

REQUIREMENTS

Calls subroutine ERDMP1

Called by subroutine SELG

RESTRICTIONS

None

SELGRO (SLGRGM)
(GETERP)

PURPOSE

To select the rotation G code

INPUT

CALL SELGRO

The following parameters and arrays in COMMON are used: DBFSEG, TABLEG, DPREVM

OUTPUT

The proper dimensional G code is stored in DBFSEG(2). The parameter GDIMUL contains the related dimensional constant.

METHOD

The rotary departure is tested to determine whether to use long, normal, short, or very short dimension G codes.

DIAGNOSTICS

ERROR 39: Rotary move is greater than maximum
departure - it should have been segmented.

REQUIREMENTS

Calls subroutines CONROT, ERDMP1, SROREC

Called by subroutines OUTPUT, SELG

RESTRICTIONS

Uses a DATA statement

SELOFS (SELOFS)
(GEPOS)

PURPOSE

To output the selected tool offset as specified by the APT statement SELECT/OFSETL

INPUT

CALL SELOFS

The input arrays ICLDAT and CLDATA are used.

OUTPUT

The offset code is stored in DBFSEG(17) and output.

METHOD

The values stored in CLDATA(5-8) are tested and DBFSEG(17) is set accordingly. Output CODE = 1.0 is also set.

DIAGNOSTICS

None

REQUIREMENTS

Calls subroutine OUTPUT

Called by subroutine SELECT

RESTRICTIONS

None

SELPAL (SLPAGM)
(GEMAXS)

PURPOSE

To process the SELECT/PALLET statement

INPUT

CALL SELPAL

The input arrays ICLDAT and CLDATA are used.

OUTPUT

The pallet select M code is output in a block by itself.

DIAGNOSTICS

None

REQUIREMENTS

Calls subroutine STOREM

Called by subroutine SELECT

RESTRICTIONS

None

SELRDR (SLRDGM)
(GMAXES)

PURPOSE

To process the SELECT/READER statement

INPUT

CALL SELRDR

The input arrays ICLDAT and CLDATA are used.

OUTPUT

The reader selected M code is output.

DIAGNOSTICS

None

REQUIREMENTS

Called by subroutine SELECT

Calls subroutine STOREM

RESTRICTIONS

None

SELTAB (SELTAB)
(GEPOS)

PURPOSE

To select the table specified by the APT statement SELECT/TABLE

INPUT

CALL SELTAB

The input arrays ICLDAT and CLDATA are used.

OUTPUT

The selected table code is stored in DBFSEG(16) and is output.

DIAGNOSTICS

None

REQUIREMENTS

Calls subroutines SELTAB, SELTAB, OUTPUT

Called by subroutine SELECT

RESTRICTIONS

None

SELTUL (SLTLGM)
(GEMAXS)

PURPOSE

To output a block which effects the selection of a tool

INPUT

CALL SELTUL

The following parameters and arrays in COMMON are used: TOLCON, CLDATA, TABLEG, GPRSLC, TOLSLC

OUTPUT

The selected tool code is set up in the parameter TOOL. Other parameters may also be set depending upon the NC machine in use.

METHOD

If CLDATA(5) is not 0, items 5, 6, 7 of CLDATA are shifted back to positions 4, 5, 6, respectively. A G code and T code are entered in DBFSEG and subroutine OUTPUT is called with CODE = -4.0. Flag TOLCON is set to -1.0 and the transformation equations are called.

DIAGNOSTICS

Comment: PREVIOUSLY SELECTED TOOL WAS NOT
LOADED - PRESENT TOOL IS SELECTED

REQUIREMENTS

Calls subroutines COMENT, TOOLNO, OUTPUT, CLASS

Called by subroutine SELECT

RESTRICTIONS

None

SEQNO (SEQNGB)
(GEBASE)

PURPOSE

To output a sequence number in accordance with the data given by the SEQNO statement

INPUT

CALL SEQNO

The input arrays ICLDAT and CLDATA are used.

OUTPUT

Flags used: SAVEN and SEQINC

The flags are set so that the parameter SEQCTR will output the desired sequence number.

DIAGNOSTICS

None

REQUIREMENTS

Called by subroutine AUXLRY

Calls no subroutines

RESTRICTIONS

None

SETUP1 (SETUGO)
(GEOUT)

PURPOSE

To set up a line for output in GEOUT1

INPUT

CALL SETUP1

The following parameters and vectors in COMMON are used: FORK, NPR, SKPFLG, BCDIMG, REGFOR, DBFSEG, OPTAB, TABLEG, NPT

OUTPUT

Flags used: NPRT, SKPCOD

The array BCDIMG is set up for printing.

METHOD

Floating point values are converted to BCD and then entered into BCDIMG for printing.

DIAGNOSTICS

None

REQUIREMENTS

Calls subroutines SETLIN, PPUNCH, CONBCD

Called by subroutine GEPRO1

RESTRICTIONS

The subroutine contains a DATA statement.

SFMO (SFMOGL)
(GELATH)

PURPOSE

To produce the appropriate programmed blocks which will give the desired value of cutting speed as specified by the given SFM conditions

INPUT

CALL SFMO

The following parameters and arrays in COMMON are used:

SFMAXR, SPNMAX, DPREVM, DPRESM, ISFMOD, SADSFM, FRMOD, FRAPID,

SYSCON, SFMDES, OPTAB, ISRNGE, NRORNG, SRTAB

OUTPUT

Flags and parameters used:

SFMAXR, SPNMIN, SPNMAX, IFIRST, IEXIT, ISCAN, NOTOCK, ISENSE,

SFMLIM, SFMCON, SPNLIM, SPNSPD, CODE, DPREVM, DPRESM

The current path will be broken up into segments, the size of which will be determined by the SFM desired, and also by the resultant spindle speed which is selected on the basis of the radius and spindle range. As each block is generated, it is stored into the segment buffer.

METHOD

The program inserts points between the computed tool centers in order to modify the spindle speed, increasing it or decreasing it as the case may be, by selecting the next highest (or next lowest) spindle speed in the given range, thereby, in effect, keeping the resultant SFM within the minimum possible variation of the given SFM. This variation limit will be maintained except when the specified limits are exceeded.

SFMO (cont'd)

METHOD (cont'd)

The SFM sequence begins with the selection of that spindle speed which is commensurate with the specified SFM conditions. As each spindle speed is selected, the spindle speed change point is located. This change point is the point at which the spindle must change to another speed in order to maintain the specified SFM. However, it is desirable to change the speed before arriving at the change point; therefore, an optimum shift point is chosen midway between. The segmentation of the current cutter path occurs because of the breaking of the path at these optimum shift points. These segments and their concomitant spindle speeds, feedrates, etc., are the resultant programmed blocks which are stored into the segment buffer.

REQUIREMENTS

Called by subroutine GOLINE, PROCQD, SADDLE, SEGMNT

Calls subroutine DPART, DSRROW, OFFARC, OUTPUT,
TSTSAF, TRUNC

RESTRICTIONS

The subroutine contains a DATA statement.

SHFTBK (SHFTBX)
(GEMULT)

PURPOSE

To shift the gears to their original setting after parking

INPUT

CALL SHFTBK(AAA,IH)

where:

AAA is the present buffer being processed

IH is the head number

The following COMMON parameters are also used:

ISHIFT = 1 Shift from feed to rapid

= 2 Shift from rapid to feed

OUTPUT

The system catch-up time for shifting gears is output along with the M code for shifting gears.

METHOD

Not applicable

DIAGNOSTICS

None

REQUIREMENTS

Called by subroutine PARK

Calls subroutines CTCHUP, RAPM, and FEDM

RESTRICTIONS

None

SHUFFL (SHUFGO)
(GEOUT)

PURPOSE

To shuffle X,Y,Z,I,J,K,A,B data to the order given by Options 59 and 60

INPUT

CALL SHUFFL

COMMON parameters used: TEMP and ISHVEC

OUTPUT

DBFSEG(3,4,5,6,7,8,9,10) are shuffled in accordance with option 59 and 60 settings.

DIAGNOSTICS

None

REQUIREMENTS

Called by subroutines GEPRO1, GEPRO2, and GEPRO3

Calls no subroutines

RESTRICTIONS

None

SPINDL(SPINGB)
(GEBASE)

PURPOSE

To process the SPINDL statement for either an RPM or SFM condition.

INPUT

CALL SPINDL

The following parameters and arrays in COMMON are used:

FORK, OPTAB, CLDATA, FLONSP, SFMRPM, MULTHD, VALUEM, ISRNGE,

SPNDIR, STOPON, RETURN, ICLDAT, TABLEG, SRTAB, SFMAXR, ENCODE,

FRMOD, FEDIPM, SFMAXI

OUTPUT

The flags are set in accordance with the conditions given in the SPINDL statement.

Flags and parameters used:

TURNON, SFMRPM, SPNSPD, SFMFLG, SFMDES, DBFSEG,

CODE, FLONSP, FLONKL, SFMLOK, SFMFLG, STATE,

ISRNGE, SPNCOM, CURNGE, MCHCON, ISFMOD, IDWLFL,

KHEAD, SPNMAX, SPNMIN, SFMAXR, FEDIPM, IFEDFL,

SFMAXI, VALUEM, FORK, CURDIR

SPINDL (cont'd)

DIAGNOSTICS

Comment: IMPROPER FORMAT, STATEMENT SKIPPED

Comment: OPTION VALUE IS ASSUMED FOR THE SPINDLE SPEED

Comment: RPM MODE ESTABLISHED

Comment: SFM MODE ESTABLISHED

Comment: REQUESTED MISCELLANEOUS FUNCTION CODE IS NOT AVAILABLE ON THIS MACHINE

Comment: RANGE REQUESTED IS NOT AVAILABLE, USE HIGHEST

Comment: LOWEST RANGE THAT SPINDLE SPEED FALLS IN IS ASSUMED

Comment: WARNING--SPINDLE DIRECTION HAS CHANGED

Comment: SPINDLE SPEED IS TOO HIGH FOR THREADING

REQUIREMENTS

Calls subroutines STOREM, DWELL, FLOAT, OUTPUT, ERDMP1, COMENT, MACSRT, LOCRNG, TSTEXT, SPTYPE, FTYPE2, FTYPE4, FTYPE6

Called by subroutine AUXLRY

RESTRICTIONS

Uses a DATA statement

SPLIT (SPLIT)
(GEMULT)

PURPOSE

To segment a data record from one head into two segments

INPUT

CALL SPLIT(RATIO, AAA, KCODE, RAD, ARCLEN, SN)

where:

RATIO = percent value indicating how the segment will be subdivided into two segments

AAA = 30 by 2 word buffer containing data to be segmented, and which are stored in (x,1) and (x,2) after segmentation

KCODE = type of data record to be segmented

1 = line cut

5 = dwell

10,11,12 = circular cut

RAD = radius value of circle before segmentation

ARCLEN = arc length before segmentation

SN = chord length before segmentation

OUTPUT

A segmented record for either head is placed in AS2(x,1) and AS2(x,2) or in AS3(x,1) and AS3(x,2).

SPLIT (cont'd)

METHOD

The record to be segmented may be a circle, dwell, or line record. Circle segmentation is done by calling subroutine CIRSEG. A dwell record is segmented by the following calculations:

$$\text{SEG1} = \text{AAA}(3,1) * \text{RATIO}$$

$$\text{SEG2} = \text{AAA}(3,1) - \text{SEG1}$$

TABLEG(5) is stored in both segments, as are also a CODE of 5 and the sequence number.

The line segmentation for 2 or 3 axes is calculated thus:

$$\text{SEG1}(\text{X, Y, or Z}) = \text{AAA}(\overset{3}{\underset{5}{4}},\ 1) * \text{RATIO}$$

$$\text{SEG2}(\text{X, Y, or Z}) = \text{AAA}(\overset{3}{\underset{5}{4}},\ 1) - \text{SEG1}(\text{X, Y, or Z})$$

A G code is then selected for both segments by calling the subroutine SELGCD. Segmentation is then completed by storing the sequence number, feedrate value, and CODE for both segments.

DIAGNOSTICS

None

REQUIREMENTS

Called by subroutines GMCIRL and GMLINE

Calls subroutines CIRSEG, SELGCD, and SRAREC

RESTRICTIONS

None

SPTYPE (STYPGB)
(GEBASE)

PURPOSE

To calculate the spindle command to be stored in SPNCOM

INPUT

CALL SPTYPE

OUTPUT

SPNCOM contains the spindle speed in command form.

METHOD

If the spindle range ISRNGE is not greater than 1, call subroutine LOCRNG to determine the lowest range the requested speed falls in. Then call subroutine DSRROW to find the row number in the range where the speed is stored. If a type 7 spindle is used, (OPTAB(19) = 7), branch to MACSRT for S-code calculation. If not, call subroutine SPNTYP to calculate the S code. If (SPNCOM ≠ DMBITS),store in STATE(2) for possible reinstatement.

DIAGNOSTICS

None

REQUIREMENTS

Called by subroutines COUPLE, SFMO, SPINDL.

Calls subroutines LOCRNG, DSRROW, MACSRT, SPNTYP.

RESTRICTIONS

Machine must have an S register

SRAREC (RNDLGB) (GEMON)

PURPOSE

To scale, round, and recover the given linear value

INPUT

CALL SRAREC (ARG)

where ARG is the number to be rounded

The following parameters and arrays in COMMON are used: OPTAB, STEP

OUTPUT

The rounded value is stored into ARG.

METHOD

$$ARG = AINT \left[\frac{|arg|}{TEMP(1)} + .5001 \right] * \frac{|ARG|}{ARG} * TEMP(1) ,$$

where:

TEMP(1) = STEP

DIAGNOSTICS

None

REQUIREMENTS

Called by subroutines CIRSEG, DRETHD, FROM, FMULT, GEOM3, GEPRO1, GEPRO2, GEPRO3, GETSFC, OFFARC, RADLIM, SADDLE, SEC, SEGMNT, SEGSAD, SELG1, SPLIT, SRFCHK, TRUNC, TURRET.

Calls no subroutines

RESTRICTIONS

Has multiple entry SROREC

SROREC (RNDRGB) (GEMON)

PURPOSE

To scale, round and recover the given rotary value

INPUT

CALL SROREC (ARG)

where ARG is the number to be rounded

The following parameters and arrays in COMMON are used: OPTAB

OUTPUT

The rounded value is stored in ARG.

METHOD

$$\text{ARG} = \text{AINT}\left[\frac{|\text{ARG}|}{\text{TEMP}(1)} + .5001\right]\left[\frac{|\text{ARG}| * \text{TEMP}(1)}{\text{ARG}}\right]$$

where:

TEMP(1) = OPTION 119

DIAGNOSTICS

None

REQUIREMENTS

Calls no subroutines

Called by subroutines GEPRO1, GEPRO2, GEPRO3, ROTABA, ROTOUT, SEGMNT, SELGRO, TRUNC

RESTRICTIONS

SROREC is a multiple entry to SRAREC.

STDMAC (STDMGI)
(GEINIT)

PURPOSE

To set up the Standard Machine conditions

INPUT

CALL STDMAC

OUTPUT

Tables TABLEM, TABLEG, OPTAB, REGSTR, REGFOR, and SRTAB for the Standard Machine are constructed.

DIAGNOSTICS

None

REQUIREMENTS

Calls no subroutines

Called by subroutine INIT

RESTRICTIONS

DATA statements are used to set up the tables.

SRFCHK (SRCHGT)
(GEBASE)

PURPOSE

To determine the plane of the circle for circular interpolation and save the circle center coordinates for later use

INPUT

CALL SRFCHK

OUTPUT

FLAGS SET: CIRFLG, IPLANE, and CIRDAT

METHOD

A record type 3000 is read and SRFCHK is called. If ICLDAT(5) = 4, a circle record has been read and the circle flag CIRFLG is set equal to 1. If not, CIRFLG is set equal to 0, and returned. The plane of the circle IPLANE is set; i.e., IPLANE = 0 for XY, 1 for ZX, and 2 for YZ. The coordinates of the center of the circle are stored in CIRDAT(1-3). These coordinates are rounded to the step size by subroutine SRAREC, and a return is made. CIRDAT is used in the circular interpolation sequence.

DIAGNOSTICS

None

REQUIREMENTS

Called by subroutine GEBASE

Calls subroutine SRAREC

RESTRICTIONS

Circular interpolation is possible in all three planes by using special programming techniques.

STOP (STOPGB)
(GEBASE)

PURPOSE

To output a block for the statement STOP which turns off the coolant, spindle, tape reader, and then stops the numerically controlled machine tool

INPUT

CALL STOP

OUTPUT

Flags used: STOPON, CURNGE, ISRNGE, FLONKL, and FLONSP

The STOP M code is output in a block by itself

DIAGNOSTICS

Comment 22: MISCELLANEOUS FUNCTION CODE NOT AVAILABLE
ON THIS MACHINE

REQUIREMENTS

Called by subroutine AUXLRY

Calls subroutines COMENT, ENTRAP, OUTPUT and STOREM

RESTRICTIONS

This subroutine uses the Multiple Entries BREAK and OPSTOP.

STOPTS (STOPTS)
(GEMULT)

PURPOSE

To store three elements of a given vector into another given vector, but only when the corresponding element of a given row of AMASK is not DMBITS

INPUT

CALL STOPTS (VEC,STOVEC,AMASK)

where:

VEC = vector to be stored

STOVEC = vector to be stored from VEC

AMASK = vector with or without DMBITS in its elements

OUTPUT

The vector is stored in STOVEC.

DIAGNOSTICS

Error 85 is indicated if VEC is found to have a non-zero value when its corresponding element in AMASK is DMBITS .

REQUIREMENTS

Called by subroutine CIRSEG

Calls subroutines ERROR

RESTRICTIONS

None

STOREM (STRMGB)
(GEBASE)

PURPOSE

To store the given auxiliary M function into the parameter VALUEM

INPUT

CALL STOREM (INDXX)

where: INDXX is the index location used to select the specified M code from TABLEM

OUTPUT

VALUEM which is in COMMON, contains the specified M code.

METHOD

If the parameter VALUEM is already occupied when the subroutine is entered, a block containing VALUEM is inserted into the segment buffer before the newly specified M code is stored into VALUEM. In many cases the program calls for TABLEM(130) to be stored into VALUEM. This is simply a device to insure that VALUEM is cleared before further program action can proceed. The result is that VALUEM is set to DMBITS since TABLEM(130) equals DMBITS.

DIAGNOSTICS

None

REQUIREMENTS

Calls subroutine OUTPUT

Called by subroutines AIR, AUXFUN, CLAMP, COMENT, COOLNT, COUPLE, CYCLE, CYCLGP, DELAY, DRESS, END, FEDOVR, FEDRAT, FROM, GEBASE, LEADER, MACHIN, OPSKIP, PICKUP, POSMOV, RAPID, RESTAT, REWIND, ROTABA, ROTABL, ROTIND, ROTMOV, SADDLE, SAFEGL, SELHED, SELPAL, SELRDR, SPINDL, STOP, THREDM, TOOLGM, TOOLL, TOOLNO, TURRET, TYPEn, UNLOAD, and WELD

RESTRICTIONS

None

TABSPD (TABSPO)
(GEBASE)

PURPOSE

To process the rotary table speed

INPUT

CALL TABSPD

The 2000 type statement processed is:

```
                  OFF        CLW
    ROTATE/TABLE, C  , RPM, CCLW, RANGE, n
```

These words have the following code numbers:

ROTATE	1066
TABLE	177
OFF	72
RPM	78
CLW	60
CCLW	59
RANGE	145

The following COMMON variables are used: CLDATA, ICLDAT, OPTAB, NWPR, SRTAB and TABLEM

OUTPUT

The proper M code is stored by calling subroutine STOREM. The table speed and code are merged and stored in DBFSEG(17). A dwell is output when the table changes direction and is a table start command block.

TABSPD (cont'd)

The following flags are set:

TABDIR the table direction:

= -1 for counterclockwise

= 1 for clockwise

IRANGF the table range flag:

= 0 if range not given

= 1 if range is given

METHOD

The table speeds are stored beginning at SRTAB(141). The speed in the table closest to the programmed speed is located.

The ROTATE/TABLE statement is searched for the modifiers OFF, CLW, CCLW, and RANGE. OFF causes the M code for turning off the table to be output. When the table changes direction, a comment to this effect is output followed by a direction M code, table speed command and dwell time all in block. A change in table range causes the following blocks to be output: a stop M code, a message telling the operator which range to shift into, and a direction M code and table command.

DIAGNOSTICS

None

REQUIREMENTS

Called by subroutine ROTABL

Calls subroutines STOREM, COMENT, and DWELL

RESTRICTIONS

Contains DATA statements

TEST1 (TTMXGB)
(GEBASE)

PURPOSE

To test the given segment length and its given feedrate to insure that the segment is sufficiently long for TMAX

INPUT

CALL TEST1 (F,S)

where:

F = given feedrate on the segment

S = given segment length

OUTPUT

F is the proper value required. The original value of F is either acceptable or is made to be the correct value.

METHOD

$F_{TMAX} = S/TMAX$, where TMAX is the maximum tape reader time.

If the given feedrate, F, exceeds F_{TMAX}, F is set equal to F_{TMAX}.

DIAGNOSTICS

None

REQUIREMENTS

Calls no subroutines

Called by subroutine RAPID

RESTRICTIONS

If F is zero or BITS, no testing is done.

TEST2 (TEST2)
(GEMULT)

PURPOSE

To test the given segment length and its given feedrate to insure that the segment is sufficiently long for TMAX

INPUT

CALL TEST2(F,S)

where:

F = given feedrate on the segment

S = given segment length

OUTPUT

F is the proper value required. The original value of F may be acceptable, but if not, it is made to be the correct value.

METHOD

$F_{TMAX} = S/TMAX$, where TMAX is maximum tape reader time

If the given feedrate, F, exceeds F_{TMAX}, F is set equal to F_{TMAX}.

DIAGNOSTICS

None

REQUIREMENTS

Called by subroutine RAPLIM

RESTRICTIONS

No testing is done if F is 0 or DMBITS.

TESTM2 (TESTM2) (GEMULT)

PURPOSE

To test any restrictions placed upon the combined mode of the two heads, and to determine whether the restrictions have been met

INPUT

CALL TESTM2(IJK)

where:

IJK is a flag set by subroutine TESTM2 to indicate whether the restrictions have been met (0 = Yes, 1 = NO)

OUTPUT

Flags used: ISAFLG, PRIMHD, and IJK

METHOD

This subroutine determines if either a spindle speed restriction or SFM restriction has been requested. If not, the subroutine returns after setting IJK = 0. If either restriction is asked for, the appropriate test is made, and IJK is set to either 0 or 1 depending upon the success or failure of the test. The spindle speed test determines the present spindle speed of both heads (DAB(3) and DAB(4)). The parameters PLUSTL and AMINTL have been set previously, and indicate the maximum spindle speed deviation allowed between the two heads. If maximum deviation is exceeded, the combined mode is not allowed. A test for SFM is likewise made.

DIAGNOSTICS

Comment: RPM ON SECONDARY HEAD NOT WITHIN DELTA RPM

Comment: SFM ON SECONDARY HEAD NOT WITHIN DELTA RPM

Comment: RESTRICTIONS FOR MERGING DATA ON HEADS
NOT MET

TEST2 (cont'd)

REQUIREMENTS

Called by subroutines GMCIRL and GMLINE

Calls subroutine COMENT

RESTRICTIONS

The APT System COMMON is used by the subroutine.

THREAD (THRDGL)
(GELATH)

PURPOSE

To set the thread-on flag and the thread-type flag

INPUT

Call THREAD

The input arrays ICLDAT and CLDATA are used.

OUTPUT

The flags are set in accordance with the THREAD statement: THRDON, THFLAG, SFMRPM, SFMFLG, ITHTYD

DIAGNOSTICS

None

REQUIREMENTS

Calls no subroutines

Called by subroutine AUXLRY

RESTRICTIONS

Uses a Data Statement

Computer Dependent

Uses multiple entry THREDO

THREDM (THREDM)
(GEMILL)

PURPOSE

To process a threading block for milling machines.

INPUT

CALL THREDM

Flags used: THRDON, THFLAG, SFMRPM, RETURN and SFMFLG

OUTPUT

DBFSEG(2), DBFSEG(5), DBFSEG(10) and the proper M codes

DIAGNOSTICS

Comment: DABVAL(11) IS GREATER THAN OR EQUAL
TO SFMAXI

REQUIREMENTS

Called by subroutine THREDM

Calls subroutines MACSRT, STOREM, and COMENT

RESTRICTIONS

This subroutine uses multiple entry features.

THREDO (THDOGL)
(GELATH)

PURPOSE

To calculate the lead components and feedrate, and select the G code for threading

INPUT

CALL THREDO

The following parameters and arrays in COMMON are used:

THLEAD, ITHTYP, RAPFLG, ENCODE, THRDON, DPRESM, DPREVM

SFMACI, OPTAB(133,165), SPNSPD, FRMAC, REGFOR(8), THRATE

OUTPUT

Lead components are output in direction cosine registers. Departures, F code, and G code are entered in BUFSEG.

CODE, MCHCON, DBFSEG(2,3,4,8,9,11), DABVAL(11)

METHOD

Lead components are in proportion to departures. If desired, lead and departure limits can be set up in subroutine MACSRT. G code selection is determined by:

(a) departure length

(b) lead length

(c) pitch type: decreasing, constant, or increasing

(d) 100S control or not

DIAGNOSTICS

Comment: MAXIPM EXCEEDED.

THREDO (cont'd)

REQUIREMENTS

Called by subroutine GOLINE

Calls subroutines MACSRT (with MCHCON = 12), COMENT

RESTRICTIONS

Uses a DATA Statement, Multiple Entry, Computer Dependent. Multiple entry is to THREAD.

TIMES (TIMES) (GEOUT)

PURPOSE

To print the total cut and dwell times

INPUT

CALL TIMES

COMMON parameters used: ITEMP, TIMCUT, TIMDWL and NOCHAR

OUTPUT

The total cut time and dwell time is written in minutes at the bottom of the last page of printed output. The tape footage is also printed out.

METHOD

$$\text{Cut Time} = \sum_{i=1}^{n} \left[\frac{L}{F}\right]_i$$

where:

L = the path length

F = the feedrate in IPM

n = the number of cut vectors

DIAGNOSTICS

None

REQUIREMENTS

Called by subroutines ABSOPR and GEPRO3

Calls no subroutines

RESTRICTION

APT System COMMON is used.

TITLE1 (TITLE1)
(GEOUT1)

PURPOSE

To printout a heading title and the register titles and page number for a new page for GEOUT1

INPUT

CALL TITLE1

The following parameters and arrays in COMMON are used:

CURMAC, TAG, IPGCTR, OPTAB, BCDREG

OUTPUT

The generated title is written on IOUTAP.

METHOD

Uses FORTRAN FORMAT

If option 143=0, a print column headed CLREC is included.

DIAGNOSTICS

None

REQUIREMENTS

Calls no subroutines

Called by subroutine GEPRO1

RESTRICTIONS

None

TITLE2 (TITLE2)
(GEOUT)

PURPOSE

To set up and write the page heading print line for GEOUT2

INPUT

CALL TITLE2

The following parameters and arrays in COMMON are used:

CURMAC, TAG, IPGCTR, DPRTNO, BCDREG

OUTPUT

Flags used: IOUTAP, CURMAC, TAG,
IPGCTR, DPRTNO, BCDING

The title is printed at the top of each new page.

DIAGNOSTICS

None.

REQUIREMENTS

Calls subroutine GEPRN2

Called by subroutines GEPRO2

RESTRICTIONS

Uses APT System COMMON

TITLE3 (TITLE3)
(GEOUT3)

PURPOSE

To print and process the generated title for GEOUT3

INPUT

CALL TITLE3

COMMON parameters used: CURMAC, TAG, IPGCTR, BCDREG, BCDIMG

OUTPUT

The desired title and headings are printed at the top of each page.

DIAGNOSTICS

None

REQUIREMENTS

Called by subroutine GEPRN1

Calls subrouting ABSOPR

RESTRICTIONS

APT System COMMON is used.

TOOLGM (TOOLGM)
(GEMILL)

PURPOSE

To output a T code block for a milling machine

INPUT

CALL TOOLGM

OUTPUT

A T code block for a milling machine is setup and output.

METHOD

This routine is called from subroutine TOOLNO to process the T code function for a milling machine. If a T code is available on the machine, a dwell block TABLEM(83) is output. Upon first entry FORK is set and the first tool length is saved in PREVTL. On subsequent entries the change in tool lengths is stored in TEMP(1), DPREVP(3) and DPREVM(3) are set to the current value and the block is output. Then an information block is output with the change in the tool length stored in DBFSEG(3), CODE = -9, and subroutine OUTPUT is called.

DIAGNOSTICS

None

REQUIREMENTS

Called by subroutine TOOLNO

Calls subroutine DWELL, STORM, OUTPUT

RESTRICTIONS

OPTAB(1) must equal O.

OPTAB(132) = 1

TOOLGP (TOOLGP)
(GEPOS)

PURPOSE

To output a T code block for a positioning machine

INPUT

CALL TOOLGP

OUTPUT

A T code block for a positioning machine is output.

METHOD

This routine is called from TOOLNO to process the T code function for a positioning machine. If a T code is available on the machine and the merge of the T code is not required (OPTAB(1) = 1), set up a TABLEG(1) block, CODE = -1 and call subroutine OUTPUT. Upon first entry FORK is set and the first tool length is saved in PREVTL. On subsequent entries OPTAB(86) is tested to determine when and how to make the tool length compensation. If OPTAB(86) = 1, make the compensation now by storing the delta tool length in TEMP(1) and adding the TEMP(1) value to the Z trans vector TRANSL(3). Save the current tool length in PREVTL before returning.

DIAGNOSTICS

None

REQUIREMENTS

Called by subroutine TOOLNO

Calls subroutine OUTPUT

RESTRICTIONS

OPTAB(1) must equal 1 or 2

OPTAB(132) must equal 2

TOOLL (TOOLL)
(GELATH)

PURPOSE

To output the T block for a lathe

INPUT

CALL TOOLL

The following parameters and arrays in COMMON are used:

REGFOR, FORK, DPREVP, DPREVM, PREVTL, TOOLEN

OUTPUT

Flags set: FORK, DPREVP, DPREVM, CODE, PREVTL

The command block for the turret is output with the T code stored at BUFSEG(13).

METHOD

Z is corrected when there is a tool length variation.

DIAGNOSTICS

None

REQUIREMENTS

Calls subroutines DWELL, STOREM, OUTPUT

Called by subroutine TOOLNO

RESTRICTIONS

Uses a Data Statement

TOOLNO (TOLMGM) (GEBASE)

PURPOSE

To process the APT statement TOOLNO, and output a T code or other M codes accordingly

INPUT

CALL TOOLNO

The input arrays ICLDAT and CLDATA are used.

OUTPUT

The determined T code is stored in DBFSEG(13). The parameters TOOL, TOOLEN, POSMAG, FFRSTL, and PREVTL may also be set.

METHOD

Initialize the magaine direction FRKDIR to zero. Store the tool number from CLDATA(4) into TOOL. If the tool change M code is available (TABLEM(7)), set it up for output by calling STOREM(7). If initial entry, save the present tool length in the previous tool length parameter PREVTL. In all cases save the current tool length in TOOLEN. Store the tool number in DBFSEG(13) for later output. If the tool length is zero, output Comment 19. If a special function is required call the subroutine MACSRT. Continue the scan of the statement.

If CCLW is programmed, call subroutine STOREM(84) to output the M code, or for CLW, call STOREM(83) to output the M code and set the direction flag FRKDIR = 1 to indicate that the direction was specified and the proper M code has been output. If OFSETL is programmed, combine the tool number TOOL and the offset number in CLDATA(N + 1) into the parameter TOOL.

After the statement is completely scanned, determine if the direction flag FRKDIR is equal to 1. If not equal to 1, direction of rotation has not been set up. Test to see if the M codes are available for direction of rotation (OPTAB(88) = negative value). If M codes are available, store the present tool number TOOL into the magazine position POSMAG. Call subroutine MINMOV to determine the shortest direction of rotation. Test the direction parameter TURDIR and set up the proper M code (TABLEM(83)) for CLW, or TABLEM(84) for CCLW.

TOOLNO (cont'd)

If the machine is a lathe (OPTAB(132) = 0), call subroutine TOOLL for final processing. If the machine is a position machine (OPTAB(1) greater than zero), call subroutine TOOLGP. If neither, the tool function is assumed to be a mill, and the subroutine TOOLGM is called. In all cases, set the initial entry flag FORK equal to 1 and save the tool number in FFRSTL and return.

REQUIREMENTS

Called by subroutine AUXLRY

Calls subroutines TOOLGM, TOOLGP, TOOLL, STOREM, MINMOV

RESTRICTIONS

None

TRANS (TRANGB) (GEBASE)

PURPOSE

To store the values of x, y, z, as given by the APT source statement TRANS, into the translation vector that is used to translate the cutter data

INPUT

CALL TRANS

The input arrays ICLDAT and CLDATA are used.

OUTPUT

The translation vector TRANSL contains the given values of x, y, z.

DIAGNOSTICS

None

REQUIREMENTS

Calls no subroutines

Called by subroutine AUXLRY

RESTRICTIONS

The tool length (TOOLEN) may be figured into z depending on the value of option 86.

TRUNC (TRNCGB)
(GEBASE)

PURPOSE

To round and truncate the present machine point

INPUT

CALL TRUNC

DPRESM(I) for I = 1 to 6 is the parameter in which the present machine point values reside.

OUTPUT

The values are modified and remain in DPRESM(I).

DIAGNOSTICS

None

REQUIREMENTS

Calls subroutines SRAREC (for linear departures), and SROREC (for rotary departures)

Called by subroutines GEOM5, LINRTY, and SFMO

RESTRICTIONS

None

TSTFCM (TSTFCM)
(GETERP)

PURPOSE

To output smaller linear segments by using smaller dimension G codes if the original segment length caused the feed command to exceed the feed command maximum.

INPUT

CALL TSTFCM

OUTPUT

A series of segments are output.

METHOD

Initially set the FORK equal to zero. If the path is a rapid, the feedrate is set negative; therefore return since rapid paths are not to be segmented. Save the CODE and the feedrate in CODE and FED. Call subroutine CONVRT to store the departures DBFSEG(3-5) into A by setting BIT values equal to zero. Call subroutine COMPFC to compute the feedrate command FCOM. Test this value against the feedrate command maximum and return if acceptable. Test the current value of GDIMUL to determine the present G code. Branch to set up values for GDIMUL and departure maximum DMIN for the next smaller G code. Segment the given path into DIV segments using the test G code. Call subroutine COMPFC to determine the feedrate command using the test G code. If this resulting FCOM is less than FCOMAX, proceed to segment the path on this basis. If the feed command is still greater than the FCOMAX, recompute the feedrate consistant with the FCOMAX, and discontinue the segmenting sequence.

DIAGNOTSICS

None

TSTFCM (cont'd)

REQUIREMENTS

Called by subroutine SELG

Calls subroutines CONVRT, COMPFC, SRAREC, and FEDLIM

RESTRICTIONS

None

TSTFLG (TFLGGB)
(GEBASE)

PURPOSE

To test flags which may have been set to produce special testing on motion records

INPUT

CALL TSTFLG

OUTPUT

Flags used: STOPON, RAPLOW, RAPFLG, RAPFED, FLRPON. FRAPID, SAFLAG, FLSFON, THMODE, THRDON and THFLAG.

As a result of testing the above, flags may be modified, or one or more of the subroutines listed below may be called.

DIAGNOSTICS

None

REQUIREMENTS

Called by subroutine MOTION

Calls subroutines RAPIDO, RAPIDX, RESTAT, SAFETO, SAFETY

RESTRICTIONS

None

TSTLIM (TLIMGB)
(GEMON)

PURPOSE

To test the slide limits for non-multiaxis machines

INPUT

CALL TSTLIM

The following COMMON variables are used:

ISHVEC, OPTAB, DPRESM, REFATL, REFBTL, SEFATL, and SEFBTL

OUTPUT

If the slide limits are exceeded on an axis, a comment to that effect is printed.

METHOD

If OPTAB(110) is not zero, slide limit testing is done. The points from the CLTAPE with their signs set for the GECENT output are tested against OPTABS (121-126). The turret offsets are added to the points before slide limit testing is done for lathes.

DIAGNOSTICS

If the slide limits are exceeded on an axis, this comment is printed:

AXIS SLIDE LIMIT HAS BEEN EXCEEDED *** WARNING

REQUIREMENTS

Called by subroutines GEOM and POSMOV

Calls subroutines COMMENT and MACSRT

RESTRICTIONS

Subroutine TSTLIM uses APT COMMON and contains a DATA statement.

TSTSAF (TSTSAF)
(GELATH)

PURPOSE

To test the SFM generated spindle speed and resultant feedrate to ensure that they are within the allowable SFM limits

INPUT

CALL TSTSAF(SFMLIM, ISENSE)

where:

SFMLIM	= 0	limit not reached
	= 1	limit reached
ISENSE	= 0	decreasing radius
	= 1	increasing radius

COMMON parameters used: SPNSPD, SPNMIN, SFMAXR, SFMAXI, FRMAX, FEDIPM, and FEDIPR

OUTPUT

Flags used: ISENSE and FRMOD

FEDIPM and SPNSPD are within the allowable range limits.

DIAGNOSTICS

Comment: MAXIPM WAS EXCEEDED

REQUIREMENTS

Called by subroutine SFMO

Calls subroutines COMENT and TSTEXT

RESTRICTIONS

None

TURRET (TURTGL) (GELATH)

PURPOSE

To process the APT statement TURRET

INPUT

CALL TURRET

The input arrays ICLDAT and CLDATA are used.

OUTPUT

Flags used: RESETF, NOWNXT, FLGROT, FLGTLP, TURDIR, IHEAD.

A turret corrective move (CODE = -3) is output if the NOW modifier is given in the TURRET statement.

DIAGNOSTICS

ERROR 41.0: Turret corrective move cannot be segmented.

Comment: TURRET CORRECTIVE MOVE EXCEEDED MAXIMUM DEPARTURE AND HAS BEEN SEGMENTED.

REQUIREMENTS

Called by subroutines AUXLRY.

Calls subroutine STOREM, TURSAD, COMTAT, DWELL, MINMOV, SRAREC, OUTPUT, ERDMP1, COMENT

RESTRICTIONS

None

TYPE00 (TY00GL)
(SPNTYP)

PURPOSE

To convert the spindle speed to command form for a Type 0 spindle

INPUT

CALL TYPE00

OUTPUT

The spindle command is stored in the parameter SPNCOM.

METHOD

The constant K is set to 10. If the number of rows in a range (number of speeds) exceeds 10, the constant K is set to 100.

Calculate the spindle command SPNCOM by taking the range row ISPDRO, calculated in subroutine DSRROW, plus (specified range*K) minus 1, plus the incremental adder OPTAB(47).

DIAGNOSTICS

None

REQUIREMENTS

Called by subroutine SPTYPE

Calls no subroutines

RESTRICTIONS

OPTAB(19) = 0

TYPE01 (TY01GL)
(SPNTYP)

PURPOSE

To convert the spindle speed to command form for a Type 1 spindle

INPUT

CALL TYPE01

OUTPUT

The spindle command is stored in the parameter SPNCOM.

METHOD

Store the spindle speed command into SPNCOM. Call subroutine EIACOM to convert the speed to the command form. The command value is returned in SPNCOM.

DIAGNOSTICS

None

REQUIREMENTS

Called by subroutine SPTYPE

Calls subroutine EIACOM

RESTRICTIONS

OPTAB(19) = 1

TYPE02 (TY02GB)
(SPNTYP)

PURPOSE

To convert the spindle speed to command form for a Type 2 spindle

INPUT

CALL TYPE02

OUTPUT

The spindle command is stored in the parameter SPNCOM.

METHOD

Initialize present range parameter IRNG and shift fork parameter FRKSHF to 1 and 0, respectively. If special shifting is required OPTAB(137) = +IREDUC where IREDUC becomes the shift row. If the requested speed row parameter ISPDRO is less than or equal to IREDUC or range parameter ISRNGE equal to 1, determine speed directly. If not, lower the speed to the shift row speed in the present range parameter IRNG upon initial entry. On subsequent entries if the range remains the same, IRNG = ISRNGE or if ISPDRO is less than or equal to IREDUC, shift directly. In all other cases a special shifting sequence is required such that a range shift is never made from a speed row higher than IREDUC, unless the speed is first lowered to the IREDUC speed and the new range speed is lower than the IREDUC speed in the new range.

DIAGNOSTICS

None

REQUIREMENTS

Called by subroutine SPTYPE

Calls subroutines GESCOM, DWELL

RESTRICTIONS

OPTAB(19) = 2

TYPE03 (TY03GB) (SPNTYP)

PURPOSE

To convert the spindle speed to the command form for a Type 3 spindle.

INPUT

CALL TYPE03

The input arrays ICLDAT and CLDATA are used.

OUTPUT

The spindle command is stored in the parameter SPNCOM.

METHOD

Initialize the old speed OLDSPD, old direction OLDIR, the range direction flag RANDIR to zero. Test to see if range and direction M codes are available (TABLEM(4,5,71)≠ DMBITS), then set flag RANDIR = 1. If the spindle direction has not been specified (SPNDIR = 0), set to the optional direction given by option 34, and output a warning comment to this effect. If a stop is required for a range change (OPTAB(89) = 0), output an M code (TABLEM(6)), and a dwell and output a comment to this effect. If a range change or direction change has taken place, determine the proper M code and output. Call subroutine DSRROW to find the row number and calculate the spindle command SPNCOM. Save the values for the M code and spindle command in the STATE vector for possible reinstatement. If the STOPON flag is set to 1, store the negative values for later output by subroutine RESTAT. If the machine has multiple feedrate ranges (OPTAB(18) = 0), output a dwell block. Save the spindle speed in OLDSPD. If RANDIR equals 0, return. If not, set up to output TABLEM(4-5) for turning on the spindle in the specified direction.

DIAGNOSTICS

Comment: NO DIRECTION GIVEN IN FIRST SPINDLE STATEMENT

Comment: WARNING-SPINDLE DIRECTION HAS CHANGED

TYPE03 (cont'd)

REQUIREMENTS

Called by subroutine SPTYPE

Calls subroutines COMENT, STOREM, DWELL, DSRROW

RESTRICTIONS

OPTAB(19) = 3

TYPE04 (TY04GL)
(SPNTYP)

PURPOSE

To convert the spindle speed to command form for a Type 4 spindle

INPUT

CALL TYPE04

OUTPUT

The spindle command is stored in the parameter SPNCOM.

METHOD

Upon entry, initialize the old speed OLDSPD and old direction OLDDIR to zero. Store the spindle speed in SPNCOM and call subroutine EIACOM to convert to the 3 digit magic code. If no direction was specified initially, select the direction from OPTAB(34) and output comment 11. If the direction has changed and a dwell is required for direction reversal (OPTAB(89) = 0, output a spindle stop M code TABLEM(6) and a dwell and output comment 12. Save the direction and range. Test to determine if special down shifting is required. If the range was not specified (CURNG = 0), assume range 1.

Determine the speed to shift to SHFSPD. If the old speed OLDSPD is greater than the SHFSPD, call subroutine EIACOM to lower the spindle speed to the SHFSPD and output a dwell, if required. Determine the location of the range change M code and call subroutine STOREM to set it up. Call subroutine EIACOM with the programmed spindle speed SPNSPD to convert it to the command form SPNCOM. Save the M code in STATE(3) and the spindle command in STATE(2) for possible reinstatement. If in the stop mode (STOPON = 1), store the minus value in the STATE vector. If the machine has multiple feedrate ranges (OPTAB(18) > 1), set up a dwell block by calling subroutine DWELL (54). Save the current spindle speed in OLDSPD and turn on the spindle (FLONSP = 1) after calling subroutine OUTPUT to store the block.

TYPE04 (cont'd)

DIAGNOSTICS

Comment: NO DIRECTION GIVEN IN FIRST SPINDLE STATEMENT

Comment: WARNING-SPINDLE DIRECTION HAS CHANGED.

REQUIREMENTS

Called by subroutine SPTYPE

Calls subroutines EIACOM, DWELL, OUTPUT

RESTRICTIONS

OPTAB(19) = 4

TYPE05 (TY05GM)
(SPNTYP)

PURPOSE

To convert the spindle speed to command form for a Type 5 spindle

INPUT

CALL TYPE05

OUTPUT

The spindle command is stored in the parameter SPNCOM.

METHOD

The requested spindle speed SPNSPD is stored in SPNCOM. Subroutine EIACOM is called to convert the speed to the command value is returned in SPNCOM.

DIAGNOSTICS

None

REQUIREMENTS

Called by subroutine SPTYPE

Calls subroutine EIACOM

RESTRICTION

OPTAB(19) = 5

TYPE08 (TY08GB)
(SPNTYP)

PURPOSE

To convert the spindle speed to command form for a Type 8 spindle

INPUT

CALL TYPE08

OUTPUT

The spindle command is stored in the parameter SPNCOM.

METHOD

The spindle command SPNCOM is calculated by adding the range row ISPDRO to the incremental adder OPTAB(47) minus 1.

DIAGNOSTICS

None

REQUIREMENTS

Called by subroutine SPTYPE

Calls no subroutines

RESTRICTIONS

OPTAB(19) = 8

TYPE09 (TY09GB)
(SPNTYP)

PURPOSE

To convert the spindle speed to command form for a Type 9 spindle

INPUT

CALL TYPE09

OUTPUT

The spindle command is stored in the parameter SPNCOM.

METHOD

The range of the spindle is determined by the tool number TOOL. If TOOL is less than 1 or greater than 12, return. If not, determine the range:

range 1 = tools 3,6,9,12

range 2 = tools 2,5,8,11

range 3 = tools 1,4,7,10

Calculate the spindle command SPNCOM by taking the spindle speed range row ISPDRO minus the number of speeds in the range NRORNG,* (range ISRNGE - 1) plus the incremental adder OPTAB(47) minus 1.

DIAGNOSTICS

None

REQUIREMENTS

Called by subroutine SPTYPE

Calls no subroutines

RESTRICTIONS

OPTAB(19) = 9

TYPE10 (TY10GL)
(SPNTYP)

PURPOSE

To convert the spindle speed to the command form for a Type 10 spindle

INPUT

CALL SPNTYP

The input arrays ICLDAT and CLDATA are used.

OUTPUT

The spindle command is stored in the parameter SPNCOM.

METHOD

Initialize the old speed OLDSPD and the old direction OLDDIR to zero. If the spindle direction has not been specified (SPNDIR=0), set up the assumed direction from option 34 and issue a comment to this effect. Now test to see if the direction has changed (SPNDIR = OLDDIR); if changed, test OPTAB(89) to output a dwell and spindle stop (if required), and output a comment to this effect. Save the spindle direction and range in parameters ISPDIR and ICUR. Based upon the settings of ISRNGE, ICUR, OLDDIR, SPNDIR, and FLONSP, branch to the proper sequence. Test OPTAB(137) for a negative value which indicates a special shift requirement. If the range is not defined, force it into range 1. Determine the maximum shift speed SHFSPD. If the old speed OLDSPD is less than SHFSPD, no special shifting is required. If a special shift is required, set up parameters, and call subroutines EIACOM and DWELL to output this SHFSPD command value in the present range. Based upon the range and the direction requested, determine the M code and output it by calling subroutine STOREM. Now call subroutine EIACOM to output the command for the speed requested. Save the M code and the spindles command in the STATE vector for late reinstatement. If presently stopped (STOPON = 1), store the negative values. If the machine has multiple feedrate ranges (OPTAB(18) = 0), output a dwell block. Save the spindle speed in OLDSPD and set the flag for a spindle off condition (FLONSP = 0), and return.

TYPE 10 (cont'd)

DIAGNOSTICS

Comment: NO DIRECTION GIVEN IN FIRST SPINDLE STATEMENT

Comment: WARNING---SPINDLE DIRECTION HAS CHANGED.

REQUIREMENTS

Called by subroutine SPTYPE

Calls subroutines COMENT, DWELL, STOREM, EIACOM

RESTRICTIONS

OPTAB(19) = 10

TYPE11 (TY11GB) (SPNTYP)

PURPOSE

To convert the spindle speed to command form for a Type 11 spindle

INPUT

CALL TYPE11

OUTPUT

The spindle command is stored in the parameter SPNCOM.

METHOD

This spindle type is a single range, double table lookup form. The spindle speed table SRTAB. Calculate ITEMP by adding the number of speeds in a range NRORNG to the row containing the requested speed ISPDRO. Set SPNCOM equal to SRTAB(ITEMP).

DIAGNOSTICS

None

REQUIREMENTS

Called by subroutine SPTYPE

Calls no subroutines

RESTRICTIONS

OPTAB(19) = 11

TYPE12 (TY12GM)
(SPNTYP)

PURPOSE

To convert the spindle speed to command form for a Type 12 spindle.

INPUT

CALL TYPE12

OUTPUT

The spindle command is stored in the parameter SPNCOM.

METHOD

If the spindle speed SPNSPD is greater than 199, the SPNSPD is modulus 10. Initially, the previous spindle speed PRVSPN is set to SPNSPD. This SPNSPD is output as the spindle command SPNCOM. On Subsequent entries a dwell block is output if the change of speed crosses the 199 limit. If no dwell block is output, RETURN is set to 0, but if a dwell block is output, RETURN is set to 1.

DIAGNOSTICS

None

REQUIREMENTS

Called by subroutine SPTYPE

Calls subroutine DWELL

RESTRICTIONS

OPTAB(19) = 12

TYPE13 (TY13GM)
(SPNTYP)

PURPOSE

To convert the spindle speed to the command form for a Type 13 spindle

INPUT

CALL TYPE13

The input arrays ICLDAT and CLDATA and OPTAB are used.

OUTPUT

The spindle command is stored in the parameter SPNCOM.

METHOD

Initialize the old speed OLDSPD and the old direction OLDDIR to zero. Calculate the various index values:

M = Index stored in SRTAB(180) where the number of speeds in range 1, head 1 is stored in SRTAB.

N = Number of speeds in range 1, head 1, SRTAB(M)

IK = Index where the multiplication FACTOR is stored in SRTAB.

FACTOR = The multiplication factor for range and head, SRTAB(IK)

M (later) = Index where the first speed in range 1, head 1 is located in SRTAB.

N(later) = Index where the last speed in range 1, head 1 is located in SRTAB.

TYPE 13 (cont'd)

METHOD (cont'd)

Compare the requested spindle speed SPNSPD with those available as a product of SRTAB(I)*FACTOR. If not exactly available, test option(90) to determine if the higher, lower, or closest value is to be used. Set SPNSPD to that value and call subroutine EIACOM to calculate the command value SPNCOM. If the spindle direction has not been specified; i.e., SPNDIR = O, set SPNDIR to the direction given by option(34), and output a warning command to this effect. If the direction has been specified, test to see if the direction has been changed. If a stop is required for a range change (OPTAB(89) = O), output an M code (TABLEM(6)) and a dwell, and output a warning comment to this effect. If a range change or direction change has taken place, determine the proper M code and output. Save the values for the M code and spindle command in the STATE vector for possible reinstatement. If the STOPON flag is set to 1, save the negative values for later output by subroutine RESTATE. If the machine has multiple feedrate ranges (OPTAB(18) ≠ O), output a dwell block. Save the spindle speed in OLDSPD and return.

DIAGNOSTICS

Comment: NO DIRECTION GIVEN IN FIRST SPINDLE STATEMENT

Comment: WARNING-SPINDLE DIRECTION HAS CHANGED

REQUIREMENTS

Called by subroutine SPTYPE

Calls subroutines EIACOM, COMENT, STOREM, DWELL, OUTPUT

RESTRICTIONS

OPTAB(19) = 13

TYPE14 (TY14GV)
(SPNTYP)

PURPOSE

To convert the spindle speed to command form for a Type 14 spindle

INPUT

CALL TYPE14

The input arrays ICLDAT and CLDATA are used.

OUTPUT

The spindle command is stored in the parameter SPNCOM

METHOD

Initialize the previous range JRNG, previous mode IMODE, and the pseudo range IRNG to zero. Test to see if the range has changed, and if so, determine if that range falls in the same mode. Ranges 1 - 10 are mode 1, and ranges 11-20 are mode 2. If the mode has changed, output a TABLEM(72 or 74) for mode 1 or 2, respectively. Determine the pseudo range IRNG, save the current actual range JRNG, and call subroutine GESCOM to determine the spindle command.

DIAGNOSTICS

None

REQUIREMENTS

Called by subroutine SPTYPE

Calls subroutines STOREM, GESCOM

RESTRICTIONS

OPTAB(19) = 14

TYPE15 (TYPE15)
(SPNTYP)

PURPOSE

To convert the spindle speed to the command form for a Type 15 spindle

INPUT

CALL TYPE15

The input arrays ICLDAT and CLDATA are used.

OUTPUT

The spindle command is stored in the parameter SPNCOM.

METHOD

Save the spindle direction in ISPDIR. Determine the M code based upon the range ISRNGE and ISPDIR. If the previous M code SAVEM is equal to the current M code VALUEM, set VALUEM to bits to suppress the redundant M code. Based upon the range ISRNGE, determine the RATIO stored in SRTAB. Calculate the spindle command, truncate and test to see if the command is exactly available. If not, select the higher, lower or closest value based upon the OPTAB(90) setting. Determine the actual command spncdm, and recalculate the actual spindle speed based upon the command value.

DIAGNOSTICS

None

REQUIREMENTS

Called by subroutine SPTYPE

Calls subroutine STOREM.

RESTRICTIONS

OPTAB(19) = 15

TYPE16 (TYPE16)
(SPNTYP)

PURPOSE

To convert the spindle to the command form for a type 16 spindle.

INPUT

CALL TYPE16

The following COMMON parameters and arrays are used:

DATACL, DMBITS, ISPDRO, ISRNGE, NRORNG, OPTAB, SPNDIR, SPNSPD, SRTAB, TABLEM, and VALUEM.

OUTPUT

The spindle command is stored in the parameter SPNCOM.

METHOD

This spindle type is identical to type 3 except that the speed codes are not unit increasing.

DIAGNOSTICS

Comment: NO DIRECTION GIVEN IN FIRST SPINDLE STATEMENT

Comment: WARNING-SPINDLE DIRECTION HAS CHANGED.

REQUIREMENTS

Called by subroutine SPTYPE

Calls subroutines COMENT, STOREM, DWELL, DSRROW

RESTRICTIONS

OPTAB(19) = 16.0

TYPE17(TYPE17)
(SPNTYP)

PURPOSE

To convert the spindles speed to commend form for a type 17 spindle.

INPUT

CALL TYPE17

The COMMON arrays ICLDAT, DATACL, OPTAB, and SRTAB are used.

OUTPUT

The spindle command is stored in the parameter SPNCOM.

METHOD

The spindle speed command, SPNCOM, is computed from the Formula:

$$SPNCOM = \frac{S_D - S_{MIN}}{S_{MAX} - S_{MIN}} \times 1000$$

DIAGNOSTICS

None

REQUIREMENTS

Called by subroutine SPTYPE.

RESTRICTIONS

OPTAB(19) = 17.0

TYPE18 (TYPE18)
SPNTYP

PURPOSE

To convert the spindle speed to command from for a Type 18 spindle.

INPUT

CALL TYPE18

The COMMON arrays ISPORQ, NRORNG, OPTAB, and SRTAB are used.

OUTPUT

The spindle command is stored in the parameter SPNCOM.

METHOD

The postprocessor stores the number of speeds per range as given in option 8, starting with SRTAB(1) and changing by the percentage of change, SRTAB(2), and continuing for the number of ranges as given in option 7. If option 137 is set, special shifting will be done as for the Type 2 spindle. The spindle command is computed by subroutine GESCOM.

DIAGNOSTICS

None

REQUIREMENTS

Called by subroutine SPTYPE

Calls subroutines DWELL and GESCOM

RESTRICTIONS

OPTAB(19) = 18.0

TYPE19(TYPE19)
SPNTYP

PURPOSE

To convert the spindle speed to the command form for a Type 19 spindle.

INPUT

CALL TYPE19

The following common parameters and arrays are used: ISRNGE, ISPDRO, OPTAB, SRTAB, TABLEG and TABLEM.

OUTPUT

The spindle command is stored in the parameter SPNCOM.

METHOD

Each time the spindle changes ranges, the postprocessor issues three command blocks. The first block is for a stop M code, the second block is a postprocessor comment which states:

A SPINDLE SPEED CHANGE OCCURS AT THIS POINT

and the third block is a non-motion block which carries the new spindle speed.

DIAGNOSTICS

Comment: A SPINDLE SPEED CHANGE OCCURS AT THIS POINT

REQUIREMENTS

Called by SPTYPE

Calls STOREM and OUTPUT

RESTRICTIONS

OPTAB(19) = 19.0

UNLOAD (UNLDGP)
(GEPOS)

PURPOSE

To output an unload M or G code

INPUT

CALL UNLOAD

OUTPUT

The proper M or G Code is output in a block by itself.

METHOD

Test OPTAB(133) to check for a special function and call subroutine MACSRT. If TLNOFF is not equal to FFRSTL, output Comment(40). If TABLEG(26) equals DMBITS, assume a TABLEM(33) is used to unload the tool. Set up a TABLEM(33), cycle off (DBFSEG(2) = TABLEG(1) 0 block and call subroutine OUTPUT. Set the tool off (TLNOFF) and tool length on tool 2 (TLEN2) equal to 0, and return. If a G code is used set up a TABLEG(26), T code = (DBFSEG(13) = TOLLOD) block, code = -4 (separate G code block) and call subroutine OUTPUT.

DIAGNOSTICS

Comment: ATTEMPTING TO UNLOAD TOOL IN WRONG MAGAZINE POCKET.

REQUIREMENTS

Called by subroutine AUXLRY

Calls subroutines MACSRT, STOREM, OUTPUT

RESTRICTIONS

OPTAB(1) must equal 1 or 2

TABLEM(33) or TABLEG(26) ≠ DMBITS

WEFREW (WEFREW)
(GEMON)

PURPOSE

To rewind and write an end-of-file on a file

INPUT

CALL WEFREW(TAPE,IND,IE)

where; TAPE = file number.

IND = error flag.

IE = 0 to rewind and open for reading

IE = 1 to rewind and open for writing

OUTPUT

The file is rewound and opened for reading or writing.

METHOD

This subroutine call the APT subroutines which writes end-of-files, rewinds files, and open files. The calling sequences and subroutine names will vary with the computer.

DIAGNOSTICS

The following unrecoverable errors are printed:

ERROR 7001: Error occurred when opening the file for reading.

ERROR 7002: Error occurred when opening the file for writing.

WEFREW (cont'd)

REQUIREMENTS

Called by subroutines ABSOPR, GMSTOR, REDTAP, GEBASE.

Calls the APT subroutines which write end-of-files, rewind files, and open files

RESTRICTIONS

APT COMMON is used. The subroutines called are computer dependent.

WELD (WELDGP)
(GEPOS)

PURPOSE

To output the miscellaneous function codes for the designated welding operations

INPUT

CALL WELD

The input arrays ICLDAT and CLDATA are used.

OUTPUT

The proper M code is made output in a command block.

METHOD

If weld is turned off (ICLDAT(4) = 72), turn off the weld flag (IGEFLG = 1), and return. If weld is turned on (ICLDAT(4) = 71), turn on the weld flag (IGEFLG = 0) and return. If TILT is specified (ICLDAT(4) = 24), either TABLEM(83) or (84) is output. If SHIFT is specified (ICLDAT(4) = 249), either TABLEM(88-92), (94) is output. If SCHEDL is specified (ICLDAT(4) = 250), either TABLEM(85), (86) is output. If FIXTUR is specified (ICLDAT(4) = 300), either TABLEM(66) (67) is output. If the code is outside this range of values, comment 22 is output.

DIAGNOSTICS

Comment: REQUESTED MISCELLANEOUS FUNCTION CODE IS NOT ON THIS MACHINE.

REQUIREMENTS

Called by subroutine AUXLRY

Calls no subroutines

RESTRICTIONS

OPTAB(1) must be positive

XOFSET (XOFSET)
(GEPOS)

PURPOSE

To output an X offset number in DBFSEG (17) in a cycle off block (G80).

INPUT

CALL XOFSET

OUTPUT

The X offset number is stored in DBFSEG(17).

METHOD

Set up a cycle off block (DBFSEG(2) = TABLEG(1)) with the specified number in DBFSEG(17). Output the block as a Code = 1 (Stop block) by calling subroutine OUTPUT.

DIAGNOSTICS

None

REQUIREMENTS

Called by subroutine AUXLRY

Calls subroutine OUTPUT

RESTRICTIONS

OPTAB(1) must equal 1 or 2.

The use of this routine usually requires an H register for the offset value.

7.0 APPENDIX

7.1 EIA "MAGIC 3" CONVERSION METHOD

Some NC machine controls require the feedrate or spindle speed commands to be in EIA a three digit code. The method of converting a number to this three digit code is given below.*

The feed or speed is expressed as a three(3) digit number. The second and third digits of this coded number are the feed or speed rounded to two digit accuracy. The first digit of the coded number is a decimal multiplier and has a value three (3) greater than the number of digits to the left of the decimal point of the feed or speed. Where there are no digits of the feed or speed to the left of the decimal point, then the number of zeros immediately to the right of the decimal point is subtracted from three (3) to provide the value of the first digit.
Example:

Feed or Speed	Coding
1728	717
150	615
15.2	515
7.82	478
.153	315
.0126	213
.00875	188
.000462	046

(Note: The second digit can never be zero unless all digits are zero.)

* Extracted in part from Electronic Industries Association Standards RS-274B Interchangeable Perforated Tape Variable Block Format for Contouring and Contouring/Positioning Numerically Controlled Machines, May 1967.

7.2 ERROR ACCUMULATION ANALYSIS

The "true" value of a point on the cutter path, i.e., the computer value, is normally an eight digit number; these are the numbers used with the vectors DPREVP and DPRESP. Departures are computed from the truncated numbers, i.e., the "true" value truncated to the number of significant digits given by the minimum step size of the NC machine; these truncated points are the numbers used with the vectors DPREVM and DPRESM. The truncated value has its last digit rounded up if the digit adjacent to the last digit is greater than 4.

Example; "True" value = 5.1235678; NC machine minimum step size = 0.0001. Therefore, the truncated value is 5.1236.

Let x be the x component value of the "true" machine point, and x' the corresponding truncated value. Then, at point 1, $x_1 = x'_1 + e_1$ where e_1 is the amount lost in truncation.

Similarly,

$$x_2 = x'_2 + e_2,$$

$$x_3 = x'_3 + e_3,$$

$$x_n = x'_n + e_n.$$

Let Δ represent the computed departures; then

$$\begin{array}{lllllll} \Delta_1 = x_1 - x_2 & & & & \Delta'_1 = x'_1 - x'_2 \\ \Delta_2 = x_2 - x_3 & & & & \Delta'_2 = x'_2 - x'_3 \\ \vdots & & & & \vdots \\ \Delta_{n-1} = x_{n-1} - x_n & & & & \Delta'_{n-1} = x'_{n-1} - x'_n \end{array}$$

$$S = \sum_{k=1}^{n} \Delta_k = x_1 - x_n \qquad S' = \sum_{k=1}^{n} \Delta'_k = x'_1 - x'_n$$

7.2 ERROR ACCUMLATION ANALYSIS (cont'd)

S and S´ are each the resultant total cutter path. The difference between the two paths gives the total error of that axis, viz.,

$$
\begin{aligned}
E = S - S' &= (x_1 - x_n) - (x'_1 - x'_n) \\
&= (x_1 - x'_1) - (x_n - x'_n) \\
&= e_1 - e_n
\end{aligned}
$$

From the above relation, it can be seen that if the cutter returns to the path beginning point, that the total accumulated error is zero since e_1 and e_n will then have the same value.

The maximum accumulated error is when $e = -e$, or $E = 2e$. But E is always less than or equal to the minimum step size of the NC machine.

The error e per step is

$$e = x - x' = x - \left[\frac{x}{STEP} + 0.5 \right] * STEP = \frac{STEP}{2} .$$

Therefore, $E = 2e_1 = STEP$, at maximum for each axis. Hence, the maximum possible vector error is $\sqrt{3} * STEP = 0.00017$ inches when $STEP = 0.0001$ inches.

7.3 DEFINITIONS AND ABBREVIATIONS

Definitions

CLTAPE The computer tape that the postprocessor reads in order to obtain the part program processed data produced by the APT system.

command The programmed coded symbols fed to the NC machine control which initiates the NC machine action indicated by the command.

modal Retention of mode. When a function is modal, the mode established is retained by the postprocessor until the mode is changed or stopped.

one-shot no retention of mode. Once the function is used, its influence ends.

speed code This is not a speed command; a speed code is code assigned one-to-one with the speeds of a spindle speed table. The speed code may or may not become part of the speed command.

Abbreviations

A/D	acceleration-deceleration sequence
AIA	Aircraft Industry Association
BCD	binary coded decimal
DMBITS	Coded number = -40404040.0
CL	cutter location; CL data refers to the generated points produced by APT
CCLW	counterclockwise
CLW	clockwise
EIA	Electronic Industry Association
EOB	end-of-block
EOF	end-of-file
IPM	inches per minute

7.3 DEFINITIONS AND ABBREVIATIONS (cont'd)

IPR	inches per revolution
RPM	revolutions per minute
SFM	surface feet per minute

7.4 DETERMINATION OF OPTIMUM LENGTH FOR A RAPID TRAVERSE MOVE

For machines that require gear shifting to obtain rapid traverse, it may be more advantageous to remain in the feedrate range and travel at the highest feedrate in that range than to incur the delay inherent with a gear change. To evaluate the situation,the postprocessor computes from the following derivation.

To traverse a path S, the time required is the time required to shift gears plus the time to traverse the distance to traverse rate.

The dwell time to shift into and out of rapid traverse is given by options 81 and 82 as,

$$T = \frac{O_{81} + O_{82}}{60} \quad \text{minutes.}$$

The time required to complete the path S at traverse rate is given by

$$T = \frac{S}{O_m}$$

where O_m is the minimum value of the rapid traverse rates stored at option 42, 43, and 44.

Therefore, the minimum time to traverse a path S is given by

$$T_1 = \left[\frac{O_{81} + O_{82}}{60}\right] + \frac{S}{O_m} \quad \text{minutes.}$$

To cover the same distance without shifting into rapid traverse will require

$$T_2 = \frac{S}{O_{39}} \quad \text{minutes,}$$

where O_{39} is the maximum feedrate which can be obtained and is taken from option 39.

7.4 DETERMINATION OF OPTIMUM LENGTH FOR A RAPID TRAVERSE MOVE (cont'd).

By equating T_1 and T_2 and solving for S, we can find that minimum distance for which a shorter distance would take longer at rapid traverse than at maximum feedrate. Performing the manipulation,we find

$$S = S_1 = \left(\frac{O_{81} + O_{82}}{60} \right) \left(\frac{O_m + O_{39}}{O_m - O_{39}} \right)$$

The above result is not affected by tape reader speed, but this must be considered also. There are two other conditions which may be more limiting. If the active block execution time is shorter than the tape reader time for the control to load the next block into buffer storage, the tool will dwell in the work. By equating the time it takes to read a maximum length block of tape (as given by option 13) with the time to traverse the path S, and the time to move S distance at maximum feedrate, we can determine two other potentially limiting path lengths.

At maximum feedrate, the minimum distance for which the tape reader would not be limiting is

$$S_2 = O_{39}O_{13}.$$

At traverse rate, the minimum distance for which the tape reader would not be limiting is

$$S_3 = \left[O_{13} - \left(\frac{O_{81} + O_{82}}{60} \right) \right] O_m$$

Therefore, the minimum optimum length for a rapid traverse move (which is stored at option 37) is the largest value of S (S_1, S_2, or S_3). Since the values of S_1, S_2, and S_3 will almost always be different, and since each is a minimum value for the case tested, the largest value of the three must be selected as the minimum value.

7.5 GECENT III POSTPROCESSOR SUPPLEMENTARY CONDITIONS OF SALES

The following terms and conditions supplement and amend the Standard Conditions of Sale of the Industrial Sales Division printed on the back of Quotation Form 13004 and published in Handbook Section 98; sale of GECENT Postprocessor and related programming services is expressly conditioned on acceptance of these supplementary terms and conditions.

1. GECENT Postprocessor System

The General Electric Company, hereinafter referred to as the Company, will supply the GECENT III Postprocessor System, which contains the following elements:

a. Source program, hereinafter called the program, consisting of a library of modular subroutines implementing the functions and features of an unlimited number of numerically controlled machines. It is supplied recorded on magnetic tape.

b. Documentation in the form of both part and computer programmer's manuals.

c. Machine subroutine, a computer subroutine which establishes the characteristics and selects from the postprocessor library those modules which are required to implement the tape controlled functions and features of the machine and control system as delineated in the terms of the contract.

d. Supplementary notes and instructions, including subroutine listing, covering the particular machine referenced in the contract.

Note: Items a. and b. are normally supplied only once to a user installation, since they can be used with many types to Mark Century equipped machines, and once installed, need not be a contract item on subsequent contracts.

Items c. and d. are supplied for each new contract and become the property of the purchaser.

The GECENT III Postprocessor Program is coded almost entirely in FORTRAN IV. To improve computer operating efficiency, certain output features of the program are writtem in machine language.

1. GECENT POSTPROCESSOR SYSTEM (cont'd)

The postprocessor program is designed to operate with the current version of APT III as released and maintained by the IIT Research Institute and in effect at date of contract unless specifically stated otherwise.

Standard APT postprocessor vocabulary words will be used in all cases. Whenever new functions and/or methods require the creation of new words, such new words will follow the accepted rules for format and syntax of the APT system. These additions will be submitted to the APT Vocabulary Review Committee for approval. Should further changes be required resulting from Committee action, these changes will be incorporated into the postprocessor.

Purchaser may reserve the right to approve the choice of new words, but only if such right is specifically stated in the contract.

GECENT postprocessor programs can be supplied for most large scale computers on which the APT computer system has been implemented. The exact computer configuration must be specified in the contract.

Users of IBM 7090/94 computers will use the GECENT III postprocessor program through a FORTRAN II/IV interface provided for the purpose.

2. Correction of Defects

Every reasonable effort is made to provide an error free program. Should programming errors in the postprocessor be detected, the purchaser will forward to the postprocessor author a complete machine memory dump, subroutine memory map, a listing of the related machine subroutine, a card copy of the part program, and any other supporting information as may be pertinent to the solution of the problem. Corrections to the program will be made only at the author's base location. Inquiries and supporting documents should be directed to GECENT Postprocessor System, Mail Drop H-8, Building 305, General Electric Company, Cincinnati, Ohio , 45215, unless advised specifically to do otherwise.

3. Postprocessor Acceptance Test

The purchaser shall forward to the postprocessor author part programs which are to be used as the basis for accepting the postprocessor. These will be used in checking out the postprocessor prior to delivery to the purchaser. These part programs must be validated programs satisfactorily processed through Section III of APT and be free of part programming errors. They may contain up to a maximum of 500 CL data points. The conversion of the part programmer's instructions (the part program) by the postprocessor into a tape containing machine language instructions in correct form and format shall constitute acceptance of the postprocessor by the purchaser. Failure of the purchaser to furnish such programs in time for postprocessor checkout to be performed and still meet the scheduled delivery of the program shall void this requirement of acceptance of the postprocessor.

4. Implementation of Postprocessor

The General Electric Company, in providing the GECENT III Postprocessor System, assumes that the user has implemented and is operating the current version of APT III as referenced in paragraph 3 above. The Company, in making available the standard GECENT program, cannot be held responsible for its satisfactory operation if it is modified by the user or implemented on other than the current version of APT, nor on special modifications of that program made by the user to suit his own particular needs and over which the Company has no control.

The user has full responsibility for assimilating the postprocessor into his computer system, including any modifications which may be required.

5. Warranty

Subject to the provisions of paragraph 6 below, General Electric Company warrants to the purchaser that the GECENT III Postprocessor System to be delivered hereunder will be of the kind and quality designated or described in the proposal. The foregoing warranty is exclusive of all other warranties, whether written, oral , or implied, including any warrany of fitness for purpose. If upon the basis of evidence submitted to the Company within 12 months from date of delivery by the Company, or from completion of the machine installation with which the program is to be used, whichever occurs later, it is shown that the material, data, analyses, programs or services

5. WARRANTY (cont'd)

delivered or rendered hereunder do not meet this warranty and the purchaser so notifies the Company, the Company shall thereupon correct any such defect or error in the postprocessor or, at its option, make available substitute material, data or programs. The foregoing shall constitute the sole remedy of the purchaser and the sole liability of the Company

If services rendered by the Company are based on a program, data, or other material supplied by the purchaser, the Company's liability hereunder shall in no case exceed 30 minutes of rerun time. In addition, it shall be the responsibility of the purchaser to include all necessary self-checking procedures in the program, data, or other material supplied to the Company.

During the above stated 12 month warranty period, the Company reserves the right to retain copies of any information, data, or reports which the purchaser has furnished under this agreement. Upon the expiration of said 12 month period, all such material, data, and reports will be disposed of as the purchaser may direct.

6. Limitation of Liability

Neither the General Electric Company nor any person acting on its behalf (a) makes any warranty or representation, express or implied, with respect to the accuracy, completeness, or usefulness of the information contained in the GECENT III Postprocessor System, or that the use of any information disclosed in the said system may not infringe privately owned rights; or (b) assumes any liabilities with respect to the use of, or for damages resulting from the use of any information disclosed in the GECENT III Postprocessor System. The Company's liability on any claim of any kind, whether in warranty, contract, negligence or otherwise, arising out of or connected with this contract, shall not exceed the price allocated to the material, data, programs or sources giving rise to the claim. In no event shall the Company be liable for special or consequential damages.

7. Computer Techniques and Skills

The Company shall not be restrained in its use of the techniques and skills of computer operation and programming acquired in the performance of any services rendered under this agreement.

8. Disclosure of Information

This order is accepted and delivery of the information and material sold hereunder is made with the express understanding and stipulation that no information transmitted by either party in connection with performance hereunder is to be regarded as secret or confidential, except as may be otherwise provided by agreement in writing signed by the authorized representatives of the parties. This provision is not intended to affect the rights of the parties under copyrights or patents now or hereafter issued under the laws of the United States or foreign countries.

9. Material and Data Furnished by Purchaser

Purchaser agrees that any punched cards, magnetic tapes, or other forms of input furnished by it to the Company in connection with the performance of this contract will be in good and usable condition. In addition, purchaser shall be responsible for the accuracy, correctness, and completeness of any material or data or programs furnished to the Company for use in the development of a computer program or in the processing of data for the purchaser.

The Company shall not be responsible for the loss of or damage to any material and data furnished to it by the purchaser. In the event such occurs, the Company's obligations under this contract shall cease unless within 10 days after notice of such loss or damage, the purchaser furnishes the Company with duplicate sets of all material and data. Such material and data will remain the property of the purchaser and, except as otherwise provided herein, will be returned to the purchaser upon completion of the services to be performed hereunder.

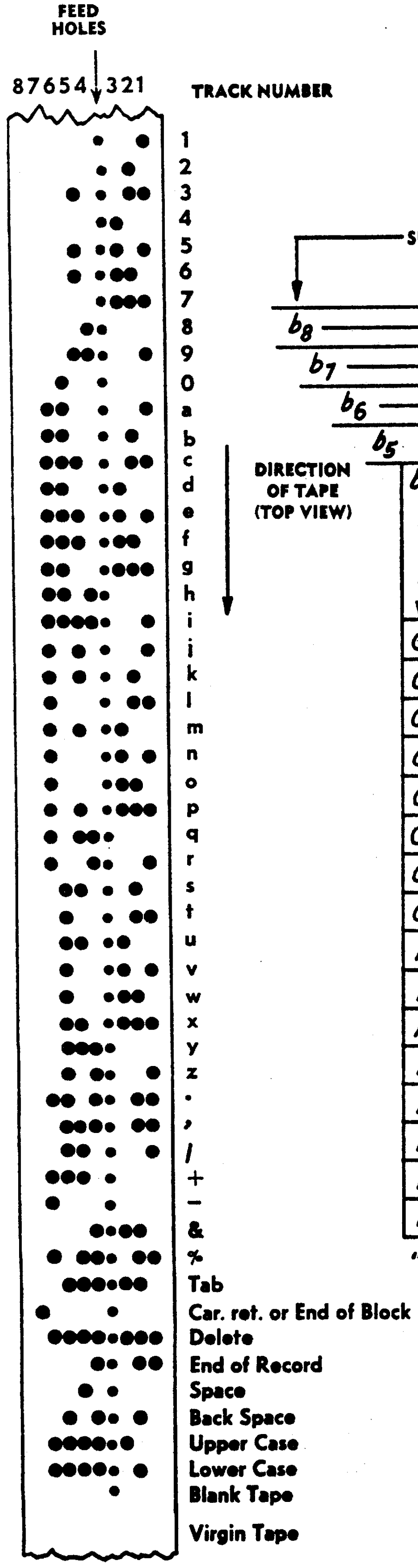
10.Title

Title to and right to possession of, without legal process, the material, data, or programs delivered hereunder shall remain with the Company until all payments hereunder, including deferred payments, whether evidenced by notes or otherwise, shall have been made in cash, and the purchaser agrees to do all acts necessary to perfect and maintain such right and title in the Company. It is the intention of the parties that the material , data or programs delivered hereunder shall remain personal property until all payments have been made in full.

10. TITLE (cont'd)]

When payment has been made in full, title to the machine subroutine protion of the postprocessor shall pass to the purchaser. Title to the GECENT III Postprocessor System, separate from the machine subroutine, remains with the Company and the Company grants to the user a nonexclusive, nontransferable license to make copies of the furnished GECENT III Postprocessor System for his own use, and not for distribution to others, in the numerical control of machines, provided such copies retain the General Electric copyright notice.

7.6 Electronic Industries Association Punched Paper Tape Code per RS-244A dated Jan 1967.



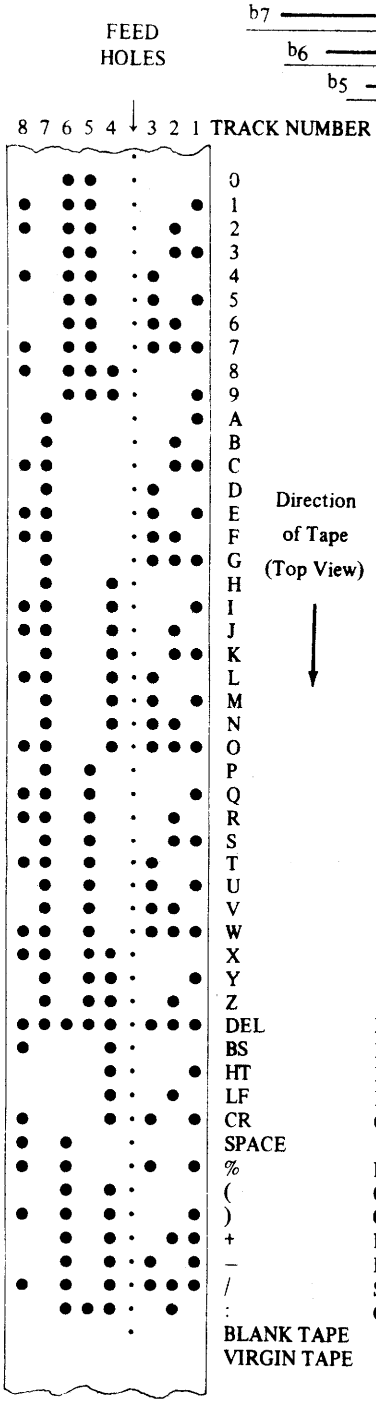
SUBSCRIPTS REFER TO TRACK NUMBERS

b_8 →				0	0	0	0	1	1	1	1
b_7 →				0	0	1	1	0	0	1	1
b_6 →				0	1	0	1	0	1	0	1
b_5 Parity Bit				SEE	# 2.	3. 1.					
b_4	b_3	b_2	b_1								
0	0	0	0	SPACE	0	—	+	CR EOB			
0	0	0	1	1	/	j	a				
0	0	1	0	2	s	k	b				
0	0	1	1	3	t	l	c				
0	1	0	0	4	u	m	d				
0	1	0	1	5	v	n	e				
0	1	1	0	6	w	o	f				
0	1	1	1	7	x	p	g				
1	0	0	0	8	y	q	h				
1	0	0	1	9	z	r	i				
1	0	1	0		BS		LC				
1	0	1	1	EOR	,	%	•				
1	1	0	0				UC				
1	1	0	1								
1	1	1	0	&	Tab						
1	1	1	1				Del				

"1" SIGNIFIES A HOLE, "0" SIGNIFIES NO HOLE

Code for Numerically
Controlled Machines
Perforated Tape

7.7 ASCII Punched Paper Tape Code per Electronic Industries Association RS-358.



Subscripts Refer To Track Numbers.

b_8					Parity Bit, See Paragraph 2.2.1							
b_7					0	0	0	0	1	1	1	1
b_6					0	0	1	1	0	0	1	1
b_5					0	1	0	1	0	1	0	1
b_4	b_3	b_2	b_1	Column → Row ↓	0	1	2	3	4	5	6	7
0	0	0	0	0	Blank Tape		Space	0		P		
0	0	0	1	1				1	A	Q		
0	0	1	0	2				2	B	R		
0	0	1	1	3				3	C	S		
0	1	0	0	4				4	D	T		
0	1	0	1	5			%	5	E	U		
0	1	1	0	6				6	F	V		
0	1	1	1	7				7	G	W		
1	0	0	0	8	BS		(	8	H	X		
1	0	0	1	9	HT		)	9	I	Y		
1	0	1	0	10	LF			:	J	Z		
1	0	1	1	11			+		K			
1	1	0	0	12					L			
1	1	0	1	13	CR		–		M			
1	1	1	0	14					N			
1	1	1	1	15			/		O			DEL

"1" Signifies a Hole, ' O" Signifies No Hole.

DEL	Delete	Ignored by Control
BS	Back Space	Ignored by Control
HT	Horizontal Tab	
LF	Line Feed	Used for End of Block
CR	Carriage Return	Ignored by Control
SPACE		Ignored by Control
%	Percent	Used for Rewind Stop
(	Opening Parenthesis	Used for Control Out
)	Closing Parenthesis	Used for Control In
+	Plus	
–	Minus	
/	Slash	Used for Block Delete (Optional)
:	Colon	Used for Reference Rewind Stop
BLANK TAPE		
VIRGIN TAPE		

SUBSET OF USA STANDARD CODE FOR INFORMATION INTERCHANGE FOR NUMERICAL MACHINE CONTROL PERFORATED TAPE

7.8 ARC TANGENT DEFINITIONS

The arc tangent function is used to evaluate the inverse trigonometric terms.
Limiting conditions are defined as follows:

$$\theta = \tan^{-1} \frac{y}{x}$$

When $x=0$,

$$\theta = \begin{array}{l} \pi/2 \text{ if } y \text{ is positive} \\ \pi/2 \text{ if } y \text{ is negative;} \end{array}$$

when both x and $y = 0$, $\theta = 0$.

7.9 GECENT III COMMON PARAMETER CROSS REFERENCE

Common Name GE635, CDC6600 Univac1108	GE635, Univac1108 CDC6600 Parameters	Common Name IBM/360	IBM/360 Parameters
GECOUT	ABSVAL(20)	D2	DABVAL
GECBAS	ADEP	D2	DEPA
GECBAS	ANGLIN	S2	ANGLIN
GECOM	AXMULT	S1	AXMULT(34)
GECOUT	BCDIMG(20)	S2	BCDIMG
GECOUT	BCDREG(20)	S2	BCDREG(20)
GECBAS	BDEP	D2	DEPB
GECOM	BITS	D1	DMBITS
GECOM	BLANKS	D1	DBLNKS
GECBAS	BUFFER(6,20)	D2	DBUFER(6,6)
GECBAS	BUFSEG(20)	D2	DBFSEG(34)
GECBAS	CIRDAT(3)	D2	DATCIR(3)
GECBAS	CIRDIR	S2	CIRDIR
GECBAS	CIRFLG	S2	CIRFLG
GECBAS	CIRPT1(3)	D2	DCRPT1(3)
GECBAS	CIRPT2(3)	D2	DCRPT2(3)
GECBAS	CIRRAD	S2	CIRRAD
GECBAS	CIRSEQ	S2	CIRSEQ
GECBAS	CLERP	S2	CLERP
GECBAS	CODE	S2	CODE
GECBAS	CRCODE	S2	CRCODE
GECOUT	CTRLIN	S2	CTRLIN
GECBAS	CURCYG	S2	CURCYG
GECOM	CURMAC	S1	CURMAC
GECBAS	CURMOD	S2	CURMOD
GECBAS	CURNG	S2	CURNGE
GECBAS	CURNTZ	S2	CURNTZ
GECOM	CUST(5)	S1	CUST(5)
GECOUT	CUTIME	S2	TIMCUT
GECBAS	CUTRAD	S2	CUTRAD
GECBAS	CUTTER	S2	CUTTER
GECBAS	CYCFLG	S2	CYCFLG
GECBAS	DECPLA	S2	XYZDEC
GECOM	DEPLIM(4)	S1	RNGDEP(4)
GECBAS	DEPMAX	S2	BIGDEP
GECOM	DEPN	S1	FACDEP
GECBAS	DIMULT	S2	GDIMUL
GECBAS	DIRFLG	S2	FLGDIR
GECBAS	DIV	S2	SDIV
GECBAS	DMS	S2	BMS
GECBAS	DRAW	S2	FLDRAW
GECBAS	DUMA(10)		
GECOUT	DUMO(10)	D1	D1OPEN(10)
GECOM	DUM(30)	D2	D2OPEN(20)
GECOUT	DWTIME	S2	TIMDWL
GECBAS	ENCODE	S2	ENCODE
GACBAS	ENDFLG	S2	ENDFLG

7.9 GECENT III COMMON PARAMETER CROSS REFERENCE (cont'd)

Common Name GE635, CDC6600 Univac1108	GE635, Univac1108 CDC6600 Parameters	Common Name IBM/360	IBM/360 Parameters
GECOM	EPSLON	S1	EPSLON
GECOM	ERROR	S1	ERROR
GECOM	FCOMAX	S1	FCOMAX
GECBAS	FDHOLD	S2	FDHOLD
GECOM	FDTRAV	S1	FDTRAV
GECBAS	FEDIPM	S2	FEDIPM
GECBAS	FEDRTR	S2	FEDIPR
GECBAS	FFRSTL	S2	PRSMAG
GECBAS	FIRST	S2	FIRST
GECOM	FLM1	S1	FLM1
GECOM	FLZ	S1	FLZ
GECOM	FL1	S1	FL1
GECOM	FL2	S1	FL2
GECOM	FL3	S1	FL3
GECOM	FL4	S1	FL4
GECOM	FL5	S1	FL5
GECOM	FL10	S1	FL10
GECOM	FL100	S1	FL100
GECOM	FL360	S1	FL360
GECBAS	FLONKL	S2	FLONKL
GECBAS	FLONSP	S2	FLONSP
GECBAS	FLRPON	S2	FLRPON
GECBAS	FLSFON		
GECBAS	FORK14		
GECBAS	FORK18		
GECBAS	FORK32		
GECBAS	FORK33		
GECBAS	FORK39		
GECOM	FORMAT(20)	S1	REGFOR(30)
GECBAS	FRAPID	S2	FRAPID
GECBAS	FRK34B		
GECOM	FRMAX	S1	FRMAX
GECOM	FRMIN	S1	FRMIN
GECBAS	FRMOD	S2	FRMOD
GECOM	FRTAB(300)	S1	FRTAB(300)
GECBAS	CRPLOD	S2	CRPLOD
GECBAS	CRPSLC	S2	CRPSLC
GECOM	HEAD3(22,3)		
GECOUT	IBLANK		
GECBAS	ICLMOD	I2	IKLMOD
GECBAS	ICODE	I2	ICODE
GECBAS	ICYTYP	I2	ICYTYP
GECOUT	IDLINE	I2	IDLINE
GECBAS	IDRAW	I2	IDRAW
GECBAS	IDWLFL	I2	IDWLFL
GECBAS	IFDRNG	I2	IFDRNG

7.9 GECENT III COMMON PARAMETER CROSS REFERENCE (cont'd)

Common Name GE635, CDC6600 Univac1108	GE635, Univac1108 CDC6600 Parameters	Common Name IBM/360	IBM/360 Parameters
GECOM	IGEFLG	I1	IGEFLG
GECBAS	IHEAD	I2	IHEAD
GECOM	INDFR	I1	INDFR
GECBAS	INDIC	I2	INDIC
GECBAS	INDPT1	I2	INDPT1
GECBAS	INDPTS	I2	INDPTS
GECOM	INT1	I1	INT1
GECOM	INT2	I1	INT2
GECOM	INT3	I1	INT3
GECOM	INT4	I1	INT4
GECOM	INT5	I1	INT5
GECOM	INT6	I1	INT6
GECOM	INT7	I1	INT7
GECBAS	INTAP(20)	I2	ICLDAT(20)
GECOM	INTZ	I1	INTZ
GECOUT	IPAGE	I2	IPAGE
GECOUT	IPGCTR	I2	IPGCTR
GECBAS	IPITCH	I2	IPITCH
GECBAS	IPLANE	I2	IPLANE
GECBAS	IRETN	I2	IRETN
GECBAS	ISAFMD	I2	ISAFMD
GECBAS	ISFMOD	I2	ISFMOD
GECOUT	ISHUFL	I2	ISHUFL
GECOUT	ISHVEC(6)	I2	ISHVEC(6)
GECOM	ISORT(20)		
GECBAS	ISPDRO	I2	ISPDRO
GECBAS	ISPTYP	I2	ISPTYP
GECBAS	ISRNGE	I2	ISRNGE
GECOM	ITEMP(5)	I1	ITEMP(5)
GECBAS	ITHTYP	I2	ITHTYP
GECOUT	IXSTOR	I2	IXSTOR
GECBAS	KTR	I2	KTR
GECOUT	LSTCOL	I2	LSTCOL
GECBAS	LSTPLN	I2	LSTPLN
GECBAS	MAFORK	I2	MAFORK
GECBAS	MAXES	I2	MAXES
GECBAS	MCHCON	I2	MCHCON
GECBAS	MODPOS	I2	MODPOS
GECOM	MULTHD	I1	MULTHD
GECOM	NAME(6)	S1	TAG(9)
GECBAS	NAXES		
GECBAS	NCOM	I2	NCOM
GECOUT	NFP(26)	I2	NFP(26)
GECOUT	NIPA(20)	I2	NIPA(20)
GECOUT	NIP(26)	I2	NIP(26)
GECOUT	NPCHAR	I2	NOCHAR

7.9 GECENT III COMMON PARAMETER CROSS REFERENCE (cont'd)

Common Name	GE635, Univac1108	Common Name	IBM/360
GE635, CDC6600	CDC6600 Parameters	IBM/360	Parameters
Univac1108			
GECBAS	NOP	I2	NOP
GECBAS	NOPTS	I2	NOPTS
GECBAS	NOSEG	I2	NOSEG
GECOUT	NPR(26)	I2	NPR(26)
GECOUT	NPTA(30)	I2	NPTA(20)
GECBAS	NRNGES	I2	NRNGES
GECBAS	NRORNG	I2	NRORNG
GECOM	NSTEP		
GECBAS	NWPR	I2	NWPR
GECBAS	OFMACH		
GECBAS	ONMACH		
GECOUT	OPRVAL(20)	S2	OPRVAL(20)
GECOM	OPTAB(250)	S1	OPTAB(250)
GECOUT	ORGIN(5)	S2	ORGIN(5)
GECOM	PARTID	S1	PARTID
GECOUT	PART(11)	D2	DPRTNO(6)
GECBAS	PL	D2	DPATH
GECOM	PLBITS	D2	DPBITS
GECBAS	POSMAG	S2	POSMAG
GECBAS	PRESMP(6)	D2	DPRESM(6)
GECBAS	PRESM(6)		
GECBAS	PRESPT(6)	D2	DPRESP(6)
GECBAS	PRESP(6)		
GECOUT	PREVF	S2	PREVF
GECOUT	PREVG	S2	PREVG
GECBAS	PREVMP(6)	D2	DPREVM(6)
GECBAS	PREVM(6)		
GECBAS	PREVPT(6)	D2	DPREVP(6)
GECBAS	PREVP(6)		
GECOUT	PREVS	S2	PREVS
GECBAS	PREVTL	S2	PREVTL
GECOUT	PREVX	S2	PREVX
GECOUT	PREVY	S2	PREVY
GECBAS	PROGK		
GECBAS	PT	S2	PT
GECBAS	RADLIN	S2	RADLIN
GECBAS	RAPFED	S2	RAPFED
GECBAS	RAPFLG	S2	RAPFLG
GECBAS	RAPLOW	S2	RAPLOW
GECBAS	RAPRNG	S2	RAPRNG
GECOUT	ROBLID(75)	S2	RDPART(75)
GECOUT	REELNO		
GECBAS	RESETF	S2	RESETF
GECBAS	RETURN	S2	RETURN
GECOM	RHSTEP	S1	RHSTEP

7.9 GECENT III COMMON PARAMETER CROSS REFERENCE (cont'd)

Common Name GE635, CDC6600 Univac1108	GE635, Univac1108 CDC6600 Parameters	Common Name IBM/360	IBM/360 Parameters
GECBAS	RMS	S2	RMS
GECOM	ROTFMN	S1	ROTFMN
GECOM	ROTFMX	S1	ROTFMX
GECOM	ROTMAX	S1	ROTMAX
GECOM	ROTRAP	S1	ROTRAP
GECBAS	ROTYPE	S2	ROTYPE
GECBAS	RPOINT	S2	RPOINT
GECBAS	SADSFM	S2	SADSFM
GECBAS	SAFLAG	S2	SAFLAG
GECBAS	SAVEN	S2	SAVEN
GECBAS	SEQCTR	S2	SEQCTR
GECOUT	SEQNEW		
GECBAS	SFMAXI	S2	SFMAXI
GECBAS	SFMAXR	S2	SFMAXR
GECBAS	SFMCIR	S2	SFMCIR
GECBAS	SFMDES	S2	SFMDES
GECBAS	SFMFLG	S2	SFMFLG
GECBAS	SFMLIM		
GECBAS	SFMLOK	S2	SFMLOK
GECBAS	SFMRAD	S2	SFMRAD
GECBAS	SFMRPM	S2	SFMRPM
GECBAS	SFMSEN		
GECOUT	SKPCOD	S2	SKPCOD
GECBAS	SKPFLG	S2	SKPFLG
GECBAS	SKPLIN	S2	SKPLIN
GECBAS	SLTOLN	S2	SLTOLN
GECBAS	SPINON	S2	SPINON
GECBAS	SPNCOM	S2	SPNCOM
GECBAS	SPNMAX	S2	SPNMAX
GECBAS	SPINMIN	S2	SPNMIN
GECBAS	SPINSPD	S2	SPNSPD
GECOM	SRTAB(300)	S1	SRTAB(300)
GECBAS	STATE(12)	S2	STATE(12)
GECOM	STEP	S1	STEP
GECBAS	STOPON	S2	STOPON
GACBAS	SYSCON	S2	SYSCON
		S1	S1OPEN(20)
		S2	S2OPEN(30)
GECOM	TABLEG(120)	S1	TABLEG(120)

7.9 GECENT III COMMON PARAMETER CROSS REFERENCE (cont'd)

Common Name GE635, CDC6600 Univac1108	GE635, Univac1108 CDC6600 Parameters	Common Name IBM/360	IBM/360 Parameters
GECOM	TABLEM(200)	S1	TABLEM(200)
GECBAS	TAPSTO(246)	D2	DATACL(246)
GECOM	TEMP (10)	D1	DTEMP (10)
GECBAS	THFLG	S2	THFLAG
GECBAS	THLEAD	S2	THLEAD
GECBAS	THMODE	S2	THMODE
GECBAS	THRATE	S2	THRATE
GECBAS	THRDON	S2	THRDON
GECBAS	TLEAD	S2	TLEAD
GECBAS	TLEN2	S2	TLEN2
GECBAS	TLHEAD	S2	TLHEAD
GECBAS	TLNOFF	S2	TLNOFF
GECOM	TMAX	S1	TMAX
GECBAS	TOLCON	S2	TOLCON
GECBAS	TOLDLN	S2	TOLDIN
GECOM	TOLIN	S1	TOLIN
GECBAS	TOLLOD	S2	TOLLOD
GECOM	TOLOUT	S1	TOLOUT
GECBAS	TOLSLC	S2	TOLSLC
GECBAS	TOOL	S2	TOOL
GECBAS	TOOLDN	S2	TOOLDN
GECBAS	TOOLEN	S2	TOOLEN
GECBAS	TRANSL(3)	D2	DTRANS (3)
GECBAS	TSTLIN	S2	TSTLIN
GECBAS	TUROFF	S2	TUROFF
GECBAS	TURPOS	S2	TURPOS
GECBAS	UPFLAG	S2	UPFLAG
GECBAS	VALUEM	S2	VALUEM
GECOM	WORDDS (20)	S1	REGSTR (30)
GECBAS	XDEP	D2	DEPX
GECBAS	YDEP	D2	DEPY
GECBAS	ZDEP	D2	DEPZ