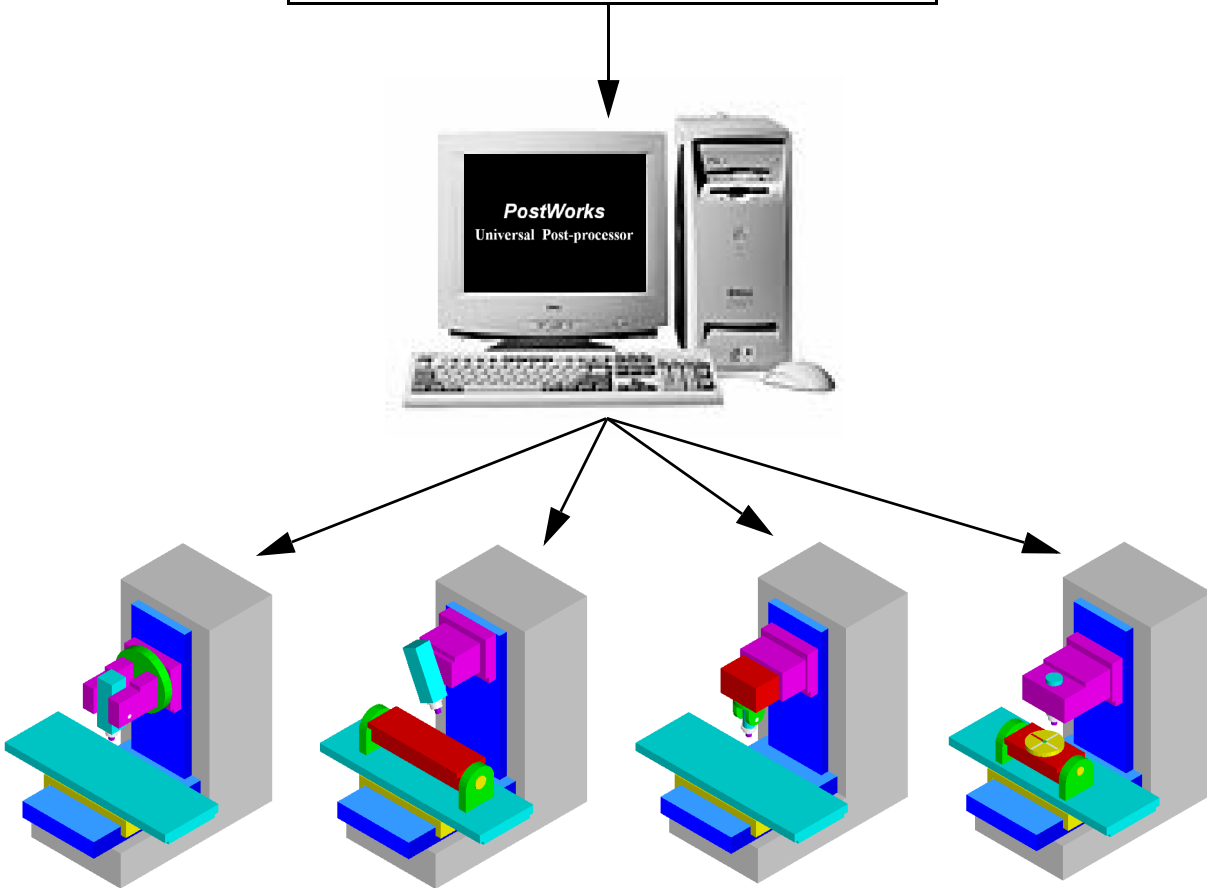


PostWorks V14.2

Universal Post-processor

```
PARTNO PostWorks 5-axis Example
MACHIN/PWORKS, 5,10,15,20,25
DISPLY/ON
MULTAX/ON
LOADTL/ 1.0000, LENGTH, 0.0000
SPINDL/11460.0000
CUTCOM/ADJUST, 54.0000
FEDRAT/ 45.8400
RAEID
GOTO / -0.5868, 1.7442, 13.2013,-0.1763002,-0.0580174, 0.9826252
GOTO / -0.5742, 1.7574, 13.2005,-0.0573679, 0.0690655, 0.9959613
GOTO / -0.5571, 1.7350, 13.2025,-0.0678441, 0.0635927, 0.9956672
GOTO / -0.5496, 1.7145, 13.2039,-0.0775203, 0.0649455, 0.9948732
GOTO / -0.5475, 1.6925, 13.2051,-0.0865240, 0.0650829, 0.9941216
GOTO / -0.5506, 1.6708, 13.2061,-0.0936203, 0.0810051, 0.9923071
GOTO / -0.5585, 1.6502, 13.2068,-0.0946931, 0.0996985, 0.9905016
GOTO / -0.5707, 1.6317, 13.2072,-0.0902180, 0.1190103, 0.9887857
GOTO / -0.5855, 1.6161, 13.2042,-0.0809151, 0.1369890, 0.9872622
GOTO / -0.6021, 1.6032, 13.1973,-0.0673507, 0.1525321, 0.9860009
GOTO / -0.6190, 1.5936, 13.1883,-0.0520649, 0.1632720, 0.9852063
GOTO / -0.6359, 1.5864, 13.1776,-0.0360976, 0.1718789, 0.9844565
GOTO / -0.6742, 1.5870, 13.1768,-0.0446210, 0.1706530, 0.9843203
```



Revision Date: December 10, 2014

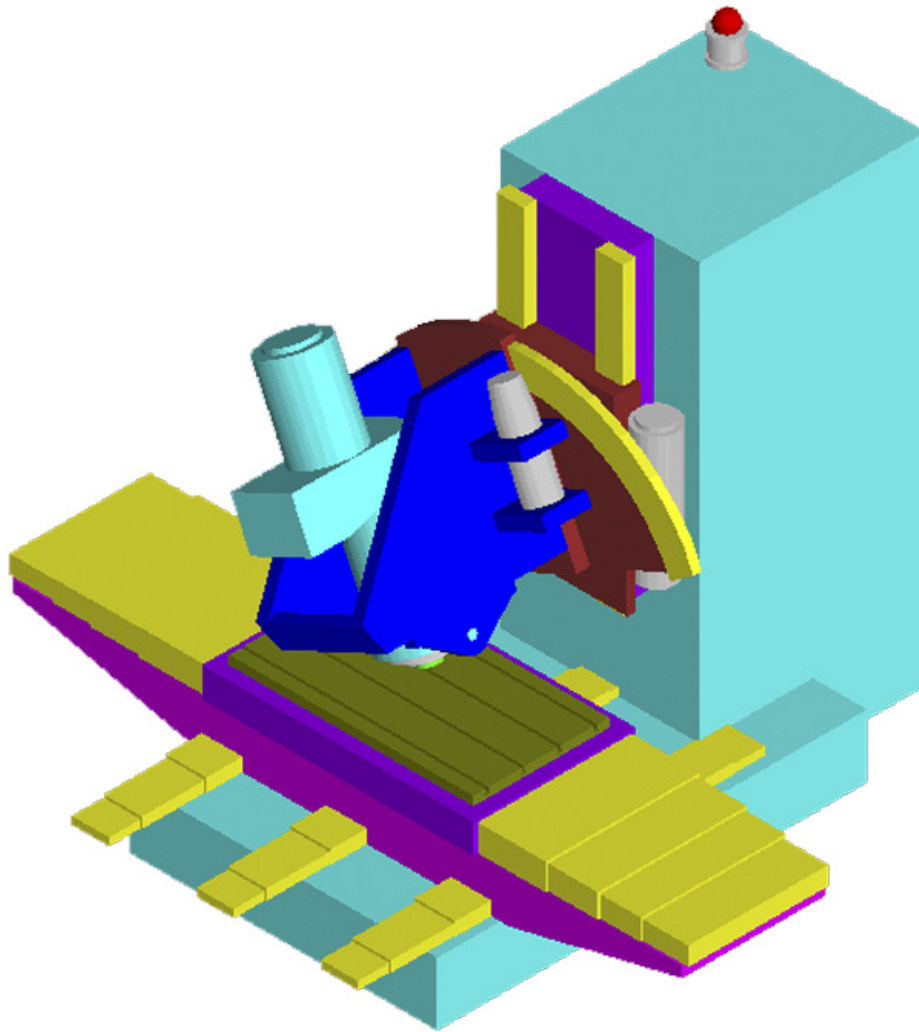
Warning and Disclaimer

Every effort has been made to make this document complete and as accurate as possible.
However, no warranty or fitness is implied.

The information is provided on an "as is" basis. Numerical Control Computer Sciences shall have neither liability nor responsibility to any person or entity with respect to any loss or damages in connection with or rising from the information contained in this document.

Copyright 1983-2014 by Numerical Control Computer Sciences
Irvine, California. Printed in the United States of America.
All rights reserved. The contents of this publication may not
be reproduced in any form or by any means, electronic
or mechanical, including photocopying, recording, or
information storage and retrieval systems, for any
purpose other than the licensee's personal use,
without prior written consent from
Numerical Control Computer Sciences.

PostWorks



Universal Post-processor Users Guide

Warning and Disclaimer

Every effort has been made to make this document complete and as accurate as possible.
However, no warranty or fitness is implied.

The information is provided on an "as is" basis. Numerical Control Computer Sciences shall have neither liability nor responsibility to any person or entity with respect to any loss or damages in connection with or rising from the information contained in this document.

Copyright 1983-2014 by Numerical Control Computer Sciences
Irvine, California. Printed in the United States of America.
All rights reserved. The contents of this publication may not
be reproduced in any form or by any means, electronic
or mechanical, including photocopying, recording, or
information storage and retrieval systems, for any
purpose other than the licensee's personal use,
without prior written consent from
Numerical Control Computer Sciences.

TABLE OF CONTENTS

CHAPTER 1 *PostWorks* Technical Specification.....1-1

1.1	Introduction.....	1-1
1.2	Configuring <i>PostWorks</i> - <i>MPost</i>	1-1
1.2.1	The HELP Facility	1-1
1.2.2	Automatic Documentation	1-1
1.2.3	Machine Configuration.....	1-2
1.2.3.1	MILL-3	1-2
1.2.3.2	MILL-4	1-2
1.2.3.3	MILL_5.....	1-2
1.2.3.4	MILL-7	1-2
1.2.3.5	LATHE-2	1-3
1.2.3.6	LATHE-4	1-3
1.2.3.7	Linear Axes Set-up	1-3
1.2.3.8	Rotary Axes Set-up.....	1-3
1.2.3.9	Home Position & Clearance Plane	1-3
1.2.3.10	Machine Limits.....	1-4
1.2.4	Control Tape Format.....	1-4
1.2.4.1	Letter Address Registers.....	1-4
1.2.4.2	G/M-code Groups	1-4
1.2.4.3	User Defined Blocks.....	1-4
1.2.4.4	Control Tape Block Format	1-5
1.2.4.5	Automatic Tape Breaks	1-5
1.2.5	Motion Block Output	1-5
1.2.5.1	Lathe Motion.....	1-5
1.2.5.2	Rotary Motion.....	1-5
1.2.5.3	Axes Output	1-6
1.2.5.4	Coordinate Translations	1-6
1.2.6	Circular Interpolation	1-6
1.2.6.1	Circular Tolerances.....	1-6
1.2.6.2	Helical Interpolation.....	1-7
1.2.7	Spline Interpolation.....	1-7
1.2.8	Motion Generation	1-7
1.2.8.1	Mutually Exclusive Axes	1-7
1.2.8.2	Rotary Axes Linearization	1-7
1.2.8.3	Polar Interpolation	1-8
1.2.8.4	Slowdown Blocks.....	1-8
1.2.8.5	Transformation Blocks	1-8
1.2.9	Mill Cycles	1-8
1.2.9.1	Cycle Simulations.....	1-9
1.2.9.2	Cycle Interruption.....	1-9
1.2.9.3	Cycle Block Output	1-9

1.2.9.4	Cycle Time Calculations	1-10
1.2.10	Lathe Cycles	1-10
1.2.11	Spindle Speed Control	1-10
1.2.12	Feed Rates	1-10
1.2.13	Rapid Motion	1-11
1.2.14	Tool Change Sequences	1-11
1.2.15	Tool Length & Fixture Offsets	1-11
1.2.16	Cutter Compensation	1-12
1.2.17	Machine Dwells	1-12
1.3	Post-processor Macros - PostMacro	1-12
1.3.1	Macro Features	1-13
1.3.2	Macro Arguments	1-13
1.3.3	Real & Text Variables	1-13
1.3.4	Post Variables	1-13
1.3.5	Operators and Functions	1-14
1.3.6	Post-processor Commands	1-14
1.3.7	Format Descriptors	1-14
1.3.8	Program Control Statements	1-14
1.3.9	Special Macros	1-14
1.3.10	Syntax Checking	1-15
1.3.11	Reading/Writing External Files	1-15
1.3.12	Look Ahead In CLfile	1-15
1.3.13	Standalone Module - PMacro	1-15
1.4	Universal Post-processor - PWorks	1-15
1.4.1	Output Files	1-16
1.4.2	Punch File Header	1-16
1.4.3	Print File Description	1-16
1.4.4	Unique Filenames	1-17
1.4.5	Multiple Clfile Support	1-17

CHAPTER 2 Getting Started2-1

2.1	Installing PostWorks	2-1
2.1.1	Installing PostWorks	2-1
2.1.2	PostWorks Runtime License	2-1
2.2	File Specification	2-2
2.3	Generating a Post-processor	2-2

CHAPTER 3 Using **MPost**3-1

3.1	Invoking MPost	3-1
3.2	MPost Runtime Options	3-1
3.2.1	Page Length - PAGE_LEN	3-2
3.2.2	Defining Input Units - UNITS	3-2
3.3	MPost Basics	3-2

3.4	Register Labels	3-2
3.5	Menus	3-4
3.5.1	The Banner Line	3-4
3.5.2	Making a Menu Selection	3-4
 CHAPTER 4 Using <i>PostMacro</i>		4-1
4.1	<i>PostComp</i> Overview	4-1
4.2	Invoking <i>PostComp</i>	4-1
4.3	<i>PostComp</i> Runtime Options	4-2
4.3.1	Documentation File - DOCUMENT	4-2
4.3.2	Listing File - LISTING	4-2
4.3.3	Object File - OBJECT	4-3
4.3.4	Page Length - PAGE_LEN	4-3
4.3.5	Quiet Process - QUIET	4-4
4.4	<i>PMacro</i> Overview	4-4
4.5	Invoking <i>PMacro</i>	4-4
4.6	<i>PMacro</i> Runtime Options	4-4
4.6.1	Input Clfile Format - CLFILE	4-5
4.6.2	Unique Output Filenames - IDENT	4-6
4.6.3	Listing File - LISTING	4-6
4.6.4	Overriding the MACHIN Card - MACHINE	4-7
4.6.5	Output Clfile - OBJECT	4-7
4.6.6	Page Length - PAGE_LEN	4-7
4.6.7	Running Custom Post-processors - POST	4-8
4.6.8	Quiet Batch Process - QUIET	4-8
 CHAPTER 5 Using <i>PWorks</i>		5-1
5.1	<i>PWorks</i> Overview	5-1
5.2	Invoking <i>PWorks</i>	5-1
5.3	<i>PWorks</i> Runtime Options	5-1
5.3.1	Clfile Transformation - ADJUST	5-2
5.3.2	Input Clfile Format - CLFILE	5-3
5.3.3	Error Messages - APTERR, WARNING, ERROR, FATAL	5-3
5.3.4	Unique Output Filenames - IDENT	5-4
5.3.5	Listing File - LISTING	5-5
5.3.6	Overriding the MACHIN Card - MACHINE	5-5
5.3.7	Overriding Machine Options - OPTION	5-5
5.3.8	Page Length - PAGE_LEN	5-6
5.3.9	Print File - PRINT	5-6
5.3.10	Punch File - PUNCH	5-6
5.3.11	Simulation File - SIMULATE	5-7
5.3.12	Quiet Batch Process - QUIET	5-7
5.3.13	Consecutive Commas - [NO]FILL[:method]	5-8

CHAPTER 6 *PostWorks* Utilities6-1

6.1	Running on Windows	6-1
6.2	Porting Machine Descriptor Files - <i>GenXfer</i>	6-1
6.3	Updating the Vocabulary Files - <i>GenWord</i>	6-2
6.4	Update the Message Files - <i>GenPrompt</i>	6-2
6.5	Updating the Help File - <i>GenHelp</i>	6-3

APPENDIX A *PostWorks* Routine Summary A-1

APPENDIX B Directory Structure and Data Files B-1

B.1	<i>PostWorks</i> Directory Structure	B-1
B.2	Data Files	B-2
B.2.1	User Vocabulary File - postvocab.syn	B-2
B.2.2	Initialization Files - routine.ini	B-3
B.2.3	Machine Descriptor File - PWORKS_#.MDF	B-3
B.2.4	Documentation Files - file.doc	B-4
B.2.5	Macro Definition Files - pmacro_#.mac	B-4
B.2.6	Macro Object File - pmacro_#.obj	B-4
B.2.7	Print Descriptor File - pworks_#.pdf	B-5
B.2.8	Punch Header File - pworks_#.phf	B-5
B.2.9	Machine Adjust File - file.maf	B-5
B.2.10	Clfiles - file.cl	B-6
B.2.11	Listing Files - file.lis	B-6
B.2.12	Print File - file.pr1	B-6
B.2.13	Punch File - file.pu1	B-7
B.2.14	Simulation File - file.sim	B-7
B.3	<i>PostWorks</i> Data Support Files	B-7
B.3.1	The Vocabulary Files - postword.WRD, postvocab.WRD	B-7
B.3.2	The Menu/Prompt File - postmenu.MSG	B-8
B.3.3	The Error Message File - posterror.MSG	B-10
B.3.4	The Automatic Documentation Message File - postdoc.MSG	B-11
B.3.5	The Help File - posthelp.HLP	B-12

APPENDIX C Vocabulary Words C-1

C.1	<i>PostWorks</i> Major Words	C-1
C.2	<i>PostWorks</i> Minor Words	C-3

APPENDIX D Error Messages D-1

D.1	Introduction	D-1
-----	--------------------	-----

D.2	PostComp Error Message Format	D-1
D.3	PostMacro Error Message Format	D-1
D.4	PWorks Error Message Format	D-2
D.5	Error Messages	D-3
APPENDIX E Glossary		E-1
APPENDIX F Machine Configuration Files		F-1
F.1	Introduction	F-1
F.2	Numbering Scheme	F-1
F.3	Demonstration Machine Configurations	F-2
APPENDIX G APT Source File Formats		G-1
G.1	Introduction	G-1
G.2	PTED APT Source File Format	G-1
G.3	CATIA APT Source File Format	G-4
G.4	Unigraphics APT Source File Format	G-7
G.5	MasterCam .NCI File G code format	G-10
G.1	Circular Interpolation Formats	G-14
G.2	Helical Interpolation Formats	G-18
G.3	Special Considerations For APT Text Commands	G-19
INDEX		1

CHAPTER 1 **PostWorks** Technical Specification

1.1 Introduction

PostWorks is both a post-processor generator and a universal post-processor for various types of machine tools, including Mills, Lathes, and Ultrasonic Cutters. It was designed from a users stand point, requiring minimum input from the user and creating output for the machine control that is more exacting to the machine requirements than from a custom post-processor.

There are 3 modules associated with the **PostWorks** post-processor, the post-processor configuration tool (**MPost**), post-processor *Macros* (**PostMacro**) and the post-processor (**PWorks**). These modules work together in such a way as to minimize the effort required to generate and use the post-processor configuration.

1.2 Configuring **PostWorks** - **MPost**

MPost is an interactive menu driven program that is used to define a *Machine Descriptor File (MDF)* for the **PostWorks** post-processor. **MPost** is used solely for developing custom post-processor configurations that will be used by **PWorks** when processing for a specific machine. Each post-processor configuration is stored in a separate *MDF* file, allowing you to generate a multitude of machine specific post-processors each configured for a specific machine.

The **MPost** interface consists of three different types of sections, a menu navigator, question and answer prompts, and forms. You can walk through each section of the **MPost** utility when defining a post-processor from scratch or go directly to a specific section and change only certain parameters when modifying an existing configuration file. In addition, an existing *MDF* file can be loaded as a default configuration and modified to create a new *MDF* file.

1.2.1 The HELP Facility

MPost's on-line help facility allows you to get help at any time during the post-processor generation. Help is available for every *Menu*, *Prompt* and *Form*. A description of the requested help entity is displayed on the screen, including a description of the acceptable input and how your choice affects the output generated by **PostWorks**.

1.2.2 Automatic Documentation

At the user's request **MPost** will create a [document file](#) containing a detailed description of the current *Machine Descriptor File*. This automatically generated description is formatted to look like custom generated post-processor documentation and includes a description of the machine configuration (linear and rotary axes set-up, machine limits, etc.), feed create and spindle tables, descriptions of the supported registers including G-code and M-code definitions, User Defined Block assignments, and a customized command summary.

Like the menu navigator, the automatic documentation file will only contain information that is pertinent to the current *Machine Descriptor File*. For example, if a home position has not been defined, then a home position will not be included in the documentation.

1.2.3 Machine Configuration

PostWorks can be configured for either a Mill or a Lathe and supports up to 6 linear axes (a primary and secondary axis for each of the X, Y and Z axes) and up to 4 rotary axes. The rotary axes can be any combination of rotary tables and/or heads and can rotate around any of the major linear axes.

PostWorks consists of multiple modules, allowing you to obtain only the support required to run your current machines in the beginning and offering an upgrade path as you add different types of machines.

1.2.3.1 MILL-3

The MILL-3 module supports Milling Machines with up to 6 linear axes. You can have a Primary and Secondary axis for each of the X, Y and Z axes. 3 axes may be active at any one time. when the machine supports both a Primary and Secondary axis, one must be locked when the other is active.

1.2.3.2 MILL-4

The MILL_4 module supports all of the features of MILL-3 and additionally supports 1 rotary axis. The rotary axis can be a positioning or contouring axis mounted on the table or on the spindle carrier.

1.2.3.3 MILL_5

The MILL_5 module supports all of the features of MILL-4, but supports up to 2 rotary axes. The rotary axes can be any combination of rotary tables and/or heads and can rotate around any of the major linear axes.

1.2.3.4 MILL-7

The MILL-7 module supports all of the features of MILL_5, but supports up to 4 rotary axes. The rotary can be any combination of rotary tables and/or heads and can rotate around any of the major linear axes. For example, 2 rotary heads and 1 rotary table, 3 rotary heads and 1 rotary table, 2 non-orthogonal rotary heads. If one of the rotary axis does not rotate around any of the X, Y or Z axis, then this will usually constitute 2 or more rotary axes.

MILL-7 also adds support for Ultrasonic Cutter machines. These machines cut various types of material with the use of a pulsating blade. Some of the features resident in **PostWorks** for Ultrasonic Cutters include the ability to guide the cutting edge of the blade along the direction of the cut, special retract logic during direction changes which prohibits the blade from breaking, varying Z-motion to prolong the life of the cutting blade, and vacuum pod control.

1.2.3.5 LATHE-2

The LATHE-2 module supports 2-axis single or twin turret Lathes. Only 1 turret may be active at any one time, merging of the two turrets is not currently supported.

1.2.3.6 LATHE-4

Combined with a MILL module, such as MILL-4, the LATHE-4 module supports 2-axis turning as well as multi-axis milling on the same machine. Milling modes supported include standard tool axis programming, cylindrical interpolation, and polar coordinate interpolation. The spindle axis (C-axis) can also be used to satisfy the Y-axis coordinates when the machine does not have a Y-axis.

1.2.3.7 Linear Axes Set-up

The machine configuration supports up to 2 linear axes for each of the major axes (X, Y, Z). The secondary axis can have 1 or 2 setups. One format for the secondary axis consists of two axes that move in the same direction, but are not connected, for example, a Z-axis quill and W-axis saddle.

The second format consists of a secondary axis that moves with the primary axis, for example a boring attachment loaded in the spindle composing an X-axis and U-axis.

1.2.3.8 Rotary Axes Set-up

Up to 4 rotary axes may be defined for the machine configuration, with simultaneous movement allowed in 2 of the rotary axes. The rotary axes can be any combination of tables and/or heads and rotate around any of the major linear axes.

The user can define the position of the rotary axis on the machine, whether the axis is a contouring or positioning only axis, and if a linear or rotary scale is used.

1.2.3.9 Home Position & Clearance Plane

A user definable home position can be defined for tool changes as well as a clearance plane for retracting the tool. The clearance plane can be an actual plane or can be a distance along the tool, added to the length of the tool, to retract.

The move to home position can be generated by **Postworks** for a *GOHOME* block acceptable to the machine control, such as G28 Z-0\$, can be output.

1.2.3.10 Machine Limits

The minimum and maximum travel limits can be set both within the **MPost** configuration tool and [within the part program](#). You can also define up to 4 interference zones, which are used by **PWorks** to determine when the spindle and machine/fixture collide.

1.2.4 Control Tape Format

The format of the machine control tape is completely user definable, including 92 registers, 11 G-code and M-code groups, and Start-of-tape and End-of-tape sequences.

1.2.4.1 Letter Address Registers

Up to 92 registers (G, X, M, F, etc.) can be defined. The user has full control of the [register format](#), including the floating point format (leading zero suppression, decimal point, etc.), beginning and ending identifying strings (up to 10 characters can be defined for each string), number of digits to the left and right of the decimal point (for both inch and metric output), and when the register should be output.

The versatility of the register description allows for a simple register format (X1.0) to the more complex formats available in today's CNC controllers, such as programmable parameters (F[10]). The [values of the letter address registers can also be preset](#) in **MPost**.

1.2.4.2 G/M-code Groups

There are 11 G-code and 11 M-code groups available for each machine configuration. Each group contains values which will cancel each other at the machine. You can define up to 14 codes in each G-code groups and up to 7-codes in each M-code group.

PostWorks will output multiple non-conflicting G/M-codes in a single control tape block or in multiple blocks depending on the requirements of the machine. G-codes and M-codes in the same block can also disallowed.

1.2.4.3 User Defined Blocks

Up to 20 user definable blocks can be generated. These blocks contain information on the format of certain tape blocks, such as, *TMARK*, automatic cycles, spindle on blocks, etc. These blocks, besides being assigned to a specific sequence, can also be output at any location within the part program at the discretion of the user.

1.2.4.4 Control Tape Block Format

The End-of-program and Control Tape rewind codes are user definable, as well as the End-of-block, operator message, rewind stop, and optional skip characters. The order of the letter address registers within the tape block is defined by the user. You may also separate each register in the tape block with a space, so that it is easier to read at the machine control.

PostWorks is also capable of including a checksum value on each tape block for machine controls that support this feature when downloading the control tape.

1.2.4.5 Automatic Tape Breaks

PostWorks will automatically break the control tape into multiple sections based on tape length, machining time or machining distance. Tape breaks are usually required for machine controls that have a limited amount of memory or when tool wear is excessive.

You have the option of creating a new file and retracting the tool away from the part at each tape break or simply outputting a block that contains the proper coding to load in the next section of the program.

1.2.5 Motion Block Output

The processing of motion blocks is determined by various settings within the machine configuration. The user can define whether the *FROM* statement is output or not, tolerances for determining when to output each axis and the default positioning mode (absolute or incremental).

1.2.5.1 Lathe Motion

When programming for a lathe, the user can choose to program in the X-axis as a radius and have **PostWorks** output it as a diameter value. **PostWorks** will also convert XY programming to ZX machine output at the discretion of the user.

1.2.5.2 Rotary Motion

The rotary axes can always be programmed in absolute, incremental, or the current absolute/incremental mode depending on the requirements of the machine. They can also be represented in either degrees or Inches/MMs. You also have the options of adjusting the input linear coordinates according to the table/head rotations.

PostWorks has the option to automatically unclamp the rotary axes prior to movement and clamp them after movement if required by the machine.

1.2.5.3 Axes Output

All of the programmable axes (X, Y, Z, A, B, etc.) can be set to move a minimum distance before being output. This feature is useful in disabling the output of the rotary axes when minor fluctuations occur in the tool axis or for support of older machines which require that moves be on an even number, for example .0002.

The maximum distance any one axis can move in a single block can also be defined. **PostWorks** will break up a move that causes the delta axis movement to exceed this limit into multiple moves.

1.2.5.4 Coordinate Translations

The linear and rotary axes can be translated or scaled by user defined values. These translations can be applied prior to or after the rotary axes adjustments or just prior to being output.

A standard twelve parameter matrix can also be used to transform the input coordinates. This matrix can be defined inside the part program or in a separate adjustment file which can be used for machine setup compensation.

1.2.6 Circular Interpolation

PostWorks supports the following 10 different formats for circular interpolation.

- | | |
|------------------------------------|---------------------------------------|
| 1. Absolute center. | 6. Radius. |
| 2. Incremental distance to center. | 7. Polar coordinates. |
| 3. Unsigned incremental distance. | 8. Dual block. |
| 4. Absolute/incremental. | 9. Absolute & Radius |
| 5. Absolute/Unsigned. | 10. Conversational Polar Coordinates. |

PostWorks will output linear moves to simulate the circular arc if the machine does not support circular motion. Circular moves can be broken up on quadrant, hemisphere or full 360 degree boundaries.

1.2.6.1 Circular Tolerances

PostWorks determines when and how to output circular interpolation blocks based on the following user defined tolerances.

1. Minimum/Maximum radius allowed.
2. Radius tolerance.
3. Minimum delta movement allowed.

The output position on the circle can also be adjusted to lie exactly on the circle, compensating for tolerancing factors in CAM systems that iterate around the circle.

1.2.6.2 Helical Interpolation

Helical interpolation can be output in a single block containing circular interpolation codes and a final depth, or **PostWorks** will generate linear moves to simulate the helical movement if the machine does not support helical interpolation.

1.2.7 Spline Interpolation

Some machine controllers have support for spline interpolation, which allows for a concise spline definition to be used to drive the machine. This eliminates the small linear moves required to approximate the spline, generating less tape and generally a smoother tool path on the machine. Spline interpolation is also beneficial to high speed machining.

PostWorks supports the output of spline records and has its own curve fitting algorithms, so no special data is required from the CAM processor. Standard linear motion is analyzed and as many points as possible are eliminated, while still holding the tolerance of the generated spline.

Spline interpolation formats currently supported are for the Siemens 880 control and the Fanuc 16 control. See “[Spline Interpolation Format](#)” section of **MPost** utility for details.

1.2.8 Motion Generation

PostWorks has built in routines that can internally generate or alter motion for certain functions that are not supported by the machine. These routines enable the programmer to expend more effort in programming the part, rather than programming for the idiosyncrasies of the machine.

1.2.8.1 Mutually Exclusive Axes

Some machines lack the capabilities to move all of the available axes at the same time. On these machines, **PostWorks** will check for mutually exclusive axes and break up the motion block so that these axes are in separate blocks.

A user definable controlling axis will determine how the move is broken up based on its positive or negative direction. You can define up to 4 sets of mutually exclusive axes.

1.2.8.2 Rotary Axes Linearization

PostWorks will linearize moves that contain rotary motion in order to keep the tool end point within a specified tolerance of the programmed move. Linearization consists of breaking up the move into smaller moves so that the tool end point moves in a straight line instead of a circular arc.

Rotary motion that takes the longest route due to machine limits can be altered as follows:

1. Retract the tool up the tool axis.
2. Position to the new rotary angles.
3. Plunge the tool back to its original position.

This is a user selectable feature which can prevent the part from being scrapped whenever radical movements in the rotary axes are programmed.

1.2.8.3 Polar Interpolation

When a machine with a rotary table is active, and the tool axis is parallel to the table's rotation axis, the programmer has the option of having the rotary table move to fulfill linear positions instead of the linear axes. This is commonly used when programming a large cylinder or spiral, providing for a better finish on the part. **PostWorks** will calculate the table positions based on the input linear positions. The linear axes may also move in conjunction with the rotary table, depending on if cylindrical or spiral motion is programmed.

1.2.8.4 Slowdown Blocks

Some older machines require a position of a move to be programmed at a slower feed rate in order to keep the tool within tolerance when changing directions. **PostWorks** will generate these slowdown blocks based on a user defined tolerance when required.

1.2.8.5 Transformation Blocks

Some newer controllers support transformation matrix that changes the default coordinate system. This is useful on machines with a rotating head. The transformation matrices can be used to align the current working plane with the standard XY-plane, thereby allowing automatic cycles, circular interpolation, etc. to be output. **PostWorks** will generate these transformation blocks based on user's specification in the **MPost** Utility and **PWorks** will modify the motion output accordingly.

1.2.9 Mill Cycles

PostWorks supports 14 different automatic/manual cycles for a mill, including drilling, boring, backboring and tapping cycles. These cycles can be output in automatic mode, for example using G81, G82, etc. or in manual mode, where **PostWorks** will generate linear moves to simulate the cycle. Up to 12 parameters can be entered in an automatic cycle.

Automatic cycles are supported in all 3 machining planes: XY, YZ and ZX. Simulated cycles are also supported in all of the machining planes and in addition in 3-axis mode, where the tool is not normal to any of the planes.

1.2.9.1 Cycle Simulations

PostWorks will simulate the cycle sequence whenever any of the following conditions are true.

- 1) The machine does not support automatic cycles.
- 2) A cycle code for the active cycle has not been defined. For example, if a code for *CYCLE/BORE9* has not been defined and *CYCLE/BORE9* is programmed, then a simulated cycle will be generated.
- 3) The programmer has explicitly requested simulated cycles by using the *CYCLE/ATUO,OFF* command.
- 4) The tool is not normal to any of the 3 machining planes. The cycle will be simulated along the tool axis using 3-axis moves.

1.2.9.2 Cycle Interruption

The *RAPID* command programmed during a cycle sequence will automatically cancel the cycle for the rapid move and reinstate the cycle following the move. A cycle interruption code can be defined, if the machine supports this feature, if not, **PostWorks** will output a cycle off code and perform the rapid move.

The *CYCLE/AVOID* command can be used to retract the tool for clamp avoidance anytime during the cycle sequence. **PostWorks** will cancel the cycle and retract the tool to a pre-defined clearance plane above the clamp. The cycle will be reinstated at the next position.

PostWorks has the capability to always position the tool in the same direction at each position during a cycle sequence if so required by the machine. When unidirectional positioning is activated, **PostWorks** will cancel the cycle, move to a position at a specified direction and distance from the hole, turn the cycle back on and position to the hole.

1.2.9.3 Cycle Block Output

The user can define complex automatic cycle blocks with the use of *User Defined Blocks*. For example, the following cycle command can output either of the two sample blocks, depending on the machine configuration.

Input

CYCLE/DEEP,FEDTO,.9.8,RAPTO,.1,STEP,.2,.1,IPM,10

OUTPUT 1

G83 X_ Y_ Z_ I_ J_ R_ F_ \$

OUTPUT 2

G79 I_ J_ R_ \$

G83 X_ Y_ Z_ F_ \$

1.2.9.4 Cycle Time Calculations

PostWorks calculates the machining time for an automatic cycle sequence based on a simulated cycle. The user can select the machining time calculation to use for each supported automatic cycle from the list of simulated cycles, resulting in a close approximation of the actual time the cycle motion will take on the machine.

1.2.10 Lathe Cycles

PostWorks supports 14 different automatic/manual cycles for a lathe, including facing, turning, drilling and thread cutting cycles. These cycles can either be output in automatic mode, for example using G75, G76, etc. or in manual mode, where **PostWorks** will generate linear moves to simulate the cycle. Up to 15 parameters can be entered in an automatic cycle.

Thread cutting cycles supported include constant lead and increasing or decreasing variable lead threads.

1.2.11 Spindle Speed Control

PostWorks supports both RPM and SFM spindle speeds. The user defines which modes the machine supports, the number of spindle ranges and how the spindle speeds should be output (including direct RPM/SFM values, a user definable spindle table or a percentage of the maximum RPM allowed).

A maximum of 3 spindle ranges are supported, each with their own minimum and maximum speeds and optional tables. Surface Feed per Minute can be output directly, if supported by the machine, or else **PostWorks** will calculate and output the spindle RPM required to maintain the programmed SFM speed.

Combination spindle and coolant on codes (M13, M14, etc.) can be output when turning both the spindle and coolant on in the same block.

1.2.12 Feed Rates

Supported feedrate modes include Feed per Minute (FPM), Feed per Spindle Revolution (FPR), Inverse Time (1/T) and rotary axes Degrees per Minute (DPM). FPM and FPR feedrates can be output as a direct value or using a predefined feed rate table. Normal and extended precision FPM

and 1/T feedrates are supported. DPM feedrates can be output as pure degrees per minute or a combination of feed per minute (for the linear axes) and degrees per minute (for the rotary axes).

The maximum feed rate can be specified separately for each axis.

PostWorks will check the rate of travel for each axis on a programmed move and slow down the feed rate whenever an axis exceeds its maximum rate.

1.2.13 Rapid Motion

Rapid moves can either be output as a high feedrate or with up to 5 codes that will cause the machine to move in rapid traverse rate. Rapid motion can be altered by **PostWorks** in the following ways, selectable by the user.

1. Along any of the major axes.
2. Along the tool axis.
3. A portion of the move can be made at the rapid rate, with the remaining portion at the programmed feed rate.

The user has the option of having **PostWorks** retract the tool to the clearance plane for each rapid move, independently of the alteration requested above. Rapid can either be a single shot command or be made modal until another feed rate is programmed, at the discretion of the user.

1.2.14 Tool Change Sequences

PostWorks can be configured for a variety of different tool change requirements, including various types of automatic tool changers. Following is a list of the tool change operations that are supported.

1. Automatic tool selection prior to loading.
2. 2 sizes of tool grippers.
3. Dual spindle capability.
4. Direct loading/unloading of tool.

The tool register can be output with only the tool number or with the tool number and the tool length offset register and/or the cutter compensation register.

1.2.15 Tool Length & Fixture Offsets

Both tool length and fixture offsets output is user selectable. Codes that enable the offsets at the control can be output, such as G43, G45, etc., or **PostWorks** will internally compensate for these offsets.

1.2.16 Cutter Compensation

PostWorks supports various forms of cutter compensation. Simple cutter compensation blocks (G41, G42) can be output or **PostWorks** will calculate and output cutcom vectors on each block with cutter compensation active. Following are some of the cutcom vector calculations that **PostWorks** supports.

1. Vector in direction of compensation with optionally 5-axis compensation allowed such as Cincinnati PQR's.
2. Vector parallel to current move.
3. Vector parallel to next move.
4. Angle of the next move (polar coordinates).

The user has the option of overriding the vector calculation values and outputting predetermined values for the cutcom vector. Cutter compensation is allowed in all 3 machining planes and with circular interpolation. **PostWorks** will also check for a maximum angular change during cutter compensation and output an error message if it is exceeded.

1.2.17 Machine Dwells

PostWorks allows dwells to be input in either number of seconds to dwell or number of spindle revolutions to dwell. Dwell output can be in one of the following user selectable formats.

1. In time (Milliseconds, Seconds or Minutes).
2. Number of spindle revolutions to dwell.
3. Multiple blocks containing a dwell. The dwell time for each dwell code is stored in the machine control.

1.3 Post-processor Macros - *PostMacro*

The **PostMacro** utility is a set of 2 routines that allow you to create post-processor *Macros* that can be used to alter and expand post-processor commands. Post-processor *Macros* are similar to programming subroutines, in that they are called from either the main program or another macro, and contain a group of commands that will be processed each time the *Macro* is called. Following is a list of some of the uses for post-processing *Macros*.

1. Build a library of post *Macros* for different types of machines. The syntax of the post-processor commands in the part program file can remain the same, but the functionality of the commands will differ depending on the machine specific post-processor requirements. This allows you to post a single part program for a multitude of machines without any modifications to the program.
2. Define post-processor macros that accept the same syntax as your current library of post-processors. These *Macros* can convert your old syntax to a syntax required by

PostWorks, allowing you to run old part programs against the **PostWorks** post-processor without making any changes to the program.

3. Define A *GOTO Macro* that will modify the cl points prior to final processing by **PostWorks**. Some of the reasons you may want to modify the cl points are: to define your own manual cycles, or to modify the tool axis vector for a fixed rotary axis or angled spindle attachment.
4. Create a post-processor tool change *Macro* that contains all of the logic and commands of your current part program tool change macro. This will help clean up the source of your part program and make it compatible with multiple machines.

1.3.1 Macro Features

Post-processor *Macros* are defined in a source file, separate from the part program file. This file is compiled into an object file by the **PostComp** routine. **PostWorks** will process this object file and merge the clfile with the post *Macros*, generating a new input clfile that will be processed for the machine.

A Macro will be called whenever the post command of the same name is encountered in the clfile (or another *Macro*) and the *Macro* has been enabled. All post-processor commands (*LOADTL*, *STOP*, etc.) can be defined as *Macros*, along with *special Macros* to handle the start-up of a program, the end of a program (*FINI*), cl points (Motion), tape breaks and error messages.

1.3.2 Macro Arguments

Minor words/values on the calling post-processor statement are passed to the called *Macro* as arguments. Minor word arguments will be stored as character strings, containing the text representation of the word ("*CLW*", "*OFF*", etc.). *Macro* arguments, both text and numeric, can be used anywhere variables can be used in the *Macro*; equations, post commands, *IF* statements, etc. The number of arguments passed to the *Macro* is also available as a numeric variable inside the *Macro*. Up to 50 minor words/values can be passed.

1.3.3 Real & Text Variables

The **PostMacro** utility supports both real and text variables. Variables can be declared as either local to the *Macro* in which they are defined or as global, accessible by all of the *Macros* in the program.

1.3.4 Post Variables

Certain variables within **PostWorks** can be accessed within a *Macro*. These variables can be used anywhere a standard variable can be used and are referenced as '%name'. Some of the

variables that can be used in a Macro are the input sequence number (*%ISN*), the cl point data (*%CLPT*), the *MULTAX* flag (*%MULTAX*) and the value of pi (*%PI*).

1.3.5 Operators and Functions

PostMacro contains a full compliment of math/text operators and functions. Numeric and text expressions may be composed of operators, functions, variables, *Macro* arguments and *Post variables*. Included are the standard operators; “+ - / * **”; and logical operators “== < > <= >= & |”. Some of the numeric functions supported are the trigonometric functions (*SIN*, *COS*, *ATAN*) and various other functions (*INDEX*, *CTOR*, *INT*, *SQRT*). Text functions include *COMMAND*, *DATE*, *TIME* and *FMTCOD*. Expressions can be statements by themselves and can appear most everywhere that a variable/value can.

1.3.6 Post-processor Commands

Standard post-processor commands, such as *LOADTL*, *CUTCOM*, *PPRINT*, etc., have exactly the same format as in the part program file. Parameters can include minor words, real & text variables, equations, *Post* variables and *Macro* arguments. In addition, *Macro* arguments may contain a range subscript specifier ‘(start:end)’, allowing easier entry of multi-parameter commands.

1.3.7 Format Descriptors

Format descriptors are used when converting a number to a character string (*RTOC*) or a character string to a number (*CTOR*). These allow you to define the floating point format (leading/trailing zero suppression or decimal point formats), the number of digits to the left and right of the (implied) decimal point, the minimum number of digits to the left and right of the (implied) decimal point to output, and whether a plus and/or minus sign is included in the character string.

1.3.8 Program Control Statements

Statements within a *Macro* are usually executed in the order in which they were written. However, you may use control statements to transfer execution to another statement in the same *Macro*. Control statements supported are the *JUMPTO/label* command, logical *IF* statements (including *IF-THEN-ELSE* structures) and repetitive *DO* loops.

1.3.9 Special Macros

Besides the normal *Macros* called by post-processor commands, you may also define *Macros* that are called at specific points while processing the clfile. The *G\$MAIN* macro is called at the very start of the clfile, before any other processing is done. The *FINI* macro is processed at the end of the clfile. The *GOTO* macro is called whenever a motion record is processed. The *ERROR* macro

allows you to selectively disable some error messages and take certain action on others. The *TAPBRK* macro lets the programmer custom design the output sequence for tape breaks.

1.3.10 Syntax Checking

The *SYNTAX* statement verifies that the post-processor command that called the current *Macro* is syntactically correct by using your specifications. The minor words/values in the calling command will be checked against the words/values in the *SYNTAX* command. If there is a conflict, then the *%ERROR Post* variable will be set to 1.

1.3.11 Reading/Writing External Files

PostMacro allows the user to read and write from/to external text files with the commands: FOPEN, FCLOSE, FREW, FREAD and FWRITE during the post processing of the input clfile.

1.3.12 Look Ahead In CLfile

The CLREAD, CLFIND commands allow the user to search ahead for user specified records in the input clfile and let the user to decide what need to be done first if the specified record is located.

1.3.13 Standalone Module - **PMacro**

PostMacro in both an integral part of **PostWorks** and is furnished as a standalone module that can be used as a pre-processor for a custom post-processor. When used as a standalone module, **PMacro** will process the input clfile and merge it with the *Macro* object file creating a new cl file to run against a custom post-processor, thereby adding the features of post-processor Macros to existing post-processors.

PMacro can automatically run the post-processor(s) whose *MACHIN* cards are encountered in a *Macro* or the input clfile. The machine specific post-processor(s) will be run against each of the output clfile(s) created. The post-processor(s) whose *MACHIN* cards are found in the input clfile will be run against each of the output clfile(s), while a *MACHIN* card contained in a *Macro* will only run the post-processor for the cl file created using the object file that contains that *Macro*.

You can also use the **PMacro** standalone module without any accompanying post-processor. It can be used solely to create output for CMM's, geometry checking programs, etc.

1.4 Universal Post-processor - **PWorks**

PWorks is the actual post-processor module within **PostWorks**. In combination with the **MPost** post-processor generator, **PostMacro** post-processor *Macros* and **PWorks** the universal post-processor, **PostWorks** can be configured to support the majority of machines in use today.

PWorks uses the *MACHIN* card in the part program to load a *Machine Descriptor File (MDF)* and a *Macro* object file prior to processing the input part program. The *MDF* file is generated with the **MPost** utility and contains a complete description of the target machine. The *Macro* object file is generated with the *v* utility and contains post-processor *Macros*, which are called by standard post-processor commands within the part program.

1.4.1 Output Files

PWorks can create four types of output files: a punch file containing the Control Tape for the machine, a user defined print file, a machine simulation file used as input to verification routines, and a listing file.

The machine simulation file contains a neutral format of the punch file and is used as input to the verification routine **NCL/IPV**. This file contains the actual positions and feed rates that will be sent to the machine. All adjustments made by **PostWorks** (linearization, cycles, helical interpolation, etc.) are output to the simulation file as standard axis positions. The verification process using the simulation file instead of a clfile or APT Source file shows the actual movement that the machine will make, instead of the positions output by the CAM processor.

The listing file contains a text representation of the input clfile after it has been modified by post-processor *Macros*. Any modifications done to the clfile because of user defined *Macros* will be contained in the listing. The programmer can also format and output messages to the listing file at any location within the part program.

1.4.2 Punch File Header

An optional leader and/or trailer section can be added to the output punch file. The definition of these sections are stored in a *Punch file Header File (PHF)*, which can contain simple text records combined with formatted post-processor variables. The header and trailer sections will usually contain identifying information about the punch file, such as creation date, post-processor version, lengths and machining times for each tool used, etc. These sections are created during a second pass after the entire part program has been processed, so any variables used, such as total machining time, will be assigned the values which they have at the end of the part program.

1.4.3 Print File Description

The format of the print file generated by **PWorks** is completely user definable. The print file description is stored in a *Print Descriptor File (PDF)*, which contains both text data and post-processor variables, along with the location of each within a record. You can define up to 10 records, each with a maximum of 6 lines per record. Each *PDF* record is assigned to a specified **PWorks** output sequence (Motion block, Non-motion block, *STOP* sequence, *FINI* sequence, etc.) which controls when it will be output. A *PDF* record can also be explicitly output at any time during the part program at the discretion of the programmer. You can have a different print file format for each post-processor configuration.

1.4.4 Unique Filenames

PWorks can process a single part program and create output for up to 10 machine configurations. A feature of **PWorks** allows you to append the machine number of the *MDF* file to the name of the output files, thus creating specific and unique file names. For example, the input file TEST.cl, which is processed for machine #33, would create the output punch file TEST_33.pu1.

1.4.5 Multiple Clfile Support

PWorks will work with the following clfile formats:

1. Binary clfiles from both **NCL** and Catia.
2. APT Source files which can be generated by virtually every CAD/CAM system in use today.
3. Text ASCII “.cls” file from Unigraphics.
4. “.nci” file from MasterCam.

CHAPTER 2 Getting Started

2.1 Installing *PostWorks*

PostWorks must be installed properly on your system before it can be used. Installation consists of loading **PostWorks** into the proper directories and letting the operating system know it is there. You will use the *INSTALL* program furnished with the **PostWorks** distribution media to load **PostWorks** onto the system.

If **PostWorks** is delivered with your CAM software, then the procedure for installing **PostWorks** detailed in this chapter may not apply. The standard installation procedure for your CAM software should automatically install **PostWorks**.

2.1.1 Installing *PostWorks*

PostWorks can run on PCs running the Windows operation system. It is delivered on flash drive or electronically through FTP or email and takes up approximately 40MB on the hard drive.

The industry standard “Setup” utility is used for the **PostWorks** installation. Simply select “SETUP” to start the installation procedure and follow the instructions on the screen.

Once the installation of **PostWorks** is completed you must enter the Software Product Key for it to run. Refer to the **PostWorks** Runtime License section on how to enter the Software Product Key.

2.1.2 *PostWorks* Runtime License

After the installation of **PostWorks** is completed, you must register a valid Software Product Key (SPK) on the computer on which you wish to run **PostWorks**. The SPK is furnished with **PostWorks** either as a license file: “*file_name.lic*” (where file-name usually is your company’s name) or as a written form. The SPK should be entered using the *NCCS_LICENSE* routine.

Use the following procedure to register the **PostWorks** SPK using the licensing routine.

- 1) Start the licensing routine.

Click Start > Programs > NCCS > License > NCCS_License

- 2) The next step depends on how the SPK is furnished.
 - a. If the SPK is furnished as a license file with the name “*file.lic*”, click “Load License” on the pull down “File” menu to open a window browser. Locate “*file.lic*”, highlight it and click “OPEN”. This will load the SPK into the form. Click “Exit” on the pull down “File” menu to exit the license routine.

- b. If the SPK is furnished as a written form, enter the data **EXACTLY** as it appears on the SPK form. After entering the entire SPK, press the “ADD” key to store it in the licensing data base. Repeat for each SPK form you are registering. After all SPKs have been entered, Click “Exit” on the pull down “File” menu to exit the license routine.

The licensing routine creates a data base file named *NCCS_LICENSE.DBA*.

2.2 File Specification

Input and output file names provided by the user must conform to a certain format. Following is the acceptable syntax for a file specification.

device:\directory\filename.typ

The only mandatory element of an input file specification is the file name. The output file name will default to the input name. The device and directory elements will default to your current defaults. The version will default to the latest version. The file type will default differently depending on the type of file.

Windows environmental variables may be used in place of the device and directory specifications.

2.3 Generating a Post-processor

This section describes the procedures for generating a post-processor configuration for a new machine and running a part program against it. It contains only an overview description of the steps involved in generating a new post-processor. See the specific reference manual, for each routine, for a complete description of the routines.

- 1) Determine a number/name scheme for the post-processor you are about to create, as **PostWorks** used these numbers/names for the purpose of identifying each file required to define a post-processor configuration. The [Machine Configuration Files Appendix](#) contains the numbering scheme used by **NCCS**.
- 2) Use the [Print File Description form](#) contained in the **MPost** Reference Manual to define the print file layout. Then edit the ‘*pworks_#.pdf*’ file to reflect this layout.
- 3) Run the **MPost** routine to generate the *Machine Descriptor File (MDF)*. You can use an existing *MDF* file as the default settings if it is similar to the post-processor you are creating, or you can generate the *MDF* file from scratch. The **MPost** chapter contains information on running the **MPost** utility. See the [MPost Reference Manual](#) for a complete description of the **MPost** utility.
- 4) Create a ‘*pmacro_#.mac*’ file that contains the post-processor *Macros* you wish to create for this post-processor. Post-processor *Macros* are not required to successfully

run **PostWorks**, but they can greatly simplify task of part programming. They are required when running only the **PostMacro (PMacro)** portion of **PostWorks**. The 'pmacro_3.mac' file is created using a standard text editor such as 'NotePad'. Be sure to assign the same number/name to the *Macro* file as you used for the *MDF* file.

- 5) Compile the 'pmacro_#.mac' or 'pmacro_name.mac' file into an object file (.OBJ) using the **PostComp utility**. Any errors generated will be displayed on the screen and in the listing file (if created). Fix any errors that occur, as the object file will be corrupted if there are any errors during compilation.
- 6) Each part program that uses this post-processor should contain a '**MACHIN/PWORKS,#**' or '**MACHIN/name**' line, where '#', 'name' is the number or name assigned to the post-processor configuration files. If you wish to run only the **PMacro** portion of **PostWorks**, in order to utilize the *Macro* features only and create a new clfile, then use the '**MACHIN/PMACRO,#**' line instead.

The machine specific post-processor generation is now complete. The following files should have been created during the generation process (where # can be a number or a name”).

pworks_#.pdf,	PWORKS_#.MDF	(will not be created for PMacro)
pmacro_#.mac,	pmacro_#.OBJ	(if <i>Macros</i> are used.)

These files can remain in your current working directory during the testing phase of your new post-processor. After the files have been proofed they can be moved to the *POWORKS_DATA* directory so that they can be utilized by all users of **PostWorks**.

CHAPTER 3 Using *MPost*

3.1 Invoking *MPost*

The following command will call up the *MPost* routine.

```
mpost [mchin [mchout]] [-option [...] -option]
```

You may also run *MPost* using the Start menu or a Shortcut icon.

‘mchin’ is the name of the *MDF* file to load when entering the *MPost* utility. ‘mchout’ is the number of the *MDF* file to create upon exiting *MPost*. The ‘mchin’ and ‘mchout’ numbers will be converted to a file name in the format of ‘*PWORKS_#.MDF*’.

Example:

```
mpost 33 34
```

In the above example, *MPost* will load the *PWORKS_33.MDF* as the default Machine Descriptor File and will change the machine number to 34, thereby creating *PWORKS_34.MDF* upon exiting.

‘options’ are runtime options that affect the initialization of *MPost*. Each option must start with the ‘-’ character. The following sections describe the runtime options in detail.

3.2 *MPost* Runtime Options

Runtime options are entered on the operating system command line when *MPost* is initially invoked. They affect the initialization of *MPost* and the default settings. Runtime options may be abbreviated to their shortest unique name, for example, *UNITS* may be shortened to *U*. In the following sections the required portion of the option name will be in capital letters, while the optional characters will be in lower case.

‘*makepost.ini*’ is a file that contains default runtime options for the *MPost* routine. When *MPost* is first invoked it will search the users directory for ‘*makepost.ini*’ and if it is found, will use the runtime options contained in the file as the defaults. If ‘*makepost.ini*’ is not found in the users’s directory, then *MPost* will search for it in the *PWORKS_DATA* directory. This file does not have to be present for *MPost* to run correctly.

3.2.1 Page Length - PAGE_LEN

Command Syntax: `mpost infile -Page_len:n`

Default: `-PAGE_LEN:60`

The *PAGE_LEN* options specifies the page length of the [automatic documentation file](#). 'n' is the number of lines to output to this file between page breaks.

3.2.2 Defining Input Units - UNITS

-Command Syntax: `mpost infile -Units:INCH`
`MM`
`SAME`

Default: `-UNITS:INCH`

The *UNITS* option defines the working units, either inch or millimeters, for this **MPost** session. An *MDF* file created in one units (*INCH* or *MM*) will be converted to the correct working units, as specified by the *UNITS* option, when loaded. *MDF* files will always be saved in the current working units.

When loading an external *MDF* file, the word *SAME* can be used to load the file using the same units that it was stored in. If you specify *SAME* and do not load an existing *MDF* file, then *INCH* will be assumed.

3.3 MPost Basics

When working with the **MPost** utility it is important to know the procedure for navigating the menu structure and entering information in prompts and forms. There are 3 different types of sections within the **MPost** utility: menus, prompts and forms. The following sections discuss the process for entering information in each type of section.

3.4 Register Labels

PostWorks supports 92 registers which need to be referenced through the **MPost** [utility](#) when defining the Control Tape output. The registers are referenced with a unique label that describes exactly which register to use.

Registers can be entered with just their identifying label or with an optional value. When a register label is specified with a value, it has the format 'reg(val)', for example 'G0(91)'.

Following is a list of physical register labels.

A1	A2	A3	B1	B2	B3	C1	C2	C3	C4	C5	C6
D	E	F1	F2	F3	G0	G1	G2	G3	G4	G5	G6
G7	G8	G9	GA	H	I1	I2	J1	J2	K1	K2	L
M0	M1	M2	M3	M4	M5	M6	M7	M8	M9	MA	N
O	P	Q	R	S	T	U1	U2	V1	V2	W1	W2
X1	X2	Y1	Y2	Z1	Z2	AA	AB	AC	AD	AE	AF
AG	AH	AI	AJ	AK	AL	AM	AN	AO	AP	AQ	AR
AS	AT	AU	AV	AW	AX	AY	AZ				

In addition to the physical register labels, **PostWorks** also supports logical register labels. Logical register labels are used to specify a type of register, for example, a G-code when the group is not known or an axis address when the mode (absolute/incremental) is not known. Following is a list and description of the logical register labels.

- G = Specifies a G-code register when the group of the G-code is not known. A value must be specified when G is used as the register label, for example G(00).
- M = Specifies an M-code register when the group of the M-code is not known. A value must be specified when M is used as the register label, for example M(02).
- \$ = The \$ character does not select one of the registers, but specifies an End-of-block. \$ is usually used in [User Defined Blocks](#) in order to generate multiple output blocks with a single User Defined Block.
- X = Either the absolute or incremental primary X-axis register. **PostWorks** will select which register to use depending on the current programming mode.
- U = Either the absolute or incremental secondary X-axis register.
- Y = Either the absolute or incremental primary Y-axis register.
- V = Either the absolute or incremental secondary Y-axis register.
- Z = Either the absolute or incremental primary Z-axis register.
- W = Either the absolute or incremental secondary Z-axis register.
- A = Either the absolute or incremental Rotary axis #1 register.
- B = Either the absolute or incremental Rotary axis #2 register.
- C = Either the absolute or incremental Rotary axis #3 register.
- @ = Either the absolute or incremental Rotary axis #4 register.

F = The active mode feed rate register. **PostWorks** will select a feed rate register depending on the current output feed rate mode (FPM, FPR, 1/T or DPM).

3.5 Menus

A menu consists of push buttons which are used to call up other menus or forms. Multiple menus and forms may be active at the same time.

3.5.1 The Banner Line

The banner line contains both a text and numeric representation of the section you are currently in and the current number for the *Machine Descriptor File (MDF)*. It is always displayed as the window banner. The banner line has the following format.

Section: **Misc Setup** *PostWorks Generator / **88** * Level: 1.

In this example, the current section is 'Misc Setup', the *MDF* file being defined is *PWORKS_88.MDF* and the numeric level is 1. The menu displayed in this section will be 1.1, 1.2, etc.

3.5.2 Making a Menu Selection

Simply push the menu button using the mouse cursor to choose a selection.

CHAPTER 4 Using *PostMacro*

4.1 *PostComp* Overview

The ***PostComp*** routine is a compiler that takes a Macro definition file as input and creates an object file that will be interpreted by either ***PMacro*** or ***PWorks***. All syntax and command validity checking is done by ***PostComp***, removing the burden from the *Macro processor* resident in both ***PMacro*** and ***PWorks***, thereby improving their performance.

PMacro uses the number(s) on the *MACHIN/PMACRO* card, and ***PWorks*** uses the number(s)/name(s) on the *MACHIN/PWORKS* or *MACHIN/name* card to generate the *Macro object file* name '*pmacro_#.OBJ*' or '*pmacro_name.OBJ*', therefore it is required to name the input *Macro definition file* '*pmacro_#.mac*' or '*pmacro_name.mac*', where '#' is a unique number identifying this file for the *Macro processor*.

PMacro allows the use of number only for the macro definition file. ***PWorks*** allows the use of either a number or a name for the macro definition file.

4.2 Invoking *PostComp*

The following command will call up the ***PostComp*** routine.

```
postcomp filename [-option [...] -option]
```

You may also run ***PostComp*** using the Start menu or a Shortcut icon. In this case an interactive window will appear asking for the input file name and runtime options.

'*filename*' is the name of the input *Macro* definition file, in the format '*pmacro_name.mac*' or '*pmacro_#.mac*'. You may specify the filename as "*name*" or "*#*" only, in which case ***PostComp*** will automatically generate the file name for you. The default file type is .mac'.

Example:

```
postcomp mypost  
or  
postcomp 128
```

In the above example, ***PostComp*** will use the file '*pmacro_mypost.mac*' or the file '*pmacro_128.mac*' as the input *Macro* definition file.

'options' are runtime options that affect the initialization of ***PostComp***. Each option must start with the '-' character. The following sections describe the runtime options in detail.

4.3 **PostComp** Runtime Options

Runtime options are entered on the operation system command line when **PostComp** is initially invoked. They affect the initialization of **PostComp** and the default settings. Runtime options may be abbreviated to their shortest unique name, for example, *PAGE_LEN* may be shortened to *PA*. In the following sections the required portion of the options name will be in capital letters, while the optional characters will be in lower case.

'*postcomp.ini*' is a file that contains default run time options for the **PostComp** routine. When **PostComp** is first invoked it will search the users directory for '*postcomp.ini*' and if it is found, will use the run time options contained in the file as the defaults. If '*postcomp.ini*' is not found in the user's directory, then **PostComp** will search for it in the *PWORKS_DATA* directory. This file does not have to be present for **PostComp** to run correctly.

4.3.1 Documentation File - DOCUMENT

Command Syntax: `postcomp infile - [NO]Document[:[file][.ext]]`

Default: **-NODOCUMENT**

PostComp has the option of creating a [documentation file](#) while processing the Macro definition file. The documentation file contains the text of all [REMARK](#) statements in the input file, thus allowing you to document the *Macros* in the source code and create a printable version to distribute to the users.

If the *NO* qualifier is specified with the *DOCUMENT* option, then a documentation file will not be created. '*file*' and '*ext*' are the name and the extension of the documentation file to create, at least one of these parameters must be specified if ':' is specified. The default file name is the input file name with an extension of '*.doc*'. If only '*file*' is specified, the documentation file name will be the input file name with an extension of '*.doc*'. If only '*ext*' is specified, the documentation file name will be the input file name with an extension of '*.ext*'. If both are specified, the documentation file name will be '*file.ext*'.

The text of the *REMARK* statements will be output exactly as is, without any modifications, except the *REMARK* major word will be removed. A form feed character will also be output at the end of each page.

4.3.2 Listing File - LISTING

Command Syntax: `postcomp infile - [NO]Listing[:[file][.ext]]`

Default: **-NOLISTING**

A [listing file](#) can optionally be created while processing the input [Macro definition file](#). The listing file will contain a header, the input line numbers, the beginning instruction's **PC** address for each statement, the input statement and any error messages generated during the compilation. A summary of the *Macros* defined and their sizes will be output at the end of the listing.

If the *NO* qualifier is specified with the *LISTING* option, then a listing file will not be created. '*file*' and '*ext*' are the name and the extension of the listing file to create, at least one of these parameters must be specified if ':' is specified. The default file name is the input file name with an extension of '*.lis*'. If only '*file*' is specified, the listing file name will be the input file name with an extension of '*.lis*'. If only '*ext*' is specified, the listing file name will be the input file name with an extension of '*.ext*'. If both are specified, the listing file name will be '*file.ext*'.

4.3.3 Object File - OBJECT

Command Syntax: `postcomp infile -[NO]Object[:[file][.ext]]`

Default: **-OBJECT**

A *Macro* object file contains binary encoded instructions which will be interpreted by **PMacro** or **PWorks** while processing a [clfile](#). Statements in the [Macro definition file](#) are compiled into executable instructions that are understood by the *Macro* processor. Most input statements will generate multiple executable instructions.

If the *NO* qualifier is specified with the *OBJECT* option, then an [object file](#) will not be created. '*file*' and '*ext*' are the name and the extension of the object file to create, at least one of these parameters must be specified if ':' is specified. The default file name is the input file name with an extension of '*.OBJ*'. If only '*file*' is specified, the object file name will be the input file name with an extension of '*.OBJ*'. If only '*ext*' is specified, the object file name will be the input file name with an extension of '*.ext*'. If both are specified, the object file name will be '*file.ext*'. It is recommended that you do not change the output file name and extension since **PMacro** will not see the object file if the object file is not in the format '*pmacro_#.OBJ*' and **PWorks** will not see the object file if the object file is not in the format '*pmacro_name.OBJ*' or '*pmacro_#.OBJ*', where *name* or '*#*' is the name or number used in the macro file '*pmacro_name.mac*' or '*pmacro_#.mac*'.

4.3.4 Page Length - PAGE_LEN

Command Syntax: `postcomp infile -Page_len:n`

Default: **-PAGE_LEN:60**

The *PAGE_LEN* option specifies the page length of the document and listing files. '*n*' is the number of lines to output to these files between page breaks.

4.3.5 Quiet Process - QUIET

Command Syntax: `postcomp infile - [NO]Quiet`

Default: `-NOQUIET`

This option only applies if **PostComp** is started from the command line. It does not have any effect if the **PostComp** graphic interface is used. At the end of the **PostComp** process, a *Process Error Log Window* will pop up to show the end result of the **PostComp** process. Using the “-Quiet” option will stop the pop up of this *Process Error Log Window* and send the process result to a log file with the name “*postcomp.log*” located in the current directory.

4.4 PMacro Overview

The **PMacro** routine is a pre-processor that takes a clfile as input and modifies it according to predefined post-processor *Macros* stored in a compiled object file created by **PostComp**. **PMacro** does the actual interpretation of the *Post Macros* and creates a modified **NCL** binary clfile to run against a custom post-processor. Optionally, the requested post-processor(s) will automatically be run against the new clfile(s).

PMacro may either be called via a runtime command or by the *MACHIN* card in the CAM processor. The *MACHIN* card may contain up to 10 numbers, specifying the *Macro object file(s)* to process the input clfile for. The *Macro* object file name must be in the format ‘*pmacro_#.OBJ*’, where ‘#’ is a number that is contained on the *MACHIN* card. **PMacro** will create a separate output clfile for each *Macro* object file that it processes the input clfile for.

4.5 Invoking PMacro

The following command will call up the **PMacro** routine.

```
pmacro filename [-option [...] -option]
```

You may also run **PMacro** using the Start menu or a Shortcut icon. In this case an interactive window will appear asking for the input file name and runtime options.

‘*filename*’ is the name of the input clfile to process. The default file type is ‘.cl’. ‘*options*’ are runtime options that affect the initialization of **PMacro**. Each option must start with the ‘-’ character. The following sections describe the runtime options in detail.

4.6 PMacro Runtime Options

Runtime options are entered on the operating system command line when **PMacro** is initially invoked. They affect the initialization of **PMacro** and the default settings. Runtime options may be abbreviated to their shortest unique name, for example, *PAGE_LEN* may be shortened to *PA*. In

the following sections the required portion of the option name will be in capital letters, while the optional characters will be in lower case.

'pmacro.ini' is a file that contains default runtime options for the **PMacro** routine. When **PMacro** is first invoked it will search the users directory for *'pmacro.ini'* and if it is found, will use the runtime options contained in the file as the defaults. If *'pmacro.ini'* is not found in the user's directory, then **PMacro** will search for it in the *PWORKS_DATA* directory. This file does not have to be present for **PMacro** to run correctly.

4.6.1 Input Clfile Format - CLFILE

Command Syntax: `pmacro infile -Clfile:n[,wid]`

Default: **-CLFILE:0**

PMacro has the ability to process input binary clfiles from **NCL** and Catia (Up to Ver. 5), and to also process textual APT Source files. The APT Source file can be in standard *GOTO/point* format or can be in the **PTED** ASCII clfile format.

The *CLFILE* option allows you to specify which format the input clfile is in. 'n' selects the clfile format and can have one of the following values.

n	Clfile Format
-	-----
0	Automatically detect file type
1	NCL binary clfile.
2	APT Source textural file.
3	Catia binary clfile (Up to Ver. 4)
4	APT Source textural file, UG circle
5	Catia binary clfile (Ver. 5)
6	MasterCam NCI file

'wid' specifies the columnar width of the input APT Source file when the *CLFILE* option is set to 2. A standard APT Source file will only use the first 72 columns to store the APT Source record. Columns 73-80 are used for record numbers, comments, etc. Some CAD/CAM systems will use all 80 columns to store the APT Source record. The default for *'wid'* is 72.

*** When running against APT Source files with Unigraphics circular interpolation records you must specify `"-CLFILE:4,132"`, since **PMacro** cannot determine this clfile type automatically using the `-CLFILE:0` runtime option. ***

4.6.2 Unique Output Filenames - IDENT

Command Syntax: `pmacro infile - [NO] Ident`

Default: **-NOIDENT**

One of the features of **PMacro** is that it allows you to process a single [clfile](#) against multiple [Macro object files](#), creating a new clfile for each of the *Macro* object files processed. Under normal circumstances this would create multiple output files with the same file name. This is totally unacceptable since the operation systems do not support multiple files with the same name in the same directory.

The *IDENT* option allows you to append the machine number of the *Macro* object file to the name of the output cl and [listing files](#), creating specific and unique file names.

Example:

Input clfile	Macro object File	Output clfile
-----	-----	-----
test.cl	pmacro_33.OBJ	test_33.cln

If the *NO* qualifier specified with the *IDENT* option, then the machine number will not be appended to the output file name(s).

4.6.3 Listing File - LISTING

Command Syntax: `pmacro infile - [NO] Listing[: [file] [.ext]]`

Default: **-NOLISTING**

A [listing file](#) can optionally be created while processing the input clfile. The listing file will contain a header, the input sequence numbers from the part program, the cl record numbers of the output clfile, a text representation of the output clfile records, output generated using the [PRINT](#) command and any error messages generated while processing the input clfile.

If the *NO* qualifier is specified with the *LISTING* option, then a [listing file](#) will not be created. '*file*' and '*ext*' are the name and the extension of the listing file to create, at least one of these parameters must be specified. If ':' is specified, the default file name is the input file name with an extension of '*.lis*'. If only '*file*' is specified, the listing file name will be the input file name with an extension of '*.lis*'. If only '*ext*' is specified, the listing file name will be the input file name with an extension of '*.ext*'. If both are specified, the listing file name will be '*file.ext*'.

4.6.4 Overriding the MACHIN Card - MACHINE

Command Syntax: `pmacro infile -Machine:n`

Default: **NONE**

PMacro processes the input *clfile* for the *Macro object file(s)* specified on the *MACHIN/PMACRO* card contained in the *clfile* by default. There may be times when you want to process the *clfile* for a machine number that is not on the *MACHIN* card or to process it for only one of the machine numbers on the *MACHIN* card.

The *MACHIN* option allows you to override the *MACHIN/PMACRO* command and to process the input *clfile* for a specific *Macro* object file. 'n' is used to call up the *Macro* object file 'pmacro_n.OBJ'. 'n' must be a number and cannot be any other characters, i.e. no named *pmacro* file is allowed.

4.6.5 Output Clfile - OBJECT

Command Syntax: `pmacro infile - [NO]Object[: [clfile] [.ext]]`

Default: **-OBJECT**

The *OBJECT* option specifies whether or not to create an output *clfile*. It is normally desirable to output a *clfile*, since this is what is used by the machine specific post-processor to create the final output for the machine control. However, there are some cases where **PMacro** will be solely used to create a listing file that will contain analysis data or input to another program.

If the *NO* qualifier is specified with the *OBJECT* option, then a *clfile* will not be created. 'clfile' and 'ext' are the name and the extension of the *clfile* to create, at least one of these parameters must be specified. If ':' is specified, the default file name is the input file name with an extension of '.cln'. If only 'clfile' is specified, the output *clfile* name will be the input file name with an extension of '.cln'. If only 'ext' is specified, the output *clfile* name will be the input file name with an extension of '.ext'. If both are specified, the output *clfile* name will be 'clfile.ext'.

4.6.6 Page Length - PAGE_LEN

Command Syntax: `pmacro infile -Page_len:n`

Default: **-PAGE_LEN:60**

The *PAGE_LEN* option specifies the page length of the listing files. 'n' is the number of lines to output between page breaks.

4.6.7 Running Custom Post-processors - POST

Command Syntax: `pmacro infile - [NO]Post`

Default: **-NOPOST**

It is often desirable to run the custom post-processor(s) against the output clfile(s) directly after **PMacro** is finished processing. The *POST* option causes **PMacro** to automatically run the post-processor(s) whose *MACHIN* cards were in the input clfile and/or *Macro* object file(s) against the output clfile(s).

When automatic post-processing is desired it is recommended that the part program file contain only the *MACHIN/PMACRO* card and that the *Macro definition file(s)* contain the machine specific *MACHIN* cards. The reason for this is that the CAM processor will most likely not know the name of the output clfile to process against the machine specific post-processor, due to the *IDENT* option and the default *'cln'* file type of the output clfile.

4.6.8 Quiet Batch Process - QUIET

Command Syntax: `pmacro infile - [NO]Quiet`

Default: **-NOQUIET**

This option only applies if **PMacro** is started from the command line. It does not have any effect if the **PMacro** graphic interface is used. At the end of the **PMacro** process, a *Process Error Log Window* will pop up to show the end result of the **PMacro** process. Using the “-Quiet” option will stop the pop up of this *Process Error Log Window* and send the process result to a log file with the name “*pmacro.log*” located in the current directory.

CHAPTER 5 Using *PWorks*

5.1 *PWorks* Overview

PWorks is a generic post-processor for milling and/or lathe applications. With the use of the **MPost** Post-processor Generator utility and **PostMacro** Macros, **PWorks** can be configured to support the majority of mills and lathes in use today.

PWorks uses the *MACHIN* card in the part program to load a *Machine Descriptor File* (MDF) and a *Macro object file* prior to processing the input part program. The *Machine Descriptor File* is generated with the **MPost utility** and contains a completed description of the target machine. The *Macro object file* is generated with the **PostComp utility** and contains post-processor *Macros*, which are called by standard post-processor commands within the part program.

5.2 Invoking *PWorks*

The following command will call up the **PWorks** post-processor.

```
pworks infile [-option [...]-option]
```

You may also run **PWorks** using the Start menu or a Shortcut icon. In this case an interactive window will appear asking for the input file name and runtime options.

'infile' is the name of the input *clfile* to process. The default file type is '.cl'. 'options' are runtime options that affect the initialization of **PWorks**. Each option must start with the '-' character. The following sections describe the runtime options in detail.

5.3 *PWorks* Runtime Options

Runtime options are entered on the operation system command line when **PWorks** is initially invoked. They affect the initialization of **PWorks** and the default settings. Runtime options may be abbreviated to their shortest unique name, for example, *PRINT* may be shortened to *PRI*. In the following sections the required portion of the option name will be in capital letters, while the optional characters will be in lower case.

'*pworks.ini*' is a file that contains default runtime options for the **PWorks** post-processor. When **PWorks** is first invoked it will search the users directory for '*pworks.ini*' and if it is found, will use the runtime options contained in the file as the defaults. If '*pworks.ini*' is not found in the user's directory, then **PWorks** will search for it in the *PWORKS_DATA* directory. This file does not have to be present for **PWorks** to run correctly.

5.3.1 Clfile Transformation - ADJUST

Command Syntax: `pworks infile -ADjust:file`

A clfile [transformation file](#) can be used to adjust the input [clfile](#) to compensate for loading the part on the machine in a different position that it was programmed for and/or for defining new lengths for tools used in the program. '*file*' is the name of the file which contains the clfile transformation data. The default file extension is '*.maf*'.

The following command procedures can be used to define the transformation data. These clfile transformations will be applied after any transformations applied using the post-processor commands [TRANS](#) and [ORIGIN](#).

- a. Transformation of the clfile positions through a matrix defined in its canonical form.

[TRANS/i1,j1,k1,d1, i2,j2,k2,d2, i3,j3,k3,d3](#)

- b. Transformation of the clfile positions by defining three positions from the programmed system and their actual position on the machine. **PWorks** will generate a transformation matrix from these three positions.

POINT/xp1,yp1,zp1, xm1,ym1,zm1

POINT/xp2,yp2,zp2, xm2,ym2,zm2

POINT/xp3,yp3,zp3, xm3,ym3,zm3

'xp,yp,zp' = coordinates of the position in the programmed system.

'xm,ym,zm' = coordinates of the position digitized on the machine.

- c. New tool lengths for programmed tools. These lengths will replace the lengths associated with the programmed tool numbers.

TOOLNO/tn [,LENGTH], tl

'tn' = Number of tool to receive new length.

'tl' = New length of tool.

5.3.2 Input Clfile Format - CLFILE

Command Syntax: `pworks infile -Clfile:n [,wid]`

Default: `-CLFILE:0`

PWorks has the ability to process input binary *clfiles* from **NCL** and Catia (Up to Ver. 5), and to also process texture APT Source files. The APT Source file can be standard *GOTO/point* format or can be in the **PTED** ASCII clfile format.

The *CLFILE* option allows you to specify which format the input clfile is in. ‘n’ selects the clfile format and can have one of the following values.

n	Clfile Format
-	-----
0	Automaticlly detect file type
1	NCL binary clfile
2	APT Source textual file
3	Catia binary clfile (Up to Ver. 4)
4	APT Source textual file, UG Circle
5	Catia binary clfile (Ver. 5)
6	MasterCam NCI file

‘wid’ specifies the columnar width (72 to 132) of the input APT Source file when the *CLFILE* option is set to 2. A standard APT source file will only use the first 72 columns to store the APT Source record. Columns 73-80 are used for record numbers, comments, etc. Some CAD/CAM systems will use all 80 columns to store the APT Source record. The default for ‘wid’ is 72.

*** When running against APT Source files with Unigraphics circular interpolation records you must specify “-CLFILE:4,132”, since **PWorks** cannot determine this clfile type automatically using the -CLFILE:0 runtime option. ***

5.3.3 Error Messages - APTERR, WARNING, ERROR, FATAL

Command Syntax: `pworks infile - [NO]APterr
Warning[:n]
Error
Fatal`

Defaults: `-APTERR
-WARNING:9999
-ERROR:9999
-FATAL:1`

PWorks has four classes of error messages, Warnings (*WARNING*), Errors (*ERROR*), FataIs (*FATAL*) and Invalid APT Source Syntax (APTERR). Warning messages are output when the error does not adversely affect the output, for example, too many parameters on a post-processor command. Error messages are output when it is likely that the error will affect the Control Tape output, for example, an ignored post-processor command. Errors generated from the *Macro* processor will always generate an error class message. Fatal messages will only be output when an error occurred that will definitely cause the output to be incorrect, for example, if the tool axis cannot be satisfied with the active rotary axes.

WARNING, *ERROR*, and *FATAL* control and output of warning, error and fatal errors messages. 'n' specifies the maximum amount of the class type messages that can be generated before **PWorks** terminates. Specify a large number if you do not want to terminate **PWorks** prematurely because of too many messages.

APTERR enables the output of the Invalid APT Source Syntax Error Message encountered when processing an input APT Source file.

If the *NO* qualifier is specified, then the specified class of messages will not be output.

5.3.4 Unique Output Filenames - IDENT

Command Syntax: `pworks infile - [NO] Ident`

Default: **-NOIDENT**

One of the features of **PWorks** is that it allows you to process a single *clfile* against multiple *machine configuration files*, creating new output files for each of the *MDF* files processed. under normal circumstances this will create multiple output files with the same file name. This is totally unacceptable since the operation systems do not support multiple files with the same name in the same directory.

The *IDENT* option allows you to append the machine number of the *MDF* file to the name of the output files, creating specific and unique file names.

Example:

Input clfile	MDF File	Output Punch file
-----	-----	-----
test.cl	PWORKS_33.MDF	test_33.pul

If the *NO* qualifier is specified with the *IDENT* option, then the machine number will not be appended to the output file name(s).

This option does not apply if the post-processor are named with names instead of numbers.

5.3.5 Listing File - LISTING

Command Syntax: `pworks infile -[NO]Listing[:[file] [.ext]]`

Default: **-NOLISTING**

A [listing file](#) can optionally be created while processing the input clfile. The listing file will contain a header, the input sequence numbers from the part program, the clfile record numbers, a text representation of the clfile records, output generated using the *PRINT* command and any *Macro* error messages generated while processing the input clfile.

If the *NO* qualifier is specified with the *LISTING* option, then a listing file will not be created. ‘*file*’ and ‘*ext*’ are the name and the extension of the listing file to create, at least one of these parameters must be specified. If ‘.’ is specified, the default file name is the input file name with an extension of ‘*.lis*’. If only ‘*file*’ is specified, the listing file name will be the input file name with an extension of ‘*.lis*’. If only ‘*ext*’ is specified, the listing file name will be the input file name with an extension of ‘*.ext*’. If both are specified, the listing file name will be ‘*file.ext*’.

5.3.6 Overriding the MACHIN Card - MACHINE

Command Syntax: `pworks infile -Machine:n`

Default: **None**

PWorks processed the input clfile for the [MDF](#) files specified on the [MACHIN/PWORKS,#](#) or [MACHIN/name](#) card contained in the [clfile](#) by default. There may be times when you want to process the clfile for a machine number or name that is not on the *MACHIN* card or to process it for only one of the machine numbers on the *MACHIN* card.

The *MACHIN* option allows you to override the *MACHIN/PWORKS,#* or *MACHIN/name* command and to process the input clfile for a specific *MDF* file. ‘n’ is used to call up the *Machine Descriptor File* ‘*PWORKS_n.MDF*’, where ‘n’ can be a number or a name.

5.3.7 Overriding Machine Options - OPTION

Command Syntax: `pworks infile -Option:n1,v1 [...] ni,vi`

Default: **None**

Only numbered “.pdf” file will be allowed to specify with the runtime option. No named “.pdf” file will be allowed.

The *OPTION* option allows you to override the options contained on the [MACHIN/PWORKS,#](#) or [MACHIN/name](#) card. These options include input/output units and the *Print Descriptor File*

(PDF) number to use. The options are entered in the same format as they are on the *MACHIN* card, first the option number 'n' is entered followed by the new value 'v'.

Example:

```
pworks test/OPTION:3,7144
```

5.3.8 Page Length - PAGE_LEN

Command Syntax: `pworks infile -PAge_len:n`

The *PAGE_LEN* option specifies the page length of the [listing](#) and [print](#) files. 'n' is the number of lines to output between page breaks.

5.3.9 Print File - PRINT

Command Syntax: `pworks infile -[NO]PRInt[:[file][.ext]]`

Default: **-PRINT**

A print file can optionally be created while processing the input clfile. The [print file](#) contents depend on the [Print Descriptor File \(PDF\)](#) used and usually contains a formatted listing of the Control Tape and certain variables that can be used to visually verify the output generated by **PWorks**.

If the *NO* qualifier is specified with the *PRINT* option, then a print file will not be created. 'file' and 'ext' are the name and the extension of the print file to create, at least one of these parameters must be specified. If ':' is specified, the default file name is the input file name with an extension of '.pr1'. If only 'file' is specified, the print file name will be the input file name with an extension of '.pr1'. If only 'ext' is specified, the print file name will be the input file name with an extension of '.ext'. If both are specified, the print file name will be 'file.ext'.

5.3.10 Punch File - PUNCH

Command Syntax: `pworks infile -[NO]PUrch[:[file][.ext]]`

Default: **-PUNCH**

The *PUNCH* option specifies whether or not to create an output [punch file](#). It is normally desirable to create a punch file, since the punch file is what is used as input to the machine control. However, there are some cases where **PWorks** may be used to create a [listing file](#) for the purpose of verifying the contents of the [clfile](#) or to test *Macros*, in which case a punch file is not necessary.

If the *NO* qualifier is specified with the *PUNCH* option, then a punch file will not be created. '*file*' and '*ext*' are the name and the extension of the punch file to create, at least one of these parameters must be specified. If '.' is specified, the default file name is the input file name with the extension specified in the corresponding *Machine Descriptor File*. If nothing is specified in the *MDF* file, the extension will be default to '.pul'.

If only '*file*' is specified, the punch file name will be the input file name with the extension specified in the corresponding *Machine Descriptor File*. If nothing is specified in the *MDF* file, the extension will be default to '.pul'.

If only '*ext*' is specified, the punch file name will be the input file name with an extension of '*ext*'. If both are specified, the punch file name will be '*file.ext*'.

5.3.11 Simulation File - SIMULATE

Command Syntax: `pworks infile - [NO] Simulate [: [file] [.ext]]`

Default: **-NOSIMULATE**

The *SIMULATE* option specifies whether or not to create an output *simulation file*. The simulation file contains a neutral format of the *punch file* which can be used as input file for verification routines (**NCL** motion playback, **NCL/IPV**). All moves generated by **PWorks**, which the input *clfile* does not contain, will be in the simulation file. These moves include cycle simulation, rotary axis linearization, rapid moves, automatic tool retraction, helical interpolation, etc.

If the *NO* qualifier is specified with the *SIMULATE* option, then a simulation file will not be created. '*file*' and '*ext*' are the name and the extension of the simulation file to create, at least one of these parameters must be specified. If '.' is specified, the default file name is the input file name with an extension of '.sim'. If only '*file*' is specified, the simulation file name will be the input file name with an extension of '.sim'. If only '*ext*' is specified, the simulation file name will be the input file name with an extension of '.ext'. If both are specified, the simulation file name will be '*file.ext*'.

5.3.12 Quiet Batch Process - QUIET

Command Syntax: `pworks infile - [NO] Quiet`

Default: **-NOQUIET**

This option only applies if **PWorks** is started from the command line. It does not have any effect if the **PWorks** graphic interface is used. At the end of the **PWorks** process, a *Process Error Log Window* will pop up to show the end result of the **PWorks** process. Using the "-Quiet" option will stop the pop up of this *Process Error Log Window* and send the process result to a log file with the name "*pworks.log*" located in the current directory.

5.3.13 Consecutive Commas - [NO]FILL[:method]

Command Syntax: `pworks infile - [NO] FILL[: REMOVE]
var`

Default: **-NOFILL**

This option specifies a method for handling consecutive commas in an input clfile is of the type APT Source Textual. It does not has any effect on other input type format.

NOFILL specifies that consecutive commas in an APT source command will not be modified and will cause an error to be generated when the command is parsed.

FILL:REMOVE Removes the extra commas from the command. The parameter spot between the commas will not exist when the command is parsed.

FILL:var ‘var’ can either be a numeric value or a vocabulary word that will be placed in the APT source command between two consuetude commas and at the end of the command if the command is terminated with a comma.

Example”

When run with the “FILL:0.” runtime option, the following APT source input line ...

`CYCLE/DRILL,FEDTO,1.5,RAPTO,.1,DWELL,,IPM,`

will be converted to the following line.

`CYCLE/DRILL,FEDTO,1.5,RAPTO,.1,DWELL,0.,IPM,0.`

CHAPTER 6 *PostWorks* Utilities

6.1 Running on Windows

You may also run the **PostWorks** utilities using the Start menu or a Shortcut icon. In this case an interactive window will appear asking for the input file name.

6.2 Porting Machine Descriptor Files - **GenXfer**

When transferring *Machine Descriptor Files* between different hardware platforms the binary *MDF* file must first be converted to a text file, because of differences in the way various hardware platforms handle binary data.

GenXfer will convert a binary *MDF* file to a text *mda* file. It will also convert a text *mda* file to a binary *MDF* file. The sample post-processor configurations released with **PostWorks** contain text *Machine Descriptor Files* that must be run against the **GenXfer** utility prior to being used.

The following command will call up the **GenXfer** utility.

```
genxfer infile
```

'*infile*' is the name of the existing *MDF* file that you wish to convert, in the format '*PWOKRS_name.MDF*' or '*PWORKS_name.mda*'. The file type, either '*.MDF*' or '*.mda*' must be specified, as this is how **GenXfer** determines if the input file is binary or text. You may specify the *infile* name simply as *name.mda* or *name.MDF*, in which case **GenXfer** will automatically generate the correct file name for you.

Use the following procedure to transfer a *Machine Descriptor File* from one hardware platform to another. '*PWORKS_33.MDF*' is used as an example *Machine Descriptor File*.

Host Computer

```
genxfer 33.MDF
```

Converts the binary file '*PWORKS_33.MDF*' to a text file '*PWORKS_33.mda*'.

Copy the following files to the target computer.

```
PWORKS_33.mda  
pmacro_33.mac  
pworks_33.pdf
```

Target Computer

genxfer 33.mda

Converts the text file 'PWORKS_33.mda' to a binary file 'PWORKS_33.MDF'.

postcomp 33

Compiles the input file 'pmacro_33.mac' and creates the object file 'pmacro_33.OBJ' Because the object file is a binary file, you cannot transfer it between different hardware platforms.

6.3 Updating the Vocabulary Files - GenWord

The [vocabulary files](#), '*postword*' and '*postvocab*' are delivered with the **PostWorks** in both their binary and text formats. The binary formats '*.WRD*' are used by the **PostWorks** routines while processing. The text formats are supplied so that the user may add new vocabulary words that are supported by their CAM processor, but not yet supported by **PostWorks**.

GenWord will convert the modified text vocabulary files into a binary format which is acceptable to the **PostWorks** routines.

The following command will call up the **GenWord** utility.

genword infile

'*infile*' can be either '*postword*' or '*postvocab*'. The default file type is '*.txt*'.

Example:

```
genword postvocab
```

In the above example, **GenWord** will use the text file '*postvocab.txt*' as input and create the binary file '*postvocab.WRD*'.

6.4 Update the Message Files - GenPrompt

The message files, '*postmenu*', '*posterror*', and '*postdoc*' are delivered with **PostWorks** in both their binary and text formats. The binary formats '*.MSG*' are used by the **PostWorks** routines while processing. The text formats are supplied so that the user may modify existing prompts or error messages and create new messages.

GenPrompt will convert the modified text message files into a binary format which is acceptable to the **PostWorks** routines.

The following command will call up the **GenPrompt** utility.

```
genprompt infile
```

'infile' can be 'postmenu', 'posterror', or 'postdoc'. The default file type is '.txt'.

Example;

```
genprompt posterror
```

In the above example, **GenPrompt** will use the text file 'posterror.txt' as input and create the binary file 'posterror.MSG'.

6.5 Updating the Help File - **GenHelp**

The [Help file](#) utilized by **MPost**, 'posthelp', is delivered with **PostWorks** in both its binary and text formats. The binary format '.HLP' is used by **MPost** when displaying help information. The text format is supplied so that the user may modify the help displayed at any of the menus, forms or prompts.

GenHelp will convert the modified text Help file into a binary format so that the modifications will be accessible to the **MPost** utility.

The following command will convert the text Help file 'posthelp.txt' to its binary format 'posthelp.HLP'.

```
genhelp posthelp
```

APPENDIX A *PostWorks* Routine Summary

genhelp posthelp.txt

Used to convert the text Help file '*posthelp.txt*' to the binary Help file '*posthelp.HLP*' which can be read by the **MPost** routine.

genprompt postmenu.txt posterror.txt postdoc.txt

Converts the text Menu/Prompt file '*postmenu.txt*', the text error message file '*posterror.txt*', and the automatic documentation message file '*postdoc.txt*' to a binary format which can be used by the **PostWorks** routines. The output file will have a file type of '*.MSG*'.

genword postword.txt postvocab.txt

Converts the text vocabulary files to a binary format which can be used by the **PostWorks** routines. The output file will have a file type of '*.WRD*'.

genxfer PWORKS_#.MDF PWORKS_#.mda

Used to transfer *Machine Descriptor Files* between different hardware platforms. **GenXfer** will convert a binary '*.MDF*' file to a text '*.mda*' file when the file type on the runtime command is '*.MDF*'. It will convert a text '*.mda*' file to a binary '*.MDF*' file when the file type is '*.mda*'. The filename '*PWORKS_#*' can be shortened to just the name '*#*' and **GenXfer** will create the filename for you.

mpost [mchin[mchout]] [-Page_len:n] [-UNITS:INCH] MM SAME

The **MPost** routine is used to generate new and modify existing post-processor configurations. It uses a *Machine Descriptor File* for both the input and output file.

postcomp infile [-[NO]Document[:[file][.ext]]] [-[NO]Listing[:[file][.ext]]] [-[NO]Object[:[file][.ext]]] [-Page_len:n] [-[NO]Quiet]

The **PostComp** routine is used to compile a *Macro* definition file into a *Macro* object file which can be used by the *Macro* processor within the **PMacro** and **PWorks** routines.

```

pmacro infile [-Clfile:n[,wid]] [- [NO] Ident]
              [- [NO] Listing[: [file] [.ext]]]
              [-Machine:n] [- [NO] Object[: [file] [.ext]]]
              [-Page_len:n] [- [NO] Post] [- [NO] Quiet]

```

The **PMacro** routine is furnished as a separate pre-processor from the **PostWorks** post-processor '**PWorks**' and is used when you want to access the *Macro* capabilities in **PostWorks** without actually running **PWorks**. **PMacro** will create both an output clfile, which is merged with the post-processor *Macros*, and a listing file. The output clfile can be used as input to a custom post-processor, thereby adding post-processor *Macros* to all of your existing post-processors.

```

pworks infile [-Adjust:file] [-Clfile:n[,wid]] [- [NO] APterr]
              [- [NO] Error[:n]] [- [NO] Fatal[:n]]
              [- [NO] FILL[:REMOVE]] [- [NO] Ident]
              var
              [- [NO] Listing[:file] [.ext]] [-Machine:n]
              [-Option:n1,v1[...]ni,vi] [-Page_len:n]
              [- [NO] PRInt[: [file] [.ext]]] [- [NO] Quiet]
              [- [NO] PUnch[: [file] [.ext]]]
              [- [NO] Simulate[: [file] [.ext]]]
              [- [NO] Warning[:n]]

```

The **PWorks** routine is a user generated post-processor. It uses the *Machine Descriptor File* from **MPost**, along with the *Macro* object file from **PostComp** to generate a machine specific Control Tape file from a clfile generated by a CAM system.

APPENDIX B Directory Structure and Data Files

B.1 *PostWorks* Directory Structure

This section describes the logical directory structure used by ***PostWorks*** along with the files stored in each directory. Some of the locations of the ***PostWorks*** files may have to be changed to match the requirements of your CAM system.

For users running ***NCL*** it is recommended that the *PWORKS_BIN* directory be the same as the *NCL_POST* directory, so that ***PostWorks*** can be accessed directly out of ***NCL***.

PWORKS_MAIN

.\Bin	(PWORKS_BIN)
.\Data	(PWORKS_DATA)
.\Dsup	(PWORKS_DSUP)

PWORKS_BIN

genhelp.exe	genprompt.exe
genword.exe	genxfer.exe
mpost.exe	postcomp.exe
pmacro.exe	pworks.exe

PWORKS_DATA

pmacro_#.mac	pmacro_#.OBJ
PWORKS_#.mda	PWORKS_#.MDF
pworks_#.pdf	pworks_#.phf
pworks.ini	makepost.ini (optional)
postcomp.ini (optional)	pmacro.ini (optional)
postvocab.syn (optional)	

PWORKS_DSUP

postdoc.txt	postdoc.MSG
posterror.txt	posterror.MSG
posthelp.txt	posthelp.HLP
postmenu.txt	postmenu.MSG
postvocab.txt	posvocab.WRD
postword.txt	postword.WRD

B.2 Data Files

PostWorks uses different types of files while processing. The most apparent to the user are the input and output files. The input and output files used by the **PostWorks** routines are listed below.

Routine -----	Input Files -----	Output Files -----
MPost	makepost.ini PWORKS_#.MDF	PWORKS_#.MDF file.doc
PostComp	postcomp.ini pmacro_#.mac	pmacro_#.OBJ pmacro_#.lis pmacro_#.doc
PMacro	pmacro.ini pmacro_#.OBJ file.cl	file.cln file.lis
PWorks	pworks.ini PWORKS_#.MDF pworks_#.pdf pworks_#.phf pmacro_#.OBJ file.maf file.cl	file.pu1 file.pr1 file.lis file.sim

The following sections discuss the input and output files used by the various **PostWorks** routines.

B.2.1 User Vocabulary File - postvocab.syn

This file contains the user defined vocabulary word lists. The format of this file is similar to the system vocabulary file (postvocab.txt) in that it consists of lines containing vocabulary words, MAJOR/MINOR designation, and word values. For example

```
DEEPHOLE  MINOR  1301
```

Unlike the system vocabulary word file, the parameters in this file can be separated by tabs or spaces. This file does not need to be compiled by the **GenPrompt** utility.

A maximum of 720 vocabulary words (system plus user) can be defined. A vocabulary word can be defined with a maximum of 24 characters.

B.2.2 Initialization Files - routine.ini

Each of the **PostWorks** routines uses an initialization file when first invoked for the purpose of setting the most frequently used runtime options. The initialization files have the following format.

routine.ini

Where:

routine = Name of **PostWorks** routine that will access this initialization file.

The runtime options contained in the initialization file have the same format as if they were entered on the runtime command line. These options can also be overridden when invoking the routine by specifying the option on the runtime command or through the Windows OS graphic interface.

The initialization files should reside in either the user's directory or in the *PWORKS_DATA* directory. Each routine will first search in the user's directory for the initialization file and if it is found, will use the runtime options in the file as the defaults. If the initialization file is not found in the user's directory, then the routine will search for it in the **PostWorks** data directory. The initialization files do not have to be present for **PostWorks** to run correctly.

B.2.3 Machine Descriptor File - PWORKS_#.MDF

The Machine Descriptor File (MDF) is a binary file containing the settings for a machine specific post-processor configuration. The *MDF* file is created using the **MPost** routine.

Each post-processor configuration you generate will be stored in separate *MDF* files, allowing you to create a multitude of machine specific post-processors, each configured for a different machine. Once created, the *Machine Descriptor Files* are used by **PWorks** to define the configuration of the target machine.

PWorks uses the number(s) or the postprocessor name contained on the *MACHIN/PWORKS* card to determine which *MDF* files to use when processing the input clfile. For example, *MACHIN/PWORKS,33* references the *PWORKS_33.MDF Machine Descriptor File*, *MACHIN/PWORKS,NCCS* references the *PWORKS_NCCS.MDF Machine Descriptor File*. The *MDF* file referenced on the *MACHIN* card must be present for **PWorks** to run, but is not used by **PMacro**.

PWorks will first search for the *Machine Descriptor File* in the user's directory. If it is not found here, **PWorks** will search for it in the *PWORKS_DATA* directory.

Machine Descriptor Files are stored in a binary format, therefore they cannot be directly transferred between different hardware platforms. You must first convert the *MDF* file to a text file (.mda) using the **GenXfer** utility prior to transferring it to a hardware platform of a different type.

B.2.4 Documentation Files - file.doc

Both **MPost** and **PostComp** create documentation files which reflect the characteristics of the post-processor configuration. **MPost** automatically generates **documentation** of an *MDF* file, including a description of the machine, G and M code definitions, and a command summary. **PostComp** uses the **REMARK** statements in the input macro file to generate the documentation file. Combined, these two documents provide a concise user's guide of the post-processor configuration.

B.2.5 Macro Definition Files - pmacro_#.mac

A *Macro* definition file is a text file that contains any Macro definitions that will be used by the *Macro* processor resident within **PMacro** and **PWorks**. A standard text editor, such as NotePad, vi or EDIT can be used to create the *Macro* definition file. Before the macros contained in this file can be used by the *Macro* processor, the file must first be compiled into a *Macro* object file.

The **PostComp** routine is used to compile the *Macro* definition file into an object file. **PostComp** will only search for the *Macro* definition file in the user's directory.

The *Macro* processor in **PMacro** and **PWorks** uses the number(s) or the postprocessor name on the *MACHIN* card to locate the *Macro* object file with the name '*pmacro_#.OBJ*', therefore it is strongly recommended to name the input *Macro* definition file '*pmacro_#.mac*', where '#' is a unique number or a postprocessor name (**PWorks** only) identifying this file for the *Macro* processor.

B.2.6 Macro Object File - pmacro_#.obj

A *Macro* object file is a binary file that is created using the **PostComp** routine from an input *Macro* definition file. This is the actual file which is used by the *Macro* processor resident within **PMacro** and **PWorks**. It is a mandatory input file for **PMacro** and is optional for **PWorks**.

The *Macro* processor will first search for the *Macro* object file in the user's directory. If it is not found here, it will be searched for in the *PWORKS_DATA* directory.

The *Macro* object file cannot be transferred between different hardware platforms, the input *Macro* definition file must be transferred and recompiled using the **PostComp** routine on the target platform.

B.2.7 Print Descriptor File - pworks_#.pdf

The *Print Descriptor File* (PDF) is a user generated text file that contains information which defines the look of the **PWorks** print file. The PDF file consists of up to 10 record definitions which actually describe the format of the print file output.

Records can be defined for various types of output generated by **PWorks**, such as the page header and trailer, motion records, the *FINI* record, etc. See the **MPost Reference Manual** for a complete description of the *Print Descriptor File*.

The **PWorks** post-processor will first search for the *Print Descriptor File* in the user's directory. If it is not found here, it will be searched for in the *PWORKS_DATA* directory.

B.2.8 Punch Header File - pworks_#.phf

PWorks has the option of including user defined header and trailer sections within the output punch file. These sections are defined in the *Punch Header File*, comprised of simple text strings and formatted post-processor variables.

The header and trailer sections are generated after the entire part program is processed, therefore any post-processor variables referenced will contain the values assigned to them at the end of the program. The current date and time, punch file name, and tool length and times are examples of data which can be included in the punch file header.

The *Punch Header File* will first be searched for in the user's directory. If it is not found here, it will be searched for in the *PWORKS_DATA* directory.

The contents of the phf file can be updated by using a Post Macro file.

B.2.9 Machine Adjust File - file.maf

The *Machine Adjust File* is a **PWorks** runtime file which can contain a transformation matrix and/or new tool length information which will be dynamically applied to the input coordinates from the input clfile. It is primarily used when the part is orientated on the machine in a different position than it was programmed for or the actual set lengths of the tools used on the machine are different than the programmed lengths.

PWorks post-processor will first search for the *Machine Adjust File* in the user's directory. If it is not found here, it will be searched for in the *PWORKS_DATA* directory.

B.2.10 Clfiles - file.cl

Both **PMacro** and **PWorks** process a clfile, generated from a CAM processor, as input. The clfile contains cl points and tool axis vectors, post-processor commands, circular records, etc., which reflect the input part program.

PMacro will load a *Macro* object file prior to processing the input cl file and merge the input clfile with the *Macros* contained in the object file. A modified clfile will be output from v.

PWorks will load a *Machine Descriptor File* and a *Macro object file* prior to processing the input clfile. The *Macros* contained in the object file will be merged with the input clfile internally within **PWorks**. All output files will reflect the changes to the input clfile caused by post-processor *Macros*.

The input clfile will only be searched for in the user's directory and must be present for both **PMacro** and **PWorks**.

B.2.11 Listing Files - file.lis

The **PostComp**, **PMacro** and **PWorks** routines have the option of creating a listing file during processing. This file usually contains a listing of the input file along with any error messages generated during processing.

The **PostComp** listing file will contain a page header, the input line numbers from the *Macro* definition file, the PC address for each statement and any error messages generated during the compilation. A summary of the *Macros* defined and their sizes will be output at the end of the listing.

The **PMacro** and **PWorks** listing files will contain a page header, the input sequence numbers from the clfile, the cl record numbers of the *Macro* modified clfile, a text representation of the modified clfile, output generated using the *PRINT* command and any error messages generated during processing.

B.2.12 Print File - file.pr1

The print file generated by **PWorks** will usually contain a formatted listing of the punch file, including the Control Tape, axis positions, feed rates, etc. A summary of the axis motions and the tool usage is usually displayed at the end of the print file.

The actual format of the print file is contained in the *Print Descriptor File* and is completely user definable. See the **MPost Reference Manual** for a complete description of the [Print Descriptor File](#).

B.2.13 Punch File - file.pu1

The punch file generated by **PWorks** contains the actual Control Tape data which will be sent to the machine control. All of the effort put into programming the part using a CAM system, defining the post-processor configuration, defining post-processor *Macros* and designing the output formats, is ultimately for the purpose of creating the punch file, as the punch file contains the information that will actually cut the part at the machine.

Once the punch file is created, it can be sent to the machine control using a variety of methods; punched tape, magnetic tape, floppy disks, DNC, etc. The format of the punch file is simple text data, any conversion required, for example, parity bits, End-of-Tape marks, etc., will be done by a transfer program such as NCP, DNC, etc.

B.2.14 Simulation File - file.sim

The simulation file generated by **PWorks** contains a neutral format of the punch file. This file is used as input to tool path verification routines such as **NCL** Motion Playback and **NCL/IPV**. It contains all axis positions contained in the punch file and additional simulations, such as cycles, linearization, helical interpolation, *ROTABL* commands, etc. Using a simulation file as input to a verification routine instead of the clfile created by the CAM processor has a distinct advantage, as you actually verifying the information that is sent to the machine.

B.3 PostWorks Data Support Files

This section discusses the data support files located in the *PWORKS_DSUP* directory. These include the vocabulary word files (*postword.WRD* and *postcovab.WRD*), the menu prompt file (*postmenu.MSG*), the error message file (*posterror.MSG*), the automatic documentation file (*postdoc.MSG*) and the Help file (*posthelp.HLP*).

These files must be present for **PostWorks** to work properly.

B.3.1 The Vocabulary Files - postword.WRD, postvocab.WRD

'*postvocab.WRD*' contains the post-processor major/minor vocabulary words and their values. It also contains the words and values of commands that **PostComp** recognized. '*postword.WRD*' contains the text of runtime options that are recognized by the **PostWorks** routines and also the vocabulary words used in the **MPost** routine.

Also furnished with the vocabulary files are the text files which contain the text representation of the vocabulary words. These files are named '*postvocab.txt*' and '*postword.txt*' which can be edited by the user, enabling you to modify the existing words and add your own vocabulary words.

The vocabulary text files contain a separate line of information for each vocabulary word. These information lines have the following format.

word <tab> type <tab> value

Where:

word = Actual text of vocabulary word. This can be from 1 to 8 characters.

type = 'type' must be either the word *MAJOR* or *MINOR*. The **PostMacro** Reference Manual contains a description of major and minor words.

value = The numeric value assigned to the vocabulary word.

<tab> = A tab character must separate each parameter on the line. You cannot use spaces instead of tabs.

The vocabulary words do not have to be in any specific order within the file, neither alphabetic nor numeric.

Once you have made any changes to the vocabulary text files, you must convert them into a format that can be read by the **PostWorks** routines. The **GenWord** routine is used to convert '.txt' vocabulary text files to '.WRD' files.

B.3.2 The Menu/Prompt File - postmenu.MSG

'postmenu.MSG' contains both the text of runtime option prompts used by the **PostWorks** routines and the menu text used by **MPost**. It also contains the headings for the listing files. The runtime option prompts are displayed when the *PROMPT* option is used.

The file 'postmenu.txt' is furnished along with the menu/prompt file and contains the text representation of the runtime prompts and Menu text. You may edit this file to change the text on any of the prompts.

The menu/prompt text file contains a separate line of information for each prompt and heading. The runtime option prompts, listing headings, each having the following format.

label\$ <tab> text

Where:

label = Label of the prompt/heading. This label is used by the **PostWorks** routines to load the actual text of the prompt/heading.

text = Actual text of the prompt/heading.

<tab> = A tab character must separate the label from the text data.

You may change the text of the prompt/heading, but do not change the label, as this is used by the **PostWorks** routines to identify the text. On some of the lines in this file you will see the text string '%s', which is used to load runtime values into the prompt/heading text and should be left in approximately the same position in the line.

The **MPost** menu items and prompts have a different format than the standard prompts, but like the stand-alone prompts the menu items are contained on separate lines and are formatted as shown below.

level# <tab> text

Where:

level = Level number of menu/prompt item. This level number is used by **MPost** to load the actual text of the menu/prompt item. For example, '1.2'.

= The level number should be terminated with a '#' character when it is a menu item with a level of prompts below it. Do not place a # character on the level number when the level below the current menu item is another menu or if a prompt is being defined.

text = Actual text of the menu/prompt item.

<tab> = A tab character must separate the level number from the text data.

You may change the text of the menu/prompt item, but do not change the level number, as this is used by **MPost** to identify the text. You can add text lines to a prompt item by placing lines after the prompt line. These text lines should be in the following format.

<tab> text

Although these text lines are placed after the prompt line in the 'postmenu.txt' file, they will be displayed prior to the prompt in **MPost**. The extra text lines are normally used to identify the answers on multiple choice type prompts, but they can also be used to add a brief explanation to the prompt, which will be displayed each time the prompt is displayed.

The stand-alone prompts should be placed before the **MPost** menu/prompt items, but other than this rule they do not have to be in any specific order within the file. The menu/prompt items must be in numeric order, with underlying prompts following directly.

Example:

2 Define Machine Configuration.
2,1# Define type of machine.

- 2.1.1 Type of machine:
 - 1. Mill
 - 2. Lathe
- 2.1.2 Enter number of linear X-axes:
 - .
 - .
 - .
- 2.2# Linear axis set-up.
 - .
 - .
 - .
- 3 Define Control Tape format.

Once you have made any changes to the '*postmenu.txt*' file, you must convert it to a format that can be read by the **PostWorks** routines. The **GenPrompt** routine is used to convert the '*postmenu.txt*' file to the '*postmenu.MSG*' file.

B.3.3 The Error Message File - *posterror.MSG*

The '*posterror.MSG*' file contains the labels and text of all error messages generated by the **PostWorks** routines. See the [Error Messages Appendix](#) for a complete list of error messages and their labels.

The file '*posterror.txt*' is furnished along with the error message file and contains the text representation of the error messages generated by the **PostWorks** routines. You may edit this file to change the text on any of the messages or you may add your own messages.

The error message file contains a separate line of information for each error message. These information lines have the following format.

labels\$ <tab> **text**

Where:

label = Label of the error message. This label is used by the **PostWorks** routines to load the actual text of the message.

text = Actual text of the error message.

<tab> = A tab character must separate the label from the error message text.

You may change the text of an error message, but do not change the label, as this is used by the **PostWorks** routines to identify the error message. On some of the lines in this file you will see the text string '%s', which is used to load runtime values into the error message text and should be left in approximately the same position within the line.

You can also add your own custom messages, simply by adding a line containing a unique label and your message text. Custom messages can be referenced via their label using the *ERROR* command and the *ERRTXT* function within a post-processor Macro.

The error messages do not have to be in any specific order within the file.

Once you have made any changes to the '*posterror.txt*' file, you must convert it into a format that can be read by the **PostWorks** routines. The **GenPrompt** routine is used to convert the '*posterror.txt*' file to the '*posterror.MSG*' file.

B.3.4 The Automatic Documentation Message File - *postdoc.MSG*

The '*postdoc.MSG*' file contains the text used when creating an automatic documentation file. See the '*Automatic Documentation*' Appendix in the **MPost** manual for an example documentation file created by **MPost**.

The file '*postdoc.txt*' is furnished along with the automatic documentation file and contains the text representation of messages output to the documentation file. You may edit this file to change the text on any of the messages.

The automatic documentation file contains a separate line of information for each message. These information lines have the following format.

```
label$ <tab> text
```

Where:

label = Label of the documentation text. This label is used by the **MPost** routine to load the actual text of the message.

text = Actual text of the documentation message.

<tab> = A tab character must separate the label from the documentation message text.

You may change the text of a documentation message, but do not change the label, as this is used by **MPost** to identify the documentation message. On some of the lines in this file you will see the text string '%s', which is used to load runtime values into the message text and should be left in approximately the same position within the line.

The documentation messages do not have to be in any specific order within the file.

Once you have made any changes to the '*postdoc.txt*' file, you must convert it into a format that can be read by the **PostWorks** routines. The **GenPrompt** routine is used to convert the '*postdoc.txt*' file to the '*postdoc.MSG*' file.

B.3.5 The Help File - *posthelp.HLP*

The '*posthelp.HLP*' file contains the Help text that is displayed whenever Help is requested by the user in the **MPost** routine. Help is available for all levels of menus, prompts and forms within **MPost**.

The file '*posthelp.txt*' is furnished along with the Help file and contains the text representation of the Help text displayed when using the **MPost** routine. You may edit this file to change the text for any of the Help messages.

The Help file has the following format.

```
~index
text
.
.
.
~index
```

Where:

- ~index = level number of the prompt that will display the Help text following the index. For example, '~3.2.1'. When defining a Help index the '~' character must be the first character of the line followed by a valid level number.
- text = Multiple lines of Help text that will be displayed whenever Help is requested at the level specified by 'index'. All lines following an '~index' line will be part of the text of for that index until another '~index' line is found.

You may change the text of the Help file, but do not change the '~index' lines, as these are used by **MPost** to locate the proper Help text.

Once you have made any changes to the '*posthelp.txt*' file, you must convert it into a format that can be read by the **MPost** routine. The **GenHelp** routine is used to convert the '*posthelp.txt*' file to the '*posthelp.HLP*' file.

APPENDIX C Vocabulary Words

C.1 *PostWorks* Major Words

This section contains a list of the major words recognized by ***PostWorks***. A brief description of the major words which constitute valid post-processor commands within ***PostWorks*** is given. The major words that are not yet supported by the ***PWorks*** post-processor can still be defined as a Macro, thereby allowing one to define customized post-processor commands for these words. Please note that not all CAM systems support all of the Major words listed here.

<u>Major</u>	<u>Value</u>	<u>Description</u>
AIR	1011	
ALIGN	1076	Controls blade alignment for Ultrasonic Cutters, or the look ahead feature of the AC style head configuration.
ARCSLP	1029	Controls the output of circular interpolation records.
AUXFUN	1022	Used to output M-codes.
BREAK	16	Controls the output of tape breaks.
CHECK	1023	Sets machine axis limits.
CHUCK	1073	
CLAMP	1060	Clamp/unclamp rotary axes.
CLEARP	1004	
CLRSRF	1057	Define the clearance plane.
COMBIN	1071	
COOLNT	1030	Controls the coolant output.
COPY	1040	
COUPLE	1049	Controls the output of helical interpolation.
CUTCOM	1007	Controls the output of cutter compensation.
CYCLE	1054	Performs automatic/manual cycle sequences.
DEBUGG	1032	
DELAY	1010	Output machine dwell blocks.
DISK	1097	
DISPLY	1021	Controls the output of operator's messages.
DRAFT	1059	
END	1	Logical end of part program.
FEDRAT	1009	Controls the cutting feed rate.
FINI		Physical end of part program.
GOHOME	17	Move the axes to their home position.
HEAD	1002	Controls the varying Z-motion feature.
INDEXF	1039	
INSERT	1046	Inserts text to the Control Tape.
INTCOD	1020	
KNIFE	1098	
LEADER	1013	Inserts leader to the Control Tape.
LETTER	1043	

<u>Major</u>	<u>Value</u>	<u>Description</u>
LIMIT	1078	
LINTOL	1067	Controls linearization tolerances.
LOAD	1075	
LOADTL	1055	Output a tool change sequence.
MACHIN	1015	Calls up the <i>PMacro</i> and <i>PWorks</i> routines.
MAXDPM	1062	Controls the maximum degrees per minute for the rotary axes.
MCHTOL	1016	Specifies circular interpolation tolerances.
MDEND	1052	
MDWRIT	1051	
MODCUT	1092	
MODE	1003	Controls programming modes and axis selections.
MOVETO	1190	
MSYS	1077	
MULTAX	1105	Controls the output of tool axis vectors.
NIBBLE	1082	
OPSKIP	1012	Controls the output of optional skip blocks.
OPSTOP	3	Optional stop.
ORIGIN	1027	Defines the part origin.
PARTNO	1045	Contains the part program description.
PEN	128	
PENDWN	130	
PENUP	129	
PITCH	1050	
PIVOTZ	1017	
PLABEL	1061	
PLOT	1041	
PLUNGE	1001	Repositions the tool after a retract.
POD	1088	Controls Vacuum Pods on Ultrasonic Cutting machines.
POSITN	1072	Positions the output axes.
POSMAP	1034	
POSTN	1024	Outputs a preset axes block or part program number.
PPLOT	1014	
PPRINT	1044	Outputs messages to the print file.
PPTOL	1068	Controls the axes output tolerances.
PPUNCH	1086	
PREFUN	1048	Used to output G-codes.
PROBE	1081	
RAPID	5	Makes the next move at the rapid traverse rate.
REPEAT	1083	
RESET	15	
RETRCT	7	Controls various retract and plunge sequences.
REVERS	1008	
REWIND	1006	Outputs rewind code.
ROTABL	1026	Positions the rotary axes.

<u>Major</u>	<u>Value</u>	<u>Description</u>
ROTATE	1066	
ROTHED	1035	Same as ROTABL.
SAFETY	1028	
SELCTL	1056	Preselects the next tool.
SELECT	1074	
SEQNO	1019	Controls the output of sequence numbers.
SET	1087	
SLOWDN	1063	Controls the output of slowdown blocks.
SMOOTH	1085	Controls spline interpolation for Ultrasonic Cutters.
SPINDL	1031	Controls the spindle speed.
STOP	2	Program stop.
SWITCH	6	
THREAD	1036	Outputs lathe threading sequences.
TITLES	1095	
TLDATA	1079	
TMARK	1005	Outputs an alignment block.
TOOLNO	1025	Controls the output of tool length compensation.
TRACUT	1038	
TRANS	1037	Modifies the input cl points and output axes.
TURRET	1033	Outputs a tool change sequence for lathes.
UAXIS	1018	
UNITS	841	
UNLOAD	10	Unloads the tool currently in the spindle.
USONIC	1093	
VACUUM	1094	
XOFSET	1084	

C.2 *PostWorks* Minor Words

This section contains a list of minor words recognized by the ***PMacro*** and ***PWorks*** routines.

<u>Minor</u>	<u>Value</u>	<u>Minor</u>	<u>Value</u>	<u>Minor</u>	<u>Value</u>	<u>Minor</u>	<u>Value</u>
AAXIS	140	ABSOL	180	ADJUST	159	ALL	51
ARC	182	AT	189	ATANGL	1	AUTO	88
AVOID	187	AXIS	132	BAR	207	BAXIS	141
BLADE	191	BORE	82	BORE5	209	BORE6	210
BORE7	211	BORE8	212	BORE9	213	BOTH	83
BRKCHP	288	CAM	237	CAXIS	142	CBORE	245
CCLW	59	CENTER	2	CIRCUL	75	CLW	60
CM	302	COARSE	195	COLLET	139	CONST	64
CROSS	3	CSINK	256	CTRLIN	126	CUTANG	160
DASH	124	DECR	65	DEEP	153	DEEPBK	216

<u>Minor</u>	<u>Value</u>	<u>Minor</u>	<u>Value</u>	<u>Minor</u>	<u>Value</u>	<u>Minor</u>	<u>Value</u>
DEEPCL	208	DELTA	188	DEPTH	510	DIAMTR	20
DITTO	127	DOTTED	125	DOWN	113	DRAG	278
DRILL	163	DS	729	DWELL	279	ENDPT	664
FACE	81	FEDTO	281	FEEDX	217	FEEDY	218
FEEDZ	219	FEET	17	FILLET	402	FINE	193
FLOOD	89	FRONT	148	GOCCLW	4050	GOCLW	4051
GRID	67	HIGH	62	HOLDER	157	IN	48
INCHES	303	INCR	66	INTENS	134	INTOF	727
INVERS	6	IPM	73	IPR	74	LARGE	7
LAST	52	LATHE	700	LEFT	8	LENGTH	9
LINCIR	95	LINEAR	76	LOCK	114	LOW	63
MANUAL	158	MAXIPM	96	MAXRPM	79	MEDIUM	61
MAIN	513	MILL	151	MILLFD	201	MILLRP	200
MINUS	10	MIRROR	56	MIST	90	MM	301
MMPM	315	MMPR	316	MODIFY	55	NEGX	11
NEGY	12	NEGZ	13	NEUTRL	166	NEXT	162
NOBACK	194	NOMORE	53	NOPOST	812	NORMAL	111
NOW	161	NOX	14	NOY	15	NOZ	16
OFF	72	OFFSET	705	OFSETL	275	OMIT	176
ON	71	OPTION	144	ORIENT	246	OUT	49
PALLET	239	PARAB	77	PARMS	934	PART	260
PAST	70	PATH	234	PERPTO	18	PLUS	19
POSX	20	POSY	21	POSZ	22	PS	728
QUILL	287	RADIUS	23	RAM	500	RAMP	854
RANDOM	183	RANGE	145	RAPTO	280	REAM	262
REAR	149	RETAIN	184	RTRCTO	295	REV	97
RIGHT	24	ROTREF	68	ROUGH	320	ROUTE	1205
RPM	78	SADDLE	150	SAME	54	SCALE	25
SFM	115	SHANK	192	SHIFT	249	SHORT	851
SIDE	94	SIZE	196	SKIP	176	SLICE	803
SLOPE	47	SMALL	26	SMM	505	SOLID	123
SPEED	272	SPLINE	628	START	57	STEP	92
SYMBOL	235	TABLE	177	TANTO	27	TAP	168
THETAR	107	THRU	152	TILT	247	TIMES	2
TO	69	TOOL	617	TORCH	172	TPI	143
TRANSL	29	TRAV	154	TRFORM	110	TURN	8
TYPE	98	UNIT	30	UP	112	XAXIS	84
XCOORD	116	XLARGE	31	XSMALL	32	XYPLAN	33
XYROT	34	XYZ	108	XYZPLN	45	YAXIS	85
YCOORD	117	YLARGE	35	YSMALL	36	YZPLAN	37
YZROT	38	ZAXIS	86	ZCOORD	118	ZIGZAG	170
ZLARGE	39	ZSMALL	40	ZXPLAN	41	ZXROT	42

APPENDIX D Error Messages

D.1 Introduction

This chapter explains the error message format and lists the messages and their corresponding labels. The error message labels are used by the *ERROR* command and *ERROR Macro*, the *PSTERR Macro* and the *ERRTXT* function.

D.2 *PostComp* Error Message Format

Error messages generated by the *PostComp* routine are displayed both to the screen and listing file. They have the following format:

```
#ERROR - 'a' at line 'b' in 'c' macro.  
message - test  
statement - text
```

All error messages have the *#ERROR* designator. A running count of the errors generated is kept by *PostComp*. 'a' is the number of errors that have occurred so far. 'b' is the line number of the statement that caused the error. The line number is contained in the listing file and is the same line number that is used by standard text editors. 'c' is the *Macro* which contains the statement that caused the error.

'*message - text*' is the text of the error message. See the [Error Messages section](#) for a description of the error message texts. '*statement - text*' is the text of the statement that generated the error.

The error message format below is generated by the 2nd pass compilation phase of *PostComp*, during which label references are resolved. All components of the message are the same, except for 'b', which references the *PC* of the instruction that generated the error message, instead of the line number of the statement.

```
#ERROR - 'a' at PC 'b' in 'c' macro.  
message - text  
statement - text
```

D.3 *PostMacro* Error Message Format

Error messages generated by the *Macro* processor in the *PMacro* and *PWorks* routines have the following format:

```
#ERROR - 'a' at (ISN='b', Clrec='c', PC='d') in 'e' macro  
message - text  
Clife: output - clfile
```

All error messages have the `#ERROR` designator. ‘a’ is the number of the errors, including this error, that have occurred so far. ‘b’ is the input sequence number, from the part program file, of the statement that called the current *Macro*.

‘c’ is the number of the next cl record to be output to the clfile. The statement that generated the error will be somewhere between the last cl record output and this cl record. Therefore, the clfile record ‘c’ will be affected by the error. The cl record number is contained in the listing file generated by **PMacro** or **PWorks**.

‘d’ is the PC of the instruction that caused the error. The beginning PC for each statement is contained in the listing file generated by **PostComp**. Since a single statement can generate more than one instruction, this PC may not coincide exactly with the PC of the statement in the listing file. In this case, the statement that caused the error will be the statement with the PC that is prior to the PC of the instruction that caused the error.

‘e’ is the name of the *Macro* which contains the statement that caused the error. The PC of the instruction (d) and the name of the *Macro* (e) will not be output if the error was generated from the input clfile and not from a *Macro* statement.

‘message -text’ is the text of the error message. See the Error Messages section for a description of the error message texts. ‘output - clfile’ is the name of the output clfile being generated when the error occurred.

D.4 **PWorks** Error Message Format

Error messages generated by the **PWorks** post-processor have the following format:

```
#'type' - a' at (ISN='b', Clrec='c').  
message1 - text  
message2 - text  
statement - text
```

PWorks has three classes of error messages, *INPUT ERRORS*, *WARNINGS*, *ERRORS* and *FATALS*. Warnings have the least impact on the output created, Errors have more impact and FATALs have the most. ‘type’ specifies the class of error that occurred. “*INPUT ERROR*” is used for errors found when loading an APT Source file.

‘a’ is the number of errors, including this error, that have occurred so far. ‘b’ is the input sequence number, from the part program file, of the statement that caused the error. ‘c’ is the number of the next cl record to be output to the clfile.

‘message - text’ is the text of the error message. See the Error Messages section for a description of the error message texts. Either one or two lines of ‘message - text’ can be output.

'*statement - text*' is output only when a post-processor command caused the error. It contains the post-processor command in question and has the parameter (minor word/value) that caused the error enclosed in quotes.

Example:

```
#ERROR - 1 at (ISN=126, Clrec=113)
Invalid minor word. This command is ignored.
FEDRAT [ /AUTO ]
```

In the above example, the minor word '*AUTO*'; is not supported by the *FEDRAT* statement.

D.5 Error Messages

This section describes the error messages and their corresponding labels. The label is specified first, delimited by a colon (:) and followed by the text of the error message. An explanation of the error message follows each message.

AXSTKOV: Output axes stack overflow. Internal bug.

The cl point look ahead filled up its internal stack prior to accomplishing its task. This is an internal bug and should be reported.

AXSTKUN: Output axes stack underflow. Internal bug.

The cl point look ahead feature tried to extract an entry on the look ahead stack which was not yet stored. This is an internal bug and should be reported.

CIRANG: Rotary movement in circular interpolation. Reverted to linear motion.

Rotary axes movement was programmed in a circular interpolation record. The circular motion will be output as a series of linear moves.

CIRMAX: Circle radius over maximum. Output as linear motion.

The radius of a circular interpolation record is greater than the maximum allowed. The calculated radius and maximum radius allowed will be output in the error message. The circular motion will be output as a series of linear moves.

CIRMIN: Circle radius under minimum. Output as linear motion.

The radius of a circular interpolation record is less than the minimum allowed. The calculated radius and minimum radius allowed will be output in the error message. The circular motion will be output as a series of linear moves.

CIRPLN: Could not calculate circular plane. Output as linear motion.

The first point of a circular interpolation record contains movement in all three of the linear axes. The circular motion will be output as a series of linear moves.

CIRPOL1: Circular interpolation is not supported in polar coordinate system.

Circular interpolation is not supported at the machine while in polar coordinate programming mode. The circular motion will be output as a series of linear moves. This error message is associated with mill/turn type machine.

CIRPOL2: Output as linear motion.

CIRRAD: Radius = 'r' Limit = 'm'

This error message is output together with the "Circle radius over maximum", "Circle radius under minimum" or "Circular move out of radius tolerance" messages. "r" denotes the actual circular radius. "m" denotes the radius limit.

CIRTOLR: Circular move out of radius tolerance. Reverted to linear motion.

The calculated radius of the current point in a circular interpolation record, as compared to the radius calculated at the beginning of the circle, exceeds the tolerance specified by the *MCHTOL* command. The first portion of the circle, up to the point that caused the error, will be output as circular motion. The remainder of the circle will be output as series of linear moves.

CIRVEC: Tool axis vector is not parallel to circle axis.

The current tool axis vector is not parallel to the axis of the circular drive surface of an APT source *GOFWD* command. This command is ignored and will not generate any motion. Refer to the APT Source File Formats appendix for a description of valid APT Source records.

CIR3AX: 3-axis movement in circular interpolation. Reverted to linear motion.

A 3-axis move was encountered in a circular interpolation record. The first portion of the circle, up to the point that caused the error, will be output as circular motion. The remainder of the circle will be output as a series of linear moves.

CLSINV: CLRSRF/START or STOP issued without any plane defined.

You must provide limit plane definitions to use prior to issuing a *CLRSRF/START* or *CLRSRF/STOP* command. You may define from one to ten limit planes.

CLSEXCC: Number of planes exceeded maximum (10) allowed for CLRSRF.

You may define a maximum of ten limit planes for the *CLRSRF/START* style command.

CLSMOD: Z-axis direction modifier expected (NEGZ,POSZ).

You must specify a reference direction to the limit plane which defines the side of the plane to keep the output axis locations. Valid direction modifiers are *NEGX*, *NEGY*, *NEGZ*, *POSX*, *POSY*, and *POSZ*.

COMAEXP: Comma expected.

A comma was expected in a statement, but was not found. For example, a comma must separate each of the parameters on a post-processor command.

COMPLXEQ: Equation is too complex.

An equation contained more than 50 operators and/or levels of parenthesis. Break the equation into smaller equations.

CONFPRM: Conflicting parameters. This command is ignored.

A post-processor command contains two or more minor words that are in conflict with one another. For example, '*FEDRAT/IPM,10,IPR,.05*'. This command will not be processed.

CUTANG: Angular change is too great while CUTCOM is active. Did not alter output.

The directional change of a move while cutter compensation is active is greater than the angular value specified in the ***MPost*** routine. This message usually means that you changed the cutting direction without changing the cutter compensation direction. The calculated angular change and the maximum change allowed will be output with this message.

CUTANG1: Angular change = 'ang1' degrees. Maximum allowed = 'ang2' degrees.

This error message is output together with the above “Angular change is too great whiel CUTCOM is active” error message. “ang1” denotes the calculated angular change. “ang2” denotes the maximum allowable angular change.

CUTENT: Initial CUTCOM move is circular motion and may not be supported by the machine.

The first motion following a *CUTCOM/ON* statement is a circular interpolation record. Most machine controls require a linear move as the first motion of cutter compensation.

CUTMOV: Move is in non-CUTCOM axis only, which may not be supported by the machine.

A move during cutter compensation has motion only in the non-active cutter compensation axis. Some machines get confused when cutter compensation is active and a move does not contain motion in either of the active cutcom axes. This is an informational message only. The output move will not be modified.

CUTPLN: CUTCOM and circular planes do not match. Reverted to linear motion.

The programmed cutter compensation plane (*XYPLAN*, *YZPLAN*, *ZXPLAN*) does not match the calculated circular interpolation plane. The circular motion will be output as a series of linear moves.

CUTVEC: CUTCOM vector component is out of range. Set to maximum value allowed.

The calculated cutter compensation vector component exceeds the maximum allowed. It will be set to the maximum value as specified in ***MPost*** which may affect the cutter compensation calculations at the machine. The calculated vector component and the maximum value allowed will be displayed with this message.

CUTVEC1: Value = 'a' Maximum = 'b'

This error message is output together with the above "CUTCOM vector component is out of range" error message. "a" denotes the calculated value and "b" denotes the allowable maximum value.

CYCACT: This command is not valid when CYCLEs are in effect. Ignored command.

A command, such as *CUTCOM*, which is not allowed while canned cycles are in effect was programmed during a canned cycle sequence.

CYCCIR: Canned cycle currently in effect. Output circular record as linear motion.

A circular interpolation record was encountered during a canned cycle sequence. The circular motion will be output as a series of linear moves.

CYCCUT: CYCLE & CUTCOM not allowed at the same time. Ignored command.

Canned cycles and cutter compensation cannot be both active at the same time.

CYCNACT: CYCLE is not currently active. Ignored command.

A command, such as *CYCLE/AVOID*, was programmed outside of a canned cycle. The command that generated the error has no effect unless a canned cycle is active.

DEFAULT: DEFAULT

DEMPART: Part may be damaged.

This error message is output together with the “[Rotary axes are taking the longest route](#)” error message.

DEPMAX: ‘a1’ axis exceeded maximum departure allowed. Did not alter output.

The specified axis exceeded the maximum movement as set by the user. No attempt to modify this move was made by **PWorks**. Normally, the post-processor will break a move into multiple moves when the maximum axis departure is exceeded, except when the user requests that this error message be output instead. This is accomplished by entering a negative value for the maximum departure allowed per individual axis.

DEPLMT: Delta = ‘a’ Limit = ‘b’

This error message is output together with the above “axis exceeded maximum departure allowed” error message. “a” denotes the delta value in this move and “b” denotes the allowable maximum departure value.

DIVZERO: Attempted divide by zero.

An attempt was made to divide a number by zero. The result will be zero.

DLYOFF: Delay per revolution programmed with spindle off. No delay output.

A machine dwell specifying the number of spindle revolutions to dwell was programmed with the spindle off. This command will be ignored.

DONOTDEF: A DO loop is not being defined.

EXCLAXS: Mutually exclusive axes ‘a1’ and ‘a2’ move at the same time.

Two mutually exclusive axes were programmed to move in the same motion block. This block will be broken up into two or more moves, separating the 'a1' and 'a2' mutually exclusive axes movement.

FMTEXP: Format specifier expected.

A format specifier (#f) was expected in the [CTOR](#) or [RTOC](#) function's argument list, but was not present.

FMTRNG: Format specifier out of range.

FPROFF: Feed per revolution programmed with spindle off. Changed to FPM mode.

An attempt was made to program the feed rate using the Feed Per Revolution mode while the spindle was turned off. The feed rate mode will revert to Feed Per Minute, using the last programmed FPM feed rate.

IFDODEEP: IF/DO structure nested too deeply.

IF-THEN-ELSE structures and *DO* loops may only be nested 10 deep.

IFDONEST: Illegal nesting of IF/DO structures.

An *IF-THEN-ELSE* structure or *DO* loop crossed the boundary of an *IF* block or *DO* loop. *IF* structures and *DO* loops must be wholly contained within an *IF* block or *DO* loop.

IFNOTDEF: An IF structure is not being defined.

INCOVL: Too many nested include files. INCLUDE command ignored.

You may have a maximum of 10 levels of include files in a *Macro* file. The maximum was exceeded with this *INCLUDE* command and the file was not included. Break out some of the *INCLUDE* commands from the lower level include files and moves them to a higher level file.

INFZON: Tool is currently in interference zone #'n'. Did not alter output.

The tool is currently positioned in an interference zone where the spindle and fixture will possibly collide.

INPRANG: Input value out of range.

A number was entered that was too low or high for what was expected. For example, the number or arguments on a *Macro* definition statement must be between 0 and 50.

INPRNG: Input value out of range. This command is ignored.

A number was programmed in a post-processor command that is too low or high for what is expected. For example, the *MODE/XAXIS,n* command will only accept 1 and 2 as valid input for 'n'.

INVADJF: Invalid syntax for Machine Adjust File record.

An error occurred while trying to parse a record from the *Machine Adjust File*. The statement which caused the error will be displayed along with the error message. Refer to the ADJUST runtime option in the “Using **PWorks**” chapter for a description of valid *Machine Adjust File* commands.

INVAPTSC: Invalid syntax for APT Source record.

An error occurred while trying to parse a record from an input APT Source file. The APT Source statement which caused the error will be displayed along with this error message. Refer to the [APT Source File Formats](#) appendix for a description of valid APT Source records.

INVARG: Invalid argument to function.

An invalid argument was passed to a numeric or text function. For example, a negative number was passed to the [SQRT](#) function.

INVCIRC: Circular interpolation records are ignored.

INVCKPL: Check plane is invalid. Could not calculate end position.

The check surface plane of an APT Source circular interpolation record (*TLON,GOFWD/circle,on,plane*) is not a valid plane or does not intersect the circle. The final location could not be calculated and the entire circular interpolation record is ignored. Refer to the [APT Source File Formats](#) appendix for a description of valid APT Source records.

INVFRMI: Invalid input for record 'n': "text".

A form field was incorrectly set in **MPost**, such as in the Register Format form. 'n' is the record within the form that the invalid input is in (A1,G0,etc.). 'text' is the incorrect field that caused the error. The form will be redisplayed and the cursor will be located at the invalid record.

INVGOFWD: GOFWD statement programmed without circle or direction statement.

A *GOFWD* command has been programmed without a circle or cylinder as the drive surface or without a preceding direction statement (*INDIRV, INDIRP*). *GOFWD* commands are valid for circular interpolation records only. This command is ignored and will not generate any motion. Refer to the [APT Source File Formats](#) appendix for a description of valid APT Source records.

INVGOTO: Invalid number of values in GOTO command.

The *GOTO* command requires a multiple of 3 values when *MULTAX* is turned off and a multiple of 6 values when *MULTAX* is turned on.

INVINSTR: **Invalid runtime instruction.**

The *Macro* processor encountered an instruction that it does not recognize. This is an internal error and should only occur when a newer version of **PostComp** is run with an older version of **PMacro** or **PWorks**.

INVLAB: **Invalid label descriptor.**

Labels must be made up of only alphanumeric characters and the first character must be a letter.

INVLEV: **Invalid level number.**

You entered an invalid level number at the 'Selection' prompt in v. Level numbers must be one or more integer numbers separated by periods. For example, '1.2'.

INVLIC: **You are not authorized to run this Machine Descriptor File.**

You attempted to reference an *MDF* file that requires a module or **PostWorks** that you do not have. For example, if the *MDF* file is configured for a rotary table, but you are only licensed to run the *MILL3* module.

INVLICO: **You are not authorized to run this software product.**

INVMAJOR: **Invalid major word. This command is ignored.**

An unrecognized post-processor command was programmed. If this command was added as a post-processor *Macro*, then it is possible that either the *Macro definition file* was not compiled or that the *Macro object file* is in the wrong directory.

INVMINOR: **Invalid minor word. This command is ignored.**

A minor word parameter was programmed on a recognized post-processor command that is not part of the valid syntax for this command.

INVMX: **Invalid matrix definition. This command is ignored.**

INVNUMPR: **Invalid number of parameters in command.**

An incorrect number of values were entered in the *TRANS*, *POINT*, or *TOOLNO* command in the *Machine Adjust File*. The statement which caused the error will be displayed along with the error message. Refer to the *ADJUST* runtime option in the “Using **PWorks**” chapter for a description of valid *Machine Adjust File* Commands.

INVOPT: Invalid runtime option: "'option'".

An unrecognized runtime option was specified. The error message will contain the option in question.

INVPARAM: Invalid parameter in command definition.

An invalid parameter was specified in a command that has fixed parameters. For example, an unrecognized floating point format in a format expression would generate this error.

INVPSIS: Unrecognized PSIS statement.

The part surface plane of an APT Source circular interpolation record (*PSIS/(PLANE/i,j,k,d)*) is not a valid plane. This statement will be ignored and **PWorks** will use the current tool position to calculate the circular interpolation part surface plane.

INVPSYN: Invalid command syntax. This command is ignored.

The syntax of a programmed post-processor command is not acceptable. This message is usually displayed when the command does not contain enough parameters.

INVPSYNW: Invalid command syntax. The rest of the command is ignored.

The syntax of a programmed post-processor command contains too many parameters. The extra parameters will be ignored.

INVRESP: Invalid response.

An unacceptable response was entered at a runtime option prompt.

INVRSP: Invalid response. Type in a '?' for help.

An invalid response to a prompt in the **MPost** routine was entered. Entering a question mark, with no other characters on the line, will display informative help text, which explains the prompt and the valid responses.

INVSYN: Invalid syntax.

This is a general error message for a statement that was not entered in the proper format. This error message will be output when none of the other error messages apply.

IVMINORW: Invalid minor word. This word is ignored.

A minor word parameter was programmed on a recognized post-processor command that is not part of the valid syntax for this command. Only this word will be ignored.

LABEXP: Label expected.

A label was expected, but not found. For example, on the *JUMPTO* or *DO* commands.

LABMAX: Maximum 24 characters per label/variable.

A variable name or label may contain from 1 to 24 characters.

LEFTVEC: Unfulfilled portion of vector = ‘i, j k’.

This means the tool axis vector “i, j, k” cannot be fulfilled with the current machine configuration.

LMTLWR: ’a’ axis exceeded negative limit. Did not alter output.

The specified axis has exceeded the negative machine axis limit. The programmed position and minimum travel limit for this axis will be displayed with this message.

LMTUPR: ’a’ axis exceeded positive limit. Did not alter output.

The specified axis has exceeded the positive machine axis limit. The programmed position and maximum travel limit for this axis will be displayed with this message.

LOCK2R: Both activated rotary axes move about the same vector. Locked 2nd axis.

LOCKALLR: Activated rotary axes move about the spindle vector.

LONGRTE: Rotary axes are taking the longest route.

When two rotary axes are active there is a possibility that a tool axis vector can be satisfied using two or more different solutions. Normally, **PostWorks** will choose the solution requiring the least amount of movement. The move that generated this error, though, caused **PostWorks** to select a solution that does not make the least amount of rotary movement, usually because the machine limits would have been violated using the shortest route. This move may be adjusted by **PostWorks** depending on the options set in **MPost**.

MACDEF: A Macro is currently being defined.

A *Macro* definition statement was encountered inside a *Macro* definition. Terminate the previous *Macro* with a *TERMAC* statement.

MACHMAX: Too many MACHIN cards. Remainder ignored.

PMacro will accept up to 10 machine specific *MACHIN* statements. The remainder will be ignored.

MACHSYN: MACHIN card syntax error. The clfile was not processed.

The *MACHIN* statement should contain at least 1 machine number, but no more than 10.

MACONLY: This statement is only valid within a Macro definition.

All statements, excluding comments, must be contained in a *Macro*.

MANYAXIS: Number of axes in motion block exceeds maximum allowed.

The number of axes moving simultaneously exceeds the maximum number specified in the **MPost** utility. The move will be broken up into two or more motion blocks.

MANYERRS: #FATAL - PostWorks terminated due to too many errors.

The part program generated too many messages of a certain class (*WARNINGs*, *ERRORs*, *FATALs*), as specified using a runtime option. **PostWorks** has been terminated prematurely, without processing the entire part program.

MANYPRMS: Too Many parameters are specified on this Command.

The post-processor command specified inside a macro definition has more than 50 parameters.

MAXARG: Too many arguments for Macro. Remainder ignored.

The post-processor command that called the *Macro* contains more parameters than were allocated using the *Macro* definition statement. The remainder of the parameters will be ignored.

MDFVER: MDF version of ‘a’ is later than program version of ‘b’.

The MDF version used for post processing is a later version and not compatible with current version of **PostWorks**. “a” denotes the version of **MPost** used to create the MDF file. “b” denotes the current version of **PostWorks** used for post processing.

MEMALL: Error trying to allocate work memory. System error.

A system error occurred while trying to allocate memory from the operating system, probably because you do not have enough resources to run this routine.

MINOREXP: Minor word expected. This command is ignored.

A numeric value was encountered on a post-processor command where a minor word was expected.

MULTBLK: This move was broken up into multiple blocks.

MULTLAB: Multiple definition of Label.

This label was previously defined in the current *Macro*. Labels within a *Macro* may not have the same name.

MULTMAC: Multiple definition of Macro.

This *Macro* was previously defined.

MULTPARM: Multiple definition of same parameter.

A parameter in a command was defined twice. For example, the floating point format was defined twice in a format expression.

MULTVAR: Multiple definition of Variable.

A variable was defined twice, using the *CHAR* or *REAL* specification statement, within a *Macro*. Variables may have the same name in different *Macros*, but not within the same *Macro* or in the *G\$MAIN Macro* and any other *Macro*.

NESTCALL: Post-processor command will generate illegal internal call.

A post-processor command was encountered in a *Macro* that does not allow post-processor commands, such as *PSTERR*.

NOAVOID: A CLRSRF/AVOID command has not yet been defined. Ignored command.

A *CYCLE/AVOID* command was programmed without first defining the cycle avoidance plane using the *CLRSRF/AVOID* command.

NOAXIS: This axis is not active in the current post configuration.

A reference was made to an axis that is not supported in the active *MDF* file. For example, *MODE/XAXIS,2* was programmed when there is only a primary X-axis.

NOCIRDIR: Cannot calculate circular direction. Output as linear motion.

There is insufficient point data in a circular interpolation record to calculate a direction from. The circular motion will be output as a series of linear moves.

NOCRAD: Calculated circle radius is zero.

The radius of a *CIRCLE* drive surface (*TLON,GOFWD/circle,on,plane*) has calculated to be zero. This command is ignored and will not generate any motion. Refer to the [APT Source File Formats](#) Appendix for a description of valid APT Source records.

NOCYCDEF: CYCLE/ON programmed without a previous CYCLE mode. Ignored command.

A *CYCLE/ON* command was programmed without programming a *CYCLE/mode* (*BORE,DRILL,etc.*) prior to it.

NOHELP: There is no help for this prompt.

The help text for this menu/prompt was not found. This error will usually be generated when the user modified the '*posthelp.HLP*' file.

NOLDTOOL: A LOADTL statement was not encountered. Ignored command.

A *SELCTL/AUTO* command was programmed, but the end of the clfile was encountered prior to finding a *LOADTL* statement. The *SELCTL* command is ignored.

NOMACHIN: A MACHIN card was not found. The clfile was not processed.

A *MACHIN* statement containing a post-processor configuration number must be located in the input clfile (part program).

NOMACRO: This Macro has not been defined.

A Macro that has not been defined was referenced in the *ENABLE* or *DISABLE* command.

NOMEMALL: A requested portion of memory has not been allocated. Internal bug.

The *Macro* processor has attempted to reference a memory location that is not in the *Macro object file*. This is an internal bug and should be reported.

NOPUNCH: A Punch file is not begin generated due to previous errors.

The punch file output is suppressed due to the corresponding MDF was specified not to output punch file when either error, warning or fatal is encountered.

NORETRCT: A RETRCT command has not been programmed yet. Ignored command.

The *PLUNGE* command repositions the tool to the location it was at prior to programming a *RETRCT* command. Therefore, a *RETRCT* command must be programmed previously in the program before programming a *PLUNGE* command.

NOROTB1: Unable to bring move within limits by rotating table.

A move was programmed that caused an axis to move outside of the machine limits while a rotary table was perpendicular to the spindle axis. **PostWorks** attempted to rotate the table to bring all axes within limits, but was unable to do so. It is possible that enabling the

MODE/ROTATE command, which uses the rotary table to fulfill a linear axis position, will alleviate this problem.

NOROTSOL: The active rotary axes could not fulfill the tool axis vector.

A solution for the tool axis using the currently active rotary axes could not be found. This error will usually occur when one or more of the rotary axes are inactive, or when using a single rotary axis machine configuration. The portion of the tool axis vector that could not be solved for will be displayed with this message.

NOSCALAR: A requested variable has not been allocated. Internal bug.

PostComp has attempted to reference a variable location that has not been allocated. This is an internal bug and should be reported.

NOSCHLEV: There is no such menu level.

A multiple menu level entry was issued at the 'Selection:' prompt in **MPost**, for example '1.5.12', for which there is no prompt.

NOTAPPLY: This prompt does not apply to the current post configuration.

A menu level entry was requested for a menu/prompt that does not apply to the current post-processor configuration. For example, selecting the Lathe Cycles menu while defining a Mill.

NOTBEGIN: Statement may not begin with this.

A statement may not begin with a number, operator or quoted text string.

NOTDESIR Output may not be what is desired.

NOTPERP: Tool axis must be perpendicular to a table. Ignored command.

The tool axis must be parallel to a rotary table rotation axis when the *MODE/ROTATE,ON* command is issued, otherwise the tool axis vector will control the rotary table and it cannot be controlled using the input linear positions.

NOTPERP1: Tool axis is not perpendicular to a table. Command MODE/ROTATE has been cancelled.

The tool axis has moved while *MODE/ROTATE* is active. Polar interpolation has been automatically cancelled. The tool axis must remain fixed and parallel to a rotary table during polar interpolation.

NOTVALID: This command is not valid for this machine. Ignored command.

A post-processor command was programmed that does not apply to the current post-processor configuration. For example, programming a *ROTABL* command when the active *MDF* file does not support a rotary axis.

NUMBEXP: Number expected. This command is ignored.

A minor word value was encountered on a post-processor command where a numeric value was expected.

NUMVAREX: Number or variable expected.

A parameter that was not a number nor a real variable was encountered in a command where one of these types were required.

OBJVER: PMacro Object version 'a' is a mismatch with program version of 'b'.

The *PostComp* utility used to generate the specified object file does not match with the current version of *PostWorks*. “a” denotes the version of *PostComp* used to create the object file. “b” denotes the current version of *PostWorks* used for postprocessing.

OPEREXP: Operator expected.

An operator must separate each number or variable in an expression.

OUTNOT: Output may be corrupt.

The output file may be corrupt and not useful. This might be due to incorrect version of MDF or obj file used.

PARENIV: Invalid nesting of parenthesis.

You can nest parenthesis no more than 50 levels. Also, a left parenthesis directly followed by a right parenthesis ')' is not allowed.

PARENLF: Left parenthesis without right parenthesis.

A statement has more left parentheses than right parentheses. Each left parenthesis must have a corresponding right parenthesis.

PARENRT: Right parenthesis without left parenthesis.

A statement has more right parentheses than left parentheses. Each right parenthesis must have a corresponding left parenthesis.

PGMLARG: Program is too large.

The *Macro* definition program is too large to be handled by **PostComp**. You should report this problem if you encounter it.

PLNEXP: Plane expected.

POSITROT: An attempt was made to move a positioning axis with contouring axes.

A move was programmed that caused a rotary axis, with no contouring capabilities, to move with one or more contouring axes. The move will be broken up into multiple blocks, with the positioning axis output in a motion block prior to the motion block containing the contouring axes.

PRCMULT: #ERROR - Multiple definition of Print Descriptor File record.

The active *PDF* file contains two or more definitions for the same record number. Record definitions have the format: '*#RECORD n#*'.

PRCSYN: #ERROR - Print Descriptor File syntax error.

An entry in the active *PDF* file is syntactically incorrect. The entry in question will be displayed with this error message.

RDPRM: Error reading Standalone Prompt record.

An error occurred trying to read either the '*postmenu.MSG*' file or the '*posterror.MSG*' file. You may have to recreate these files using the **GenPrompt** routine.

RETRDISA: Automatic retract feature is disabled.

This error message will only be displayed when the rotary axes are taking the longest route and the automatic retract feature is disabled. In this case, the machine will move directly to the calculated position without taking any precautionary measures to ensure part integrity. If the tool is in contact with the part, it is most likely that it will destroy the part to get to the new position.

You can either enable the automatic retract feature, so that *v* will clear the tool from the part before moving to the new position, or on some machines, you can set the rotary axis limits so that the tool will follow the part around to the new position. It is recommended that you use the automatic retract feature.

ROTNOSAT: The rotary axis could not be moved in the requested direction.

A rotary axis, whose direction was requested using the *ROTABL/NEXT* command, could not be moved in this direction, most likely because of machine axes limits.

ROTXFM: **There is rotary axes movement when a transformation block is active.**

SETTINGS: **Some of the settings may be incorrect.**

SFMMAX: **Spindle SFM speed exceeded maximum allowed. Set to maximum.**

A *SPINDL/SFM* command was programmed that set the Surface Feed per Minute greater than the maximum value allowed for this post-processor configuration. The *SFM* value was set to the maximum value allowed.

SFMMIN: **Spindle SFM speed exceeded minimum allowed. Set to minimum.**

A *SPINDL/SFM* command was programmed that set the Surface Feed per Minute less than the minimum value allowed for this post-processor configuration. The *SFM* value was set to the minimum value allowed.

SLWSHORT: **Distance too short for slowdown span. Output entire move at slowdown feed.**

A move programmed while post-processor generated slowdown blocks are in effect is shorter than the distance required to perform the slowdown span. The entire move will be output using the slowdown feed rate.

SMDNOPT: **The second mandatory tool end coordinate was not programmed.**

SMDNPERP: **The forward vector and tool axis vector are not perpendicular.**

SMDNPRP3: **The drilling vector for Head 3 is not perpendicular to the stringer.**

SMDNPRP4: **The drilling vector for Head 4 is not perpendicular to the stringer.**

SMDNSAME: **Coordinates must be programmed in pairs and be the same location.**

SMROTAX1: **Both active rotary axes move about the same vector.**

Both of the active rotary axes are positioned in such a way that they both rotate around the same vector, making it impossible to calculate which axis to move in order to satisfy the tool axis vector.

SPAWN: **Error trying to spawn sub-process "'post'".**

An error occurred while trying to run a machine specific post-processor. The name of the 'post' will be displayed with the error message.

SPNLMT: Spindle speed = ‘a’ Limit = ‘b’

This error message is output together with the “Spindle RPM/SFM speed exceeded maximum/minimum allowed. Set to maximum/minimum.” error message. “a” denotes the specified spindle speed and “b” denotes the allowable maximum/minimum spindle speed.

SPNMAX: Spindle RPM speed exceeded maximum allowed. Set to maximum.

A *SPINDL/RPM* command was programmed that set the spindle Revolutions Per Minute greater than the maximum value allowed for this post-processor configuration. The *RPM* value was set to the maximum value allowed.

SPNMIN: Spindle RPM speed exceeded minimum allowed. Set to minimum.

A *SPINDL/RPM* command was programmed that set the spindle Revolutions per Minute less than the minimum value allowed for this post-processor configuration. The *RPM* value was set to the minimum value allowed.

STENDCON: End of file reached while searching for continuation record.

The last line of the *Macro definition file* has a continuation character (\$).

STLONG: Statement cannot be longer than 512 characters.

A statement, including its continuation lines, cannot be longer than 512 characters.

SUBEXP: Subscript expected.

A real variable that has been declared as an array must be referenced with a subscript. Also, Post variable arrays must have a subscript.

SUBMANY: Too many subscripts specified.

SUBNEED: Not enough subscripts specified.

SUBBOUND: Subscript out of bounds.

A reference was made to a variable array, where the subscript was less than 1 or greater than the dimensioned size of the array.

SUBRANG: Subscript range not allowed.

SYNMACH: Invalid Machine number: "n".

An invalid number was entered with the *-MACHINE* runtime option. “n” is the machine number in question.

SYNOPT: Invalid syntax for runtime option: "option".

The syntax of a recognized runtime option is incorrect. The error message will contain the option in question.

TAB1ST: Tables must be defined prior to heads.

When defining the rotary axes configuration in ***MPost*** it is mandatory to specify the rotary tables prior to the rotary heads. The rotary axes should actually be defined in the following order.

1. Table riders.
2. Table carriers.
3. Head carriers.
4. Head riders.

TABCYL: Cylinder axis must be a table type rotary axis.

When enabling cylindrical interpolation for a mill/turn machine using the ***MODE/MILL,DIAMTR*** command, the rotary axis used for cylindrical interpolation must be defined as a rotary table and not a rotary head.

TABCYL1 Invalid 3-rd axis specified in machine plane definition.

TOOLDIF1: This tool number does not match the tool selected using SELCTL.

The tool number in a ***LOADTL*** command does not match the tool number used in the previous ***SELCTL*** command. The ***SELCTL*** tool number will be used.

TOOLDIF2: The SELCTL parameter will be used. SELCTL = 't'

This error message is output together with the above "This tool number does not match the tool selected using SELCTL." error message. "t" denotes the SELCTL tool number.

TOOLEN1: This tool does not have the same tool length as previously programmed.

The tool length specified in a ***LOADTL*** or ***SELCTL*** command does not match the tool length previously programmed for this tool. ***PostWorks*** will display the old programmed tool length with this message and will use the newly programmed tool length.

TOOLEN2: The new length will be used. Previous length = 'len'

This error message is output together with the above "This tool does not have the same tool length as previously programmed." error message. "len" denotes the previous tool length.

TOOLGRP1: This tool gripper does not match the gripper selected using SELCTL.

The gripper selection (*LARGE*, *SMALL*) specified in the *LOADTL* command does not match the gripper size used in the previous *SELCTL* command. The *SELCTL* gripper size will be used.

TOOLMAX: Tool number over maximum allowed. Maximum = 'n'.

The tool number programmed on a *SELCTL*, *LOADTL* or *UNLOAD* command exceeds the maximum number allowed for this machine configuration. The maximum value will be used.

TOOLNSEL: This tool was not previously selected. A SELCTL was issued internally.

A *LOADTL* command was programmed without a previous *SELCTL* command to preselect the tool. On machines that support a tool selection sequence, you must select the tool prior to loading it. **PostWorks** will automatically issue a *SELCTL* command when it is not programmed.

TOOLOVR1: You have exceeded the local tool storage.

A maximum of 120 tools can be used in a single part program. The last defined tool data stored will be overwritten with the newly selected tool.

TOOLRD1: This tool does not have the same tool radius as previously programmed.

The tool radius specified in a *TURRET* command does not match the tool radius previously programmed for this tool. **PostWorks** will display the old programmed tool radius with this message and will use the newly programmed tool radius.

TOOLRD2 The new radius will be used. Previous radius = "r"

This error message is output together with the above "This tool does not have the same tool radius as previously programmed." error message. "r" denotes the previous radius.

TOOLSPN1: This tool spindle does not match the spindle selected using SELCTL.

The tool spindle selection (*HIGH*, *LOW*) specified in the *LOADTL* command does not match the spindle selected in the previous *SELCTL* command. The *SELCTL* spindle selection will be used.

TXTVAREX: Text variable expected.

A parameter that was not a text variable or text string was encountered in a command where one of these types are required.

UNCLSDIF: Unclosed IF/DO structure.

An *IF-THEN-ELSE* structure was not terminated with an *ENDIF* statement, or the ending label for a *DO* loop was not found.

UNDEFLAB: Undefined label.

A reference was made to a label, either in a *JUMPTO* or *DO* statement, that does not exist.

UNDEFVAR: Undefined variable.

A variable name was encountered that was not declared using the *REAL* or *CHAR* specification statements. All variables used in the program must first be declared.

UNRECCL: Unrecognized cfile record. Type = 'm' Sub-type = 'n'.

A cfile record was encountered that is not recognized by **PostWorks**. The record type 'm' and subtype 'n' will be contained in the error message. This error should reported.

UNRECCMD: Unrecognized command.

A post-processor command was encountered in a *Macro* that is not recognized nor supported by **PostComp**. This may be because you are running an earlier version of **PostComp** than you expect.

VAREXP: Real variable expected.

VARVOCNM: Variable, vocabulary word or number expected.

A parameter that is not a variable, vocabulary word or number was encountered in a command where one of these types are required.

VECPLN: Tool axis does not intersect clearance plane. Ignored command.

A *RETRCT* command was programmed when the tool axis is parallel to the clearance plane, as defined by the *CLRSRF* command.

VOCEXP: Vocabulary word expected.

A parameter that is not a vocabulary (minor) word was encountered in a command where a vocabulary word is required.

VE CZERO: Calculated vector has a length of zero.

The *INDIRV/INDIRP* statement or *CIRCLE* definition contain a vector definition which does not have a length. This entire circular interpolation record will be ignored. Refer to the *APT Source File Formats* appendix for a description of valid APT Source records.

WRDNUMRG: Vocabulary word, number or range expected.

The *SYNTAX* command does not accept variables or expressions; only vocabulary words, numbers and ranges may be specified.

ZPARLEL: The Z-axis is parallel to the XY plane. The output point is incorrect.

The Z-axis, which rotates with the rotary axis (as specified in the *MPost* utility), is positioned so that it moves parallel to the XY plane. It is impossible to calculate the tool location in this situation.

APPENDIX E Glossary

Alignment Block

A block which contains enough information (tape codes) to restart the Control Tape from at the machine. A startup block.

APT Source File

A text file which contains cutter centerline points, tool axis vectors, post-processor commands and circular interpolation records. Most CAM systems can generate APT source files which can be used as input to **PMacro** and **PWorks**. An APT source file usually has a file extension of '.as' or '.cla'.

Array

A variable that is dimensioned so that it can contain more than one value. Arrays are referenced using a subscript that identifies the entity within the array to access.

Automatic Cycle

A canned cycle whose output consists of codes that will cause the machine to generate the moves required for the cycle sequence.

Block

A logical group of codes, terminated with an End-of-block code, which constitute a single line of information for the machine control.

Canned Cycle

A single output block or group of output blocks, generated from a single input statement, which will generate a series of moves at the machine. Cycles are usually used to drill holes or rough cut the part.

Checksumming

The process of adding a summation value to each block of information transmitted to a machine, so that the machine can verify the contents of what has been sent.

Circular Interpolation

A single block of output which will cause the machine to move in a circular arc. A circular interpolation block usually contains a direction code (G02, G03), the center of the circle (I, J, K) and the final point on the arc (X, Y, Z).

Clfile

A binary file that contains cutter centerline points, tool axis vectors, post-processor commands and circular records. This type of file is used as input to **PMacro** and **PWorks**, and is also created as an output file from **PMacro**. An input clfile, created by a CAM system, usually has the extension '.cl', while an output clfile, created by v, has the extension '.cln'.

Code

A single entity of information which is usually composed of a register designator and a value. For example, G02, X1.543, S100, etc.

Control Character

A special character that is generated by pressing the CTRL key and a letter key simultaneously. Control characters are considered a single keystroke and are used when editing prompt input at runtime. A control character is usually referenced with a circumflex (^) and the character. For example, Control C would be referenced as ^C.

Control Tape

The blocks of information which constitute a sequence of operations that will cut a part at the machine.

Custom Post-processor

A post-processor that is customized for a specific machine control or family of controls.

Cutter Compensation

The process where the machine control will compensate for over/under sized cutters as compared to the diameter of the programmed cutter.

Cylindrical Interpolation

Multiple blocks of information which contain linear axis moves in conjunction with rotary axis movement. Cylindrical interpolation is used with mill/turn machines to ease the programming of round paths while in milling mode. The path is usually programmed as a flat shape and the post-processor will convert it to a cylindrical shape.

Form

A group of entry fields that are contained on a single page. The user can randomly enter responses in various fields within a form.

Helical Interpolation

A circular interpolation block that also contains movement for a third linear axis, thereby creating a helix at the machine.

Input File

A file which is read by the **PostWorks** routines, such as *Macro definition files*, *Machine Descriptor Files (MDF)*, *clfiles*, etc.

Instruction

Executable statements in a Macro definition file are compiled into basic commands understood by the *Macro* processor. These commands are binary instructions that contain only the necessary information to perform the function as specified by the input statement. A single statement may generate multiple instructions.

Interference Zone

An area on the machine in which the spindle and fixture will collide.

Linear Axis

An axis on the machine that moves in a straight line. For example, the X, Y, and Z axes.

Linear Interpolation

A single block of output that will cause the machine to move in a straight line.

Linearization

The process where a move containing rotary axes movement is broken up into smaller linear moves for the purpose of keeping the tool end point within a certain tolerance.

LMDP Mode

An acronym (*Lathe Mill Diameter Programming*) for Cylindrical Interpolation mode on a Mill/Turn type machine.

LMFP Mode

An acronym (*Lathe Mill Face Programming*) for Polar Coordinate Interpolation mode on a Mill/Turn type machine.

Machine Adjust File

A file which can contain a transformation matrix and/or new programmed tool lengths. This file is used to adjust the post-processor output when the part is placed on the machine in a different position or when the actual tool lengths differ from the input program. The clfile does not have to be regenerated from the CAM system to perform these adjustments.

Machine Specific Post-processor

The post-processor that converts a clfile to the format required by the machine control. This may be a general post-processor or a custom post-processor for a specific machine control.

Macro

A Macro contains group of statements that will be executed whenever its corresponding post-processor command is encountered in the input clfile. Post-processor *Macros* are contained in a separate source file that is compiled by **PostComp** and should not be confused with macros resident in a CAM system. References to post-processor Macros will always be referred to in this manual with a capital M.

Macro Argument

Minor words/values in post-processor commands which call *Macros* will be passed to the *Macro* as arguments. *Macro* arguments are referenced as *%ARG(n)* and can contain real values or minor words. *Minor words* will be stored as text strings.

Macro Definition File

A text file that is used as input to the **PostComp** utility and contains Macro definitions.

Macro Object File

A binary file that is created by **PostComp** after compiling a [Macro definition file](#). The [Macro object file](#) contains the binary instructions that will be used by the *Macro* processor resident in **PMacro** and **PWorks**.

Macro Processor

The post-processor *Macro* interpreter located within the **PMacro** and **PWorks** routines which processes the compiled *Macro* definitions contained in the *Macro* object file.

Major Word

A recognized vocabulary word that begins a post-processor statement, such as *LOADTL*, *PPRINT*, *FEDRAT*.

Manual Cycle

A canned cycle where the post-processor will generate the moves required to perform the cycle sequences.

Menu

An interactive display that allows you to choose from a multiple of entries. Each entry displayed in the menu will either select another menu, a form or a group of prompts.

Mill/Turn Machine

A Mill/Turn is a machine which is designed for both turning and milling operations. It is typically setup as a lathe with a milling head attachment. When in mill mode, the spindle can be programmed as a rotary axis.

Minor Word

A recognized vocabulary word that is used as a parameter on a post-processor statement, such as *CLW*, *NOW*, *ON*. Text variables may also be used to represent minor words in some circumstances.

Motion Block

A Control Tape block of information which contains linear and/or rotary axes movement.

Mutually Exclusive Axes

Two or more axes that cannot be programmed to move in the same motion block.

Number

A string of numeric characters including the plus, minus and period characters. All numeric strings will be treated as real numbers.

Operator

Character(s) that perform a specific operation in a numeric or text expression. For example, addition (+), subtraction (-), etc.

Output File

A file that is created by the **PostWorks** routines, such as [Macro object files](#), listing files, print files, etc.

Parameter

The sub-part(s) of a statement. Statements usually consist of a command name and modifiers that support the command. Parameters are considered the command modifiers.

Part Program

The input file to the CAM processor that contains all of the statements required to define and cut the actual part.

PC

Macro statements are compiled into simple instructions that can be interpreted by the *Macro* processor. The PC or “Program Counter” is a pointer to the location of an instruction.

Polar Interpolation

Rotary table positions which are output to satisfy input linear moves rather than to fulfill a tool axis vector. The linear axes contain the radius of the polar coordinate, while the rotary table contains the angle. This mode can be used with both mills and Mill/Turn type machines.

Post Variable

Variables that are internal to **PostWorks** and can be accessed as real and/or text variables within a *Macro*. *Post variables* are referenced as *%name*.

Post-processor Command

A statement consisting of a major word and optional minor words/values. These statements are passed to the clfile to be processed by the machine specific post-processor and may call post-processor *Macros*.

Preset Axis Registers Block

A Control Tape block which contains axis registers and values that will be stored as the current position for each programmed axis in the block. Used to translate the machine zero to the programmed zero location.

Prompt

A single question, displayed on the screen, that requires input from the user.

Real Expression

A statement containing real numbers/variables, numeric operators and/or functions that equate to a single value. Real expressions may be used in most places where a real number/variable can appear.

Real Variable

A symbolic name that represents a stored real number or array of numbers. Real variables are defined using the *REAL* statement and can be 1 to 8 characters in length.

Register

A storage location within a machine control which contains numeric values. Each register in the machine control will perform a separate function, for example, moving an axis, performing preparatory and miscellaneous functions, etc. Registers are usually designated by a single letter (A, B, G, M, etc.) and a value.

Rotary Axis

An axis on the machine that moves in a circular arc. For example, the A and B axes.

Spline Interpolation

A group of output blocks which contain the definition of a spline. The control will use this definition to cut the actual spline on the machine. Spline interpolation will usually generate less data and create smoother motion than a series of linear moves, lending itself to high speed machining.

Subscript

A pointer within an array that determines the array entity to access. Subscripted arrays are referenced using the 'var(sub)' syntax.

Tape Break

A break in the Control Tape due to tape length, machining time, or any other reason. A tape break can consist of breaking the logical Control Tape into multiple physical tapes, or simply inserting a tape block at the location of the tape break.

Text Expression

A statement containing text strings/variables, text operators and/or functions that equate to a single text string. Text expressions may be used in most places where a text string/variable can appear.

Text File

A file that is composed of human-readable characters, such as letters, numbers and punctuation. A *Macro* definition file is a text file.

Text Post-processor Command

A post-processor command that requires a text string parameter instead of minor words/values. Text post-processor commands are: *INSERT*, *LETTER*, *PARTNO* and *PPRINT*. These commands will accept an unquoted text string as their parameter when the major word starts in the first column.

Text String

A group of characters that enclosed in quotes. Text strings may be used to represent vocabulary words or simply as output strings to the listing file.

Text Variable

A symbolic name that represents a stored text string. Text variables are defined using the *CHAR* statement and can be 1 to 8 characters in length.

Tool Change Sequence

The entire series of operations that are required to change the cutting tool at the machine.

Ultrasonic Cutter

This type of machines uses a pulsating knife or disk cutter to cut a variety of materials (cloth, honeycomb, wood, etc.). When using a knife cutter, a programmable rotary head is used to keep the blade pointed in the same direction as the cut.

User Defined Block

A *User Defined Block* describes the “look” of a special Control Tape block. It contains a list of registers and optional values that will be output whenever the *User Defined Block* is output.

Vacuum Pods

Vacuum pods are a flexible fixturing alternative to standard bolts and clamps. The pod itself consists of a vacuum tube connected to a suction cup at the end. The part is placed on the suction cups and a vacuum is applied to hold down the part. The height can be set independently for each pod and some pods have swivel joints so that they can hold contoured parts.

APPENDIX F Machine Configuration Files

F.1 Introduction

PostWorks is delivered with a library of demonstration machine/post-processor configuration files that support a variety of machine tools. Some of the demo files are set up to emulate existing custom post-processors. The demo files consist of *Machine Descriptor Files* (*PWORKS_#.mda*), *Macro* files (*pmacro_#.mac*) and *Print Descriptor Files* (*pworks_#.pdf*).

In order to conserve space on the distribution media, the *Machine Descriptor Files* are being delivered in their text format (*.mda*) instead of their binary format (*.MDF*). You must run these files against the **GenXfer** Utility Program prior to using them. Also, the *Macro object files* (*.OBJ*) are not furnished. You must run the *Macro definition files* (*.mac*) against the **PostComp** utility prior to using them.

F.2 Numbering Scheme

It is recommended that you develop a numbering scheme for naming the post-processor configuration files (*PWORKS_#.MDF*, *pmacro_#.mac*, *etc.*). The numbering scheme used should reflect the type of machine and/or control that the post-processor configuration is targeted for.

NCCS uses a numbering scheme when developing post-processor configuration files for **PostWorks**. Following is an explanation of this numbering scheme.

Number:	8	1	001	3	2	
						-Number of rotary axes.
						- Number of linear axes.
						- Machine make.
						- Machine type.
						- 8 = Demo file generated by NCCS .

Machine type can be one of the following:

- 1 = Mill
- 2 = Lathe
- 3 = Ultrasonic Cutter
- 4 = Mill/Turn

Machine makes uses the following numbering convention to designate the make of the machine and/or control.

001 - 054	Cincinnati
055 - 108	Fanuc
109 - 162	Allen Bradley
163 - 189	Bostomatic
190 - 214	Hurco
215 - 240	Omnimil
241 - 260	Tree
261 - 300	Kearney & Trecker
301 - 320	Ferrari
321 - 340	GFM
341 - 360	Dorries Scharmann

F.3 Demonstration Machine Configurations

Following is a list of demo files, including the **PostWorks** number, machine description and the custom post-processor which is emulated, if any.

<u>Number</u>	<u>Machine</u>	<u>Custom Post</u>
8100132	Cinci 5-axis 20V w/Acrumatic 950 Control	Cin950
8100232	Cinci 5-axis Hydro-Tel w/Acrumatic CNC PC	CIN5AX
8100331	Cinci 4-axis T-30 w/Acrumatic 950 Control	
8400421	Cinci Avenger 250MT Mill/Turn w/850 CNC	
8105532	SNK FSP-120V 5-axis Profiler w/Fanuc 11M	SNK5AX
8105632	Okuma Howa 5-axis VMP-8 w/Fanuc 15M	FANUC5
8105730	Generic 3-axis Mill w/Fanuc 11M	
8100930	Ekstrom/Carlson 3-axis Router w/AB 8600	
8116332	Bostomtic 5-axis w/SPC II Control	BMSPC5
8121532	OM3 Vertical 5-axis Omnimil w/Sundstrand	OMSNS5
8121632	OM3 Vertical 5-axis Omnimil w/Sundstrand	BXOM35
8126131	Kearney & Trecker KT/CNC Control Type C	
8130133	Ferrari 6-axis Machining Center	
8224120	Tree Ultra Prec. 1000 CNC Turning Center	TRUTL1
8332133	GFM Ultrasonic Core & Knife w/CNC 6000	
8434133	Dorries Scharmann VC4400 w/Siemens 880T	

It is recommended that you rename the demo files to a numbering scheme which has been set up at your facility before using any of the demo files. You should also verify the output created using the demo files that emulate custom post-processors, as the options that are used may not be the same options that you use. You should reference the *Macro definition file (pmacro_#.mac)* for a description of the options which were used as default settings and which commands are supported.

APPENDIX G APT Source File Formats

G.1 Introduction

An APT Source file is a text file which contains cutter centerline points, tool axis vectors, post-processor commands and circular interpolation records. The APT Source file is used as input to **PMacro** and **PWorks**. **PostWorks** supports generic APT Source files (as produced by most CAD/CAM systems and APT Source files generated by **PTED**). This chapter describes the various APT Source file formats which are supported.

G.2 PTED APT Source File Format

```
N1      , PARTNO      THIS IS AN ASCII CL EXAMPLE
N2      , MACHIN/ PWORKS, OPTION, 4., 1000.
N3      , COOLNAT/ ON
N4      , SPINDL/ 300., CLW
N5      , FEDRAT/ IPM, 10.
N6      , MULTAX/ ON
N20     , PPRINT *****~
*****~
N21     , PPRINT
N22     , PPRINT      MULTAX/ON WITH CIRCULAR
N23     , PPRINT
N24     , PPRINT *****~
*****~
N26     , CUTTER/ 0.
N26     , FEDRAT/ IPM, 10.
N27     , FROM/      0.      ,      4.      ,      0.      , $
          0.      ,      0.      ,      1.
N29     , GOTO/      0.      ,      2.      ,      0.      , $
          0.      ,      0.      ,      1.
N30     , CIRCLE/ 2., 2., 0., 0., 0., 1., 2.
N30     , GOTO/      0.00394654,      1.87441893,      0.      , $
          0.      ,      0.      ,      1.
          GOTO/      0.1577055,      1.74933351,      0.      , $
          0.      ,      0.      ,      1.
          GOTO/      0.03542554,      1.62523736,      0.      , $
          0.      ,      0.      ,      1.
          GOTO/      0.06283367,      1.5026202 ,      0.      , $
          0.      ,      0.      ,      1.
          GOTO/      0.09788692,      1.38196601,      0.      , $
          0.      ,      0.      ,      1.
          GOTO/      0.14044702,      1.26375087,      0.      , $
          0.      ,      0.      ,      1.
          GOTO/      0.19034588,      1.14844145,      0.      , $
          0.      ,      0.      ,      1.
          GOTO/      0.24738657,      1.03649266,      0.      , $
          0.      ,      0.      ,      1.
          GOTO/      0.31134415,      0.92834626,      0.      , $
          0.      ,      0.      ,      1.
```

	GOTO/	0.38196599,	0.82442947,	0.		, \$
		0.	0.	1.		
	GOTO/	0.45897353,	0.72515198,	0.		, \$
		0.	0.	1.		
	GOTO/	0.54206288,	0.63090571,	0.		, \$
		0.	0.	1.		
	GOTO/	0.63090599,	0.54206261,	0.		, \$
		0.	0.	1.		
	GOTO/	0.72515225,	0.45897338,	0.		, \$
		0.	0.	1.		
	GOTO/	0.82442975,	0.38196585,	0.		, \$
		0.	0.	1.		
	GOTO/	0.92834675,	0.31134388,	0.		, \$
		0.	0.	1.		
	GOTO/	1.036493 ,	0.24738643,	0.		, \$
		0.	0.	1.		
	GOTO/	1.14844179,	0.19034574,	0.		, \$
		0.	0.	1.		
	GOTO/	1.26375133,	0.14044688,	0.		, \$
		0.	0.	1.		
	GOTO/	1.38196653,	0.09788679,	0.		, \$
		0.	0.	1.		
C30	GOTO/	1.50262082,	0.06283354,	0.	(8)	, \$
		0.	0.	1.		
	GOTO/	1.625238 ,	0.03542542,	0.		, \$
		0.	0.	1.		
	GOTO/	1.74933422,	0.01577055,	0.		, \$
		0.	0.	1.		
	GOTO/	1.8744197 ,	0.00394654,	0.		, \$
		0.	0.	1.		
	GOTO/	1.99999976,	0.	0.		, \$
		0.	0.	1.		
N32	, PPRINT	*****~				

N33	, PPRINT					
N34	, PPRINT	MULTAX/OFF WITHOUT CIRCULAR				
N35	, PPRINT					
N36	, PPRINT	*****~				

N38	, MULTAX/ OFF	(9)				
N39	, GOTO/	2.12915826,	0.03075678,	0.		
	GOTO/	2.25473666,	0.0688605 ,	0.		
	GOTO/	2.37857127,	0.11480334,	0.		
	GOTO/	2.49921393,	0.16811167,	0.		
	GOTO/	2.61640358,	0.22865151,	0.		
	GOTO/	2.72970319,	0.29619893,	0.		
	GOTO/	2.83865142,	0.37047425,	0.		
	GOTO/	2.94295406,	0.45128167,	0.		
	GOTO/	3.04210591,	0.53819144,	0.		
	GOTO/	3.13858151,	0.63144416,	0.		
	GOTO/	3.34466028,	0.83780634,	0.		
	GOTO/	3.43135571,	0.91841102,	0.		
	GOTO/	3.52175188,	0.99392092,	0.		
	GOTO/	3.62035918,	1.06395745,	0.		

	GOTO/	3.70932937,	1.11494637,	0.
	GOTO/	3.79211926,	1.15169382,	0.
	GOTO/	3.8718183 ,	1.17745304,	0.
	GOTO/	3.95006561,	1.19389391,	0.
	GOTO/	1.02803946,	1.20195758,	0.
	GOTO/	4.10590458,	1.20185661,	0.
	GOTO/	4.18362951,	1.19406557,	0.
	GOTO/	4.263978 ,	1.17865992,	0.
	GOTO/	4.34895897,	1.15526342,	0.
	GOTO/	4.44172001,	1.12293065,	0.
	GOTO/	4.54821253,	1.07959545,	0.
	GOTO/	4.82110882,	0.96139657,	0.
	GOTO/	4.92559004,	0.92267281,	0.
	GOTO/	5.024508 ,	0.89359397,	0.
	GOTO/	5.12563467,	0.87220365,	0.
	GOTO/	5.23160553,	0.85700464,	0.
	GOTO/	5.35145998,	0.84629267,	0.
	GOTO/	5.49503994,	0.83895594,	0.
	GOTO/	5.70051813,	0.83074659,	0.
	GOTO/	5.83352327,	0.82169169,	0.
	GOTO/	5.95785618,	0.80693001,	0.
	GOTO/	6.08438396,	0.78271842,	0.
	GOTO/	6.19659472,	0.75301135,	0.
	GOTO/	6.31317854,	0.71471387,	0.
	GOTO/	6.43770456,	0.66655028,	0.
	GOTO/	6.5747714 ,	0.60659111,	0.
C39	GOTO/	6.73438025,	0.53057396,	0.
	GOTO/	6.94783735,	0.42530948,	0.
	GOTO/	7.1788435 ,	0.31666842,	0.
	GOTO/	7.41463518,	0.21491478,	0.
	GOTO/	7.64902401,	0.12256801,	0.
	GOTO/	7.88612135,	0.03771998,	0.
	GOTO/	8. ,	-0.00000021,	0.
N40	, COOLNT/	OFF		
N41	, SPINDL/	OFF		
N42	, STOP			
N43	, FINI			

10

Description:

- ① = Each cl record has the associated input source line number at the front of the record. This is the actual part program line number that generated the cl record. The ISNs are stored as N-values and are separated from the actual cl record with a comma.
- ② = Post-processor commands are stored exactly as they would appear in a part program. Minor words are stored as the actual words and variables are stored as floating point numbers. If the post-processor command is too long to fit in 72 columns, a dollar sign (\$) is appended to the end of the first line and the command is continued on the following line.

- ③ = MULTAX and CUTTER statements are stored exactly as they would appear in a part program. Note that the GOTO points following MULTAX/ON all have tool axis vectors associated with them.
- ④ = The INSERT, LETTER, PARTNO and PPRINT commands contain an ASCII text string instead of minor words and variables. When a text command is longer than 72 characters (including the ISN), it is continued on the next line. The tilde (~) character is used to denote that the command is continued.
- ⑤ = All motion records are stored as either a FROM or GOTO record. GOFWD, GOLFT, GODLTA, etc. generated motion will be stored as GOTO records. When a tool axis vector (MULTAX/ON) is stored with the motion point, a dollar sign (\$) is appended to the end of the GOTO point and the tool axis is stored on the next line.
- ⑥ = Circular records are stored with their canonical form: 'CIRCLE/x, y, z, i, j, k, r'. circular records are always followed by one or more motion records. The circular interpolation record actually consists of the circle record and the motion record(s) immediately following the circle record.
- ⑦ = GOTO point records without an ISN are part of a multiple point cl record. Multiple point cl records are generated from a single command (GOLGT, GOFWD, etc.). Circular records are usually multiple point records (except on very small circles).
- ⑧ = When a single motion command generates more points than will fit on a single cl record, a motion continuation record is output. When the ISN is stored with a C-values instead of an N-value, it denotes that the cl record is a motion continuation record and not a new motion record.
- ⑨ = MULTAX has been turned off. Note that the motion records following the MULTAX/OFF command do not have tool axis vectors associated with them.
- ⑩ = The last command in a clfile is always the FINI command.

G.3 CATIA APT Source File Format

```

PARTNO      THIS IS AN ASCII CL EXAMPLE
MACHIN/ PWORKS,2, OPTION, 4, 1000
COOLNT/ ON
SPINDL/ 300, CLW
READ/2,STRTUP
TLC = MACRO/TN=1,TL,SPIN
      SPINDL/OFF
      COOLNT/OFF
      LOADTL/TN,LENGTH,TL
      SPINDL/SPIN,CLW
TERMAC
FEDRAT/ IPM, 10

```

①
②
③

```

MULTAX/ ON
PPRINT *****

PPRINT
PPRINT CIRCULAR INTERPOLATION RECORD #1
PPRINT
PPRINT *****
CALL/TLC, TN=1, TL=3.5, SPIN=1200
CUTTER/ 1.000000, .000000, .500000, .000000, .000000, .000000,
        2.000000
FEDRAT/ IPM, 10.0000
FROM/ 0.00000, 4.00000, 0.00000, $
      0.000000, 0.000000, 1.000000
GOTO/ 0.00000, 2.00000, 0.00000, $
      0.000000, 0.000000, 1.000000
INDIRV/ .00000, 1.00000, .00000
TLON,GOFWD/ (CIRCLE/ 2.00000, 2.00000, .00000,$
             2.00000),ON,(LINE/ 2.00000, 2.00000, .00000,$
                               2.00000, .00000, .00000)
GODLTA/ .00000, .00000, 2.00000, $
        0.000000, 0.000000, 1.000000
PPRINT *****
PPRINT
PPRINT CIRCULAR INTERPOLATION RECORD #2
PPRINT
PPRINT *****
MULTAX/ OFF
GODLTA/ .00000, .00000, -2.00000
PSIS/(PLANE/0,0,1,0)
INDIRP/ 2.25470, .06890, .0000
TLON,GOFWD/(CIRCLE/CANON, 2.00000, 2.00000, .00000,$
            .00000, .00000, 1.00000, 2.00000),$
            ON,(PLANE/ .00000, 1.00000, .00000, 2.00000)

RAPID
GOTO/ 0.00000, .00000, 5.00000
COOLNT/ OFF
SPINDL/ OFF
STOP
FINI

```

Description:

- ① = Post-processor commands are stored exactly as they would appear in a part program. Minor words are stored as the actual words and variables are stored as floating point numbers. If the post-processor command is too long to fit in 72 columns, a dollar sign (\$) is appended to the end of the first line and the command is continued on the following line.
- ② = The READ/2 command behaves like an INCLUDE command. The file name specified after the '2' parameter is included into the APT source file. **PostWorks** will first try to open the file as specified in the READ command, if it is not found

then it will be searched for in the *PWORKS_DATA* directory. The default file extension is '.mac', since this command is usually used to load system *Macros*.

- ③ = Standard Macro definitions with or without parameters are allowed. Each Macro parameter may or may not have a default value. The contents of the Macro are ignored until called by a CALL statement. The each command of the Macro is processed, with **PostWorks** replacing each parameter used within the Macro with its assigned value, either from the Macro definition or the CALL statement. Only simple post-processor commands may be used within Macros. Variables (other than Macro parameters), If statements, and looping logic (DO) are not allowed.
- ④ = MULTAX and CUTTER statements are stored exactly as they would appear in a part program. Note that the GOTO points following MULTAX/ON all have tool axis vectors associated with them.
- ⑤ = The INSERT, LETTER, PARTNO and PPRINT commands contain an ASCII text string instead of minor words and variables. A text command should always fit within 72 columns.
- ⑥ = Standard Macro calls are allowed. macros can be fully contained within the APT source file as long as they're defined prior to the Macro CALL. They may also be stored in an external file. **PostWorks** will first try to call an internal Macro, then try to load an external file as specified by the Macro name, and last try to load an external file located in the *PWORKS_DATA* directory. The default extension is '.mac'
- ⑦ = All motion records, except GODLTA and circular records are stored as either a FROM or GOTO record. When a tool axis vector (MULTAX/ON) is stored with the motion point, the motion record will be longer than 72 columns.
- ⑧ = This format of circular record assumes that the motion is always in the XY plane. It consists of a directional vector (INDIRV/INDIRP) and a motion command (TLON, GOFWD/circle,on,line). Notice that the circle does not have a vector associated with it, nor is the check surface a plane. See #11 for a variation of acceptable circular records.
- ⑨ = GODLTA motion records are stored as delta distances from the previous position. When a tool axis vector (MULTAX/ON) is stored with the motion point, the motion record will be longer than 72 columns. This tool axis vector is an absolute position and not an incremental distance from the previous tool axis vector.
- ⑩ = MULTAX has been turned off. Note that the motion records following the MULTAX/OFF command do not have tool axis vectors associated with them.
- ⑪ = This format of circular record allows for circular motion in any machining plane. It consists of a PSIS statement, directional vector (INDIRV/INDIRP) and a motion

command (TLON, GOFWD/circle, on, line). Notice that the circle has an axis vector associated with it and the check surface is a plane. The circle axis must be parallel to the tool axis vector for circular motion to be calculated. The PSIS statement is not necessary, as **PostWorks** will automatically calculate the part surface plane.

⑫ = The last command in a cfile is always the FINI command.

G.4 Unigraphics APT Source File Format

```

TOOL PATH/1st Tool Path
PARTNO      THIS IS AN ASCII CL EXAMPLE
MACHIN/ PWORKS,2, OPTION, 4, 1000
FEDRAT/ IPM, 10
CALL//usr/macro/om/tlc,tn=1,tl=2.5,spin=1200
COOLNT/ ON
PAINT/PATH
MULTAX/ ON
DISPLY/1.0 x .25cr E-Mill
FEDRAT/ IPM, 10.0000
FROM/  0.00000,    4.00000,    0.00000, $
      0.000000,    0.000000,    1.000000
GOTO/  0.00000,    2.00000,    0.00000, $
      0.000000,    0.000000,    1.000000
CIRCLE/2.0000, 2.0000, .0000, .0000, .0000, 1.0000, $
      2.0000, .0005, 1.0000, 1.0000, 0.0000
GOTO/  2.00000,    0.00000,    0.00000, $
      0.000000,    0.000000,    1.000000
GODLTA/ .00000,    .00000,    2.00000, $
      0.000000,    0.000000,    1.000000
DISPLY/*****
DISPLY/      MULTAX/OFF WITHOUT CIRCULAR
DISPLY/*****
MULTAX/OFF
GOTO/2.1292,    0.0308,    0.0000
      2.2547,    0.0689,    0.0000
      2.3786,    0.1148,    0.0000
      2.4992,    0.1681,    0.0000
      2.6164,    0.2287,    0.0000
      2.7297,    0.2962,    0.0000
      2.8387,    0.3705,    0.0000
      2.9430,    0.4513,    0.0000
      3.0421,    0.5382,    0.0000
      3.1386,    0.6314,    0.0000
      3.3447,    0.8378,    0.0000
      3.4314,    0.9184,    0.0000
      3.5218,    0.9939,    0.0000
      3.6204,    1.0640,    0.0000
      3.7093,    1.1149,    0.0000
      3.7921,    1.1517,    0.0000
      3.8718,    1.1775,    0.0000

```



```

3.9501,    1.1939,    0.0000
1.0280,    1.2020,    0.0000
4.1059,    1.2019,    0.0000
4.1836,    1.1941,    0.0000
4.2640,    1.1787,    0.0000
4.3490,    1.1553,    0.0000
4.4417,    1.1229,    0.0000
4.5482,    1.0796,    0.0000
4.8211,    0.9614,    0.0000
4.9256,    0.9227,    0.0000
5.0245,    0.8936,    0.0000
5.1256,    0.8722,    0.0000
5.2316,    0.8570,    0.0000
5.3515,    0.8463,    0.0000
5.4950,    0.8390,    0.0000
5.7005,    0.8307,    0.0000
5.8335,    0.8217,    0.0000
5.9579,    0.8069,    0.0000
6.0844,    0.7827,    0.0000
6.1966,    0.7530,    0.0000
6.3132,    0.7147,    0.0000
6.4377,    0.6666,    0.0000
6.5748,    0.6066,    0.0000
6.7344,    0.5306,    0.0000
6.9478,    0.4253,    0.0000
7.1788,    0.3167,    0.0000
7.4146,    0.2149,    0.0000
7.6490,    0.1226,    0.0000
7.8861,    0.0377,    0.0000
8.0000,    0.0000,    0.0000

```

```

RAPID
GOTO/  0.00000,    .00000,    5.00000
COOLNT/ OFF
SPINDL/ OFF
STOP
END-OF-PATH
FINI

```

⑩①

Description:

- ① = 'TOOL PATH', 'PAINT', and 'END-OF-PATH' statements are ignored.
- ② = Post-processor commands are stored exactly as they would appear in a part program. Minor words are stored as the actual words and variables are stored as floating point numbers. If the post-processor command is too long to fit in 72 columns, a dollar sign (\$) is appended to the end of the first line and the command is continued on the following line.
- ③ = Standard Macro calls are allowed. In this example, the Macro is stored in an external file, 'usr/macro/om/tlc'. The Macro name is the same as the file name ('tlc'). The file could contain the following Macro definition.

```

TLC=MACRO/TN=1,TL,SPIN
SPINDL/OFF
COOLNT/OFF
LOADTL/TN,LENGTH,TL
SPINDL/SPIN,CLW
TERMAC

```

Macros can be fully contained within the APT source file as long as they were defined prior to the Macro CALL. They may also be stored in an external file. **PostWorks** will first try to call an internal Macro, then try to load an external file as specified by the Macro name, and last try to load an external file located in the *PWORKS_DATA* directory. The default file extension is '.mac'.

- ④ = MULTAX and CUTTER statements are stored exactly as they would appear in a part program. Note that the GOTO points following MULTAX/ON all have tool axis vectors associated with them.
- ⑤ = The DISPLY/text command is converted to a PPRINT command. INSERT, LETTER, PARTNO, PPRINT, and DSIPLY commands contain an ASCII text string instead of minor words and variables. A text command should always fit within 72 columns.
- ⑥ = All motion records, except GODLTA and circular records are stored as either a FROM or GOTO record. When a tool axis vector (MULTAX/ON) is stored with the motion point, the motion record will be longer than 72 columns.
- ⑦ = Circular records are stored with their canonical form: 'CIRCLE/x, y, z, i, j, k, r, t, t, f, d, e' along with requested circular interpolation tolerances of 'r,t' and the cutter diameter and corner radius of 'd,e'. Circular records are always followed by a single motion record which defines the ending location of the circular motion.
- ⑧ = GODLTA motion records are stored as delta distances from the previous position. When a tool axis vector (MULTAX/ON) is stored with the motion point, the motion record will be longer than 72 columns. This tool axis vector is an absolute position and not an incremental distance from the previous tool axis vector.
- ⑨ = MULTAX has been turned off. Note that the motion records following the MULTAX/OFF command do not have tool axis vectors associated with them.
- ⑩ = The last command in a cfile is always the FINI command.

G.5 MasterCam .NCI File G code format

Following shows the MasterCam .NCI file G codes supported by **PostWorks** and the corresponding **PostWorks** commands.

1. G-code: 0 or 1 (Linear/RAPID Move)

Format: **g**
 Par1, Par2, Par3, ..., Par10

Where: g = 0 means RAPID Move
 g = 1 means Linear Feed Move

Corresponding **PostWorks** Commands:

COOLNT
CUTCOM
RAPID or FEDRAT
GOTO

2. G-code: 2 or 3 (Circular Move)

Format: **g**
 Par1, Par2, Par3, ..., Par10

Where: g = 0 means Clockwise Circular Move
 g = 1 means Counter Clockwise Circular Move

Corresponding **PostWorks** Commands:

COOLNT
CUTCOM
RAPID or FEDRAT
COUPLE
GOTO

3. G-code: 4 (SPINDL, DWELL)

Format: **4**
 Par1, Par2, Par3

Corresponding **PostWorks** Commands:

DELAY
SPINDL

4. G-code: 11 (5-Axis Move)

Format: **11**
 Par1, Par2, Par3, ..., Par12

Corresponding **PostWorks** Commands:

MULTAX
COOLNT
CUTCOM
RAPID or FEDRAT
GOTO

5. G-code: 80 (Canned CYCLE Cancel)

Format: **80**
 Blank Line

Corresponding **PostWorks** Command:

CYCLE/OFF

6. G-code: 81 (Canned CYCLE Sart)

Format: **81**
 Par1, Par2, Par3, ..., Par19

Where: “Par1” specifies the type of canned cycles.

Corresponding **PostWorks** Commands:

CYCLE/DRILL	Par1 = 0, without DWELL
CYCLE/FACE	Par1 = 0, with DWELL
CYCLE/THRU	Par1 = 1
CYCLE/DEEP	Par1 = 2
CYCLE/TAP	Par1 = 3
CYCLE/REAM	Par1 = 4, without DWELL
CYCLE/BORE9	Par1 = 4, with DWELL
CYCLE/Bore	Par1 = 5
CYCLE/SHIFT	Par1 = 6
CYCLE/BORE8	Par1 = 7
CYCLE/BORE7	Par1 = 8
CYCLE/MILL	Par1 = 9
CYCLE/REVERS	Par1 = 10
MULTAX	
GOTO	

7. G-code: 100 (Canned CYCLE Repeat Position)

Format: **100**
 Par1, Par2, Par3, Par4, Par5

Corresponding **PostWorks** Commands:

CYCLE
GOTO

8. G-code: 1000 (Null Tool Change)

Format: **1000**
 Par1, Par2, Par3, ..., Par18

Corresponding **PostWorks** Commands:

SPINDL
COOLNT

9. G-code: 1001, 1002 (Tool Change)

Format: **g**
 Par1, Par2, Par3, ..., Par18

Where: g = 1001 means Start of the Tool change
 g = 1002 means Tool Change

Corresponding **PostWorks** Commands:

LOADTL
PPRINT TLN,
SPINDL
TOOLNO/ADJUST
COOLNT

10. Gcode: 1003 (End of File)

Format: **1003**
 Par1, Par2, Par3

Corresponding **PostWorks** Command.

FINI

11. Gcode: 1004 (Cancdl Cutter Compensation)

Format: **1003**
 Blank Line

Corresponding **PostWorks** Command:

CUTCOM/OFF

12. Gcode: 1005, 1006, 1007, 1008 (Comment)

Format: **g**
 Comment

Where: g = 1005 means Before.
 g = 1006 means After.
 g = 1007 means With.
 g = 1008 means Tool Operation Comment
 Comment = Text to be output

Corresponding **PostWorks** Command:

PPRINT

13. Gcode: 1027 (Working Coordinate System)

Format: **1027**
 Par1, Par2, Par3, ..., Par12

Corresponding **PostWorks** Command:

TRANS/mx

14. Gcode: 1050 (NCI Version Header)

Format: **1050**
 Par1, Par2, Par3, ..., Par12

Corresponding **PostWorks** Command:

REMARK/header

15. Gcode: 20001 (Tool Definition, Tool Name)

Format: **20001**
 Tool Name

This is output as part of tool change sequence as PPRINT TLN command.

16. Gcode: 20007 (Tool Definition, Holder Parameters)

Format: **20001**
 Par1, Par2, Par3, ..., Par11

Only Par3 (Overall Length) and Par7 (Holder length) will be used to calculate the tool set length.

G.1 Circular Interpolation Formats

As can be seen in the previous sections, circular interpolation records differ in most APT Source files created by various CAM systems. This section describes all of the formats of APT Source circular interpolation records which are supported by **PostWorks**.

```
PSIS/ (PLANE/i,j,k,d)
INDIRV/i1,j1,k1
TLON, GOFWD/ (CIRCLE/CANON,x,y,z, i,j,k, r), ON, [n,INTOF,]
          (PLANE/i2,j2,k2,d2)
```

This is the most common form of circular interpolation. It is formatted using the standard APT syntax and can be used with any APT system.

The floor or part surface is defined using the PSIS statement. This statement is not required by **PostWorks**.

The direction of cut is set with either an INDIRV/vector or an INDRIP/point statement.

The circle is finally driven using the TLON,GOFWD/dsf,ON,csf statement. In this example, the CIRCLE is the drive surface and the PLANE is the check surface. you can use either a CIRCLE or CYLNDR as the drive surface and either a PLANE or a LINE as the check surface.

The optional 'n,INTOF' clause allows a circle of more than 180 degrees to be driven. 'n' specifies the intersection of the drive and check surfaces to position at. The default is the first intersection when this clause is not specified.

```

PSIS/ (PLANE (POINT/x,y,z) , PERPTO, (VECTOR/i,j,k)
INDIRV/i1,j1,k1
TLON, GOFWD/ (CYLNDR/x,y,z, i,j,k, r) , ON,
            (PLANE/PERPTO, (PLANE/ (POINT/x,y,z,) ,
            PERPTO, (VECTOR/i,j,k) ) , (POINT/x,y,z) ,
            (POINT/x1,y1,z1))

```

This type of circular interpolation record is generated by CATIA Ver 4 and later.

The floor of the part surface is defined using the PSIS statement which is defined by defining a plane passing the circular motion center with a normal vector parallel to the CYLNDR's axis.

The direction of cut is set with the INDIRV/vector statement.

The circle is finally driven using the TLON,GOFWD/dsf,ON,csf statement. The drive surface is a CYLNDR entity with a radius of 'r', the center located at (x, y, z), and a circular plane vector (i, j, k). The check surface is defined as a plane normal to the circular plane and passing through two points with one point passing through the center of the circle.

```

AUTOPS
INDIRV/i,j,k
TLON, GOFWD/ (CIRCLE/x,y,z, r) , ON,
            (LINE/x1,y1,z1, x2,y2,z2)

```

This type of circular interpolation record is generated by CATIA Ver 4 and later.

The floor of the part surface is defined using the AUTOPS statement.

The direction of cut is set with the INDIRV/vector statement.

The circle is finally driven using the TLON,GOFWD/dsf,ON,csf statement. The drive surface is a CIRCLE with a radius of 'r', the center located at (x, y, z). The check surface is defined by a LINE with the two end points located at (x1,y1,z1) and (x2,y2,z2).

```

AUTOPS
INDIRV/i,j,k
TLON, GOFWD/ (CIRCLE/x,y,z, r) , TANTO,
            (LINE/x1,y1,z1, x2,y2,z2)

```

This type of circular interpolation record is generated by Computer Vision.

The floor of the part surface is defined using the AUTOPS statement.

The direction of cut is set with the INDIRV/vector statement.

The circle is finally driven using the TLON,GOFWD/dsf,TANTO,csf statement. The drive surface is a CIRCLE with a radius of 'r', the center located at (x, y, z). The check surface is defined by a LINE with the two end points located at (x1,y1,z1) and (x2,y2,z2).

```
N5 , CIRCLE/x,y,z, i,j,k, r
N6 , GOTO/x1,y1,z1, i,j,k
      .
      .
      .
GOTO/xn,yn,zn, i,j,k
```

This type of circular interpolation record consists of a CIRCLE record which contains the canonical form of the circle and a group of GOTO commands which contain iterations around the circle. The circular iterations continue until the next sequence number.

This circular interpolation format usually comes from the clfile editing routine **PTED**.

```
CIRCLE/x,y,z, i,j,k, r
GOTO/x1,y1,z1, i,j,k
      .
      .
      .
xn,yn,zn, i,j,k
```

This type of circular interpolation is similar to the previous format, except that the circular iteration points following the first point do not contain the GOTO major word. These iteration points continue until the next statement containing a major word is encountered.

```
ARCDAT/x,y,z, i,j,k, r
ARCMOV/CLW , x,y,z, ang
      CCLW
```

This is a more concise circular interpolation format. It contains the circle definition in the ARDCAT statement and the movement direction, ending position, and delta angular movement in the ARCMOV statement. It does not actually drive a circle nor does it contain a group of circular iteration points.

The VARIMETRIX CAM product generates this type of circular interpolation records.

Note: **PostWorks** will interpret the records as helical interpolation instead of circular interpolation if the following

ARCSLP/depth

statement is added before the *ARCDAT* and *ARCMOV* statements.

```
CW /i,j,k, xs,ys,zs, xc,yc,zc, xe,ye,ze, r
CCW
```

This circular interpolation record is generated by Solution 3000 and consists of the movement direction, circle axis, starting point, circle center, ending position, and circle radius. It does not actually drive a circle nor does it contain a group of circular iteration points.

```
$$ CIRCLE/x,y,z, i,j,k, r
GOTO/x1,y1,z1, i,j,k
.
.
.
GOTO/xn,yn,zn, i,j,k
```

This is a very basic type of circular interpolation and is usually generated by an APT system. The comment (\$\$) contains the canonical form of the circle and is followed by a series of GOTO points. **PostWorks** will analyze each of the following points to determine when the circular interpolation record ends, as the APT Source file does not contain any delimiter at the end of the circular record.

```
CIRCLE/x,y,z, i,j,k, r, t,f, d,e [,TIMES,n]
GOTO/x1,y1,z1, i,j,k
```

This type of circular interpolation record consists of a CIRCLE record which contains the canonical form of the circle (x, y ,z, i, j, k ,r), circular interpolation tolerances (r, t), and the cutter parameters (d, e). A single GOTO command which contains the final circular interpolation position follows the CIRCLE command. The direction of the normal vector (i ,j, k) determines the direction of the circular interpolation record.

“TIMES,n” are optional parameters specify the number of circular motion around the circle. The tool will travel around the circle “n-1” times plus the partial arc programmed by the actual moves. If the programmed arc is 360 degrees, then the tool will travel exactly “n” times around the circle.

Unigraphics II will typically generate this type of circular interpolation record.

```
CIRCLE/x,y,z, i,j,k, r
GOTO/x1,y1,z1, i,j,k
```

This type of circular interpolation record consists of a CIRCLE record which contains the canonical form of the circle (x, y ,z, i, j, k ,r). A single GOTO command which

contains the final circular interpolation position follows the CIRCLE command. The direction of the normal vector (i, j, k) determines the direction of the circular interpolation record.

This type of circular interpolation record is usually generated by Unigraphics.

```
MOVARC/x,y,z, i,j,k, r, ANGLE,ang  
GOTO/x1,y1,z1, i,j,k
```

This type of circular interpolation record consists of a CIRCLE record which contains the canonical form of the circle (x, y, z, i, j, k, r) and the delta angle “ang”. A single GOTO command which contains the final circular interpolation position follows the CIRCLE command. The direction of the normal vector (i, j, k) determines the direction of the circular interpolation record.

This type of circular interpolation record is usually generated by Unigraphics.

```
GOCLW /xc,yc,zc, i,j,k, r, xe,ye,ze  
GOCCLW
```

This circular interpolation record is generated by Gerber and consists of the movement direction, circle center, circle axis, circle radius and ending position. It does not actually drive a circle nor does it contain a group of circular iteration points.

G.2 Helical Interpolation Formats

Helical interpolation records differ in most APT Source files created by various CAM systems. This section describes all of the formats of APT Source circular interpolation records which are supported by **PostWorks**.

```
GOTO/x,y,z,i,j,k  
HELICAL/xc,yc,zc,ti,tj,tk,i,j,k,pitch,rad[,ang,hgt,n],xe,ye,ze
```

This type of helical interpolation record is generated by CATIA Ver 5.

“x, y, z”	Specifies the starting point of the helical motion.
“xc, yc, zc”	Specifies the helical circle center coordinates.
“ti, tj, tk”	Specifies the direction of the initial circular motion (tangent vector).
“i, j, k”	Specifies the circle axis vector
“pitch”	Specifies the pitch of the helix per 360 degrees.
“rad”	Specifies the radius of the circle.

“ang”	An optional parameter specifies the total angle of the circular move. “ang” will be greater than 360 degrees if more than a full circle is output.
“hgt”	An optional parameter specifies the helix height.
“n”	An optional parameter specifies the helix angle.
“xe, ye, ze”	Specifies the final position of the helical motion.

Note: The “ang, hgt, n” parameters are optional, but if one is specified all three must be specified.

```
GOTO/x1,y1,z1, i,j,k
CIRCLE/x,y,z, i,j,k, r, t,f, d,e [,TIMES,n]
GOTO/x2,y2,z2, i,j,k
```

Normally this set of input is considered as circular motion. However, if the GOTO point following the CIRCLE record is on a different planar level than the GOTO point preceding the circle (the start point), then the circular motion will be considered helical in nature. This type of helical interpolation record consists of a CIRCLE record which contains the canonical form of the circle (x, y, z, i, j, k, r), circular interpolation tolerances (r, t), and the cutter parameters (d, e). A single GOTO command which contains the start helical position preceding the CIRCLE command and a single GOTO command which contains the final helical interpolation position follows the CIRCLE command. The direction of the normal vector (i, j, k) determines the direction of the circular interpolation record.

“TIMES,n” are optional parameters specify the number of helical motion around the circle. The number of helix will be “n-1” times plus the partial arc programmed by the actual moves. If the programmed arc is 360 degrees, then there will be exactly “n” helix.

Unigraphics II will typically generate this type of helical interpolation record.

G.3 Special Considerations For APT Text Commands

PWorks allows a maximum of 132 characters (including the command word itself) in a text command (PARTNO, PPRINT, INSERT, LETTER). However, if the total number of characters exceeds the maximum input line length parameter “wid” specified by the **PWorks** run time option

```
-cl[file]:n[,wid]
```

the corresponding text command can be separated into two lines with the first line ending with a “\$” or “~” character at any column not to exceed “wid” and the rest at next line.

Example:

```
PPRINTThis PPRINT command is separated $  
into 2 lines
```

```
PPRINTThis PPRINT command is also separated ~  
into 2 lines
```

The PARTNO command can only output a maximum of 74 characters to the punch file unless output as a message block.

INDEX

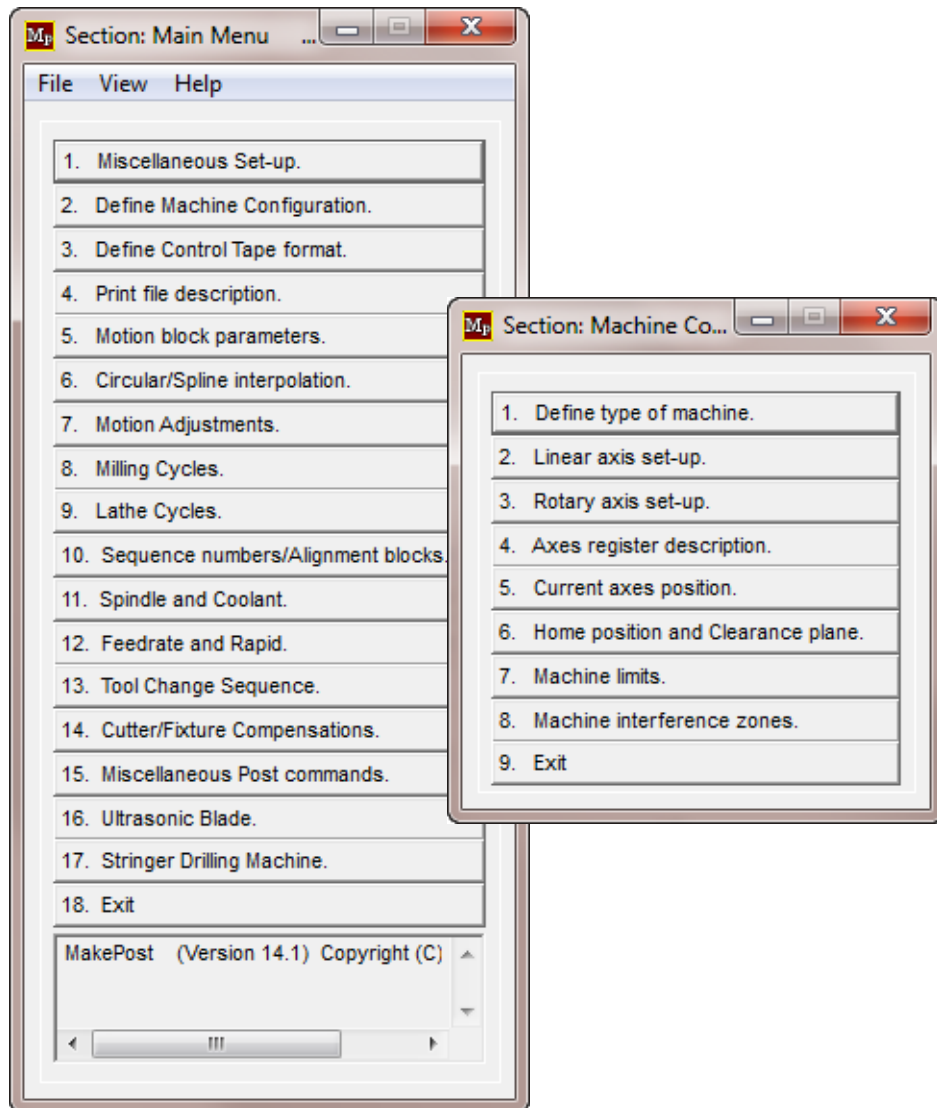
ADJUST	5-2
Alignment Block	E-1
APT Source File	4-5, E-1, G-1
APTERR	5-3
Array	E-1
Automatic Cycle	E-1
Automatic Documentation	B-11
Banner Line	3-4
Batch Process	4-8, 5-7
Block	E-1
Canned Cycle	E-1
Checksumming	E-1
Circular Interpolation	1-6, E-1
Clearance Plane	1-3
CLFILE	4-5, 5-3, B-6, E-2
Code	E-2
Consecutive Commas	5-8
Control Character	E-2
Control Tape	B-7, E-2
Custom Post-processor	4-4, 4-8, E-2
Cutter Compensation	1-12, E-2
Cylindrical Interpolation	E-2
Data File	B-2
Directory Structure	B-1
Documentation Files	B-4
Dwell	1-12
ERROR	5-3
Error Message File	B-10
Error Messages	5-3, B-10, D-1
FATAL	5-3
File Specification	2-2
FILL	5-8
Form	E-2
GenHelp	6-3, B-12
GenPrompt	6-2, B-10, B-11
GenWord	6-2, A-1, B-8
GenXfer	6-1, A-1, B-4
GOTO	D-9
Helical Interpolation	1-7, E-3
Help File	B-12

Home Position	1-3
IDENT	4-6, 5-4
Initialization File	B-3
Input File	E-3
Instruction	E-3
Interference Zone	E-3
Lathe Cycle	1-10
LATHE-2	1-3
LATHE-4	1-3
Linear Axes	1-3
Linear Axis	E-3
Linear Interpolation	E-3
Linearization	1-7, E-3
LISTING	4-2, 4-6, 5-5
Listing File	4-2, 4-6, B-6
LMDP Mode	E-3
LMFP Mode	E-3
Logical Register	3-3
MACHIN	1-16, 2-3, 4-7, 4-8, 5-1, 5-5
MACHINE	4-7, 5-5
Machine Adjust File	B-5, E-4
Machine Descriptor File	3-1, 5-1, B-3
Machine Limits	1-4
Machine Specific Post-processor	E-4
Macro	1-12, E-4
Macro Argument	1-13, E-4
Macro Definition File	4-1, 4-3, B-4, E-4
Macro Object File	4-1, 4-3, 4-4, 4-7, B-4, E-4
Macro Processor	4-1, 4-3, E-4
Major Word	C-1, E-5
makepost.ini	3-1
Manual Cycle	E-5
Menu	E-5
Menu/Prompt File	B-8
Menus	3-4
Mill Cycle	1-8
Mill/Turn Machine	E-5
MILL-3	1-2
MILL-4	1-2
MILL-5	1-2
MILL-7	1-2
Minor Word	C-3, E-5
Motion Block	E-5
MPost	1-1, 2-2, 3-1, A-1, B-3
MULTAX	G-4
Mutually Exclusive Axes	E-5

NCL	B-1
Number	E-5
OBJECT	4-3, 4-7
Operator	E-5
OPTION	5-5
Output File	E-6
PAGE_LEN	3-2, 4-3, 4-7, 5-6
Parameter	E-6
Part Program	E-6
PC	E-6
Physical Register	3-3
PMacro	1-15, 2-3, 4-4, A-1, A-2
pmacro.ini	4-5
Polar Interpolation	1-8, E-6
Post Variable	E-6
PostComp	2-3, 4-1, A-1, B-4
postcomp.ini	4-2
PostMacro	1-1, 1-12, 2-3, 4-1
Preset Axis Registers Block	E-6
PRINT	5-6
Print Descriptor File	5-6, B-5, F-1
Print File	1-16, 5-6, B-6
Prompt	E-7
PTED	G-1
PUNCH	5-6
Punch File	1-16, B-7
Punch Header File	B-5
PWorks	1-1, 1-15, 5-1, A-1, A-2, B-3
pworks.ini	5-1
QUIET	4-4, 4-8, 5-7
Rapid Motion	1-11
READ	G-5
Real Expression	E-7
Real Variable	E-7
Register	1-4, E-7
Register Labels	3-2
REMARK	4-2
Rotary Axes	1-3
Rotary Axis	E-7
Runtime License	2-1
Runtime Options	3-1, 4-2, 4-4, 5-1
SIMULATE	5-7
Simulation File	1-16, 5-7, B-7
Software Product Key	2-1
Spindle Speed Control	1-10
Spline Interpolation	1-7, E-7

Subscript	E-7
Tape Break	1-5, E-7
Text Expression	E-8
Text File	E-8
Text Post-processor Command	E-8
Text String	E-8
Text Variable	E-8
Tool Change	1-11
Tool Change Sequence	E-8
Ultrasonic Cutter	1-3, E-8
UNITS	3-2
User Defined Block	E-8
Vacuum Pods	E-9
Vocabulary File	B-7
Vocabulary Words	6-2, B-7
WARNING	5-3

MPost



Post-processor Generator Reference Manual

Warning and Disclaimer

Every effort has been made to make this document complete and as accurate as possible.
However, no warranty or fitness is implied.

The information is provided on an "as is" basis. Numerical Control Computer Sciences shall have neither liability nor responsibility to any person or entity with respect to any loss or damages in connection with or rising from the information contained in this document.

Copyright 1983-2014 by Numerical Control Computer Sciences
Irvine, California. Printed in the United States of America.
All rights reserved. The contents of this publication may not
be reproduced in any form or by any means, electronic
or mechanical, including photocopying, recording, or
information storage and retrieval systems, for any
purpose other than the licensee's personal use,
without prior written consent from
Numerical Control Computer Sciences.

TABLE OF CONTENTS

CHAPTER 1 Miscellaneous Setup1-1

1.0	Miscellaneous Setup	1-1
1.1	Post-processor Name	1-1
1.2	Machine Description	1-1
1.3	Input Units	1-1
1.4	Output Units	1-1
1.5	Enter Inch selection Code	1-1
1.6	Enter Millimeter Selection Code	1-2
1.7	Punch File Extension	1-2
1.8	Suppress Punch File When Error Level Is Encountered.	1-2

CHAPTER 2 Machine Configuration2-1

2.1	Define Type Of Machine	2-1
2.1.1	Type Of Machine	2-1
2.1.2	Does Milling Mode Support Polar Interpolation	2-1
2.1.3	Does Milling Mode Support Cylindrical Interpolation.	2-1
2.1.4	Is Cylindrical Interpolation Supported In User Defined Plane.	2-1
2.1.5	Is Cylindrical Interpolation Supported In Only A Single Plane	2-2
2.1.6:8	Enter Number Of Linear 'Axes'	2-2
2.1.9	Are Rotary Axes Output As Tool Axis Vector	2-2
2.1.10	Enter Number Of Rotary Axes	2-2
2.1.11	Enter Number Of Interference Zones	2-2
2.1.12	Enter Default Spindle Tool Axis	2-2
2.2	Linear Axis Set-up	2-3
2.2.1:3	Enter Calculation Type For Secondary 'Axis'	2-3
2.2.4:6	Enter Distance Between Primary And Secondary 'Axes'	2-3
2.2.7:9	Default 'Axis' At Start Of Program	2-3
2.3	Rotary Axis Set-up	2-4
2.3.1:4	Rotary Axis #n Is A Table/Head.	2-4
2.3.5:8	Rotary Axis #n Rotates Around Which Axis [X,Y,Z]	2-4
2.3.9	Does The Z-axis Rotates With The Rotary Head	2-4
2.3.10	Are The Rotary Tables Connected	2-4
2.3.11:14	Enter Number of Physical Rotary Axes #n.	2-5
2.3.15:18	Rotary Axis #n Is A Contouring/Positioning Axis	2-5
2.3.19:22	Rotary Axis #n Is Output On A Linear/Rotary Scale	2-5
2.3.23:26	Initial Orientation For Rotary Axis #n	2-6
2.3.27	Does Positioning Rotary Axis Have Feedrate Control	2-6
2.3.28:31	Enter Circumference Of Rotary Axis #n	2-6
2.3.32:35	A Positive Number Will Move Rotary Axis #n In A CLW/CCLW Direction.	2-6

2.3.36	Default Active Rotary Axes At Start Of Program (Max. 2)	2-7
2.3.37:40	Distance To Center Of Rotary Axis #n [X,Y,Z]	2-7
2.4	Axes Register Description	2-7
2.4.1:11	Register For Absolute Linear Axis	2-7
2.4.2:12	Register For Incremental Linear Axis	2-7
2.4.13:22	Register For Internal Rotary Axis #n	2-8
2.4.14:23	Register For Absolute Rotary Axis #n	2-8
2.4.15:24	Register For Incremental Rotary Axis #n	2-8
2.4.25:27	Register For Tool Axis Vector Component	2-8
2.4.28	Register For Absolute Polar Interpolation Y-axis	2-8
2.4.29	Register For Incremental Polar Interpolation Y-axis	2-9
2.4.30	Register For Internal Cylindrical Interpolation Rotary Axis	2-9
2.4.31	Register For Absolute Cylindrical Interpolation Rotary Axis	2-9
2.4.32	Register For Incremental Cylindrical Interpolation Rotary Axis	2-9
2.5	Current Axes Position	2-9
2.5.1:10	Current Position For 'Axis'	2-10
2.6	Home Position And Clearance Plane	2-10
2.6.1:10	'Axis' Home Position	2-10
2.6.11	Default Clearance Plane Mode	2-10
2.6.12	Default Clearance Plane	2-10
2.6.13	Retract Distance To Add To Current Tool Length	2-10
2.6.14	Retract Tool Tip Or Pivot Point To Clearance Plane	2-11
2.6.15	Enter Retract Direction Logic	2-11
2.7	Machine Limits	2-11
2.7.1:10	'Axis' Travel Limits [Min, Max]	2-11
2.7.11:14	Maximum Value For Rotary Axis #n	2-12
2.7.15:18	Code To Reset Rotary Axis #n When Register Limit Is Reached	2-12
2.8	Machine Interference Zones	2-12
2.8.1:40	'Axis' Interference Area For Zone 'n'	2-12
2.9	Exit	2-13

CHAPTER 3 Define Control Tape Format3-1

3.0	Walk Through Control Tape Format	3-1
3.1	Tape Register Format	3-1
3.2	Define G-code Groups	3-3
3.3	Define M-code Groups	3-3
3.4	Store Register Values	3-3
3.5	Define Tape Register Order	3-3
3.6	User Defined Tape Blocks	3-4
3.7	Define Tape Block Format	3-4
3.7.1	Multiple Non-conflicting G-codes Can Be In A Single Block	3-5
3.7.2	Multiple Non-conflicting M-codes Can Be In A Single Block	3-5
3.7.3	G-codes And M-codes Can Be In The Same Block	3-5
3.7.4	Word Address/Tab Sequential Format	3-5

3.7.5	End-of-block Character(s)	3-5
3.7.6	Start Of Message Character(s)	3-5
3.7.7	End Of Message Character(s)	3-6
3.7.8	Leader Character	3-6
3.7.9	Tab Character	3-6
3.7.10	Optional Skip Character	3-6
3.7.11	Default Mode For CHECK/LENGTH	3-6
3.7.12	Enter Checksum Register (CHECK/LENGTH)	3-6
3.7.13	Maximum Value For Checksum Register	3-6
3.8	Define Punch File Format	3-7
3.8.1	Punch File Is Packed/Unpacked	3-7
3.8.2	Maximum Length Of An MCD Block	3-7
3.8.3	PARTNO Card Output	3-7
3.8.4	Separate Tape Registers With A Space In Punch File	3-7
3.8.5	Separate Macro Parameters With A Space In Punch File	3-8
3.9	Define Start/End Sequences	3-8
3.9.1	Enter Punch File Header File (.phf) Number	3-8
3.9.2	Amount Of Leader At Start Of Punch File [Inch/Mm]	3-8
3.9.3	Amount Of Leader At End Of Punch File [Inch/Mm]	3-8
3.9.4	Rewind Stop Character(s)	3-8
3.9.5	Output An EOB With Rewind Stop Code	3-8
3.9.6	User Defined Starting Block	3-9
3.9.7	Code To Output With The REWIND Card	3-9
3.9.8	Code To Output With FINI Card	3-9
3.9.9	Start Of Tape Character(s)	3-9
3.9.10	End Of Tape Character(s)	3-9
3.10	Tape Break Sequences	3-9
3.10.1	Enter Default Mode For BREAK Command	3-9
3.10.2	Create A New Punch File At Tape Breaks	3-10
3.10.3	Retract The Tool At Tape Breaks	3-10
3.10.4	Issue TMARK After Tape Break	3-10
3.10.5	Default Tape Length For Tape Breaks	3-10
3.10.6	Default Machining Time For Tape Breaks	3-11
3.10.7	Default Delta Linear Distance For Tape Breaks	3-11
3.10.8	Default Retract Rate For Tape Breaks	3-11
3.10.9	Default Plunge Rate For Tape Breaks	3-11
3.11	Exit	3-11

CHAPTER 4 Print File Description4-1

4.0	Print File Description	4-1
4.1	Enter Default Print Descriptor File (.pdf) Name	4-1
4.2	Enter Page Header Record(s)	4-1
4.3	Enter Page Trailer Record(s)	4-1
4.4	Enter Motion Block Record(s)	4-1

4.5	Enter Non-motion Block Record(s)	4-1
4.6	Enter INSERT Record(s)	4-2
4.7	Enter STOP Record(s)	4-2
4.8	Enter OPSTOP Record(s)	4-2
4.9	Enter LOADTL Record(s)	4-2
4.10	Enter FINI Record(s)	4-2
4.11	Enter PPRINT Record(s)	4-2

CHAPTER 5 Motion Block Parameters5-1

5.0	Walk Through Motion Block Parameters	5-1
5.1	Special Event Handling	5-1
5.1.1	Specify Handling Of FROM Statement	5-1
5.1.2	The X-axis Is Input As Radius/Diameter	5-1
5.1.3	The X-axis Is Output As Radius/Diameter	5-1
5.1.4	Output The X-axis As Negative Positions	5-1
5.1.5	The Part Is Programmed In XY And Should Be Output In ZX	5-2
5.1.6	Default Positioning Mode [ABS/INCR]	5-2
5.1.7	FORCE Current Position When Changing From INCR To ABS Mode	5-2
5.1.8	FORCE Output Of All Axes When One Is Output	5-2
5.2	Rotary Motion	5-2
5.2.1	Use Tool Axis Vector To Control Rotary Axis Positions:	5-2
5.2.2	Adjust Input Points For Rotary Table(s)	5-3
5.2.3	Adjust Input Points For Rotary Head(s)	5-3
5.2.4	Force Output Of All Rotary Axes When One is Output	5-3
5.2.5:8	Absolute/Incremental Output For Rotary Axis #n	5-3
5.2.9	Enter Value For A CCLW Move To 0 Degree For The Rotary Axes ..	5-4
5.2.10	Enter Value For A CLW Move To 0 Degree For The Rotary Axes. ...	5-4
5.2.11	Enter Rotary Axes Size Change Code	5-4
5.2.12	Output Rotary Axes Size Change As	5-4
5.2.13:16	Enter Register For Rotary Axis #n Size Change	5-5
5.2.17:23	Enter CLW Direction Code For Rotary Axis #n	5-5
5.2.18:24	Enter CCLW Direction Code For Rotary Axis #n	5-5
5.3	Clamping Logic	5-5
5.3.1	Enter Clamping Logic For Rotary Axes	5-5
5.3.2:8	Enter Clamping Code For Rotary Axis #n	5-6
5.3.3:9	Enter Unclamping Code For Rotary Axis #n	5-6
5.3.10	Clamping Codes Should Be Output In A Block By Themselves	5-6
5.3.11:17	User Defined Block For Clamping Rotary Axis #n	5-6
5.3.12:18	User Defined Block For Unclamping Rotary Axis #n	5-6
5.4	Define Axes Tolerances	5-7
5.4.1:10	Minimum Linear Increment Allowed For 'Axis' Movement	5-7
5.4.11:20	Tolerance For Determining Output On 'Axis'	5-7
5.5	Default Output Axes Adjustments	5-7
5.5.1:10	Default TRANS/-AXIS For 'Axis'	5-7

5.5.11:20	Default TRANS/SCALE,-AXIS For 'Axis'	5-7
5.6	Motion Register Description.	5-8
5.6.1	Linear Motion Code	5-8
5.6.2	Linear Motion Code In LMFP Mode	5-8
5.6.3	Linear Motion Code In LMDP Mode	5-8
5.6.4	Absolute Positioning Mode	5-8
5.6.5	Incremental Positioning Mode	5-8
5.6.6	Code(s) For Setting Major Tool Axis To XYPLAN	5-8
5.6.7	Code(s) For Setting Major Tool Axis To ZXPLAN	5-9
5.6.8	Code(s) For Setting Major Tool Axis To YZPLAN	5-9
5.6.9	LMFP/LMDP Mode Minimum Span Register	5-9
5.6.10	Enable Mill/Turn Milling Mode Code	5-9
5.6.11	Disable Mill/Turn Milling Mode Code	5-9
5.6.12	Enter Mill/Turn Polar Interpolation Code	5-9
5.6.13	Enter Mill/Turn Cylindrical Interpolation Code	5-10
5.6.14	Register For Circle Unit In Cylindrical Interpolation	5-10
5.6.15	Is Cylinder Axis Code Required In LMDP Enable/Disable Block ...	5-10
5.6.16	Is Cylinder Axis Value Required In LMDP Enable/Disable Block ...	5-10
5.6.17	User Defined 1st Motion Block	5-10
5.6.18	User Defined Non-motion Block	5-10
5.7	Exit	5-10

CHAPTER 6 Circular/Spline Interpolation6-1

6.0	Walk Through Circular/Spline Interpolation	6-1
6.1	Circular Interpolation Format	6-1
6.1.1	Does The Machine Controller Support Circular Interpolation	6-1
6.1.2	Enter Format Of Circular Interpolation	6-1
6.1.3	Enter Format Of Circular Interpolation (LMFP/LMDP Mode)	6-2
6.1.4	Incremental Distance Should Be From Start Point To Center	6-2
6.1.5	The Center Point Should Be Output As Radius/Diameter	6-3
6.1.6	Circular Interpolation Is Allowed In All 3 Planes (XY, YZ, ZX)	6-3
6.1.7	Is 3-axis Circular Interpolation Supported	6-3
6.1.8	Output Final Point Of 3-axis Circular On Separate Block	6-3
6.1.9	Enter Format of 3-axis Circular Intermediate Point	6-3
6.1.10	Enter Planes Where Circular Interpolation Is Allowed (XY, YZ, ZX) ..	6-4
6.1.11	Circular Moves Should Be Broken Up	6-4
6.1.12	Force Position On Circular Interpolation Record	6-4
6.1.13	Force Direction Code On Circular Interpolation Record	6-4
6.1.14	Estimate Delta Distance During Circular Interpolation	6-4
6.2	Circular Interpolation Tolerances	6-5
6.2.1	Radius Tolerance For Circular Interpolation	6-5
6.2.2	Minimum Linear Movement To Output As Circular Interpolation	6-5
6.2.3	Minimum Radius Allowed For Circular Interpolation	6-5
6.2.4	Maximum Radius Allowed For Circular Interpolation	6-5

6.2.5	Adjust Final Point To Lie Exactly On The Circle	6-5
6.3	Helical Interpolation Format.	6-6
6.3.1	Does The Machine Controller Support Helical Interpolation.	6-6
6.3.2	Output Final Linear Depth With Helical	6-6
6.3.3	Helical Final Depth Is Output As	6-6
6.3.4	Output Delta Linear Movement With Helical	6-6
6.3.5	Tolerance For Generating Helical Movement	6-7
6.4	Circular Interpolation Registers	6-7
6.4.1	Polar Coordinate Linear Motion Code	6-7
6.4.2	Polar Coordinate Rapid Motion Code	6-7
6.4.3	Clockwise Circular Interpolation Code	6-7
6.4.4	Counter-Clockwise Circular Interpolation Code	6-7
6.4.5	Conversational Circle Center Code	6-7
6.4.6	Conversational Circular Record Code	6-7
6.4.7	Conversational Delta Angle Register	6-8
6.4.8	Clockwise Circular Interpolation Code In LMFP Mode	6-8
6.4.9	Counter-Clockwise Circular Interpolation Code In LMFP Mode	6-8
6.4.10	Clockwise Circular Interpolation Code in LMDP Mode	6-8
6.4.11	Counter-Clockwise Circular Interpolation Code In LMDP Mode	6-8
6.4.12	Clockwise Helical Interpolation Code	6-8
6.4.13	Counter-Clockwise Helical Interpolation Code	6-8
6.4.14	Clockwise 3-axis Circular Interpolation Code	6-8
6.4.15	Counter-Clockwise 3-axis Circular Interpolation Code	6-9
6.4.16	X-axis Circle Center Register	6-9
6.4.17	Y-axis Circle Center Register	6-9
6.4.18	Z-axis Circle Center Register	6-9
6.4.19	Circle Radius Register	6-9
6.4.20	X-axis Intermediate Point Register	6-9
6.4.21	Y-axis Intermediate Point Register	6-9
6.4.22	Z-axis Intermediate Point Register	6-10
6.4.23	Polar Coordinate Angle Register	6-10
6.4.24	Output X And Y Circle Center Registers In All Planes	6-10
6.4.25	XY-plane Selection Code	6-10
6.4.26	ZX-plane Selection Code	6-10
6.4.27	YZ-plane Selection Code	6-10
6.4.28	User Defined Plane Selection Code	6-10
6.4.29	Are Axis Code Values Required With Plane Selection	6-10
6.4.30	Output Dwell Code With Plane Selection	6-11
6.4.31	Amount Of Dwell To Output With Plane Selection Code	6-11
6.4.32	Output Plane Selection Code In Circular Block	6-11
6.5	Spline Interpolation Format	6-11
6.5.1	Does The Machine Controller Support Spline Interpolation	6-11
6.5.2	Enter Codes For Enabling High Precision Contouring Mode	6-11
6.5.3	Enter Codes For Disabling High Precision Contouring Mode	6-12
6.5.4	Enter The Tolerance For Fitting Spline Through Points	6-12
6.5.5	Enter Chordal Tolerance For Spline Interpolation	6-12

6.5.6	Enter Spline Interpolation Code	6-12
6.5.7	Force Linear/Circular Interpolation Code After Spline Mode	6-12
6.5.8	Output Spline Interpolation Code In A Block By Itself	6-13
6.5.9	Output Knot Values Or Node Distances	6-13
6.5.10	Enter Register For Knot Values/Node Distances	6-13
6.5.11	Include Start Point In Spline Interpolation	6-13
6.6	Exit	6-14

CHAPTER 7 Motion Adjustments7-1

7.0	Walk Through Motion Adjustments	7-1
7.1	Miscellaneous Adjustments	7-1
7.1.1	Maximum Number Of Axes Which Can Move In A Single Block	7-1
7.1.2	Priority For Output Axes	7-1
7.1.3	Number Of Mutually Exclusive Axes Sets	7-2
7.1.4:16	Enter nth Set Of Mutually Exclusive Axes	7-2
7.1.5:17	Controlling Axis For nth Set	7-2
7.1.6:18	Axes Priority With Positive Axis Move For nth Set	7-2
7.1.7:19	Axes Priority With Negative Axis Move For nth Set	7-3
7.1.20	Hold Back Motion Where Only 1 Axis Moves	7-3
7.2	Rotary Axes Linearization	7-3
7.2.1	Default Linearization Tolerance	7-4
7.2.2	Default Rapid Linearization Tolerance	7-4
7.2.3	Adjust Tool Axis Vector To Minimize Rotary Movement	7-4
7.2.4	Linear Tolerance To Use For Tool Axis Adjustment	7-5
7.2.5	Angular Cone To Consider Tool Axis Adjustment For	7-5
7.2.6	Minimum Angular Delta Movement To Consider For Tool Axis Adjustment	7-5
7.2.7	Default Tool Length To Consider For Tool Axis Adjustment	7-5
7.2.8	Tolerance For Truncating Tool Axis Vector Components	7-5
7.2.9	Retract Tool When Taking Longest Route	7-6
7.2.10	Allow Tool Retraction With Linearization Disabled	7-6
7.2.11	Add Tool Length To Retract Distance	7-6
7.2.12	Automatic Retract Distance (In Addition To Tool Length)	7-7
7.2.13	Automatic Retract Plane	7-7
7.2.14	Offset Tool Prior To Retract	7-7
7.2.15	Default Offset Distance	7-7
7.2.16	Retract Feedrate For Automatic Retract	7-7
7.2.17	Plunge Feedrate For Automatic Retract	7-8
7.2.18	Blade Positioning Feedrate For Automatic Retract	7-8
7.2.19	Offset Feedrate For Automatic Retract	7-8
7.2.20	Enter Logic To Use When Determining The Rotary Shortest Route . .	7-8
7.2.21	Controlling Axis For Shortest Route Determination	7-9
7.3	Rotary Axes Look-Ahead	7-9
7.3.1	Does This Machine Require Look-Ahead Positioning	7-9

7.3.2	Rotary Axes Look-Ahead Mode	7-10
7.3.3	Vector Component Deviation	7-10
7.3.4	Look-ahead For Limit Violations	7-10
7.3.5	Allow Rotation Of Axis In Single Block	7-10
7.3.6	Retract Tool When Rotated In Single Block	7-11
7.4	Slowdown Spans	7-11
7.4.1	Does The Control Support Slowdown Codes	7-11
7.4.2	Should Slowdown Codes Replace Linear/Circular Codes	7-11
7.4.3	Are Slowdown Codes Required On Every Active Motion Block	7-11
7.4.4	Number Of Slowdown Modes Supported	7-12
7.4.5	Minimum Angular Change To Generate Slowdown Span	7-12
7.4.6:9	Linear Slowdown Span Code #n	7-12
7.4.10:16	CLW Circular Slowdown Span Code #n	7-12
7.4.11:17	CCLW Circular Slowdown Span Code #n	7-12
7.4.18	Slowdown Off Code	7-13
7.4.19	Default Slowdown Tolerance	7-13
7.4.20	Default Slowdown Mode	7-13
7.5	Maximum Axis Departure	7-13
7.5.1	Apply Maximum Axis Departures During Rapid Moves	7-13
7.5.2:11	Maximum Axis Departure Allowed For 'Axis'	7-13
7.6	Acceleration Blocks	7-14
7.6.1	Does The Control Require Acceleration Spans	7-14
7.6.2	Default Mode For Acceleration Blocks	7-14
7.6.3	Maximum Vector Velocity	7-14
7.6.4	Capped Vector Velocity	7-14
7.6.5	Minimum Feed Rate Step	7-14
7.6.6	Maximum Feed Rate Step	7-15
7.7	Transformation Blocks	7-15
7.7.1	Does The Control Support Transformation Codes	7-15
7.7.2	Are Transformation Base On Active Transformations	7-15
7.7.3	Enter Transformation Rotation Output Logic	7-15
7.7.4	Use Table Rotations In Transformation Block Calculation	7-16
7.7.5	Use Output Rotary axis positions for coordinate transformations	7-16
7.7.6	Allow Rotary Axes Movement When A Transformation Block Is Active	7-16
7.7.7	Automatically Cancel Cutcom With Enable Transformations Block. .	7-17
7.7.8	Automatically Cancel Transformations Prior To Motion Block	7-17
7.8	Transformation Output	7-17
7.8.1	Enter Enable Translations Code	7-17
7.8.2	Enter Enable Rotations Code	7-17
7.8.3	Enter Disable Translations Code	7-17
7.8.4	Enter Disable Rotations Code	7-18
7.8.5	Enter X-Translation Register	7-18
7.8.6	Enter Y-Translation Register	7-18
7.8.7	Enter Z-Translation Register	7-18
7.8.8	Enter X-Rotation Register	7-18

7.8.9	Enter Y-Rotation Register	7-18
7.8.10	Enter Z-Rotation Register	7-18
7.8.11	Enter Rotation Vector I-Component Register	7-18
7.8.12	Enter Rotation Vector J-Component Register	7-18
7.8.13	Enter Rotation Vector K-Component Register	7-19
7.8.14	Enter Rotation Register	7-19
7.8.15	Enter Non-positioning Behavior Code	7-19
7.8.16	Output XYZ Translation Registers When Canceling Translations ...	7-19
7.8.17	Output Transformation Block In Relation To Current Motion Block .	7-19
7.9	Exit	7-20

CHAPTER 8 Milling Cycles8-1

8.0	Walk Through Milling Cycles	8-1
8.1	Cycle Definition Codes	8-1
8.1.1	Does This Machine Support Automatic Cycles	8-1
8.1.2	Automatic Cycles Are Allowed In All 3 Planes (XY,YZ, ZX)	8-1
8.1.3	Output Cycles As Macro Call	8-1
8.1.4	Macro Call Code	8-2
8.1.5	Enter CYCLE/OFF Code	8-2
8.1.6:18	Enter CYCLE/Mode Code	8-2
8.1.19	XY-plane Selection Code	8-2
8.1.20	ZX-plane Selection Code	8-2
8.1.21	YZ-plane Selection Code	8-2
8.1.22	Enter RAPID Cycle Interrupt Code	8-2
8.1.23	Enter RETRACT/ON Code	8-3
8.1.24	Enter RETRACT/OFF Code	8-3
8.2	Cycle Parameter Codes	8-3
8.2.1	Enter Final Depth Register	8-3
8.2.2	Enter Top-of-Part Register	8-4
8.2.3	Enter RAPTO Plane Register	8-4
8.2.4	Enter Threads Per Inch Register	8-5
8.2.5	Enter DWELL Parameter 1 Register	8-5
8.2.6	Enter DWELL Parameter 2 Register	8-6
8.2.7	Enter STEP Parameter 1 Register	8-6
8.2.8	Enter STEP Parameter 2 Register	8-6
8.2.9	Enter Single Parameter OFFSET Register	8-7
8.2.10	Enter X-axis OFFSET Register	8-7
8.2.11	Enter Y-axis OFFSET Register	8-8
8.2.12	Enter Z-axis OFFSET Register	8-8
8.2.13	Should XY Register Always Be Output As OFFSET Registers	8-8
8.2.14	Enter RTRCTO Parameter 1 Register	8-8
8.2.15	Enter RTRCTO Parameter 2 Register	8-8
8.2.16	Enter REPEAT Parameter Register	8-9
8.3	Cycle Positioning Parameters	8-9

8.3.1	Output A Unidirectional Move Prior To Each Cycle Point	8-9
8.3.2	Enter Positioning Direction Vector.	8-9
8.3.3	Enter Positioning Distance	8-10
8.3.4	Position Above Hole Prior To Issuing CYCLE/DEEP Block	8-10
8.3.5	Mode For Final Depth Value	8-10
8.3.6	The Final Depth Distance Is Relative To	8-10
8.3.7	Mode For Top-of-Part.	8-11
8.3.8	Output Format For STEP (Peck) Parameters	8-11
8.3.9	Default Rapid Rate For Cycles	8-11
8.4	Rapto And Retract Parameters	8-11
8.4.1	Mode For Rapto Plane Value	8-11
8.4.2	Enter Calculation Type For Rapto Plane	8-12
8.4.3	Enter Gage Height Difference Between Rapto Plane And Top Of Part	8-12
8.4.4	Mode For Retract Plane Value	8-12
8.4.5	Enter Calculation Type For Retract Plane	8-13
8.4.6	Enter The Final Retract Logic To Use	8-13
8.4.7	Should The 1st RTRCTO Value Override Normal Retract Logic	8-13
8.4.8	Output A Retract Level Code	8-13
8.4.9	Output A Retract Block When The Rapto Plane Increases	8-14
8.4.10	Enter The Format For A Retract Block	8-14
8.4.11	Retract Tool At Each Level Of CYCLE/THRU.	8-14
8.4.12	Enter Motion Alteration For Cycle Generated Rapid Moves.	8-15
8.4.13	Retract/Plunge Rapid Moves Along Tool Axis	8-15
8.5	Cycle Block Output	8-15
8.5.1	Force Cycle Definition Code On The Initial Cycle Block	8-15
8.5.2	Force Active Linear Axes On The Initial Cycle Block	8-15
8.5.3	Force Linear Axes Output On Every Cycle Block	8-16
8.5.4	Force Input Cycle Parameters On Every Cycle Block	8-16
8.5.5	Output A Feedrate With Cycle Blocks	8-16
8.5.6	Force Feedrate On The Initial Cycle Block	8-16
8.5.7	Should TPI Value Be Output As A -ve Number With CYCLE/REVERS.	8-16
8.5.8	Should A Full Cycle Block Be Output When Depth Changes	8-17
8.5.9	Automatically Lock Feedrate/Spindle Overrides With CYCLE/TAP.	8-17
8.5.10	Force Linear Interpolation Code After CYCEL/OFF.	8-17
8.6	Cycle User Defined Blocks.	8-17
8.6.1	CYCLE/OFF User Defined Block	8-17
8.6.2	CYCLE/Mode User Defined Block	8-17
8.7	Cycle Machining Times	8-19
8.7.1:13	Enter CYCLE/Mode Time Calculations.	8-19
8.7.14	Use The First STEP Parameter In Time Calculations	8-19
8.7.15	Use The Second STEP Parameter In Time Calculations	8-19
8.7.16	Estimate Delta Distances During Automatic Cycles	8-19
8.8	Exit	8-20

CHAPTER 9 Lathe Cycles9-1

9.0	Walk Through Lathe Cycles	9-1
9.1	Cycle Definition Codes	9-1
9.1.1	Does This Machine Support Automatic Cycles	9-1
9.1.2	Enter Constant Lead Thread Cutting Code	9-1
9.1.3	Enter Increasing Variable Lead Thread Cutting Code	9-1
9.1.4	Enter Decreasing Variable Lead Thread Cutting Code	9-1
9.1.5	Enter CYCLE/OFF Code	9-2
9.1.6:18	Enter CYCLE/Mode Code	9-2
9.1.7:19	Enter CYCLE/Mode With STEP Code	9-2
9.2	Cycle Parameter Codes	9-2
9.2.1	Enter Z-axis Lead Register	9-3
9.2.2	Enter X-axis Lead Register	9-3
9.2.3	Enter Increasing Variable Lead Register	9-3
9.2.4	Enter Decreasing Variable Lead Register	9-3
9.2.5	Enter Final Z-axis Position Register	9-3
9.2.6	Enter Final X-axis Position Register	9-3
9.2.7	Enter FEDTO Parameter Register	9-4
9.2.8	Enter OFFSET Parameter 1 Register	9-4
9.2.9	Enter OFFSET Parameter 2 Register	9-4
9.2.10	Enter RAPTO Parameter Register	9-4
9.2.11	Enter REPEAT Parameter Register	9-4
9.2.12	Enter RTRCTO Parameter 1 Register	9-4
9.2.13	Enter RTRCTO Parameter 2 Register	9-4
9.2.14	Enter STEP Parameter 1 Register	9-4
9.2.15	Enter STEP Parameter 2 Register	9-5
9.2.16	Enter TOOL Parameter 1 Register	9-5
9.2.17	Enter TOOL Parameter 2 Register	9-5
9.2.18	Enter Chamfer ON Code	9-5
9.2.19	Enter Chamfer OFF Code	9-5
9.3	Cycle Positioning Parameters	9-5
9.3.1	Position To Taper Position Prior To Issuing CYCLE/THREAD,STEP Block	9-5
9.3.2	Position To Taper Position Prior To Issuing CYCLE/TURN,STEP Block	9-6
9.3.3	Mode For Automatic Cycles	9-6
9.3.4	Output Format For STEP (Peck) Parameters	9-6
9.3.5	Default Rapid Rate For Cycles	9-6
9.4	Cycle Block Output	9-6
9.4.1	Force Cycle Definition Code On The Initial Cycle Block	9-7
9.4.2	Force Active Linear Axes On The Initial Cycle Block	9-7
9.4.3	Force Input Cycle Parameters On Every Cycle Block	9-7
9.4.4	Force Feedrate On The Initial Cycle Block	9-7
9.4.5	Force Linear Interpolation Code After CYCEL/OFF	9-7
9.4.6	Automatically Cancel Cycle After First Move	9-8

9.5	Cycle User Defined Blocks	9-8
9.5.1	CYCLE/OFF User Defined Block	9-8
9.5.2:15	CYCLE/Mode User Defined Block	9-8
9.6	Cycle Machining Times	9-9
9.6.1:14	Enter CYCLE/Mode Time Calculations	9-9
9.7	Exit	9-9

CHAPTER 10 Sequence Numbers/Alignment Blocks10-1

10.0	Sequence Numbers/Alignment Blocks	10-1
10.1	Output Sequence Numbers By Default	10-1
10.2	Register For Sequence Numbers	10-1
10.3	Beginning Sequence Number	10-1
10.4	Sequence Number Increment	10-1
10.5	Output Sequence Numbers Every nth Block	10-1
10.6	Maximum Sequence Number	10-1
10.7	Register For Alignment Block Number	10-1
10.8	User Defined Block For Non-motion Alignment Blocks	10-2
10.9	User Defined Block For Motion Alignment Blocks	10-2
10.10	User Defined Block For Automatic Cycle Blocks	10-2

CHAPTER 11 Spindle & Coolant11-1

11.0	Walk Through Spindle & Coolant Parameters	11-1
11.1	Spindle Set-up	11-1
11.1.1	Enter Spindle Speed Output Format	11-1
11.1.2	Does The Machine Controller Support SFM Spindle Control	11-1
11.1.3	Enter The Number Of Spindle Ranges [1-3]	11-1
11.1.4	Low Range Spindle Speed Limits [Min, Max]	11-2
11.1.5	Medium Range Spindle Speed Limits [Min, Max]	11-2
11.1.6	High Range Spindle Speed Limits [Min, Max]	11-2
11.1.7	SFM Spindle Speed Limits [Min, Max]	11-2
11.1.8	Tolerance For RPM Output When In SFM Mode	11-2
11.1.9	Low Range Spindle Speed Limits For Mill Head [Min, Max]	11-2
11.1.10	Medium Range Spindle Speed Limits For Mill Head [Min, Max]	11-3
11.1.11	High Range Spindle Speed Limits For Mill Head [Min, Max]	11-3
11.2	Spindle Registers	11-3
11.2.1	Spindle RPM Speed Register	11-3
11.2.2	Spindle SFM Speed Register	11-3
11.2.3	Spindle CLW Code	11-3
11.2.4	Spindle CCLW Code	11-3
11.2.5	Spindle OFF Code	11-3
11.2.6	Spindle ORIENT Code	11-4
11.2.7	Spindle BOTH Code	11-4
11.2.8	Low Range Spindle Code	11-4

11.2.9	Medium Range Spindle Code	11-4
11.2.10	High Range Spindle Code.	11-4
11.2.11	RPM Spindle Speed Control Code	11-4
11.2.12	SFM Spindle Speed Control Code	11-4
11.2.13	SFM Maximum Spindle Speed Control Code	11-5
11.2.14	SFM Spindle Radius Register.	11-5
11.2.15	Disable Spindle Speed Overrides Code	11-5
11.2.16	Enable Spindle Speed Overrides Code	11-5
11.2.17	User Defined Block For Spindle On Blocks.	11-5
11.2.18	Output Spindle Off Codes In A Block By Themselves	11-5
11.3:5	Spindle Speed Range Tables.	11-6
11.6	Coolant Set-up	11-6
11.6.1	Coolant Mist Code	11-6
11.6.2	Coolant Flood Code	11-6
11.6.3	Coolant Air Code	11-6
11.6.4	Coolant OFF Code	11-6
11.6.5	Combination Spindle CLW And Coolant Mist Code	11-7
11.6.6	Combination Spindle CCLW And Coolant Mist Code	11-7
11.6.7	Combination Spindle CLW And Coolant Flood Code	11-7
11.6.8	Combination Spindle CCLW And Coolant Flood Code	11-7
11.6.9	Combination Spindle CLW And Coolant Air Code	11-7
11.6.10	Combination Spindle CCLW And Coolant Air Code	11-7
11.6.11	Output Coolant On Codes In A Block By Themselves	11-8
11.6.12	Output Coolant Off Codes In A Block By Themselves	11-8
11.7	Exit	11-8

CHAPTER 12 Feedrate & Rapid12-1

12.0	Walk Through Feedrate & Rapid Parameters.	12-1
12.1	Miscellaneous Feedrate Set-up	12-1
12.1.1	Does The Machine Controller Support FPM Feedrates	12-1
12.1.2	Does The Machine Controller Support FPR Feedrates	12-1
12.1.3	Does The Machine Controller Support DPM Feedrates	12-1
12.1.4	Does The Machine Controller Support Inverse Time Feedrates	12-1
12.1.5	Default Feedrate Mode	12-1
12.1.6	Force Current Feed Output After FEDRAT Statement	12-2
12.1.7	Force Current Feed Output After Feedrate Mode Change	12-2
12.1.8	Feedrate Mode For Rotary Axes Movement	12-2
12.1.9	Output Average Rotary Axes Radius With Feedrate	12-3
12.1.10:13	Rotary Axis #n Radius Register	12-3
12.1.14	Disable Feedrate Overrides Code	12-3
12.1.15	Enable Feedrate Overrides Code	12-3
12.2	Feed Per Minute Feedrates	12-3
12.2.1	Output FPM Feedrates As FPR/INVERS.	12-3
12.2.2	Output FPM Feedrates Using A Predefined Table	12-4

12.2.3	Output FPM Feedrates Based On The Control Point	12-4
12.2.4	FPM Register	12-4
12.2.5	Set FPM Mode Register And Value	12-4
12.2.6:7	Pulse Weight For 'Type' Axes	12-4
12.2.8	Are Extended Precision FPM Feedrates Supported	12-4
12.2.9	Multiplication Constant For Extended Precision FPM Feedrates	12-5
12.2.10	Extended Precision FPM Register	12-5
12.2.11	Set Extended Precision FPM Mode Register And Value	12-5
12.3	Feed Per Revolution Feedrates	12-5
12.3.1	Output FPR Feedrates As FPM/INVERS.	12-5
12.3.2	Output FPR Feedrates Using A Predefined Table	12-5
12.3.3	FPR Register	12-5
12.3.4	Set FPR Mode Register And Value	12-6
12.3.5	Should FPR Feedrates Be Forced When The Z-axis Moves	12-6
12.3.6	Force FPR Feedrate When FPM Feedrate Is Less Than.	12-6
12.4	Degrees Per Minute Feedrates	12-6
12.4.1	DPM Register	12-6
12.4.2	Set DPM Mode Register And Value.	12-6
12.4.3	Should DPM Mode Register Override Linear Register	12-6
12.5	Inverse Time Feedrates.	12-7
12.5.1	How Should Inverse Time Feedrates Be Calculated	12-7
12.5.2	Inverse Time Feedrate Calculation For Circular Interpolation.	12-8
12.5.3	Enter Constant For Inverse Time Calculations.	12-8
12.5.4	Enter Command Pulse Weight For Inverse Time Calculations	12-8
12.5.5	Unit Of Time For Inverse Time Feedrates	12-8
12.5.6	Inverse Time Feedrate Register	12-8
12.5.7	Set Inverse Time Mode Register And Value	12-8
12.5.8	Are Extended Precision Inverse Time Feedrates Supported	12-9
12.5.9	Multiplication Constant For Extended Precision 1/T Feedrates.	12-9
12.5.10	Extended Precision Inverse Time Register.	12-9
12.5.11	Set Extended Precision Inverse Time Mode Register And Value	12-9
12.5.12	Enter Constant For Rotary Delta Movement In Inches	12-9
12.5.13	Enter Constant For Rotary Delta Movement In Millimeters	12-9
12.6:7	Feed Per 'Unit' Feedrate Table.	12-10
12.8	Rapid Set-up	12-10
12.8.1	Does The Machine Controller Support Rapid Mode	12-10
12.8.2	Enter Codes For Setting Rapid Mode.	12-10
12.8.3	Does The Machine Controller Support Rapid Mode In LMDP Mode.	12-10
12.8.4	Enter Code For Setting Rapid In LMDP Mode	12-11
12.8.5	Does The Machine Controller Support Rapid Mode In LMDP Mode.	12-11
12.8.6	Enter Code For Setting Rapid In LMDP Mode	12-11
12.8.7	Output Linear Interpolation Code On Rapid Block	12-11
12.8.8	Reset The Programmed Feedrate After A Rapid Move	12-11
12.8.9	Reset The Feedrate Mode After A Rapid Move.	12-11

12.8.10	Enter Motion Alteration For Rapid Moves	12-11
12.8.11	Retract/Plunge Rapid Moves Along Tool Axis	12-12
12.8.12	Retract To Clearance Plane On Rapid Moves	12-12
12.8.13	Mode For Rapid Moves	12-12
12.8.14	Modify Rapid Moves To Complete Fastest Axis First During Simulation.	12-13
12.9	Output Feedrate Limits	12-13
12.9.1:6	'Mode' Feedrate Limits [Min, Max].	12-13
12.9.7	Enter Unit Scale Factor For Feedrate Limit Values	12-13
12.10	Machine Feedrate Limits	12-13
12.10.1:10	Maximum 'Axis' Feedrate	12-14
12.10.11:20	'Axis' Rapid Rate	12-14
12.10.21	Minimum Amount Of Time Allowed Per Move In Seconds	12-14
12.10.22	Adjust Output Feedrates For Maximum Axes Feedrates	12-14
12.11	Accel/Decl Time Adjustments	12-15
12.11.1	Adjust Machining Time For Acceleration/Deceleration Spans	12-15
12.11.2:11	Maximum Velocity For 'Axis'	12-15
12.12	Exit	12-15

CHAPTER 13 Tool Change Sequence13-1

13.0	Walk Through Tool Change Sequence	13-1
13.1	Tool Change Support	13-1
13.1.1	Maximum Tool Number Allowed	13-1
13.1.2	Does The Machine Support 'Select Tool' Sequence	13-1
13.1.3	Does The Machine Have Both A SMALL And LARGE Gripper	13-1
13.1.4	Does The Machine Have Both A MAIN And AXIAL Spindle	13-1
13.1.5	Support SELECT/TOOL And LOAD/TOOL Commands	13-2
13.1.6	Number Of Turrets	13-2
13.1.7	Distance Between Turrets	13-2
13.1.8	Default Turret At Beginning Of Program	13-2
13.2	Loading The Tool	13-2
13.2.1	Should The Tool To Unload Be Output With A Tool Change	13-2
13.2.2	Should An Unload Code Be Output For A Tool Change	13-3
13.2.3	Output Tool Number With A Tool Unload Block	13-3
13.2.4	Output An Alignment Block On Tool Change Blocks	13-3
13.2.5	Output An Alignment Block After A Tool Change Block	13-4
13.2.6	Turn Off Tool Length Compensation Prior To Tool Change	13-4
13.2.7	Turn On Tool Length Compensation After A Tool Change	13-4
13.2.8	Is A Tool Change Considered An End-of-Sequence	13-4
13.2.9	Look Ahead For LOADTL When PPRINT Encountered	13-4
13.2.10	Calculate Separate Times For Each Instance The Same Tool Is Loaded.	13-5
13.2.11	Amount Of Time For Tool Change Sequence In Seconds	13-5
13.3	Tool Change Codes	13-5

13.3.1	Pick Up Tool (With Small Gripper) Code	13-5
13.3.2	Pick Up Tool With Large Gripper Code.	13-5
13.3.3	Select Tool Register	13-6
13.3.4	Tool Number Register	13-6
13.3.5	Output The Tool Register On A Block By Itself	13-6
13.3.6	Tool Change Code	13-6
13.3.7	Manual Tool Change Code.	13-7
13.3.8	Unload Tool (With Small Gripper) From Main Spindle Code.	13-7
13.3.9	Unload Tool (With Small Gripper) From Axial Spindle Code	13-7
13.3.10	Unload Tool With Large Gripper From Main Spindle Code	13-7
13.3.11	Unload Tool With Large Gripper From Axial Spindle Code	13-7
13.3.12	Force Turret CLW Code.	13-7
13.3.13	Force Turret CCLW Code	13-8
13.3.14	User Defined Block For Tool Changes.	13-8
13.4	Tool Length Offsets	13-8
13.4.1	Add Tool Lengths To Output Axes	13-8
13.4.2	Register For Tool Length Offsets.	13-8
13.4.3	Base Number For Tool Length Offsets.	13-9
13.4.4	Code For Enabling Tool Length Offsets.	13-9
13.4.5	Code For Disabling Tool Length Offsets	13-9
13.4.6	Code For PLUS Tool Length Offsets	13-9
13.4.7	Code For MINUS Tool Length Offsets	13-9
13.4.8	Output MINUS Offsets As Negative Values	13-9
13.4.9:11	'Axis' Tool Offset Register	13-10
13.4.12	Output Plane Selection Code With Tool Length Offset Register.	13-10
13.4.13	Output Tool Length Offset On Codes In A Block By Themselves	13-10
13.4.14	Output Tool Length Offset Off Codes In A Block By Themselves	13-10
13.4.15	User Defined Block For Tool Length Offsets	13-10
13.5	Exit	13-11

CHAPTER 14 Cutter/Fixture Compensation14-1

14.0	Walk Through Cutter/Fixture Compensation	14-1
14.1	Cutter Compensation Support.	14-1
14.1.1	Cutter Compensation Output Format	14-1
14.1.2	Unitize Cutcom Vector	14-2
14.1.3	Is Cutter Compensation Allowed In All 3 Planes.	14-2
14.1.4	Is 3-dimensional Cutter Compensation Supported	14-2
14.1.5	Is 5-axis Cutter Compensation Allowed.	14-2
14.1.6	Are Approach/Departure Codes Supported	14-2
14.1.7	Maximum Angular Change When Cutcom Is Active.	14-2
14.1.8	Maximum Vector Component Value Allowed.	14-3
14.2	Cutter Compensation Control	14-3
14.2.1	Output Cutcom On Codes In A Block By Themselves	14-3
14.2.2	Output Cutcom Off Code In A Block By Itself	14-3

14.2.3	Output Directional Vector On Initial Cutcom On Block	14-3
14.2.4	Output Directional Vector On Circular Block	14-3
14.2.5	Output Directional Vector Between Multiple Circular Blocks	14-3
14.2.6	Output An Offset Register With A Value Of '0' With Cutcom Off. . .	14-4
14.2.7	Output An Error Message With Move In Non-Cutcom Axis.	14-4
14.3	Cutter Compensation Codes	14-4
14.3.1	Cutcom LEFT Code	14-4
14.3.2	Cutcom RIGHT Code	14-4
14.3.3	Cutcom OFF Code	14-4
14.3.4	Cutter Compensation Offset Register	14-5
14.3.5	Base Number Of Cutter Compensation Register	14-5
14.3.6	Cutcom Vector Definition Code	14-5
14.3.7:9	'Axis' Vector Component Register	14-5
14.3.10	Cutcom Angular Direction Register	14-5
14.3.11	CLW Circular Cutcom Direction Code	14-6
14.3.12	CCLW Circular Cutcom Direction Code	14-6
14.3.13:15	Approach/Departure Code #n	14-6
14.3.16	Approach/Departure Distance Register	14-6
14.3.17	3-D Cutter Compensation Code	14-6
14.3.18	Tool Radius Register	14-6
14.3.19	Tool Corner Radius Register	14-7
14.3.20	Rotary Axis Position Mode Register	14-7
14.3.21	Maximum Feedrate Register	14-7
14.4	Fixture Offsets	14-7
14.4.1	Register For Fixture Offsets	14-7
14.4.2	Base Number For Fixture Offsets	14-7
14.4.3	Code For Enabling Fixture Offsets	14-8
14.4.4	Code For Disabling Fixture Offsets	14-8
14.4.5	Code For PLUS Fixture Offsets	14-8
14.4.6	Code For MINUS Fixture Offsets	14-8
14.4.7	Output MINUS Offsets As Negative Values	14-8
14.4.8:10	'Axis' Fixture Offset Register	14-8
14.4.11	Output Fixture Offset ON Codes In A Block By Themselves	14-9
14.4.12	Output Fixture Offset OFF Codes In A Block By Themselves	14-9
14.5	Exit	14-9

CHAPTER 15 Miscellaneous Post Commands15-1

15.0	Walk Through Misc. Post Commands	15-1
15.1	Messages And INSERT	15-1
15.1.1	Add Sequence Numbers To Message Blocks	15-1
15.1.2	Add Sequence Numbers To INSERT Blocks	15-1
15.1.3	Output An End-of-Block On Message Blocks	15-1
15.1.4	Output An End-of-Block On INSERT Blocks	15-1
15.1.5	Convert Message Spaces To Tabs	15-1

15.1.6	Align Message Blocks At Column	15-2
15.1.7	Process PPRINT IPV Statements Normally	15-2
15.2	OPSKIP, OPSTOP, And STOP	15-2
15.2.1	Enter Code For OPSTOP	15-2
15.2.2	Enter Code For STOP	15-2
15.2.3	User Defined Block After OPSKIP/OFF	15-3
15.2.4	User Defined Block After OPSTOP	15-3
15.2.5	User Defined Block After STOP	15-3
15.2.6	Should OPSTOP Be Considered An End-of-Sequence	15-3
15.2.7	Should STOP Be Considered An End-of-Sequence	15-3
15.3	DELAY	15-3
15.3.1	Enter Delay Definition Code	15-3
15.3.2	Enter Delay Time Register	15-4
15.3.3	Enter Spindle Revolution Delay Register	15-4
15.3.4	Enter Output Format For Time Delay Blocks	15-4
15.3.5	Unit To Output For Time Delays	15-4
15.3.6	Enter Time Interval In Seconds For Each Delay Block	15-5
15.3.7	Does The X-axis Need To Be Output In A Delay Block	15-5
15.3.8	Reset The Programmed Feed Rate After A Delay Block	15-5
15.4	GOHOME	15-5
15.4.1	Enter Code For GOHOME/CHECK	15-5
15.4.2	Enter Code For GOHOME/AUTO	15-5
15.4.3	Enter Code For GOHOME/FROM	15-5
15.4.4	Enter Code For GOHOME/NEXT	15-6
15.4.5:14	Enter Register For GOHOME/'Axis',n	15-6
15.5	POSTN	15-6
15.5.1	Enter Register For POSTN Program Number	15-6
15.5.2	Enter Code For POSTN/NORMAL	15-6
15.5.3	Enter Code For POSTN/INCR	15-6
15.5.4	Enter Code For POSTN/OFF	15-7
15.5.5	Output POSTN/OFF Code In A Block By Itself	15-7
15.5.6:15	Enter Register For POSTN/'Axis',n	15-7
15.6	Exit	15-7

CHAPTER 16 Ultrasonic Blade Configuration.....16-1

16.0	Walk Through Ultrasonic Blade Configuration	16-1
16.1	Ultrasonic Blade Cutter Set-up	16-1
16.1.1	Enter Default Blade Direction Vector	16-1
16.1.2	Register For Internal Blade Rotary Axis	16-1
16.1.3	Register For Absolute Blade Rotary Axis	16-1
16.1.4	Register For Incremental Blade Rotary Axis	16-1
16.1.5	Blade Rotary Axis Is Output On A Linear/Rotary Scale	16-2
16.1.6	Current Position For Blade Rotary Axis	16-2
16.1.7	Maximal Change Of Blade Rotary Axis	16-2

16.1.8	Blade Rotary Axis Travel Limits [Min, Max]	16-2
16.2	Ultrasonic Blade Alignment	16-2
16.2.1	Automatic Blade Alignment Mode	16-3
16.2.2	Minimum Span To Apply Align Alteration	16-3
16.2.3	Minimum Blade Angle Which Will Have Align Alteration	16-3
16.2.4	Span Distance At Which To Align Blade	16-3
16.2.5	Blade Alignment Direction	16-3
16.3	Ultrasonic Blade Positioning	16-4
16.3.1	Blade Alignment Positioning Mode	16-4
16.3.2	Distance Above Location To Align Blade	16-4
16.3.3	Enter Feed Rate To Position The Tool	16-4
16.3.4	Enter Reed Rate To Plunge The Tool	16-4
16.4	Smooth Cutting Command Set-up	16-5
16.4.1	Smooth ON Code	16-5
16.4.2	Smooth OFF Code	16-5
16.4.3	Enable Curve Blending Codes	16-5
16.4.4	Disable Curve Blending Codes	16-5
16.4.5	Curve Blending Limit Angle Register	16-6
16.4.6	Maximal Curve Blending Direction Change	16-6
16.4.7	Initial Span For Initiating Curve Blending Mode	16-6
16.5	Vacuum Pods	16-6
16.5.1	Pod Vacuum Pump OFF Code	16-7
16.5.2	Pod Vacuum Pump ON Code	16-7
16.5.3	Pod Vacuum OFF Code	16-7
16.5.4	Disable Pod Vacuum OFF Code	16-7
16.5.5	All Pods Down Code	16-7
16.5.6	All Pods Low Pressure Code	16-7
16.5.7	All Pods High Pressure Code	16-8
16.5.8:9	Pod Row/Column Address Registers	16-8
16.5.10	Address Single Pod Code	16-8
16.5.11	Addressed Pod Unclamp Code	16-8
16.5.12	Addressed Pod Clamp Code	16-8
16.5.13	Addressed Pod No Pressure Code	16-8
16.5.14	Addressed Pop Up Code	16-8
16.5.15	Addressed Pod Up/Clamp Code	16-9
16.6	Exit	16-9

CHAPTER 17 Stringer Drilling Machine17-1

17.0	Walk Through Ultrasonic Blade Configuration	17-1
17.1	Stringer Drill Set-up	17-1
17.1.1	Enter Active Head At Start Of Program	17-1
17.1.2	Enter Offset Distance For Head 2 Calculations	17-1
17.1.3	Enter Pivot Offset Distance For Head 3 And 4 Calculations	17-1
17.2	Axes Register Description	17-1

17.2.1	Define registers for Head #	17-1
17.2.2	Register for X-axis	17-1
17.2.3	Register for Y-axis	17-2
17.2.4	Register for Z-axis	17-2
17.2.5	Register for Rotary axis #1	17-2
17.2.6	Register for Rotary axis #2	17-2
17.2.7	Register for Rotary axis #3	17-2
17.3	Stringer Codes.	17-2
17.3.1	User Defined Block For Head #1 Activation	17-2
17.3.2	User Defined Block For Head #2 Activation	17-2
17.3.3	User Defined Block For Head #3 Activation	17-3
17.3.4	User Defined Block For Head #4 Activation	17-3
17.3.5	User Defined Block For Head #1 Home Move	17-3
17.3.6	Code For Clip holder Slide Home Position	17-3
17.3.7	Spindle ON Code For Drilling Head	17-3
17.3.8	Spindle OFF Code For Drilling Head	17-3
17.3.9	Coolant ON Code For Drilling Head	17-3
17.3.10	Coolant OFF Code For Drilling Head	17-3
17.3.11	Code For Activating The Milling Head	17-4
17.3.12	Code For Activating The Drilling Heads	17-4
17.4	Current Axes Position	17-4
17.4.1	Current Axes Positions For Head #	17-4
17.4.2	Current Position For X-axis	17-4
17.4.3	Current Position For Y-axis	17-4
17.4.4	Current Position For Z-axis	17-4
17.4.5	Current Position For Rotary Axis #1	17-4
17.4.6	Current Position For Rotary Axis #2	17-4
17.4.7	Current Position For Rotary Axis #3	17-5
17.5	Home Position.	17-5
17.5.1	Home Positions For Head #	17-5
17.5.2	X-axis Home Position	17-5
17.5.3	Y-axis Home Position	17-5
17.5.4	Z-axis Home Position	17-5
17.5.5	Rotary Axis #1 Home Position	17-5
17.5.6	Rotary Axis #2 Home Position	17-5
17.5.7	Rotary Axis #3 Home Position	17-5
17.6	Machine Limits	17-6
17.6.1	Machine limits for Head #	17-6
17.6.2	X-axis Travel Limits [Min, Max]	17-6
17.6.3	Y-axis Travel Limits [Min, Max]	17-6
17.6.4	Z-axis Travel Limits [Min, Max]	17-6
17.6.5	Rotary Axis #1 Travel Limits [Min, Max]	17-6
17.6.6	Rotary Axis #2 Travel Limits [Min, Max]	17-6
17.6.7	Rotary Axis #3 Travel Limits [Min, Max]	17-6
17.7	Exit	17-6

APPENDIX A Print Descriptor File..... A-1

A.1	Introduction.....	A-1
A.2	Record Definitions	A-1
A.2.1	Line Definitions	A-1
A.2.2	Text Data.....	A-2
A.2.3	Variable Data	A-2
A.3	Example Print Descriptor File	A-4

APPENDIX B Forms..... B-1

B.1	Introduction.....	B-1
B.2	Machine Description Form.....	B-1
B.3	Axis Considerations	B-3
B.4	Interference Zones	B-5
B.5	Register Format.....	B-6
B.6	G/M-Codes	B-7
B.7	Register Defaults.....	B-8
B.8	Spindle / Feedrate Tables	B-8
B.9	User Defined Blocks.....	B-9
B.10	Cycles	B-9
B.11	Print Descriptor Records.....	B-10

APPENDIX C Automatic Documentation File..... C-1

APPENDIX D Punch Header File..... D-1

D.1	Introduction.....	D-1
D.2	The Header Section.....	D-1
D.3	The Trailer Section	D-1
D.4	Data Lines.....	D-1
D.5	Looping Region	D-2
D.6	Example Punch Header File	D-2

INDEX..... 1

CHAPTER 1 Miscellaneous Setup

1.0 Miscellaneous Setup

This section is used to load a [Machine Descriptor File](#) (*.MDF*), set the current machine number/name and to setup the default units.

1.1 Post-processor Name

Enter the identifying name for the current *MDF* file. All settings will be saved to '*PWORKS_name.MDF*' when exiting the **MPost** facility. If you load an *MDF* file and make changes to it in order to generate another machine description, you must enter a new name here, otherwise you will overwrite the *MDF* file you originally loaded. The post-processor name can be up to 40 characters long.

1.2 Machine Description

Enter the textual description for this machine configuration. The machine description can be up to 80 characters long.

1.3 Input Units

Enter the default clfile input units, either INCH or MM, that the current post configuration will expect as input. This setting has no effect on the units used as input to the **MPost** facility. To change these units use the [UNITS](#) switch on the input command line or in the initialization file (*makepost.ini*). The default input units can be changed with the [MACHIN/...,OPTION,1,n](#) command in the part program.

1.4 Output Units

Enter the default units, either INCH or MM, that the current post configuration will output to the Control Tape. The default output units can be changed with the [MACHIN/...,OPTION,2,n](#) command in the part program.

1.5 Enter Inch selection Code

Enter the code to output to the Control Tape when Inch output has been selected. Enter the code in the form 'r(v)'. Example: G(70). This code will be output at the beginning of the tape. This code may be omitted by leaving this entry blank.

1.6 Enter Millimeter Selection Code

Enter the code to output to the Control Tape when Millimeter output has been selected. Enter the code in the form 'r(v)'. Example: G(71). This code will be output at the beginning of the tape. This code may be omitted by leaving this entry blank.

1.7 Punch File Extension

Enter the default file extension for punch files generated using this Machine Descriptor File. The extension entered here will be the default extension instead of the normal default extension '.pul'. It can be overridden by using the '-PUNCH:.ext' runtime switch in either the *pworks.ini* file, on the command line or the **PWorks** utility interface. Be sure the '.' character is included in the specification.

The command line or the **PWorks** utility interface run time switch has the highest priority in specifying the default punch file extension. The 'pworks.ini' file has the second highest priority and this **MPost** menu setting has the lowest priority in specifying the output file extension.

1.8 Suppress Punch File When Error Level Is Encountered

Typically **PWorks** will output a Punch file regardless of how many warnings, errors, or fatals are encountered. You can change this behavior and suppress the output of the Punch file whenever a warning, error, or fatal is encountered. If the Punch file output is suppressed, then a Punch file will not be created. The only exception being if multiple Punch files are being output, then Punch files completed prior to encountering an error will still be output.

The following options can be selected.

- "NO" will always create the Punch file output when Punch file output has been enabled.
- "WARNING" will suppress the Punch file output whenever a warning, error, or fatal is encountered.
- "ERROR" will suppress the Punch file output whenever an error or fatal is encountered.
- "FATAL" will suppress the Punch file output whenever a fatal is encountered.

CHAPTER 2 Machine Configuration

This selection will lead you through all of the aspects involved in defining the machine configuration, including type of machine, linear axes configuration, rotary axes configuration, axes travel limits, etc.

2.1 Define Type Of Machine

This section defines the type of machine for this post.

2.1.1 Type Of Machine

Enter the number that corresponds to the type of machine that you wish to support.

- 1 = Mill
- 2 = Lathe
- 3 = Ultrasonic Cutter
- 4 = Mill/Turn
- 5 = Stringer Drilling Machine

2.1.2 Does Milling Mode Support Polar Interpolation

Some Mill/Turn machines support a mode where the rotary axis (spindle) takes over as the Y-axis while cutting the face of the part. The rotary axis is usually output using rectangular coordinates, with the control calculating the actual rotary angle. Enter "YES" if your machine supports this feature.

2.1.3 Does Milling Mode Support Cylindrical Interpolation

Cylindrical interpolation is supported for Mill/Turn machines only, and allows a round part to be programmed in a flat plane. **PostWorks** will wrap the flat part onto a cylinder and output the correct linear and rotary positions to cut the part. One of the advantages of cylindrical interpolation is that the control supports circular interpolation on the round part during this mode.

Enter YES if your machine supports this feature.

2.1.4 Is Cylindrical Interpolation Supported In User Defined Plane

Enter YES if the machine supports cylindrical interpolation in planes other than the standard lathe plane (rotation about the Z-axis). Most machines turn the part around the Z-axis and do not support cylindrical interpolation in any other plane.

2.1.5 Is Cylindrical Interpolation Supported In Only A Single Plane

Most Mill/Turn machines have only one rotary table, which allows for cylindrical interpolation in only a single plane. Enter YES at this prompt if this describes your machine. If the machine has multiple rotary tables (spindles), which rotate in different planes, then enter NO.

2.1.6:8 Enter Number Of Linear 'Axes'

Enter the number of linear axes that this machine has which move along the specified axis. You will be prompted for each of the linear (XYZ) axes. Each major linear axis can support up to 2 axes, a Primary and Secondary axis.

2.1.9 Are Rotary Axes Output As Tool Axis Vector

Enter YES if the tool axis vector should be output directly to the Control Tape instead of the rotary axes positions. In this case, it is assumed that the machine controller is intelligent and will handle the conversion of the tool axis vector to the actual rotary axis positions. The typical answer is NO, which allows **PostWorks** to calculate and output all rotary axes. In this case the user must specify the position, orientation, and all other parameters controlling the configuration and output of the rotary axes.

2.1.10 Enter Number Of Rotary Axes

Enter the number of rotary axes that this machine has. You can enter from 0 to 4 rotary axes, though a maximum of two rotary axes may be active at any one time. The remainder rotary axes must be locked, though they may be locked in any position.

2.1.11 Enter Number Of Interference Zones

Enter the number of zones that will cause the spindle and machine to collide. The axes position will be checked for entry to these zones and **PostWorks** will output a warning message anytime an interference zone is breached. You can define up to 4 interference zones. The actual interference zones are defined in the "[Machine interference zones](#)" section.

2.1.12 Enter Default Spindle Tool Axis

Enter a vector that represents the primary (spindle) tool axis when all rotary axes are at 0 degrees. The primary tool axis will be used internally by **PostWorks** to calculate the rotary angles, perform cycle operations, adjust for the programmed tool length, etc.

A spindle tool axis vector of 0,0,1 is normally used, instances where you may want to specify a different vector is when attaching a 90 degree head to the spindle or when manually rotating the spindle.

The *MODE/TOOL* command can be used to change the primary tool axis from within the part program.

2.2 Linear Axis Set-up

This section defines the attributes of the primary and secondary linear axes (X, Y, Z, U, V, W). This section will be disabled unless a secondary axis exists for at least one of the primary axis.

2.2.1:3 Enter Calculation Type For Secondary 'Axis'

Enter the number that corresponds to the calculation required for generating the secondary linear axis. This prompt will only be displayed when 2 linear axes of this type are supported.

1. Secondary_n = Input_n - Fixed_distance - Primary_n
2. Secondary_n = Input_n - Previous_Input_n
3. Secondary_n = Input_n

Calculation type 1 is usually when the primary and secondary axes are not connected, for example a quill and saddle.

Calculation type 2 is usually used when the secondary axes moves with the primary axis, for example a boring attachment loaded in the spindle.

Calculation type 3 is used when the machine control is intelligent enough to calculate the primary and secondary axis positions based on an input position relative to the part.

2.2.4:6 Enter Distance Between Primary And Secondary 'Axes'

Enter the distance between the primary and secondary linear axis when both axes are at 0. This distance is used as the 'Fixed_distance' when 2 linear axes of the same type are supported and calculation type 1, from the previous prompt, was selected. This prompt will only be displayed when calculation type 1 was chosen for the specified axis.

2.2.7:9 Default 'Axis' At Start Of Program

Enter which of the linear axis should be active at the start of the program, by default, either PRIMARY, SECONDRY, or BOTH. The active axis can be changed at any time during the part program by using the *MODE/-AXIS* command. This prompt will only be displayed when 2 linear axes of the same type are supported.

BOTH selects both linear axes to be output by default and is usually only selected when calculation type 3 was chosen for this axis.

2.3 Rotary Axis Set-up

This section defines the attributes of the rotary axes. Whether it is a rotary table or head, the location of the rotary axis on the machine, if it moves on a linear or rotary scale, etc. This section will be disabled if there are no rotary axes.

2.3.1:4 Rotary Axis #n Is A Table/Head

Specify whether the rotary axis is a rotary table (TABLE) or if the spindle carrier rotates (HEAD). Rotary axis definition must be in the following order.

1. Table riders.
2. Table carriers.
3. Head carriers.
4. Head riders.

A carrier axis is defined as a rotary axis that has another rotary axis mounted on it. The mounted rotary axis will change position, relative to the machine, as the carrier axis rotates. A rider axis is a rotary axis that is mounted on the carrier axis. The rotation of the rider axis does not affect that of the carrier axis.

Given the above set of rules, if the machine has 2 rotary axes, a table and a head, then TABLE should be entered as Rotary #1, and HEAD for Rotary axis #2. The manner in which the rotary axes are referenced throughout **PostWorks** will be exactly as stated above.

This prompt will only be displayed when at least one rotary axis has been specified.

2.3.5:8 Rotary Axis #n Rotates Around Which Axis [X,Y,Z]

Enter the primary linear axis, either X, Y, or Z that this rotary axis rotates about when all rotary axes are at 0 degrees. Keep in mind the order in which rotary axes are defined. This prompt will only be displayed when at least one rotary axis has been specified.

2.3.9 Does The Z-axis Rotates With The Rotary Head

Enter YES if the Z-axis is mounted on the rotary head and its direction changes when the head rotates. This prompt will only be displayed when at least one of the rotary axes is defined as a HEAD.

2.3.10 Are The Rotary Tables Connected

Enter YES if the rotary tables are connected in a carrier/rider type configuration, where one table is mounted on top of the other. This is the most typical configuration.

Enter NO if the rotary tables are mounted on the machine separately. The rotation of either rotary axes will not affect the position of the other. This configuration is not common and usually entails having separate parts mounted on each rotary table.

2.3.11:14 Enter Number of Physical Rotary Axes #n

Some multi-head or multi-table machines require that the rotary axes associated with each head or table be addressed separately, even if the position will always be the same.

Entering a value greater than one at this prompt will cause the post to output multiple registers for this particular rotary axis to control the separate rotary axes associated with each head or table. The same register will be used to output each axis, but with a different number designation in the register's label.

The starting characters for the rotary axis register must contain a '#' character in order to take advantage of this feature. If it does not, then only a single register without any modification will be output with this rotary axis (the same as when there are no duplicate axes).

Example:

Register: C#=

Output: C1=90. C2=90. etc.

The '#' character will be replaced with the number of the rotary axis that is active. Up to nine duplicate axes for each rotary axis are allowed. Enter a value of one if the machine does not have duplicate axes for this rotary axis.

By default all duplicate rotary axes are disabled, except for the first one (1). Use the *ROTABL/ON* command to enable multiple duplicate axes and *ROTABL/OFF* to disable the axes.

2.3.15:18 Rotary Axis #n Is A Contouring/Positioning Axis

Enter CONTOUR if this rotary axis is a contouring axis, meaning that it can move in conjunction with the linear axes. If the rotary axis can only be moved by itself or will not maintain a constant velocity in conjunction with the other axes at the programmed feed rate, then enter POSITION. This prompt will only be displayed when at least one rotary axis has been specified.

2.3.19:22 Rotary Axis #n Is Output On A Linear/Rotary Scale

The rotary axis moves on a linear scale when a value of 720 degrees will move the axis 2 revolutions from 0 degrees. A rotary scale will always be output between 0 and 360 degrees, usually with the sign (+-) of the value determining the direction. Enter either LINEAR or ROTARY. This prompt will only be displayed when Rotary axis #n is a contouring axis.

2.3.23:26 Initial Orientation For Rotary Axis #n

If this rotary axis rotates about a vector other than one of the major axes (X,Y,Z), then you can specify its initial orientation here. Most specifications will define the orientation of this type of axis referencing angles from a major axis, therefore this prompt requests that the initial orientation be specified as angle rotation(s) around major axes in the following format.

X30,Z45

The above specification will position the defined axis at 30 degrees about the X-axis and 45 degrees about the Z-axis. The order of the rotation axes is important and follows the same rules as when defining the physical rotary axes (table riders, table carriers, head riders, head carriers). You do not have to specify two rotations for the initial orientation, if the physical axis is only rotated around one major axes, then only this axis has to be specified. For example, for a nutating head the answer could be as follows.

X40

2.3.27 Does Positioning Rotary Axis Have Feedrate Control

Enter YES if the positioning rotary axis can be output with a feed rate to control its speed. If the rotary axis rotation speed cannot be controlled and will always move at one speed, then enter NO. This prompt will only be displayed when at least one of the rotary axes supported is defined as a positioning only axis.

2.3.28:31 Enter Circumference Of Rotary Axis #n

Enter the value that will cause one full rotation for this rotary axis. This will usually 360.0 for rotary axes that are output using degrees. Some machines, for example the Bostomatic, output the rotary angle as a circumference instead of in degrees. The value may be different in this case. This prompt will only be displayed when Rotary axis #n moves on a linear scale

2.3.32:35 A Positive Number Will Move Rotary Axis #n In A CLW/CCLW Direction

Enter the direction that this rotary axis will move when a positive number is output, either clockwise (CLW) or counter-clockwise (CCLW). A positive number will move most rotaries in the clockwise direction. This prompt will only be displayed when at least one rotary axis has been specified.

2.3.36 Default Active Rotary Axes At Start Of Program (Max. 2)

This prompt is displayed only when 3 or more rotary axes are supported. Although **PostWorks** can supported up to 4 rotary axes, only 2 can be active at any one time. Enter which rotary axes (1-4) are active, by default, at the beginning of the program. You can enter 1 or 2 axes separated by a comma. Example, 1, 2. The active rotary axes can be changed at any place within the part program by using the [MODE/AXIS](#) command.

2.3.37:40 Distance To Center Of Rotary Axis #n [X,Y,Z]

Enter the distance from the programmed 0,0,0 location to the center of this rotary axis when all linear and rotary axes are at zero. The center of all rotary axes are required by **PostWorks** when calculating the machine coordinate points from the programmed cl points. Enter the X, Y and Z coordinates separated by commas, for example, 0,10.5,0.

The following rules must be taken into consideration when 2 or more rotary heads are configured.

1. When multiple rotary heads share the same pivot point, then the rotary center must be defined for head rider.
2. The head carrier rotation center point should be defined as a distance from the head rider center point.

You can use the [ROTABL/ORIGIN](#) command to change these values at any place within the part program. This prompt will only be displayed when at least one rotary axis has been specified.

2.4 Axes Register Description

This section defines the registers to use for each of the output linear and rotary axes.

2.4.1:11 Register For Absolute Linear Axis

Enter the register descriptor for the absolute linear axis position, for example X1, X2, etc. You must enter a register even if the machine only supports incremental mode, as this register is used internally by **PostWorks** in calculating the incremental value for the linear axis. The X1, U1, Y1, V1, Z1 and W1 registers have initially been reserved for the absolute linear axes.

2.4.2:12 Register For Incremental Linear Axis

Enter the register descriptor to use for incremental positioning of the linear axis, for example X1, X2, etc. You do not need to enter a register if the machine does not support incremental mode. The X2, U2, Y2, V2, Z2 and W2 registers have initially been reserved for the incremental registers.

DO NOT enter the same register for absolute and incremental positions, as **PostWorks** uses the absolute positioning register to calculate the incremental move.

2.4.13:22 Register For Internal Rotary Axis #n

Enter the register descriptor which will be used internally by **PostWorks** to keep track of the absolute position of this rotary axis. This register will not be output and is used by **PostWorks** to keep track of the rotary axis on a linear scale of 360 degrees, no matter how the axis is actually output.

This register must be unique (not used as an output register for any other address). The A1, B1, C1 and C4 registers have initially been reserved as the internal rotary axis registers.

2.4.14:23 Register For Absolute Rotary Axis #n

Enter the register descriptor for the absolute rotary axis, for example, A2, B2, etc. You must enter a register even if the machine only supports incremental mode, as this register is used internally by **PostWorks** in calculating the incremental value for the rotary axis. The A2, B2, C2 and C5 registers have initially been reserved as the absolute rotary axis registers.

2.4.15:24 Register For Incremental Rotary Axis #n

Enter the register descriptor to use for incremental positioning of this rotary axis, for example A3, B3, etc. You do not need to enter a register if the machine does not support incremental mode. The A3, B3, C3, and C6 registers have initially been reserved for the incremental rotary axis registers.

DO NOT enter the same register for absolute and incremental positions, as **PostWorks** uses the absolute positioning register to calculate the incremental move.

2.4.25:27 Register For Tool Axis Vector Component

Enter the register descriptor for each of the tool axis components, for example I2. The I2, J2, and K2 registers have initially been reserved for these registers. These prompts are only displayed when the tool axis is output instead of the actual rotary axis positions.

2.4.28 Register For Absolute Polar Interpolation Y-axis

Enter the register descriptor for the absolute Y-axis position when in polar interpolation. This register will contain the Y-axis rectangular position which actually controls the rotary axis during polar interpolating on a Mill/Turn type machine.

This prompt will only be displayed when the machine type is set to a Mill/Turn and polar interpolation is supported.

2.4.29 Register For Incremental Polar Interpolation Y-axis

Enter the register descriptor to use for incremental positioning of the polar interpolation Y-axis. You do not need to enter a register if the machine does not support incremental mode.

DO NOT enter the same register for absolute and incremental positions, as **PostWorks** uses the absolute positioning register to calculate the incremental move.

This prompt will only be displayed when the machine type is set to a Mill/Turn and polar interpolation is supported.

2.4.30 Register For Internal Cylindrical Interpolation Rotary Axis

Enter the register descriptor which will be used internally by **PostWorks** to keep track of the absolute position of the cylindrical interpolation rotary axis. This register will not be output and is used by **PostWorks** to keep track of the rotary axis on a linear scale of 360 degrees, no matter how the axis is actually output.

This register must be unique (not used as an output register for any other address). The A1, B1, C1 and C4 registers have initially been reserved as the internal rotary axis registers.

This prompt will only be displayed when the machine type is set to a Mill/Turn and cylindrical interpolation is supported.

2.4.31 Register For Absolute Cylindrical Interpolation Rotary Axis

Enter the register descriptor for the absolute cylindrical interpolation rotary axis. The A2, B2, C2 and C5 registers have initially been reserved as the absolute rotary axis.

2.4.32 Register For Incremental Cylindrical Interpolation Rotary Axis

Enter the register descriptor to use for incremental positioning of this rotary axis. The A3, B3, C3 and C6 registers have initially been reserved as the incremental rotary axis registers.

DO NOT enter the same register for absolute and incremental positions, as **PostWorks** uses the absolute positioning register to calculate the incremental move.

2.5 Current Axes Position

This section defines the default position at the beginning of the program for each of the linear and rotary axes.

2.5.1:10 Current Position For 'Axis'

Enter the default position for the requested axis at the start of the program. This location is input as the actual machine position and not the tool tip position.

2.6 Home Position And Clearance Plane

This section defines the machine Home position, for use by the *GOHOME* command and the default clearance plane, for use by the *RETRCT* command.

2.6.1:10 'Axis' Home Position

Enter the machine's home (reference point) position for the specified axis. The *GOHOME* command can be used to position the machine to its home position. This location is input as the actual machine position, not as the tool end point position.

2.6.11 Default Clearance Plane Mode

Enter either 'NORMAL' or 'TOOL'. 'NORMAL' sets the clearance plane type to be an actual plane, the tool will be retracted along the tool axis vector to this plane when a *RETRCT* command is issued. 'TOOL' specifies that the tool will retract a defined distance from its current position when a *RETRCT* command is issued.

2.6.12 Default Clearance Plane

Enter the default clearance plane to use at the beginning of the program. The clearance plane should be entered using the format 'i, j, k, d' or just 'd'. If only a single number is entered, 'd', then a plane vector of '0, 0, 1' will be used.

This prompt will only be displayed when the default clearance mode was set to 'NORMAL'. You can use the *CLRSRF/NORMAL* command to change the clearance plane anywhere within the part program.

2.6.13 Retract Distance To Add To Current Tool Length

Enter the distance along the current tool axis vector in addition to the tool length to be used at the beginning of the program for the default retract distance. The tool will retract this distance plus the currently programmed tool length whenever a *RETRCT* command is issued.

This prompt will only be displayed when the default clearance mode was set to 'TOOL'. You can use the *CLRSRF/TOOL* command to change the clearance distance anywhere within the part program.

2.6.14 Retract Tool Tip Or Pivot Point To Clearance Plane

Enter the tool/spindle location which will move to the clearance plane when a *RETRACT* command is issued. You can enter either 'TOOL' or 'PIVOT'. 'TOOL' will position the tool end point at the defined clearance plane. 'PIVOT' will move the tool stop or rotary head pivot point, whichever applies, to the clearance plane. This setting will become the default retract reference location and can be changed using the *RETRACT/(PAST,TO)* command.

2.6.15 Enter Retract Direction Logic

Enter the value that corresponds to the way you want retract moves output to the Control Tape by default.

1. Retract tool along the X-axis.
2. Retract tool along the Y-axis.
3. Retract tool along the Z-axis.
4. Retract tool along the current tool axis vector.

Selecting option 1, 2, or 3 results in the tool retracting along the specified axis (X, Y, or Z), without regards to the tool axis vector. Option 4 will retract the tool along the current spindle vector.

Regardless of the retract logic chosen here, the tool will always retract in relationship to the machine axes configuration. This means that any adjustments to the linear axes for table rotations will be made first. These adjusted locations will be used as a basis for the retraction direction and locations.

When the tool retracts along the tool axis (4), the tool spindle vector (the portion of the tool axis vector remaining after being adjusted for table rotations) will be used.

2.7 Machine Limits

This section defines the axes travel limits of the machine for all of the supported axes.

2.7.1:10 'Axis' Travel Limits [Min, Max]

Enter the minimum and maximum travel limits, separated by a comma, for the specified axis. These limits can be changed using the *CHECK/-AXIS* command.

PWorks will try to generate a solution which will satisfy the specified travel limits. If no such solution can be found, **PWorks** will output an "Travel Limits Exceeded" error message.

PWorks determines whether to output "Travel Limits Exceeded" error messages by comparing the specified travel limits with the *internally calculated axes position* (The internally calculated

positions might not be the same as the values output to the punch file). **PWorks** will compare the specified travel limits with the absolute output registers value for the linear axes and compare the internal registers values for the rotary axes. It must be noted that the absolute output registers values for linear axes and the internal register values for rotary axes can be modified by the *TRANS/-AXIS* and *TRANS/SCALE,-AXIS* commands.

2.7.11:14 Maximum Value For Rotary Axis #n

Enter the maximum value that can be output for Rotary Axis #n. This value is different than the axis limit in that it only controls the maximum value that the axis register can contain and is not an actual machine limit. The post-processor will actually output a code or an orientation block whenever this value is exceeded, so that the machine controller will reset this axis to be at a position close to zero degrees, while the axis itself will not physically move.

This prompt/feature is only valid for rotary axes that are output on a linear scale.

2.7.15:18 Code To Reset Rotary Axis #n When Register Limit Is Reached

Enter the register and its value that will be output to reset this rotary axis back/close to zero degrees when the maximum output value is reached.

Entering a register without any value will output this register with the reset position of the rotary axis as its value. Leaving this prompt blank will output an orientation block (POSTN). Specifying both a register and its value will output this register in a block by itself.

This prompt/feature is only valid for rotary axes that are output on a linear scale.

2.8 Machine Interference Zones

This section defines spindle and fixture collision zones on the machine. Up to 4 collision zones can be defined. The number of collision zones is defined in the Type of machine section.

2.8.1:40 'Axis' Interference Area For Zone 'n'

Enter the lower and upper limit that will define the specified axis collision area for this interference zone. This axis will be considered inside the interference zone when its location is equal to or greater than the lower limit and less than or equal to the higher limit. All axes defined as part of the interference zone must be within their limits for the interference zone to be considered breached.

You may enter either a lower and upper limit, separated by a comma, when an area of the specified axis is part of the interference zone, or you may enter a single number, if the axis must be at this location to be in the interference zone, then leave this prompt blank.

These interference zones can be changed or defined by using the *CHECK/OUT* command.

2.9 Exit

Exits the Machine configuration section and returns you to the previous section.

CHAPTER 3 Define Control Tape Format

3.0 Walk Through Control Tape Format

This selection will lead you through the Control Tape characteristics, including register descriptions, G-code and M-code Groups, User Defined Tape Blocks and the Punch File Format.

3.1 Tape Register Format

The Tape register format form enables you to modify the format of each of the 92 registers. You can change the beginning characters, floating point format, sign output, the minimum number of digits to output and the ending characters for each register.

The registers are assigned a function throughout the rest of the **MPost** routine. Following is an explanation and list of acceptable input for each of the different headings in the Tape register format.

Reg

This field cannot be modified and contains the designator for each register as it should be referenced throughout the **MPost** routine.

Begin

Defines the starting characters for each register, which will be placed prior to the register value when formatting it for output. You may define anywhere from 0 to 24 characters. On most machines this will consist of a single letter (X, F, G, M, etc.).

If the starting characters of the register address must be followed by a space preceding the register value, then use the backslash character '\ ' to denote this space. Spaces in the midst of the starting characters can simply be represented with a space ' ' character. To place a backslash character at the end of the starting characters you must use a double backslash '\\'. this rule holds true for the ending characters of the register.

Format

Defines the floating point format for each of the registers. Valid formats are as follows:

LEADING	=	Leading zero suppression.
TRAILING	=	Trailing zero suppression.
FLOATING	=	Number contains a decimal point.
DECIMAL	=	Number contains a decimal point, except whole numbers which will drop the decimal point.

Sign

Controls the output of the sign (+-) character. Valid control characters are as follows:

(Blank)	=	Negative numbers will not contain a sign.
+	=	Positive numbers will contain a “+” sign.
-	=	Negative numbers will contain a “-” sign.
*	=	Negative numbers and zero will contain a “-” sign.
PN	=	Positive numbers will contain “P(“ as the sign. Negative numbers will contain “N(“ as the sign.

When “PN” is specified, be sure to add a “)” as the ending character for this register.

Left

Enter the number of digits to the left of the (implied) decimal point, for both Inch and Metric modes.

Right

Enter the number of digits to the right of the (implied) decimal point, for both Inch and Metric modes.

Min Left

Enter the minimum number of digits to the left of the (implied) decimal point that the output number must contain.

Min Right

Enter the minimum number of digits to the right of the (implied) decimal point that the output number must contain.

End

Defines the ending characters for each register, which will be placed after the register value when formatting it for output. You may define anywhere from 0 to 24 characters. On most machines this will consist of no characters.

Control

The control entry determines when this particular register will be output to the Control Tape. Following is a list of acceptable input to this field.

NOT_USED	=	This register will never be output.
ALWAYS	=	This register will be output on every block.

SET	=	This register will be output whenever it is internally set by PWorks .
CHANGED	=	This register will be output whenever its value changes.
NONZERO	=	This register will be output whenever it has value other than zero.
MOTION	=	This register will only be output on motion blocks when its value has changed.

3.2 Define G-code Groups

The G-code Groups form allows you to define up to 10 G-code groups, each with up to 14 values. These groups are used by **PostWorks** to determine which G-codes cancel other G-codes.

Each G-code group should contain values, without the G designator, which cancel each other on the machine, though Group 0 is usually used for non-modal G-codes, not G-codes which cancel one another. **PostWorks** also uses these groups to determine when to output certain G-codes.

3.3 Define M-code Groups

The M-code Groups form allows you to define up to 10 M-code groups, each with up to 7 values. these groups are used by **PostWorks** to determine which M-codes cancel other M-codes.

Each M-code group should contain values, without the M designator, which cancel each other on the machine, though Group 0 is usually used for non-modal M-codes, not M-codes which cancel one another. **PostWorks** also uses these groups to determine when to output certain M-codes.

3.4 Store Register Values

The Preset Registers form allows you to define the default value for each register. The register designator (A1, B2, C3,etc.) appears along with a default value. You can only change the value, not the designator.

3.5 Define Tape Register Order

The Register Order Form allows you to define the order in which each register appears in the output Control Tape block. The register order is listed from left to right on each line. You only need to define the registers that are supported with the current machine configuration. Any registers that are used but do not appear in this form will be appended to the end of the block.

The register order can be dynamically changed by using the **PostMacro** command “**REGORD**”.

3.6 User Defined Tape Blocks

The User Defined Blocks form contains the definitions for up to 20 User Defined Tape Blocks, each including up to 20 registers. These block definitions are referenced in other sections of the **MPost** routine, such as the [beginning tape block](#), the [first motion block](#), [non-motion blocks](#), the first block after a [STOP](#), [OPSTOP](#) and [OPSKIP/OFF](#) command, etc. They can also be output using the [FORCE/BLOCK](#) command within a post-processor *Macro*.

For each block, enter a list of registers with an optional value separated by commas. These registers will output whenever this block is forced out during the part program. You can specify both physical (G1, X1, etc.) and logical registers (X, F).

When specifying a value with a register, the value must be enclosed in parenthesis, G(90). You can also specify the \$ character to signify an End-of-block. The registers to the left of the \$ will be output in a single block and the registers to the right will be output in the following block.

Example:

G(17),G(9),X1,Y1,Z1,F,M1,\$,G(00),Z2

A “-” sign can be inserted in front of any register in the list. This minus sign denotes that the register(s) referred to will not be output with this tape block definition.

User defined blocks for macro calls, such as cycles on a Siemens 840D controller, are defined by a sequence of numbers that represent the order of the macro parameters in the call block. All parameter values defined in a macro call will automatically be output, since the parameters are position dependant. Any parameters that are not specified in the User Defined Block order will be output in their numeric order after the parameters defined in the User Defined Block are output. You cannot mix registers with numeric macro parameters in a single User Defined Block.

Example:

1,2,3,4,12,7,18

Output:

CYCLE83 (parm-1,parm-2,parm-3,parm-4,parm-12,parm-7,parm-18)

3.7 Define Tape Block Format

This section defines certain attributes of an output Control Tape block. End-of-block characters, the leader character, optional skip character and how G-codes and M-codes are handled can be defined.

3.7.1 Multiple Non-conflicting G-codes Can Be In A Single Block

Enter YES if G-codes from multiple groups can be output in single Control Tape block. G-codes that are in the same group, meaning they cancel one another, will still be output in separate blocks.

3.7.2 Multiple Non-conflicting M-codes Can Be In A Single Block

Enter YES if M-codes from multiple groups can be output in single Control Tape block. M-codes that are in the same group, meaning they cancel one another, will still be output in separate blocks.

3.7.3 G-codes And M-codes Can Be In The Same Block

Enter NO if an M-code cannot be output in the same Control Tape block as a G-code.

3.7.4 Word Address/Tab Sequential Format

Enter the number that corresponds to the addressing mode of a Control Tape block.

1. Word Address
2. Fixed Tab Sequential

Word address specifies that each register has a character designator assigned to it that identifies it.

Fixed tab sequential specifies that each register is identified by its location within the Control Tape block and each register will be preceded by a tab character.

3.7.5 End-of-block Character(s)

Enter between 0 and 5 characters that will be output as the End-of-block designator. This is usually the \$ character, though when direct downloading of the Control Tape to the machine is used, without running through a convertor, then it is usually set to be no characters. When there are no End-of-block characters, then you must also specify an unpacked punch file.

3.7.6 Start Of Message Character(s)

Enter between 0 and 10 characters that define the start of an operator message. Operator messages are output with the *PPRINT* statement when *DISPLY/ON* has been specified.

3.7.7 End Of Message Character(s)

Enter between 0 and 10 characters that define the end of an operator message. Operator messages are output with the *PPRINT* statement when *DISPLY/ON* has been specified.

3.7.8 Leader Character

Enter a single character that will be output as Control Tape leader. This is usually the space character.

3.7.9 Tab Character

Enter a single character that will be output for tabs. Since the tab is a non-printable character, it is normally replaced with a printable character, usually an asterisk (*), and then run through a convertor (NCP, DNC, etc.) before being sent to the machine. Tabs are used with Tab sequential output and sometimes in message blocks.

3.7.10 Optional Skip Character

Enter a single character that will be output at the beginning of all Control Tape blocks after an *OPSKIP/ON* command has been encountered. This is usually the slash (/) character.

3.7.11 Default Mode For CHECK/LENGTH

Enter YES if you want **PostWorks** to perform a checksumming of each tape block. When **checksumming is enabled**, **PWorks** will total the decimal values of each character in the tape block and will add a code to the block that contains the result of this summation.

Checksumming is primarily used to verify the contents of the Control Tape when transmitting it to the machine control. Verify that the machine supports checksumming prior to enabling it.

3.7.12 Enter Checksum Register (CHECK/LENGTH)

Enter the register that will contain the checksum value when checksumming is enabled. The E register is usually used.

3.7.13 Maximum Value For Checksum Register

Enter the maximum value that can be placed in the checksum register. When checksumming is enabled and this value is reached during the summation process, then the summation value will be rolled back to zero and the summation process continued.

For example, if the summation total calculates to 1234 and the maximum value allowed is 999, then the checksum register will contain 235.

3.8 Define Punch File Format

This section defines the format of the output Control Tape file. The output of the *PARTNO* card and packed or unpacked output are some of the options.

3.8.1 Punch File Is Packed/Unpacked

Enter either PACKED or UNPACKED, depending on how you want the punch file output. A packed punch file has the Control Tape blocks packed into 72 column lines, with columns 73-80 reserved for an ID number. Unpacked punch files contain a single Control Tape block per line.

3.8.2 Maximum Length Of An MCD Block

Enter the maximum size of an output MCD or Control Tape block in number of characters. **PostWorks** will break up any block which exceeds this size into multiple blocks. A valid response is in the range of 10 to 512 This prompt only applies if the punch file is output in unpacked format.

This size can be changed in the part program file by using the Command: "MACHIN/PWORKS,name,OPTION,5,n".

3.8.3 PARTNO Card Output

Enter the number that corresponds to the way you want to handle the output of the *PARTNO* card.

0. Do not output the PARTNO card.
1. Output the first PARTNO card as is.
2. Output the first PARTNO card in a message block.
3. Output all PARTNO cards as is.
4. Output all PARTNO cards in a message block.

Normally the *PARTNO* card is output as is (1).

3.8.4 Separate Tape Registers With A Space In Punch File

Some machines allow the registers in a Control Tape block to be separated with leader character(s) in order to make the display easier to read. Enter YES if you want the registers in each block separated with a single leader character.

3.8.5 Separate Macro Parameters With A Space In Punch File

Some block formats are output using a Macro call with parameters. The parameters are typically contained within parenthesis and separated by commas. YES specifies the parameters are separated by a single space following the comma. NO specifies the Macro parameters will be output in a compressed format, without spaces between them.

3.9 Define Start/End Sequences

This section controls the output of certain sequences at the beginning and end of the output Control Tape file. The amount of leader output, rewind stop characters, *FINI* code and User Defined Block at the start of the program can be defined.

3.9.1 Enter Punch File Header File (.phf) Number

Enter the *PHF* file name that contains the punch file header and trailer sections. Up to 40 characters can be entered, the actual file name loaded will be '*pworks_name.phf*'. Leaving this field blank will disable the output of a header and/or trailer section within the punch file. The [Punch Header File Appendix](#) contains a complete description of the *PHF* file.

3.9.2 Amount Of Leader At Start Of Punch File [Inch/Mm]

Enter the amount of leader you want output to the Punch file following the *PARTNO* card. The value will be in Inches or Millimeters, depending on the *UNITS* runtime option.

3.9.3 Amount Of Leader At End Of Punch File [Inch/Mm]

Enter the amount of leader you want output to the Punch file following the *FINI* card. The value will be in Inches or Millimeters, depending on the *UNITS* runtime option.

3.9.4 Rewind Stop Character(s)

Enter between 0 and 5 characters that define the rewind stop code. The rewind stop code will be output automatically by the *PARTNO* card and at the programmers discretion using the *TMARK/AUTO* command. If you do not wish a rewind stop code at the beginning of the tape, then do not enter any character here.

3.9.5 Output An EOB With Rewind Stop Code

Enter YES if the rewind stop code should be output with an End-of-Block code. The rewind stop code is usually output in a block by itself, terminated by an End-of-Block code.

3.9.6 User Defined Starting Block

Enter the number of the User defined starting block you want to be output as the first block of the control tape. A value of 0 will disable this feature.

3.9.7 Code To Output With The REWIND Card

Enter the register and optional value you want output with the *REWIND* command. You can leave this prompt blank, in which case a code will not be output with *REWIND*. M(30) is usually output with the *REWIND* command.

3.9.8 Code To Output With FINI Card

Enter the register and optional value you want output with the *FINI* card. You can leave this prompt blank, in which case a code will not be output with *FINI*. M(02) is usually output as the End-of-Tape code.

3.9.9 Start Of Tape Character(s)

Enter between 0 and 5 characters that define the Start-of-Tape mark. Usually the rewind stop code will suffice to mark the beginning of the tape, though some machines also require a special string at the start of the Control Tape.

3.9.10 End Of Tape Character(s)

Enter between 0 and 5 characters that define the End-of-Tape mark. Usually the code output with the *FINI* card will suffice to mark an End-of-Tape, though some machines also require a special string at the end of the Control Tape.

3.10 Tape Break Sequences

This section defines the actions to take when the Control Tape is broken up into multiple tapes due to the *BREAK* command. Options covering the default *BREAK* mode, multiple punch files, automatic tool retraction, etc. are included in this section.

3.10.1 Enter Default Mode For BREAK Command

Enter the mode that will be active at the start of the program for tape breaks. You can enter one of the following modes.

OFF - Disables tape breaks.

- ON - Enables tape breaks when a *BREAK* command is encountered and the tape length is greater than the maximum allowed.
- AUTO - Enables tape breaks when the tape length is greater than the maximum allowed.
- TIMES - Enables tape breaks when the machining time is greater than the maximum allowed.
- DELTA - Enables tape breaks when the linear axes machining distance is greater than the maximum allowed.

See the *BREAK/mode* command in the **PWorks** Reference Manual for a detailed description of tape breaks.

3.10.2 Create A New Punch File At Tape Breaks

Enter YES if you want the Control Tape to be broken up into multiple files whenever a tape break sequence is output. Each new punch file will be formatted the same as the original punch file (*PARTNO* card output, leading/trailing leader, etc.).

3.10.3 Retract The Tool At Tape Breaks

Enter Yes if you want the tool to be automatically retracted during a tape break sequence. The tool will be retracted prior to creating a new punch file (if requested) and repositioned to its original position afterwards.

3.10.4 Issue TMARK After Tape Break

Enter YES if you want *TMARK* command automatically issued after a tape break. The first block following a tape break will be an alignment block. The *TMARK* command will be output just prior to repositioning the tool to its original position.

3.10.5 Default Tape Length For Tape Breaks

Enter the maximum tape length allowed before a tape break sequence will be output. This length should be entered in feet or meters, depending on the current input units. **PostWorks** will output a tape break sequence whenever the tape length exceeds this value.

This prompt will only be displayed when the default *BREAK* mode has been set to 'ON' or 'AUTO'.

3.10.6 Default Machining Time For Tape Breaks

Enter the maximum machining time allowed before a tape break sequence will be output. This value should be entered in minutes. **PostWorks** will output a tape break sequence whenever the machining time exceeds this value.

This prompt will only be displayed when the default *BREAK* mode has been set to '*TIMES*'.

3.10.7 Default Delta Linear Distance For Tape Breaks

Enter the maximum linear machining distance allowed before a tape break sequence will be output. This value should be entered in either inches or millimeters, depending on the current input units. **PostWorks** will output a tape break sequence whenever the machining distance exceeds this value.

This prompt will only be displayed when the default *BREAK* mode has been set to '*DELTA*'.

3.10.8 Default Retract Rate For Tape Breaks

Enter the feed rate at which to retract the tool during a tape break sequence. A value of 0 will retract the tool at the rapid traverse rate. This value should be entered in Feed per Minute.

This prompt will only be displayed when tool retracts in a tape break sequence have been enabled.

3.10.9 Default Plunge Rate For Tape Breaks

Enter the feed rate at which to reposition the tool after it has been retracted during a tape break sequence. A value of 0 will plunge the tool at the rapid traverse rate. This value should be entered in Feed per Minute.

This prompt will only be displayed when tool retracts in a tape break sequence have been enabled.

3.11 Exit

Exits the Control Tape format section and returns you to the previous section.

CHAPTER 4 Print File Description

4.0 Print File Description

This section defines the [Print Descriptor File](#) (.pdf) that the current post configuration will use. The actual format of the Print Descriptor File is described in the [Print Descriptor File Appendix](#).

4.1 Enter Default Print Descriptor File (.pdf) Name

Enter the *PDF* file name that you wish to load as the default print file description for the current post configuration. The PDF file name can be up to 40 characters long. The actual file name loaded will be '*pworks_name.pdf*'. The *PDF* file must exist when running the **PWorks** post-processor. You can change the *PDF* file in the part program by using the [MACHIN/PWORKS,OPTION,3](#) command. You can also override this number at runtime by using the '*OPTION,n*' runtime switch.

4.2 Enter Page Header Record(s)

Enter the record number(s) from the Print Descriptor File (*PDF*) that will be output each time the print file advances to a new page. You can enter up to 2 records. If you do not enter any record numbers, then the print file will not contain any page headers.

4.3 Enter Page Trailer Record(s)

Enter the record number(s) from the Print Descriptor File (*PDF*) that will be output prior to the print file advancing to a new page. You can enter up to 2 records. If you do not enter any record numbers, then the print file will not contain any page trailers.

4.4 Enter Motion Block Record(s)

Enter the records number(s) from the Print Descriptor File (*PDF*) that will be output when the control tape block contains axes movement. You can enter up to 2 records. If you do not enter any record numbers, then motion blocks will not be output to the print file.

4.5 Enter Non-motion Block Record(s)

Enter the records number(s) from the Print Descriptor File (*PDF*) that will be output when the control tape block does not contains axes movement. You can enter up to 2 records. If you do not enter any record numbers, then non-motion blocks will not be output to the print file.

4.6 Enter INSERT Record(s)

Enter the records number(s) from the [Print Descriptor File \(PDF\)](#) that will be output with an *INSERT* record. You can enter up to 2 records. If you do not enter any record numbers, then *INSERT* records will not be output to the print file.

4.7 Enter STOP Record(s)

Enter the records number(s) from the Print Descriptor File (*PDF*) that will be output after a *STOP* command is processed. You can enter up to 2 records. If you do not enter any record numbers, then the *STOP* command will not output a print file record.

4.8 Enter OPSTOP Record(s)

Enter the records number(s) from the Print Descriptor File (*PDF*) that will be output after a *OPSTOP* command is processed. You can enter up to 2 records. If you do not enter any record numbers, then the *OPSTOP* command will not output a print file record.

4.9 Enter LOADTL Record(s)

Enter the records number(s) from the Print Descriptor File (*PDF*) that will be output after a *LOADTL* command is processed. You can enter up to 2 records. If you do not enter any record numbers, then the *LOADTL* command will not output a print file record.

4.10 Enter FINI Record(s)

Enter the records number(s) from the Print Descriptor File (*PDF*) that will be output after the *FINI* command is processed. You can enter up to 2 records. If you do not enter any record numbers, then the *FINI* card will not output a print file record.

4.11 Enter PPRINT Record(s)

Enter the records number(s) from the Print Descriptor File (*PDF*) that will be output after a *PPRINT* command is processed. You can enter up to 2 records. If you do not enter any record numbers, then the *PPRINTs* will not be output to the print file.

CHAPTER 5 Motion Block Parameters

5.0 Walk Through Motion Block Parameters

This selection will lead you through all the aspects of defining how motion blocks are handled, including the processing of the *FROM* statement, absolute/incremental programming modes, rotary motion output, and axes tolerance.

5.1 Special Event Handling

This section defines the *FROM* statement processing, lathe radius/diameter programming, and absolute/increment programming modes.

5.1.1 Specify Handling Of FROM Statement

Enter the number that corresponds to the way you want to handle the processing of the *FROM* statement.

- 1 = *FROM* is not output now and will force output of all axes with the next motion block.
- 2 = *FROM* is not output now and will set the current values for each axis.
- 3 = *FROM* is treated the same as *GOTO*.

Normally the *FROM* statement sets the current axes position (2).

5.1.2 The X-axis Is Input As Radius/Diameter

This prompt will only be displayed for lathes. Enter either *RADIUS* or *DIAMETER*, depending on how the input parts will be programmed, this will usually be *RADIUS*.

5.1.3 The X-axis Is Output As Radius/Diameter

This prompt will only be displayed for lathes. Enter either *RADIUS* or *DIAMETER*, depending on how the machine expects to see the X-axis. **PostWorks** will double the X-axis values when the part is programmed as radius values but output as diameter values.

5.1.4 Output The X-axis As Negative Positions

Enter *YES* if the output coordinates should be mirrored through the X-axis, causing the X-axis to be output as negative positions. This question will only be asked when a lathe is being defined.

5.1.5 The Part Is Programmed In XY And Should Be Output In ZX

This prompt will only be displayed for lathes. Enter YES if the input coordinates are programmed in the XY-plane. This is the normal manner for programming lathes. Enter NO if the input coordinates are programmed in the ZX-plane. The output points will always be in the ZX-plane.

5.1.6 Default Positioning Mode [ABS/INCR]

Enter the default positioning mode, either ABS (absolute positions) or INCR (incremental distances) for the output axes. The positioning mode can be changed during the part program by using the *MODE/INCR* command.

5.1.7 FORCE Current Position When Changing From INCR To ABS Mode

Enter the number that corresponds to the way you want to handle the change from incremental to absolute output modes.

- 1 = Do not output the current position.
- 2 = Output the current axes position immediately.
- 3 = Force output of all axes on next motion block.

It is sometimes desirable to force the motion block following a change to absolute programming to include all of the supported axes positions. If this is desirable, enter either 2 or 3.

5.1.8 FORCE Output Of All Axes When One Is Output

Some programmers like to have all of the machining axes output on each motion block in case the NC program has to be restarted on the machine at a random block in the program. Enter Yes at this prompt if you want all of the machining axes output in every motion block.

5.2 Rotary Motion

This section handles the output of rotary motion, including output point adjustments for rotary movement, absolute/incremental programming modes, and the clamping and unclamping of the rotary axes. It can only be entered when at least one rotary axis is supported.

5.2.1 Use Tool Axis Vector To Control Rotary Axis Positions:

Enter YES when the part is defined in one coordinate system and the tool axis should be used to control the position of the rotary axes.

If the part geometry is defined in all planes of the rotary axes movement and only the [ROTABL](#) commands will be used to position the rotary axes, then enter NO.

5.2.2 Adjust Input Points For Rotary Table(s)

Enter YES when the part is defined in one coordinate system and the tool axis is used to move the rotary axes. In this case, the input points will be adjusted according to the table rotations prior to being output.

If the part geometry is defined in all planes of rotary table movement and the [ROTABL](#) command will be used to position the rotary table(s), then enter NO.

This prompt will only be displayed when at least one of the rotary axes is a table.

You can use the [ROTABL/ADJUST,TABLE](#) command to change the setting within the part program.

5.2.3 Adjust Input Points For Rotary Head(s)

When the milling machine contains a rotary head, the output points are usually the location of a rotary pivot point that does not move when the rotary head moves. Enter YES if you want to adjust the input points to lie on this set position at the machine. The [Machine Configuration section](#) and the [ROTABL/ORIGIN](#) command can be used to define the location of the pivot point.

This prompt will only be displayed when at least one of the rotary axes is a head.

You can use the [ROTHED/ADJUST,HEAD](#) command to change the setting within the part program.

5.2.4 Force Output Of All Rotary Axes When One is Output

Enter YES is all rotary axes must be output in a block whenever a single rotary axis would be output. Most machine controls only require that each axis that actually changes be output in a block. In this case enter NO at this prompt.

5.2.5:8 Absolute/Incremental Output For Rotary Axis #n

Enter the number that corresponds to the output format for this rotary axis.

1. Output in current mode.
2. Output is always absolute.
3. Output is always incremental.

Some machines, though they support both absolute and incremental modes, require that the rotary axes be output in only one mode. When this is the case, it is usually the absolute mode.

Enter 1 when the machine will allow this rotary axis to be output in both absolute and incremental modes, as requested by the *MODE/INCR* command.

5.2.9 Enter Value For A CCLW Move To 0 Degree For The Rotary Axes

This prompt will only be displayed when at least one rotary axis is output on a rotary scale, since a rotary scale uses the sign (+-) of the number to determine the direction the axis moves.

When a rotary axis moves to 0 degrees in a counter-clockwise direction the value output is -0. Some machines are not able to determine the difference between a negative zero and a positive zero. For these machines, enter the value to output for a CCLW move to 0 degrees. This will usually be 360.

5.2.10 Enter Value For A CLW Move To 0 Degree For The Rotary Axes

This prompt will only be displayed when at least one rotary axis is output on a rotary scale, since a rotary scale uses the sign (+-) of the number to determine the direction the axis moves.

When a rotary axis moves to 0 degrees in a clockwise direction the typical value output is 360. Some machines do not support a rotary angle value of 360 degrees. These machines usually require a value of 0 degrees.

5.2.11 Enter Rotary Axes Size Change Code

Enter the register and its value that defines a "rotary axis size change" block. The circumference of the rotary axes is usually 360 degrees and cannot be changed. Though, some machines, such as the Bostomatic, accept the rotary axis input as a percentage of its programmed circumference. These machines will usually allow you to change the programmed rotary axes circumference.

Enter a size change code here only if your machine supports circumference rotary axes programming. G(25) is normally used. you can also use the *ROTABL/SIZE* command to change the size of the rotary axis.

5.2.12 Output Rotary Axes Size Change As ...

Enter the number that corresponds to the units in which the machine expects to see rotary size changes from the following units.

1. Radius.
2. Diameter.
3. Circumference.

This prompt will only affect the value output to the Control Tape when changing the programmed size of a rotary axis, the *ROTABL/SIZE* command will still expect to see the new size entered as the circumference.

5.2.13:16 Enter Register For Rotary Axis #n Size Change

Enter the register that will be used to output the newly programmed size for this rotary axis. This register will contain the new size and will be output with the rotary axes size change code specified previously.

5.2.17:23 Enter CLW Direction Code For Rotary Axis #n

Some machine controls require that a code be output with the rotary axis position in order to specify the rotation direction. If this control requires a separate rotary direction code, then enter the register and its value here for moving Rotary axis #n in the clockwise direction.

5.2.18:24 Enter CCLW Direction Code For Rotary Axis #n

Some machine controls require that a code be output with the rotary axis position in order to specify the rotation direction. If this control requires a separate rotary direction code, then enter the register and its value here for moving Rotary axis #n in the counter-clockwise direction.

5.3 Clamping Logic

This section handles the clamping and unclamping logic of the rotary axes. It can only be entered when at least one rotary axis is supported.

5.3.1 Enter Clamping Logic For Rotary Axes

Enter the number that corresponds to the logic required to automatically clamp and unclamp the rotary axes.

1. Clamping/unclamping of rotary axes is not necessary.
2. Each rotary axis should be unclamped prior to movement and immediately clamped after each movement.
3. Each rotary axis should be unclamped prior to movement and clamped prior to the specific rotary axis not moving.
4. Each rotary axis should be unclamped prior to any rotary output and clamped prior to pure linear output with no rotary output.
5. Each rotary axis should be unclamped prior to any rotary movement and clamped prior to pure linear axes movement (no rotary movement).

Some machines require that rotary axes be clamped when they are not moving, in order to maintain a constant position. **PostWorks** has the ability to automatically unclamp the rotary axis prior to movement and to clamp it when there is no movement. Number 3 is the most common type of clamping logic used when clamping of the rotary axes is necessary.

The difference between numbers 4 and 5 is that selecting number 4 will check for rotary axis output on a block and does not count whether there is actual movement or not. Selecting number 5 requires that there is actual rotary axis movement in order to trigger the clamping/unclamping logic.

You can also manually clamp and unclamp the rotary axes using the **CLAMP** command.

5.3.2:8 Enter Clamping Code For Rotary Axis #n

Enter the rotary and value required to clamp this rotary axis, for example, M(10). This prompt will only be displayed when automatic clamping of the rotary axes has been enabled.

5.3.3:9 Enter Unclamping Code For Rotary Axis #n

Enter the register and value required to unclamp this rotary axis, for example, M(11). This prompt will only be displayed when automatic clamping of the rotary axes has been enabled.

5.3.10 Clamping Codes Should Be Output In A Block By Themselves

Enter YES the codes required to clamp and unclamp the rotary axes should be output in a block by themselves. Otherwise, they will be output on the motion block that caused the automatic clamping/unclamping. The prompt will only be displayed when automatic clamping of the rotary axes has been enabled.

5.3.11:17 User Defined Block For Clamping Rotary Axis #n

Enter the number of the User defined block you want to be output when Rotary axis #n is clamped. A value of 0 will disable this feature. This prompt will only be displayed when automatic clamping of the rotary axes has been enabled.

5.3.12:18 User Defined Block For Unclamping Rotary Axis #n

Enter the number of the User defined block you want to be output when Rotary axis #n is unclamped. A value of 0 will disable this feature. This prompt will only be displayed when automatic clamping of the rotary axes has been enabled.

5.4 Define Axes Tolerances

This section defines the minimum increment allowed for axes output and the tolerance for determining when each axis should be output.

5.4.1:10 Minimum Linear Increment Allowed For 'Axis' Movement

Enter the minimum delta movement allowed for the specified axis. All output for this axis will be on an even increment. For example, if .0002 is the minimum increment allowed and .0005 is requested to be output, then the actual output value may be .0004 or .0006.

Enter a value of zero if the minimum increment corresponds to the accuracy specified for the axis' register. For example, if 4 digits to the right of the decimal point has been specified and the minimum increment is .0001, then enter 0.

5.4.11:20 Tolerance For Determining Output On 'Axis'

Enter the tolerance for determining when the specified axis should be output. This axis will not be output until the delta movement is greater than or equal to the tolerance entered. This feature is useful on multi-axis machines where a slight fluctuation in the tool axis will generate output positions that are close but not exactly at the same location.

You can use the *PPTOL* command to change the tolerance values within the part program.

5.5 Default Output Axes Adjustments

This section defines the default *TRANS/-AXIS* and *TRANS/SCALE,-AXIS* values.

5.5.1:10 Default TRANS/-AXIS For 'Axis'

Enter a value which you want to be added to the specified axis prior to being output. This value will be added to the axis until a *TRANS/-AXIS* command is encountered in the program. This value is normally 0.

5.5.11:20 Default TRANS/SCALE,-AXIS For 'Axis'

Enter a scale factor to multiple the specified axis by prior to being output. The scale will be applied to the axis until a *TRANS/SCALE,-AXIS* command is encountered in the program. This value is normally is 1.

5.6 Motion Register Description

This section defines the codes for contouring motion, absolute/incremental modes, and the User Defined Blocks for the 1st motion block and non-motion blocks.

5.6.1 Linear Motion Code

Enter the register and its value for setting linear interpolation mode. The linear motion code is usually G(01).

5.6.2 Linear Motion Code In **LMFP** Mode

Enter the register and its value for setting linear interpolation mode while in polar interpolation on a Mill/Turn machine. This code is usually D(11).

5.6.3 Linear Motion Code In **LMDP** Mode

Enter the register and its value for setting linear interpolation mode while in cylindrical interpolation on a Mill/Turn machine. This code is usually D(21).

5.6.4 Absolute Positioning Mode

Enter the register and its value for setting absolute positioning mode. Normally, the absolute positioning code for machines which support both absolute and incremental modes is G(90). If you do not enter a register, then no code will be output when switching to absolute mode.

5.6.5 Incremental Positioning Mode

Enter the register and its value for setting incremental positioning mode. Normally, the absolute positioning code for machines which support both absolute and incremental modes is G(91). If you do not enter a register, then no code will be output when switching to incremental mode.

5.6.6 Code(s) For Setting Major Tool Axis To XYPLAN

Enter the code(s) that will be output when the **MODE/TOOL,XYPLAN** command is used. The **MODE/TOOL,XYPLAN** sets the primary (spindle) tool axis to 0,0,1. You can enter anywhere from 0 to 3 codes.

Example:

G(17), M(53)

5.6.7 Code(s) For Setting Major Tool Axis To ZXPLAN

Enter the code(s) that will be output when the *MODE/TOOL,ZXPLAN* command is used. The *MODE/TOOL,ZXPLAN* sets the primary (spindle) tool axis to 0,1,0. You can enter anywhere from 0 to 3 codes.

Example:

G(18), M(54)

5.6.8 Code(s) For Setting Major Tool Axis To YZPLAN

Enter the code(s) that will be output when the *MODE/TOOL,YZPLAN* command is used. The *MODE/TOOL,YZPLAN* sets the primary (spindle) tool axis to 1,0,0. You can enter anywhere from 0 to 3 codes.

Example:

G(19), M(54)

5.6.9 LMFP/LMDP Mode Minimum Span Register

Enter the register to use for the minimum span length while in either polar interpolation or cylindrical interpolation mode on a Mill/Turn type machine. This span length is used to keep the tool within tolerance of the part when the rotary axis is used to satisfy the Y-axis on machines which do not have a physical Y-axis. The L register is usually used.

5.6.10 Enable Mill/Turn Milling Mode Code

Enter the code that changes the mode on a Mill/Turn machine from lathe programming to milling. This is the same code which enables the spindle as a rotary axis and usually M(19).

5.6.11 Disable Mill/Turn Milling Mode Code

Enter the code that changes the mode on a Mill/Turn machine from milling to lathe programming. This code disables the spindle from being programmed as a rotary axis and usually M(20).

5.6.12 Enter Mill/Turn Polar Interpolation Code

Enter the code which enables the polar interpolation feature on a Mill/Turn type machine. Polar interpolation is a simplified way to program a milling operation on the face of the part. This code is usually G(79).

5.6.13 Enter Mill/Turn Cylindrical Interpolation Code

Enter the code which enables the cylindrical interpolation feature on a Mill/Turn type machine. Cylindrical interpolation is a simplified way to mill grooves on the diameter of the part. This code is usually G(70).

5.6.14 Register For Circle Unit In Cylindrical Interpolation

On a Mill/Turn machine which supports cylindrical interpolation, it is sometimes required for the program to define the unit length of the part circumference. This register will contain the arc length of one degree around the circumference of the part. If your machine does not require this value, then leave this prompt blank.

5.6.15 Is Cylinder Axis Code Required In LMDP Enable/Disable Block

Enter YES if the rotary axis address should be output when entering and exiting cylindrical interpolation mode. The rotary axis address will be output in the block containing both the [LMDP](#) enable code and the block cancelling LMDP mode.

5.6.16 Is Cylinder Axis Value Required In LMDP Enable/Disable Block

Most machines require only the rotary axis address (C, A, B, etc.) to be output on an LMDP enable/disable block. If your machine does not need a value to be output with this address, then enter NO at this prompt, otherwise enter YES. The value output with this address will be the value currently stored in this register.

5.6.17 User Defined 1st Motion Block

Enter the number of the User Defined Block you want to be output as the first motion block of the Control Tape. Usually this will at least contain all axis positions and the initial feedrate. A value of 0 will disable this feature.

5.6.18 User Defined Non-motion Block

Enter the number of the User Defined Block you want to be output for each non-motion block. For example, [SPINDL](#), [STOP](#) and [FINI](#) blocks. A value of 0 will disable this feature.

5.7 Exit

Exits the Motion block parameters section and returns you to the previous section.

CHAPTER 6 Circular/Spline Interpolation

6.0 Walk Through Circular/Spline Interpolation

This selection will lead you through all of the aspects of defining circular interpolation output, including format, radius limits, helical interpolation, and register descriptions. It will also lead you through the definition of spline interpolation.

6.1 Circular Interpolation Format

This section defines the output format for circular interpolation; absolute center point, incremental distances to center point, polar coordinates, etc. It also includes circular boundary definitions.

6.1.1 Does The Machine Controller Support Circular Interpolation

Enter YES if the machine will accept a single block which will generate circular movement on the machine. If you answer NO, then all input circular records will be output as linear moves.

6.1.2 Enter Format Of Circular Interpolation

Enter the number that corresponds to the format of the circular interpolation block the machine expects to see.

- 1 = The absolute center point position of the circle will be output in a circular interpolation block.
- 2 = The signed incremental distances to the center point of the circle, for each linear axis that moves, will be output in a circular interpolation block.
- 3 = The unsigned incremental distances to the center point of the circle, for each linear axis that moves, will be output in a circular interpolation block.
- 4 = When in absolute mode, the absolute center point of the circle will be output. When in incremental mode, the signed incremental distances to the center of the circle will be output.
- 5 = When in absolute mode, the absolute center point of the circle will be output. When in incremental mode, the unsigned incremental distances to the center of the circle will be output.
- 6 = The radius of the circle will be output in a circular interpolation block.
- 7 = Both the linear move prior to and the circular interpolation block will be output as polar coordinates, including the center point of the circle and the angular position of the move.

- 8 = The absolute center point of the circle will be output in one block, with the ending point of the arc output in the following block.
- 9 = The absolute center point of the circle and the circle's radius will be output in a circular interpolation block.
- 10 = The conversational polar coordinate format of circular interpolation consists of two blocks of output per circular record. The first block contains the absolute coordinates of the circle center and the second block contains the delta angle of the circular move and the circular direction.

6.1.3 Enter Format Of Circular Interpolation (LMFP/LMDP Mode)

Enter the number that corresponds to the format of the circular interpolation block the Mill/Turn machine expects to see when in either polar interpolation or cylindrical interpolation mode.

- 1 = The absolute center point position of the circle will be output in a circular interpolation block.
- 2 = The signed incremental distances to the center point of the circle, for each linear axis that moves, will be output in a circular interpolation block.
- 3 = The unsigned incremental distances to the center point of the circle, for each linear axis that moves, will be output in a circular interpolation block.
- 4 = When in absolute mode, the absolute center point of the circle will be output. When in incremental mode, the signed incremental distances to the center of the circle will be output.
- 5 = When in absolute mode, the absolute center point of the circle will be output. When in incremental mode, the unsigned incremental distances to the center of the circle will be output.
- 6 = The radius of the circle will be output in a circular interpolation block.
- 7 = Both the linear move prior to and the circular interpolation block will be output as polar coordinates, including the center point of the circle and the angular position of the move.
- 8 = The absolute center point of the circle will be output in one block, with the ending point of the arc output in the following block.
- 9 = The absolute center point of the circle and the circle's radius will be output in a circular interpolation block.

6.1.4 Incremental Distance Should Be From Start Point To Center

This prompt will only be displayed when signed incremental distances to the center point are output on circular interpolation blocks. Enter YES if the distances output should be from the starting point of the arc to the center point. This is the standard output.

Some machines require the distance from the center point to the start of the arc. Enter NO to generate these distances.

6.1.5 The Center Point Should Be Output As Radius/Diameter

Enter either RADIUS or DIAMETER, depending on how the machine expects to see the X-axis center location for circular interpolation. Most lathes, even though they accept diameter positioning for the X-axis, expect to see the X-axis center location programmed in reference to the part radius.

6.1.6 Circular Interpolation Is Allowed In All 3 Planes (XY, YZ, ZX)

Enter NO if your machine only supports circular interpolation in the certain planes. In this case, the following question will appear requesting the planes which circular interpolation is allowed in.

6.1.7 Is 3-axis Circular Interpolation Supported

Some machines support circular interpolation in a plane other than one of the major planes (XY, YZ, ZX). In this case, the circular direction code, an intermediate point, and the final point of the circular arc will be output. Enter YES if your machine supports circular interpolation outside of the major plane.

6.1.8 Output Final Point Of 3-axis Circular On Separate Block

Enter YES if 3-axis circular interpolation output should be broken up into two separate blocks, with the first block containing the circular direction code and the intermediate point, and the second block containing the final point of the circular arc. Enter NO if the intermediate and final points can be in the same block.

Example:

```
YES:  G02.4 X Y Z $  
      X Y Z$  
NO:   CIP X Y Z I1= nJ1= K1=$
```

6.1.9 Enter Format of 3-axis Circular Intermediate Point

Enter the number that corresponds to the format (Absolute/Incremental) of the 3-axis circular intermediate point when it is output in a single block with the final circular location.

- 1 = Use the same format as the circular center point.
- 2 = Use the circular end point format (MODE/INCR,ON-OFF).

When the 3-axis intermediate point is output in a separate line and not combined with the final point the programmed mode (MODE/INCR) will always be used.

6.1.10 Enter Planes Where Circular Interpolation Is Allowed (XY, YZ, ZX)

Enter all planes which support circular interpolation. Valid responses are XY, YZ, and ZX. Separate each plane with a comma, for example: XY, ZX. All circular moves in a plane which does not support circular interpolation will be output as linear moves.

6.1.11 Circular Moves Should Be Broken Up

Enter the number that corresponds to the boundary limitations of circular interpolation output.

- 1 = on quadrant boundaries.
- 2 = on hemisphere boundaries.
- 3 = on full 360 degree boundaries.

6.1.12 Force Position On Circular Interpolation Record

Enter YES if you wish to force the output of all active linear axes on a circular interpolation block. if you enter NO, then the output of the linear axis will follow the normal rules, usually meaning they will not be output if they do not change.

6.1.13 Force Direction Code On Circular Interpolation Record

Enter YES if the machine requires a direction code (CLW, CCLW) on every circular block output. The circular direction code usually is only output on the first circular block following a linear move or when the direction changes. On some machines, though, the circular direction codes are not modal and the control will revert to linear interpolation if the direction code is not encountered in a motion block.

If you enter NO, then the output of the circular direction codes will follow the normal rules, usually meaning they will not be output when the circular direction remains the same.

6.1.14 Estimate Delta Distance During Circular Interpolation

Normally **PostWorks** will only use the output blocks during circular interpolation to calculate the machining distances (axis delta movements), which can be useful when comparing the [Slide Motion Recaps](#) from one run to another, since the delta summations will be different when comparing circular interpolation output to linear output. At times it is necessary to see a more exact estimate of the machining distances during circular interpolation, for example when estimating tool life.

Enter YES at this prompt if the total machining distances during circular interpolation should be included in the axis delta summations kept by **PostWorks**.

6.2 Circular Interpolation Tolerances

This section defines the minimum and maximum radius allowed, the minimum amount of linear movement to output as circular interpolation, and whether or not to adjust the final point on a circular move to lie exactly on the circle.

6.2.1 Radius Tolerance For Circular Interpolation

Enter the maximum amount of deviation that a cl point in a circular interpolation record can have as compared to the radius calculated by using the starting point of the arc. If the cl point exceeds this tolerance, then the remainder of the circular record will be output as linear moves.

You can use the *MCHTOL* command within the part program to change this tolerance.

6.2.2 Minimum Linear Movement To Output As Circular Interpolation

Enter the minimum amount of linear movement that can be output as a circular interpolation block. If the delta movement from the starting point to the ending point is less than this value, then this portion of the circular interpolation record will be output as linear moves.

You can use the *MCHTOL* command within the part program to change this distance.

6.2.3 Minimum Radius Allowed For Circular Interpolation

Enter the minimum radius of a circle that can be output as circular interpolation. If the circle radius is less than this value, then the circular interpolation record will be output as linear moves. This is usually a machine limitation.

6.2.4 Maximum Radius Allowed For Circular Interpolation

Enter the maximum radius of a circle that can be output as circular interpolation. If the circle radius is greater than this value, then the circular interpolation record will be output as linear moves. This is usually a machine limitation.

6.2.5 Adjust Final Point To Lie Exactly On The Circle

Some machines require that the final point of the circular arc lie exactly on the circle as defined by the starting point and circle center point, without even the smallest deviation. Since the CAM software iterates around the circle, there is a possibility that the final point is off the circle, but within a certain tolerance.

Enter YES if your machine must have the final point lie exactly on the circle. One side effect of having **PostWorks** adjust the final point is that you can accumulate positioning errors when driving multiple circles, because the adjusted point of the previous arc will become the starting point of the next arc, thereby changing the radius of the next arc slightly. This is not a problem when there is a linear move between circles.

6.3 Helical Interpolation Format

This section defines the output format for helical interpolation and whether the machine supports helical interpolation.

6.3.1 Does The Machine Controller Support Helical Interpolation

Enter YES if your machine will generate helical movement automatically. A helical interpolation block usually consists of circular interpolation codes along with the third linear axis depth and optionally the lead of the depth.

If you enter NO, then **PostWorks** will generate the helical motion as a series of linear moves.

6.3.2 Output Final Linear Depth With Helical

Enter YES if helical interpolation blocks should contain the final depth of the third linear axis. This is normally the case. This prompt will only be displayed when the machine supports helical interpolation.

6.3.3 Helical Final Depth Is Output As

Enter the number corresponding to the output format for the helical final depth.

- 0 = Current programming mode.
- 1 = Always absolute.
- 2 = Always incremental.

6.3.4 Output Delta Linear Movement With Helical

Enter the number corresponding to the distance travelled along the arc for the third linear axis with helical interpolation.

- 0 = Do not output the delta linear movement of the third axis.
- 1 = Output the linear movement per radian for the third axis.
- 2 = Output the linear movement per 360 degrees for the third axis.

This prompt will only be displayed when the machine supports helical interpolation.

6.3.5 Tolerance For Generating Helical Movement

Enter the tolerance to maintain for linear moves which **PostWorks** generates for helical interpolation. This prompt will only be displayed when the machine does not support helical interpolation.

6.4 Circular Interpolation Registers

This section defines the registers to use for all codes output for circular interpolation blocks, including the plane selection codes.

6.4.1 Polar Coordinate Linear Motion Code

Enter the register and its value for the linear move immediately preceding a circular interpolation move when polar coordinate circular interpolation is supported. This code is usually G(11).

6.4.2 Polar Coordinate Rapid Motion Code

Enter the register and its value for a rapid move that immediately precedes a circular interpolation move when polar coordinate circular interpolation is supported. This code is usually G(10).

6.4.3 Clockwise Circular Interpolation Code

Enter the register and its value for setting circular interpolation mode in the clockwise direction. This code is usually G(02).

6.4.4 Counter-Clockwise Circular Interpolation Code

Enter the register and its value for setting circular interpolation mode in the counter-clockwise direction. This code is usually G(03).

6.4.5 Conversational Circle Center Code

Enter the register and its value that defines the circle center definition block when conversational polar coordinates are output for circular interpolation. This code is usually CC.

6.4.6 Conversational Circular Record Code

Enter the register and its value that defines the circular motion block when conversational polar coordinates are output for circular interpolation. This code is usually CP.

6.4.7 Conversational Delta Angle Register

Enter the register descriptor for the incremental angular movement when conversational polar coordinate circular interpolation is supported. This register is usually IPA.

6.4.8 Clockwise Circular Interpolation Code In LMFP Mode

Enter the register and its value for setting circular interpolation mode in the clockwise direction while in polar interpolation mode on a Mill/Turn type machine. This code is usually D(12).

6.4.9 Counter-Clockwise Circular Interpolation Code In LMFP Mode

Enter the register and its value for setting circular interpolation mode in the counter-clockwise direction while in polar interpolation mode on a Mill/Turn type machine. The code is usually D(13).

6.4.10 Clockwise Circular Interpolation Code in LMDP Mode

Enter the register and its value for setting circular interpolation mode in the clockwise direction while in cylindrical interpolation mode on a Mill/Turn machine. This code is usually D(22).

6.4.11 Counter-Clockwise Circular Interpolation Code In LMDP Mode

Enter the register and its value for setting circular interpolation mode in the counter-clockwise direction while in cylindrical interpolation mode on a Mill/Turn type machine. The code is usually D(23).

6.4.12 Clockwise Helical Interpolation Code

Enter the register and its value for setting helical interpolation mode in the clockwise direction. This code is usually G(02).

6.4.13 Counter-Clockwise Helical Interpolation Code

Enter the register and its value for setting helical interpolation mode in the counter-clockwise direction. This code is usually G(03).

6.4.14 Clockwise 3-axis Circular Interpolation Code

Enter the register and its value for setting circular interpolation mode outside of the major planes in the clockwise direction. This code is usually CIP or G(02.4).

6.4.15 Counter-Clockwise 3-axis Circular Interpolation Code

Enter the register and its value for setting circular interpolation mode outside of the major planes in the counter-clockwise direction. This code is usually CIP or G(03.4).

6.4.16 X-axis Circle Center Register

Enter the register descriptor for the X-part of the circle center point for circular interpolating blocks. The I1 register usually used.

6.4.17 Y-axis Circle Center Register

Enter the register descriptor for the Y-part of the circle center point for circular interpolating blocks. The J1 register usually used.

6.4.18 Z-axis Circle Center Register

Enter the register descriptor for the Z-part of the circle center point for circular interpolating blocks. The K1 register usually used.

6.4.19 Circle Radius Register

Enter the register descriptor for the circle radius for circular interpolation blocks. The R register is usually used. This prompt will only be displayed when the circle radius is output in circular interpolation blocks.

6.4.20 X-axis Intermediate Point Register

Enter the register descriptor for the X-part of the intermediate point when 3-axis circular interpolation is active. The I2 register is usually used. This prompt will only be displayed when 3-axis circular interpolation is supported.

6.4.21 Y-axis Intermediate Point Register

Enter the register descriptor for the Y-part of the intermediate point when 3-axis circular interpolation is active. The J2 register is usually used. This prompt will only be displayed when 3-axis circular interpolation is supported.

6.4.22 Z-axis Intermediate Point Register

Enter the register descriptor for the Z-part of the intermediate point when 3-axis circular interpolation is active. The K2 register is usually used. This prompt will only be displayed when 3-axis circular interpolation is supported.

6.4.23 Polar Coordinate Angle Register

Enter the register descriptor for the angular position when polar coordinate circular interpolation is supported.

6.4.24 Output X And Y Circle Center Registers In All Planes

Some machines require that the same circle center point registers be output for all machining planes of circular interpolation, for example, always I & J. Enter YES if your machine only accepts the same two circle center registers for all machining planes.

6.4.25 XY-plane Selection Code

Enter the register and its value that selects the XY machining plane. This is usually G(17).

6.4.26 ZX-plane Selection Code

Enter the register and its value that selects the ZX machining plane. This is usually G(18).

6.4.27 YZ-plane Selection Code

Enter the register and its value that selects the YZ machining plane. This is usually G(19).

6.4.28 User Defined Plane Selection Code

Some Mill/Turn machines support cylindrical interpolation ([LMDP](#)) in user selected planes. These machines require a user defined plane selection block to be output when enabling LMDP mode. Enter the register and its value which defines a user defined plane block. This code is usually G(16).

6.4.29 Are Axis Code Values Required With Plane Selection

User defined plane selection blocks contain a definition code and the axis addresses which define the machining plane. Usually the axis addresses without any values (X, Y, etc.) are output in this

block. In this case enter NO at the prompt. If the machine required a value to be output with these addresses, then enter YES. The last output value for each register will be output.

6.4.30 Output Dwell Code With Plane Selection

This prompt is not yet supported.

6.4.31 Amount Of Dwell To Output With Plane Selection Code

This prompt is not yet supported.

6.4.32 Output Plane Selection Code In Circular Block

Enter YES if the plane selection code (G17, etc.) can be output in the same block as the circular interpolation codes (G02, etc.). Enter NO if the plane selection code should be output in the block preceding the circular interpolation block.

6.5 Spline Interpolation Format

This section defines the output format for spline interpolation, including controller support, tolerances, and register definitions.

6.5.1 Does The Machine Controller Support Spline Interpolation

Enter YES if the machine will accept a spline definition as input and generate the actual motion which drives the spline. If you enter YES to this question, then **PostWorks** will evaluate the input coordinates and generate a spline definition whenever they are within tolerance of a smooth spline. The spline definition is actually calculated by **PostWorks** and requires no special input from the CAM system.

Spline interpolation formats currently supported are for the Siemens 880 control and the Fanuc 16 control.

6.5.2 Enter Codes For Enabling High Precision Contouring Mode

Enter up to 5 registers and their values which will enable high precision contouring control mode. This mode must be enabled on Fanuc controls when spline interpolation is in effect and is usually enabled using the following codes.

Fanuc	-	G(5),P(10000)
Siemens	-	(none)

The Siemens control does not use high precision contouring control mode.

6.5.3 Enter Codes For Disabling High Precision Contouring Mode

Enter up to 5 registers and their values which will disable high precision contouring control mode. This mode must be enabled on Fanuc controls when spline interpolation is in effect and is usually disabled using the following codes.

Fanuc	-	G(5),P(0)
Siemens	-	(none)

6.5.4 Enter The Tolerance For Fitting Spline Through Points

Enter the tolerance to use when creating splines. This tolerance will determine the number of points used to define the spline. A looser tolerance will generate larger splines with fewer control points, reducing the amount of tape blocks.

6.5.5 Enter Chordal Tolerance For Spline Interpolation

This tolerance specifies the maximum amount of deviation that a spline can have between two definition points as compared to the programmed cl points. The chordal tolerance is used to gauge the maximum distance a straight line cl move deviates from the calculated spline. If the distance is greater than the chordal tolerance, then the spline will be split into two or more splines, or linear interpolation will be reinstated.

6.5.6 Enter Spline Interpolation Code

Enter the register and its value used to enable spline interpolation mode at the controller. It usually has the following format.

Fanuc	-	G06.2
Siemens	-	BSPLINE

The Siemens code should be setup in the Register Format section as having the start characters 'BSPLI', the end characters 'NE' and 0 digits output to the left and right of the decimal point.

6.5.7 Force Linear/Circular Interpolation Code After Spline Mode

Enter YES if an interpolation code must be reinstated after spline interpolation. If you enter NO at this prompt, then the normal output control for the interpolation register will be used to determine when it will be output.

6.5.8 Output Spline Interpolation Code In A Block By Itself

Enter YES if the spline interpolation definition code must be output in a block by itself. Otherwise, it will be output with the first coordinate in the spline definition.

Typical Answer:

Fanuc - NO
Siemens - YES

6.5.9 Output Knot Values Or Node Distances

Along with the control points output with spline interpolation, the machine control will usually require additional data to define the spline. The Fanuc control requires the knot values, while the node distance is the length along the curve between spline coordinates and is required by the Siemens control.

Typical Answer:

Fanuc - KNOT
Siemens - NODE

6.5.10 Enter Register For Knot Values/Node Distances

Enter the register descriptor that will contain either the knot value or the node distance for each spline definition coordinate.

Typical Answer:

Fanuc - K
Siemens - PL=

A register must be setup to the appropriate starting characters and numeric format and entered at this prompt, for example, K2.

6.5.11 Include Start Point In Spline Interpolation

The starting point for spline interpolation is the machine location output previous to the spline code output. The Fanuc control requires that this point be repeated as the first control point of the spline. The Siemens control does not expect this location to be output. The following answers are typical for the two controls.

Fanuc - YES
Siemens - NO

6.6 Exit

Exits the Circular/Spline interpolation section and returns you to the previous section.

CHAPTER 7 Motion Adjustments

7.0 Walk Through Motion Adjustments

This selection will lead you through all of the different types of adjustments that can be applied to machining motion, including mutually axes linearization, and slowdown blocks.

7.1 Miscellaneous Adjustments

This section includes support for the maximum amount of axes which can move in single motion block, mutually exclusive axes, and holding back motion when only a single axis moves.

7.1.1 Maximum Number Of Axes Which Can Move In A Single Block

Some machines only allow so many axes to move in a single block. If your machine has this limitation, then enter the maximum number of axes which can move in a single block. Otherwise, if your machine has no such limitation, enter 10.

The move will be broken into two or more moves when the number of axes in a single block exceeds this number.

7.1.2 Priority For Output Axes

This prompt will only be displayed when the maximum number of axes allowed to move in single block is less than the number of axes supported, both linear and rotary. Enter each supported axis, separated with a comma, in the order of which you want them output when the number of axes exceeds the maximum allowed.

When the move is broken up, the axes appearing first in this list will be output on the first block, with the remaining axes output on the following blocks.

Following is a list of acceptable axis designators.

X1	=	Primary X-axis.	X2	=	Secondary X-axis.
Y1	=	Primary Y-axis.	Y2	=	Secondary Y-axis.
Z1	=	Primary Z-axis.	Z2	=	Secondary Z-axis.
A1	=	Rotary axis #1.	A2	=	Rotary axis #2.
A3	=	Rotary axis #3.	A4	=	Rotary axis #4.

7.1.3 Number Of Mutually Exclusive Axes Sets

Some machines lack the capabilities to move all of the available programmable axes at the same time. **PostWorks** has the capability to check for mutually exclusive axes (axes which cannot be programmed to move simultaneously in the same motion block) and break up the motion block so that these axes are in separate blocks. An error message will also be output when the move is broken up.

You can enter up to 4 sets of mutually exclusive axes. If all of the programmable axes can move together simultaneously, then enter 0 at this prompt.

7.1.4:16 Enter nth Set Of Mutually Exclusive Axes

Enter the machine axes, linear and/or rotary, that cannot move simultaneously. You can enter from 2 to 4 mutually exclusive axis descriptors.

Following is a list of acceptable axis designators.

X1	=	Primary X-axis.	X2	=	Secondary X-axis.
Y1	=	Primary Y-axis.	Y2	=	Secondary Y-axis.
Z1	=	Primary Z-axis.	Z2	=	Secondary Z-axis.
A1	=	Rotary axis #1.	A2	=	Rotary axis #2.
A3	=	Rotary axis #3.	A4	=	Rotary axis #4.

7.1.5:17 Controlling Axis For nth Set

When breaking a move, where 2 or more mutually exclusive axes were programmed to move together, into 2 or more motion blocks, **PostWorks** has the capability to determine how the move is to be broken up based on whether a specified axis moves in the positive or negative direction.

Enter the axis descriptor whose direction will determine how a move will be broken up when 2 or more mutually exclusive axes are programmed to move simultaneously.

7.1.6:18 Axes Priority With Positive Axis Move For nth Set

Enter the priority that will be used when breaking up a move which contains mutually exclusive axes programmed to move simultaneously and the controlling axis for this set moves in the positive direction. The order in which you want the axes output, with each axis descriptor separated by a comma, should be entered.

You can specify that certain axes be output in one block, while other axes are output in a separate block, by specifying the \$ character in the axis description list. Again, the \$ character must be separated by commas.

Example

X1,Y1,\$,Z1,\$,A1,A2

In the above example the primary X and Y axes will be output in the first block, the primary Z axis in the second block, and rotary axes #1 and #2 in the final block, as shown below.

X1.34 Y8.5\$
Z6.3\$
A90.0 B100.83\$

In this example the \$ character is assumed to be the End-of-Block character, and may be different for your machine configuration. Any remaining axes that are not contained in this list will be output in the final block.

7.1.7:19 Axes Priority With Negative Axis Move For nth Set

Enter the priority that will be used when breaking up a move which contains mutually exclusive axes programmed to move simultaneously and the controlling axis for this set moves in the negative direction. The order in which you want the axes output, with each axis descriptor separated by a comma, should be entered.

You can specify that certain axes be output in one block, while other axes are output in a separate block, by specifying the \$ character in the axis description list. Again, the \$ character must be separated by commas.

7.1.20 Hold Back Motion Where Only 1 Axis Moves

Enter YES if you want to disable the output of any supported axis while it is the only axis moving, until another axis moves. This feature is generally used when you have at least one rotary axis. In some cases motion is generated that will contain multiple points and tool axis vectors, though the output from the post-processor will only generate rotary axis movement, in this case, the rotary axis movement will not be output until another axis moves or the rotary axis changes direction. So instead of multiple blocks of rotary moves, you will only have one.

7.2 Rotary Axes Linearization

This section defines the tolerance for rotary axes linearization and the support for an automatic tool retract when the rotary axes are positioned using the longest route. This section can only be entered when at least one rotary axis is supported.

7.2.1 Default Linearization Tolerance

PostWorks has the capability to break up a move that consists of rotary angle changes, when a specified tolerance has been exceeded. The move will be broken up into smaller moves in order to keep the tool end point on the same line as the previous and new positions.

Enter the largest acceptable deviation from the programmed straight line move that can occur. Linearization is recommended under certain conditions, because the tool end point will actually move in a circular fashion whenever a large rotary axis is programmed, while the linear machine slides make a linear move. Linearization will cause the tool end point to move within the specified tolerance of a straight line, while the linear machine slides move in a circular fashion.

The *LINTOL* command can be used within the part program to change the linearization tolerance. Enter a value of 0 to disable linearization.

7.2.2 Default Rapid Linearization Tolerance

Moves programmed with *RAPID* in effect usually take place away from the part due to the inconsistency of axes movements during this mode. Therefore, rapid moves are not normally linearized. There are times when *RAPID* is used when the tool is in close proximity to the part, in this case enter the default tolerance to use for linearization when *RAPID* is in effect. Enter a value of 0 to disable linearization during rapid moves.

The *LINTOL/RAPID* command can be used to change the linearization tolerance used with rapid moves from within the part program.

7.2.3 Adjust Tool Axis Vector To Minimize Rotary Movement

PostWorks has the capability of adjusting the tool axis vector by moving the ending position of the top of the tool back along the forward vector whenever the tool axis is near the apex point of an AC-style rotary axis. This feature is used to keep the rotary axes from rotating excessively when the apex point is neared. The tool tip will still move to the programmed position and the top of the tool will be within a user specified tolerance of the programmed position.

Of the various “adjustments” that can be made to the tool axis when it is near the apex point, this is the recommended feature to use.

The *LINTOL/ADUST* command can be used within the part program to turn on and turn off this adjustment.

7.2.4 Linear Tolerance To Use For Tool Axis Adjustment

Enter the tolerance value to use to position the top of the tool when tool axis adjustments to minimize rotary movement have been enabled. The top of the tool will always remain within this tolerance of the programmed position.

The *LINTOL/ADUST* command can be used within the part program to modify the tolerance value.

7.2.5 Angular Cone To Consider Tool Axis Adjustment For

Enter the angular distance that the tool axis vector must be within in reference to the apex point of an AC-style rotary axis before **PostWorks** will consider adjusting the tool axis. This is typically a small value (less than 5 degrees) since the further away the tool axis is from the apex point, the less noticeable the fluctuations in the rotary axes will be and the less benefit this feature will provide.

The *LINTOL/ADUST* command can be used within the part program to modify the angular value.

7.2.6 Minimum Angular Delta Movement To Consider For Tool Axis Adjustment

Enter the minimum angular delta movement that the rotary axes must move prior to considering adjusting the tool axis vector. This limit is used to keep from adjusting the tool axis vector when the rotary axes do not fluctuate greatly. This is typically a higher value (10 degrees or more)

7.2.7 Default Tool Length To Consider For Tool Axis Adjustment

Enter the value to use for the tool length when an actual tool length has not been programmed. The tool length is used to determine where the top of the tool is when performing tool axis adjustments.

The *LINTOL/ADUST* command can be used within the part program to modify the default tool length value.

7.2.8 Tolerance For Truncating Tool Axis Vector Components

Enter the tolerance to use for the tool axis vector components (IJK) to determine when a component should be set to zero. Whenever a tool axis component is less than this value, it will be set to zero. This feature is useful for locking out a tool axis component during 4-axis motion.

The *PPTOL/VECTOR* command can be used to change this value within the part program.

7.2.9 Retract Tool When Taking Longest Route

In many instances there is a possibility that two sets of rotary angles can satisfy the input tool axis vector. Usually **PostWorks** will choose the set of angles that require the least amount of movement. An exception to this rule is when the closest set of angles will cause the machine to exceed an axis travel limits. In this case **PostWorks** will choose the rotary axis with the longest movement, as long as it keeps the machine within limits.

When linearization is enabled and **PostWorks** choose the set of rotary angles which require the longest movement, then you have the option of performing the following sequence.

1. Optionally shift the tool.
2. Retract the tool up the tool axis a specified distance.
3. Position to the new rotary angles in rapid mode.
4. Plunge the tool back to its position prior to retraction.
5. Optionally shift the tool back to its original position.

Select *DISTANCE* at this prompt if you would like **PWorks** to perform these functions and retract the tool to a defined distance whenever the rotary axes take the longest route. This feature can save the part from being scrapped whenever radical movements in the rotary axes are output.

Selecting *PLANE* will retract the tool to a predefined plane. This plane is defined in the machine axes so that the tool end point will retract to this plane after being adjusted for any defined rotary tables.

Remember, **PostWorks** will only automatically retract the tool when linearization has been enabled (*LINTOL/ON*). The RETRACT/TOOL command can be used to set the automatic retract options in the part program.

7.2.10 Allow Tool Retraction With Linearization Disabled

Enter YES to this prompt if you want **PWorks** to perform the automatic tool retraction sequence described in the previous prompt when the rotary axes are taking the longest route and linearization is disabled. Entering NO will only retract the tool when tool retraction is enabled and linearization is active.

7.2.11 Add Tool Length To Retract Distance

Enter YES to this prompt if you want the tool length used as part of the distance to retract the tool when the rotary axes take the longest route to fulfill the tool axis. If you enter NO, then only the distance specified by at the following prompt will be used for the retract distance. This prompt will only be displayed when automatic tool retraction has been enabled.

7.2.12 Automatic Retract Distance (In Addition To Tool Length)

The value entered at this prompt, added to the currently programmed tool length, is the distance the tool will retract when the rotary axes take the longest route to fulfill the tool axis. Entering a value of 0 will cause only the tool length to be used as the retract distance. This prompt will only be displayed when automatic tool retraction has been enabled.

7.2.13 Automatic Retract Plane

Enter the default clearance plane to use when automatically retracting the tool when the rotary axes take the longest route to fulfill the tool axis. The retract plane should be entered using the format “i, j, k, d” or just “d”. If only a single number is entered, “d”, then a plane vector of “0, 0, 1” will be used.

This prompt will only be displayed when the default retract logic is set to *PLANE*.

7.2.14 Offset Tool Prior To Retract

You have the option of shifting the tool away from the part prior to having it retracted when the rotary axes take the longest route. The tool can be shifted either left or right of the forward direction, as specified by the user. Since **PostWorks** does not have any information on where the part is, due care should be used when using this feature so that the tool does not violate the part when shifting the tool. **PostWorks** will check for a closed angled wall situation and temporarily disable the tool offset feature when it determines that the tool will violate the part due to this condition. This is the only checking that is performed for part violation.

UP disables the tool offset feature. LEFT offsets the tool to the left of the tool path and RIGHT offsets it to the RIGHT of the tool path. The *RETRACT/TOOL,...,OFFSET* command can be used to change this setting during the programming of the part.

7.2.15 Default Offset Distance

Enter the default distance for offsetting the tool when it is shifted prior to retracting when the rotary axes take the longest route. It is recommended that a minimal distance be used, since the goal of this movement is only to minimize dwell marks.

7.2.16 Retract Feedrate For Automatic Retract

Enter the feed rate, in Feed Per Minute, for retracting the tool when the rotary axes take the longest route to fulfill the tool axis. Entering a value of 0 will cause the machine to retract the tool in rapid mode. This prompt will only be displayed when automatic tool retraction has been enabled.

7.2.17 Plunge Feedrate For Automatic Retract

Enter the feed rate, in Feed Per Minute, for plunging the tool back to its original position when the rotary axes take the longest route to fulfill the tool axis. Entering a value of 0 will cause the machine to plunge the tool in rapid mode. The prompt will only be displayed when automatic tool retraction has been enabled.

7.2.18 Blade Positioning Feedrate For Automatic Retract

Enter the feed rate, in Feed Per Minute, for rotating the blade to its calculated position when the tool is retracted due to the angular change of the blade direction exceeding the maximum specified. A value of 0 will cause the blade to be positioned in rapid mode. This prompt will only be displayed when the machine type is defined to be an Ultrasonic Cutter.

7.2.19 Offset Feedrate For Automatic Retract

Enter the feed rate, in Feed Per Minute, for shifting the tool prior to tool retraction when the rotary axes take the longest route to fulfill the tool axis. Entering a value of 0 will cause the machine to shift the tool in rapid mode. This prompt will only be displayed when automatic tool retraction and tool offsetting have been enabled.

7.2.20 Enter Logic To Use When Determining The Rotary Shortest Route

When two or more rotary axes are active there is a possibility that a tool axis vector can be satisfied using two or more different axis rotations. **PostWorks** will usually select the position which causes the least amount of rotary movement. Selecting the shortest route can be accomplished in different ways. The user has control over how the determination is made.

Enter the value which corresponds to the shortest route determination logic to use.

1. Combination of all rotary axis movements.
2. Comparison of largest rotary axis movements.
3. Summation of all rotary movements.
4. Using a specific rotary axis.

Selection 1 will combine and average the movements of all active rotary axes and use this as the delta movement. Selection 2 will compare the single largest movements from the different sets of rotary axis calculations. Selection 3 will total all rotary axis movements. Selection 4 will base the total rotary movement on a single user defined axis.

Example

Let's assume the following delta movement for two different sets of rotary axes which satisfy the programmed tool axis. The resultant delta movement calculations and the set chosen for each selection listed above are shown.

	Deltas		Delta used for comparison				(B-axis)
	A-axis	B-axis	1	2	3	4	
Set #1	50	90	103	90	140	90	
Set #2	110	0	110	110	110	0	
			---	---	---	---	
Set chosen			1	1	2	2	

The *ROTABL/SHORT* command can also be used to change the shortest route determination logic anywhere within the part program.

7.2.21 Controlling Axis For Shortest Route Determination

Enter the rotary axis number whose delta movements alone will be used to determine the shortest route taken. This prompt will only be displayed when 'Using a specific rotary axis' was chosen at the previous prompt.

7.3 Rotary Axes Look-Ahead

This section defines the rotary axes look-ahead features for machines configuration with at least two rotary axes such that when the secondary rotary is at 0 degrees, the primary rotary axis can rotate to any position without affecting the tool axis orientation to the part. An AC-style head is the most common rotary configuration with this feature and will be used as a reference in this documentation.

7.3.1 Does This Machine Require Look-Ahead Positioning

Some rotary axis configurations are such that when the secondary rotary is at 0 degrees, the primary rotary axis can rotate to any position without affecting the tool axis orientation to the part. An AC-style head is the most common rotary configuration with this feature and will be used as a reference in this documentation.

PostWorks contains a look-ahead provision for these types of machines, where it will scan the clfile when the secondary axis (A-axis) is at zero looking for the next change in the primary axis (C-axis). It will then position the C-axis during a move where the A-axis is at zero, so that it is in position for the next move.

If this machine has this style of rotary axes, then enter YES at this prompt. The actual look-ahead scanning features are controlled by the remaining prompts in this section and with the *ALIGN/AXIS* command. Entering AUTO will cause **PostWorks** to determine if the machine has AC-style rotaries. This method is compatible with previous versions of **PostWorks**.

Note: AUTO is only present as an option so that old MDF files will run in the same manner (shown below) as they did before, so it is not recommended to use this setting if you are changing any other look ahead options.

- Rotate to center of limit when positioning C-axis (does not scan clfile).
- Previous and current move must be at 0 degrees to position C-axis.
- Leave tool down when rotating C-axis.

7.3.2 Rotary Axes Look-Ahead Mode

Select the look-ahead method to use by default for AC-style rotaries. OFF disables the look-ahead feature, AUTO will look for a change in the C-axis when the A-axis is at zero, and RAPID will only perform the look-ahead when the A-axis is at zero and a rapid move is in effect.

AUTO is typically used with machines that have an AC-style rotary head. The RAPID look-ahead mode is typically used with machines with at least one rotary table or a nutating head.

7.3.3 Vector Component Deviation

Enter the tolerance to use for the I and J components of the tool axis vector to determine when the A-axis should be at zero. Whenever both of these components are less than this value, they will be set to zero. This feature can be used to help prevent the C-axis from swinging wildly as the A-axis is almost at zero.

7.3.4 Look-ahead For Limit Violations

Enter YES if the look-ahead feature should continue to scan the clfile until either the A-axis moves back to zero or an axis limit is reached. Enabling this feature allows the C-axis to be optimally positioned nearest the limit that will allow it the greatest amount of movement before actually violating a limit.

7.3.5 Allow Rotation Of Axis In Single Block

PostWorks will normally require that the previous move have the A-axis at zero along with the current move prior to scanning the clfile for the next C-axis movement. This method ensures that the C-axis will be positioned during other axis movement and will not dwell in-place during positioning.

Enter YES at this prompt if you require the C-axis always positions itself while the A-axis is at zero, even in a single block without any other axis movement. This will happen anyway if linearization is active, since it is necessary in keeping the tool within tolerance of the move, but by enabling it during look-ahead you have the option of retracting the tool during this positioning.

7.3.6 Retract Tool When Rotated In Single Block

This feature retracts the tool, positions the C-axis, and plunges the tool back to its original position when the C-axis is positioned in a single block during the look-ahead sequence. The positioning method used is the same that is used when retracting the tool due to the longest route being taken. The same distances and feed rates will be used.

Enter YES at this prompt if you want the tool retracted during C-axis only positioning. Entering NO will cause the C-axis to be positioned in-place.

7.4 Slowdown Spans

This section defines slowdown support, including post generated slowdown blocks and machine controlled slowdown codes.

7.4.1 Does The Control Support Slowdown Codes

Enter YES if the machine control will accept codes to enable and disable slowdown adjustments on the machine. If the machine does not support these codes and you want **PostWorks** to generate slowdown blocks, then enter NO at this prompt.

7.4.2 Should Slowdown Codes Replace Linear/Circular Codes

If you enter YES to this prompt, then normal linear and circular interpolation codes (G01, G02 for example) will not be output when slowdowns are enabled. The defined codes for the current slowdown mode will replace the normal interpolation codes. Slowdown codes will usually be output in conjunction with the normal interpolation codes.

This prompt will only be displayed when the machine supports slowdown codes.

7.4.3 Are Slowdown Codes Required On Every Active Motion Block

Enter YES if the active slowdown code is non-modal and must be forced out in every motion block when slowdowns are enabled. Slowdown enable codes are usually modal and need to be output only once when activated, with a slowdown off code being output when slowdowns are cancelled.

This prompt will only be displayed when the machine supports slowdown codes.

7.4.4 Number Of Slowdown Modes Supported

Enter the number of slowdown modes supported by the machine. Various modes of slowdown support usually consist of various methods of acceleration at the start of the move and deceleration at the end of the move. Some controls also support an automatic corner rounding mode, which should be considered a slowdown mode in **PostWorks**. You can enter from 1 to 4 slowdown modes. See the machine's programming guide for description of the various slowdown modes that are supported.

This prompt will only be displayed when the machine supports slowdown codes.

7.4.5 Minimum Angular Change To Generate Slowdown Span

Enter the minimum angular change, in degrees, that is required for a slowdown block to be output when slowdowns are enabled. Slowdowns will only be enabled when the directional angular change of a move is greater than the value entered here and **SLOWDN/ON** is in effect. Enter a value of 0 to enable slowdown blocks on every move between **SLOWDN/ON** and **SLOWDN/OFF**.

This prompt will only be displayed when the machine supports slowdown codes. the tolerance for slowdown spans will be used to determine when a slowdown blocks need to be output, when **PostWorks** generates the slowdown blocks.

7.4.6:9 Linear Slowdown Span Code #n

Enter the register and optional value that will enable slowdown mode 'n' for the machine control. This code will be output when slowdown mode 'n' is enabled and a linear interpolation block is output.

7.4.10:16 CLW Circular Slowdown Span Code #n

Enter the register and optional value that will enable slowdown mode 'n' when a circular interpolation move in the clockwise direction is output. If the machine control does not require separate codes for linear and circular slowdown codes, then enter the same code that was used for linear interpolation slowdown blocks.

7.4.11:17 CCLW Circular Slowdown Span Code #n

Enter the register and optional value that will enable slowdown mode 'n' when a circular interpolation move in the counter-clockwise direction is output. If the machine control does not require separate codes for linear and circular slowdown codes, then enter the same code that was used for linear interpolation slowdown blocks.

7.4.18 Slowdown Off Code

Enter the register and optional value that will disable all slowdown modes at the machine. This prompt will only be displayed when the machine supports a slowdown codes.

7.4.19 Default Slowdown Tolerance

Enter the cornering tolerance that should be maintained when **PostWorks** generates slowdown blocks. Slowdown spans will be calculated by **PostWorks** and output at a feed rate that will keep the tool within this tolerance.

When a move has a feed rate that is too fast to keep the tool within the specified tolerance, then **PostWorks** will output two motion blocks. The first block will contain a calculated span at the programmed feed rate and the second block will contain the rest of the programmed move at a slower feed rate.

The prompt will only be displayed when the machine does not support slowdown codes.

7.4.20 Default Slowdown Mode

Enter ON if you want slowdown blocks enabled by default at the start of the program, otherwise enter OFF. On machines that support slowdown codes, you will normally enter OFF. On machine that require post-processor generated slowdown spans, ON is the typical response.

7.5 Maximum Axis Departure

This section defines the maximum departure allowed for each axis and does it apply to *RAPID* move.

7.5.1 Apply Maximum Axis Departures During Rapid Moves

In most cases maximum axis departures are setup to deal with machine control limitations and therefore require even rapid moves to adhere to these maximum distances, but there are machine configurations that the departure distances are limited due to other circumstances and while only in cutting mode. For example, a multi-axis machine with an [AC style head](#). Since rapid moves are not cutting moves, answering NO to this prompt will not apply the maximum axis departures during rapid moves.

7.5.2:11 Maximum Axis Departure Allowed For 'Axis'

Enter the maximum distance that the specified axis can move in a single block. If a positive number is entered and the axis exceeds this movement, then the move will be broken up into

multiple blocks. If a negative value is entered and the axis exceeds the absolute value of this movement, then an error message will be output and no adjustment to the move will be made. Enter an extremely large number (99999999) if there is no such limitation.

7.6 Acceleration Blocks

This section defines the acceleration blocks if required.

7.6.1 Does The Control Require Acceleration Spans

Acceleration blocks within **PostWorks** are used to limit the velocity of the tool when accelerating in order to keep the torque on the tool within an acceptable range. They are not used to compensate for limitations of the machine control servos. The acceleration blocks should be enabled if there is a possibility that undue pressure can be applied to the tool during machine acceleration. Acceleration blocks are especially useful when transitioning from multi-axis moves, where the feed rates are slowed down dramatically due to the rotary axes speeds, to straight linear moves, where the feed rate is set back to the programmed feed rate.

7.6.2 Default Mode For Acceleration Blocks

Acceleration blocks can be either enabled (ON) or disabled (OFF) by default at the start of the program. The mode can be changed anytime by using the *SLOWDN/RAMP,ON* or the *SLOWDN/RAMP,OFF* commands.

7.6.3 Maximum Vector Velocity

Enter the maximum acceleration velocity of the machine. This will be used by **PostWorks** as the vector velocity and not as the velocity of any single physical axis. It is recommended that the velocity of the fastest axis be entered here as units per second. This value can be changed by using the *SLOWDN/RAMP,LIMIT,...* command.

7.6.4 Capped Vector Velocity

Enter the maximum acceleration velocity allowed for a move. **PostWorks** will calculate acceleration blocks based on this velocity as compared to the physical vector velocity of the machine. This value is entered in units per second. This value can be changed by using the *SLOWDN/RAMP,LIMIT,...* command.

7.6.5 Minimum Feed Rate Step

Enter the minimum feed rate step that will be output in an acceleration block. The feed rate in the acceleration block will be at least the previous feed rate value plus this amount. This value is

entered in units per minute. This value can be changed by using the *SLOWDN/RAMP,STEP...* command.

7.6.6 Maximum Feed Rate Step

Enter the maximum feed rate step that will be output in an acceleration block. The feed rate in the acceleration block will not be over the previous feed rate value plus this amount. This value is entered in units per minute. This value can be changed by using the *SLOWDN/RAMP,STEP...* command.

7.7 Transformation Blocks

This section defines the transformation blocks if required.

7.7.1 Does The Control Support Transformation Codes

Enter YES if the machine control can accept a transformation matrix that changes the default coordinate system. Transformation matrices can be used on machines with a rotating head to align the current working plane with the standard XY-plane, thereby allowing automatic cycles, circular interpolation, etc. to be output.

When transformation codes are supported, the *TRANS/matrix* and *TRANS/TOOL* commands are used to output the transformation matrix.

7.7.2 Are Transformation Base On Active Transformations

Some controls require that the transformation matrices be based on the default coordinate system, regardless if a modified coordinate system is in effect. Enter NO at this prompt in this case.

Other machine controls will reference the transformation matrix based on the active coordinate system. In this case, the matrices are considered accumulative in nature. Enter YES at this prompt, if the transformation matrix is based on the active coordinate system.

7.7.3 Enter Transformation Rotation Output Logic

1. Rotations are about a single vector.
2. Rotations are about the major axes in the ZYX order.
3. Rotations are about the major axes in the XYZ order.
4. Rotations are about the major axes output in separate blocks.
5. Use output rotary axes rotations.
6. Use negative output rotary axes rotations.

PostWorks will output transformation matrices in three different formats.

The first (1) format will output the XYZ translations and a single rotation about a defined vector. This format is typically used with the Fanuc controller.

The second (2) format consists of XYZ translations combined with XYZ rotations. The translations are applied first, with the rotations calculated in Z first, Y second, and X last. This format is typically used with the Siemens controller.

The third (3) format consists of XYZ translations combined with XYZ rotations. The translations are applied first, with the rotations calculated in X first, Y second, and Z last. This format is typically used with the Heidenhain controller.

The fourth (4) format consists of XYZ translations, but with each axis rotation being output in a separate block containing the enable transformation rotations code, the IJK components of the major axis that the rotation is to occur about, and the actual rotation. This is a hybrid of format (1) and format (2) in that a vector is output with the rotations, but the rotation must occur around a major axis. This format is typically used with the Mazatrol controller.

The fifth (5) and sixth (6) formats will use the active rotary axes positions as the transformation rotation values. The transformation rotations will be the same as the output rotary axes (the rotation angles will match up with the axis that each rotary axis rotates about), with the only exception in that the angles will be output in a reverse direction when format (6) is specified.

7.7.4 Use Table Rotations In Transformation Block Calculation

Controllers that are able to perform their own machine axes rotation calculations when the rotary axes move typically require that the table rotations be included in the transformation block calculations. Controllers that require that **PostWorks** perform the rotary axes adjustments for the linear axes will not typically require the table rotations in the transformation calculations, since these adjustments are already handled.

7.7.5 Use Output Rotary axis positions for coordinate transformations

This prompt will only be displayed when the transformation rotations are output using the rotary axes position values. Entering Yes to this prompt will create an internal matrix that will match the rotary transformations as they are output to the transformation block, the same as the rotary axis positions.

Entering No to this prompt will create an internal matrix that matches the standard method for calculating the transformation matrix when the rotary transformations are output around the major axes or a single vector. The Heidenhain controller requires this calculation method.

7.7.6 Allow Rotary Axes Movement When A Transformation Block Is Active

Answering No to this prompt will cause **PostWorks** to behave in the same manner as answering Yes to the previous prompt as described above when TRANS/TOOL,AUTO is in effect. The

positioning move will be output prior to the transformation block. If a transformation block is already active when the tool axis changes, then the transformation block will be canceled, the positioning move made, and then the new transformation block will be activated.

When TRANS/TOOL,NOW is in effect, then a warning message will be output whenever the tool axis changes and a transformation block is active.

7.7.7 Automatically Cancel Cutcom With Enable Transformations Block

Enter Yes at this prompt if cutter compensation should be canceled prior to outputting an enable transformations block and immediately enabled after this block.

7.7.8 Automatically Cancel Transformations Prior To Motion Block

Enter Yes at this prompt if the active transformation block should be canceled prior to outputting the programmed motion block when the transformation block is automatically calculated using the programmed tool end point and tool axis vector. Enabling this option will cancel the active transformation, output the motion block, and enable the new transformations.

Answer No at this prompt if the active transformation does not have to be deactivated or if code in the XFORM Macro will handle the canceling of the active transformation.

7.8 Transformation Output

This section defines the transformation output.

7.8.1 Enter Enable Translations Code

Enter the register and its value that define a translation block. The translation block will contain the XYZ translation values. Typical codes are G(68) or ATRANS.

7.8.2 Enter Enable Rotations Code

Enter the register and its value that define a rotation block, the rotation block will contain the XYZ rotations or a vector definition and a single rotation. It is possible that the translation block code and the rotation block code are the same. In this case, the translations and rotations will be output in the same block. Typical codes are G(68) or AROT.

7.8.3 Enter Disable Translations Code

Enter the register and its value that will disable the translation values. This code can be the same as the “Enable Translations” code, in which case it will be output in a block by itself, without the translation values. Typical codes are G(69) or ATRANS.

7.8.4 Enter Disable Rotations Code

Enter the register and its value that will disable the rotation values. This code can be the same as the “Enable Rotation” code, in which case it will be output in a block by itself, without the rotation values. Typical codes are G(69) or AROT.

7.8.5 Enter X-Translation Register

Enter the register that defines the X-translation value of a transformation matrix.

7.8.6 Enter Y-Translation Register

Enter the register that defines the Y-translation value of a transformation matrix.

7.8.7 Enter Z-Translation Register

Enter the register that defines the Z-translation value of a transformation matrix.

7.8.8 Enter X-Rotation Register

Enter the register that defines the X-rotation value of a transformation matrix. This prompt will only be displayed when the rotations are not about a single vector.

7.8.9 Enter Y-Rotation Register

Enter the register that defines the Y-rotation value of a transformation matrix. This prompt will only be displayed when the rotations are not about a single vector.

7.8.10 Enter Z-Rotation Register

Enter the register that defines the Z-rotation value of a transformation matrix. This prompt will only be displayed when the rotations are not about a single vector.

7.8.11 Enter Rotation Vector I-Component Register

Enter the register that defines the I-component of a transformation matrix rotation vector. This prompt will only be displayed when the rotations are about a single vector.

7.8.12 Enter Rotation Vector J-Component Register

Enter the register that defines the J-component of a transformation matrix rotation vector. This prompt will only be displayed when the rotations are about a single vector.

7.8.13 Enter Rotation Vector K-Component Register

Enter the register that defines the K-component of a transformation matrix rotation vector. This prompt will only be displayed when the rotations are about a single vector.

7.8.14 Enter Rotation Register

Enter the register that defines the rotation value about the transformation matrix vector. This prompt will only be displayed when the rotations are about a single vector.

7.8.15 Enter Non-positioning Behavior Code

Enter the register and its value that will be output to set the positioning behavior of the transformation matrix to keep the tool positioned at its current location. This is the only mode currently supported. The typical text of the register is “STAY” for the Heidenhain controller.

7.8.16 Output XYZ Translation Registers When Canceling Translations

Some controllers handle translations in a more standard way (such as fixture offsets) than through the transformation process. When output using a translation block rather than a transformation block, then it is usually required to output the XYZ translation registers with a value of 0 when canceling translations. In this case, enter YES at this prompt. If translations are canceled using a transformation cancel code and the XYZ translation registers are not required, then enter NO.

7.8.17 Output Transformation Block In Relation To Current Motion Block

1. Output transformation block after motion block.
2. Output transformation block prior to motion block.
3. Transformation block is output prior to and after motion block output.

This prompt determines when the transformation codes should be output or the XFORM Macro called in relation to when the motion block is output.

Selecting “1” at this prompt will cause **PostWorks** to first output a positioning move when the tool axis changes and then output the transformation block when TRANS/TOOL,AUTO is in effect. For example, in the following sequence.

```
N10 G01 X2.0 Y-2.0 Z3.0 B45.0 F100.0$  
N20 G68 I0.0 J-1.0 K0.0 R-45.0$
```

When selecting “2” at this prompt, the transformation block will be output first and then the positioning move, as follows.

```
N10 G68 I0.0 J-1.0 K0.0 R-45.0$  
N20 G01 X2.0 Y-2.0 Z3.0 F100.0$
```

“3” is typically only selected at this prompt when an XFORM Macro is defined. It causes the XFORM Macro to be called prior to the motion block and then again after the motion block. This setting can be used to output the motion block in the middle of the transformation block output, such as when the transformation block needs to be canceled prior to outputting the motion block and then enabled after the motion block is output.

7.9 Exit

Exits the Motion Adjustments section and returns you to the previous section.

CHAPTER 8 Milling Cycles

8.0 Walk Through Milling Cycles

This selection will lead you through all of the sections used for defining milling cycles, including cycle definition codes, cycle parameter codes, positioning options, User Defined Blocks for cycles, and machining time calculations.

8.1 Cycle Definition Codes

This section defines the cycle definition codes, for example, the codes output for *CYCLE/OFF*, *CYCLE/DRILL*, *CYCLE/BORE*, etc.

8.1.1 Does This Machine Support Automatic Cycles

Enter YES if this machine control will accept automatic (canned) cycle blocks and will perform the cycle simulation itself. An automatic cycle block usually contains a cycle definition code, cycle positions, and parameter codes, which will define the cycle to be performed.

Example:

```
G81 X1.0 Y2.0 Z-.3 R.1 F7.5$
```

If you enter NO at this prompt, then **PostWorks** will output a series of linear moves that will simulate each cycle.

8.1.2 Automatic Cycles Are Allowed In All 3 Planes (XY,YZ, ZX)

Enter NO if your machine only supports automatic cycles in the XY plane. In this case, all cycle sequences in the YZ and ZX planes will be simulated by outputting a series of linear moves.

8.1.3 Output Cycles As Macro Call

Some machine controllers include support for automatic cycles as macros or subroutines, the most notable being the Siemens 840D controllers. Enter YES to this prompt if your machine requires cycles to be called in the following format. See *CYCLE/...PARAMS,m1*. for details of how to use this feature.

Example:

```
MCALL CYCLE81 (p1, p2, ..., pn)
```

8.1.4 Macro Call Code

Enter the register and its value that is used to call modal automatic cycle macros. The text “MCALL” is usually used, so a register can be setup to have these characters as the starting string with no digits to the left or right of the decimal point (strictly text output).

8.1.5 Enter CYCLE/OFF Code

Enter the register and its value that will cancel an automatic cycle sequence. This code is usually G(80).

8.1.6:18 Enter CYCLE/Mode Code

Enter the register and its value that will enable an automatic cycle for this mode at the machine. This code will be output on *CYCLE/mode* sequences. If you do not enter a value here, then **PostWorks** will assume that the machine does not support this *CYCLE* mode and will simulate the cycle whenever *CYCLE/mode* is programmed.

Following is a list of valid CYCLE modes and the code that is normally associated with them.

BORE /	G(86)	DRILL /	G(81)	SHIFT /	G(76)
BORE7 /	G(87)	FACE /	G(82)	TAP /	G(84)
BORE8 /	G(88)	MILL /	G(79)	THRU /	G(87)
BORE9 /	G(89)	REAM /	G(85)	DEEP /	G(83)
REVERS/	G(74)				

8.1.19 XY-plane Selection Code

Enter the register and its value that selects the XY automatic cycle plane. This is usually G(17).

8.1.20 ZX-plane Selection Code

Enter the register and its value that selects the ZX automatic cycle plane. This is usually G(18).

8.1.21 YZ-plane Selection Code

Enter the register and its value that selects the YZ automatic cycle plane. This is usually G(19).

8.1.22 Enter RAPID Cycle Interrupt Code

A *RAPID* command programmed within any cycle sequence will suspend the cycle to output the *RAPID* move. The move programmed with *RAPID* will be output as a positioning move and the cycle sequence will be reinstated immediately after this move.

Some machines have a special code that supports this feature. Enter the register and its value that will interrupt the active cycle for a single move. If the machine does not support this feature, then leave this prompt blank and **PostWorks** will cancel the cycle and output the move using the rules governing *RAPID* moves.

8.1.23 Enter RETRACT/ON Code

Enter the register and its value that will be output with the *RETRACT/ON* command. *RETRACT/ON* specifies for the tool to fully retract to the level it was at prior to the cycle being activated.

The *RETRACT/ON* code is usually G(98). Please note that not all machines support this feature.

8.1.24 Enter RETRACT/OFF Code

Enter the register and its value that will be output with the *RETRACT/OFF* command. *RETRACT/OFF* specifies for the tool to retract to the rapto plane as defined by the '*RAPTO,r*' cycle parameter.

The *RETRACT/OFF* code is usually G(99). Please note that not all machines support this feature.

8.2 Cycle Parameter Codes

This section defines the cycle parameter codes, for example the codes output for '*RAPTO,r*', '*STEP,peck*', etc.

8.2.1 Enter Final Depth Register

A) Standard Register

Enter the register that will contain the final depth for an automatic cycle. Normally you would leave this prompt blank and **PostWorks** will use the appropriate axis register for the final depth value.

Some machines, though, require a value other than the final position to define the final cycle depth. In this case enter a register other than an axis register that will contain the final depth value.

B) Macro Parameter

If automatic cycles are output as macro calls, then the macro parameter number that contains the final depth should be entered at this prompt. For example, in the following macro call, the final depth is the 4th parameter, so 4 should be entered at this prompt.

CYCLE81 (x,x,x,DP,x)

There are some cases when the same cycle register will be in different parameter positions in different cycle macros. You can use the User Defined Block definitions for each cycle to rearrange the order of the parameters.

Note: Per Siemens 840D programming manual, the 4th parameter is for the absolute depth output and the 5th parameter is for the incremental depth output. **PWorks** will automatically output the incremental depth to the 5th parameter instead of the 4th parameter. Therefore always specify the absolute parameter position for the final depth register and disregard the setting of the final depth mode ([Menu 8.3.5 : Mode for final depth value.](#)) in the corresponding mdf file.

8.2.2 Enter Top-of-Part Register

A) Standard Register

Normally the position at the top of the part is not output in a cycle sequence. Some machines, though, require this position in order to perform the cycle simulation. In this case enter a register other than an axis register that will contain the position at the top of the part.

If you do not enter a register at this prompt, then this position will not be output in a cycle sequence.

B) Macro Parameter

If automatic cycles are output as macro calls, then the macro parameter number that contains the top of part should be entered at this prompt. For example, in the following macro call, the top of part is the 2nd parameter, so 2 should be entered at this prompt.

CYCLE81 (x,RFP,x,x,x)

There are some cases when the same cycle register will be in different parameter positions in different cycle macros. You can use the User Defined Block definitions for each cycle to rearrange the order of the parameters.

8.2.3 Enter RAPTO Plane Register

A). Standard Register

Enter the register that will contain the rapto plane value, as specified using the '[RAPTO,r](#)' cycle parameter. This register is usually 'R'.

B) Macro Parameter

If automatic cycles are output as macro calls, then the macro parameter number that contains the rapto plane should be entered at this prompt. For example, in the following macro call, the rapto plane is the 3rd parameter, so 3 should be entered at this prompt.

CYCLE81 (x,x,SDIS,x,x)

There are some cases when the same cycle register will be in different parameter positions in different cycle macros. You can use the User Defined Block definitions for each cycle to rearrange the order of the parameters.

8.2.4 Enter Threads Per Inch Register

A) Standard Register

Some machines allow for thread tapping cycles to contain the number of threads per inch/mm, instead of a feed rate, to control the tapping feed. If the machine has this feature, then enter the register that will contain the threads per inch/mm value for tapping cycles.

B) Macro Parameter

If automatic cycles are output as macro calls, then the macro parameter number that contains the thread size should be entered at this prompt. For example, in the following macro call, the thread size is the 8th parameter, so 8 should be entered at this prompt.

CYCLE84 (x,x,x,x,x,x,x,MPIT)

There are some cases when the same cycle register will be in different parameter positions in different cycle macros. You can use the User Defined Block definitions for each cycle to rearrange the order of the parameters.

8.2.5 Enter DWELL Parameter 1 Register

A) Standard Register

Enter the register that will contain the dwell time specified by the '*DWELL,dwll*' cycle parameter. '*dwll*' usually defines the machine dwell at the bottom of the hole.

B) Macro Parameter

If automatic cycles are output as macro calls, then the macro parameter number that contains the dwell value should be entered at this prompt. For example, in the following macro call, the dwell time is the 6th parameter, so 6 should be entered at this prompt.

CYCLE85 (x,x,x,x,x,DTB)

There are some cases when the same cycle register will be in different parameter positions in different cycle macros. You can use the User Defined Block definitions for each cycle to rearrange the order of the parameters.

8.2.6 Enter DWELL Parameter 2 Register

A) Standard Register

Enter the register that will contain the dwell time specified by the '*DWELL,dwl2*' cycle parameter. '*dwl2*' usually defines the machine dwell at the top of the hole for multiple pass cycles.

B) Macro Parameter

If automatic cycles are output as macro calls, then the macro parameter number that contains the second dwell value should be entered at this prompt. For example, in the following macro call, the dwell time is the 10th parameter, so 10 should be entered at this prompt.

CYCLE83 (x,x,x,x,x,x,x,x,DTS)

There are some cases when the same cycle register will be in different parameter positions in different cycle macros. You can use the User Defined Block definitions for each cycle to rearrange the order of the parameters.

8.2.7 Enter STEP Parameter 1 Register

A) Standard Register

Enter the register that will contain the initial depth of cut as specified by the '*STEP,peck1*' cycle parameter. '*peck1*' usually defines either the depth of the first cut or the depth of all cuts in a multiple pass cycle.

B) Macro Parameter

If automatic cycles are output as macro calls, then the macro parameter number that contains the step value should be entered at this prompt. For example, in the following macro call, the step value is the 6th parameter, so 6 should be entered at this prompt.

CYCLE83 (x,x,x,x,x,FDEP)

There are some cases when the same cycle register will be in different parameter positions in different cycle macros. You can use the User Defined Block definitions for each cycle to rearrange the order of the parameters.

8.2.8 Enter STEP Parameter 2 Register

A) Standard Register

Enter the register that will contain the final depth of cut as specified by the '*STEP,peck2*' cycle parameter. '*peck2*' usually defines the depth of the final cut in a multiple pass cycle.

B) Macro Parameter

If automatic cycles are output as macro calls, then the macro parameter number that contains the second step value should be entered at this prompt. For example, in the following macro call, the step value is the 8th parameter, so 8 should be entered at this prompt.

CYCLE83 (x,x,x,x,x,x,x,DAM)

There are some cases when the same cycle register will be in different parameter positions in different cycle macros. You can use the User Defined Block definitions for each cycle to rearrange the order of the parameters.

8.2.9 Enter Single Parameter OFFSET Register

A) Standard Register

Enter the register that will contain the offset value when the '*OFFSET,off1*' cycle parameter is specified without '*off2*'. '*off1*' by itself usually defines the offset distance opposite the spindle orientation direction for shifting the tool in a fine boring or back boring cycle.

B) Macro Parameter

If automatic cycles are output as macro calls, then the macro parameter number that contains the offset value should be entered at this prompt. For example, in the following macro call, the offset value is the 8th parameter, so 8 should be entered at this prompt.

CYCLE86 (x,x,x,x,x,x,x,RPA)

There are some cases when the same cycle register will be in different parameter positions in different cycle macros. You can use the User Defined Block definitions for each cycle to rearrange the order of the parameters.

8.2.10 Enter X-axis OFFSET Register

Enter the register that will contain the X-axis offset value as specified by the '*OFFSET*' cycle parameter. '*off1*' will contain the X-axis offset value when the cycle is programmed in the XY plane. '*off2*' will contain the X-axis offset value when the cycle is programmed in the ZX plane. This offset value will usually be used in a fine boring or back boring cycle.

8.2.11 Enter Y-axis OFFSET Register

Enter the register that will contain the Y-axis offset value as specified by the '*OFFSET*' cycle parameter. '*off2*' will contain the Y-axis offset value when the cycle is programmed in the XY plane. '*off1*' will contain the Y-axis offset value when the cycle is programmed in the YZ plane. This offset value will usually be used in a fine boring or back boring cycle.

8.2.12 Enter Z-axis OFFSET Register

Enter the register that will contain the Z-axis offset value as specified by the '*OFFSET*' cycle parameter. '*off1*' will contain the Z-axis offset value when the cycle is programmed in the ZX plane. '*off2*' will contain the Z-axis offset value when the cycle is programmed in the YZ plane. This offset value will usually be used in a fine boring or back boring cycle.

8.2.13 Should XY Register Always Be Output As OFFSET Registers

Enter YES if the offset registers for the X and Y axes contain the offset values, as specified in the '*OFFSET*' cycle parameter, no matter what plane the cycle is activated in. If you enter NO to this prompt, then the offset registers corresponding to the axes in the active cycle plane will be output.

8.2.14 Enter RTRCTO Parameter 1 Register

A) Standard Register

Enter the register that will contain the value specified by the '*RTRCTO,ret*' cycle parameter. '*ret*' usually defines the level above each cut to retract to in a multiple pass cycle.

B) Macro Parameter

If automatic cycles are output as macro calls, then the macro parameter number that contains the retract plane should be entered at this prompt. For example, in the following macro call, the retract plane is the 1st parameter, so 1 should be entered at this prompt.

CYCLE81 (RTP,x,x,x,x)

There are some cases when the same cycle register will be in different parameter positions in different cycle macros. You can use the User Defined Block definitions for each cycle to rearrange the order of the parameters.

8.2.15 Enter RTRCTO Parameter 2 Register

A) Standard Register

Enter the register that will contain the value specified by the '*RTRCTO,plng*' cycle parameter. '*plng*' usually defines the level above the previous cut to position at after retracting the tool in a multiple pass cycle.

B) Macro Parameter

If automatic cycles are output as macro calls, then the macro parameter number that contains the retract value should be entered at this prompt. For example, in the following macro call, the retract value is the 15th parameter, so 15 should be entered at this prompt.

CYCLE83 (x,x,x,x,x,x,x,x,x,x,x,x,x,x,VRT)

There are some cases when the same cycle register will be in different parameter positions in different cycle macros. You can use the User Defined Block definitions for each cycle to rearrange the order of the parameters.

8.2.16 Enter REPEAT Parameter Register

Enter the register that will contain the value specified by the '*REPEAT,rep*' cycle parameter. '*rep*' usually defines the number of repetitions for each cycle block to perform at equal increments.

8.3 Cycle Positioning Parameters

This section contains support for unidirectional positioning, the format for certain positioning parameters, such as the final position and top of part modes, step parameter output, and defines the default rapid rate for cycles.

8.3.1 Output A Unidirectional Move Prior To Each Cycle Point

Enter YES if the machine requires the axes to always move in the same direction when positioning above the hole for a cycle operation.

When unidirectional positioning has been enabled, **PostWorks** will cancel the active cycle, move to a position at a specified distance and direction from the hole, move to the hole and enable the cycle.

This feature is usually only required on older machines that require the axes to always position in the same direction in order to hold tolerance.

8.3.2 Enter Positioning Direction Vector

Enter the direction that the cycle axes must move when positioning above the hole for a cycle operation. Enter the direction in the format of a 3 dimensional vector, 'i, j, k'. This prompt will only be displayed when unidirectional positioning has been enabled.

8.3.3 Enter Positioning Distance

Enter the distance away from the hole that the tool should position at prior to moving to the hole and performing the cycle, when making a unidirectional positioning move.

8.3.4 Position Above Hole Prior To Issuing CYCLE/DEEP Block

Some machines require that the tool already be positioned above the hole prior to programming a chip breaking cycle block. If you enter YES at this prompt, then **PostWorks** will automatically cancel the cycle and position above the hole prior to outputting the *CYCLE/DEEP* parameter block.

Enter NO to this prompt if your machine does not have this requirement. Machines that require this feature usually use some of the axis registers to specify cycle parameters.

8.3.5 Mode For Final Depth Value

Enter the number that corresponds to the output format for the final depth value in a cycle block.

1. Same as currently programmed mode.
2. Always absolute.
3. Always incremental.

Some machines, though they support both absolute and incremental modes, require that the final depth in a cycle block be programmed in only one mode. When this is the case, it is usually the incremental mode.

8.3.6 The Final Depth Distance Is Relative To

Enter the number that corresponds to the plane which the final depth will reference when it is output in incremental mode. The three modes are:

- 1 = The incremental final depth value will come from the rapto plane as defined by the '*RAPTO,r*' cycle parameter.
- 2 = The incremental final depth value will come from the clearance plane, defined as the tool position prior to activating the cycle.
- 3 = The incremental final depth value will come from the top of the part, as defined in the *GOTO* point motion for this hole. Only use this option if the top of the top of the part is output in a cycle block.

8.3.7 Mode For Top-of-Part

Enter the number that corresponds to the output format for the top of the part, when this value is output on a cycle block.

1. Same as currently programmed mode.
2. Always absolute.
3. Always incremental.

Some machines, though they support both absolute and incremental modes, require that the top of the part in a cycle block be programmed in only one mode. This prompt will only be displayed when a register has been defined for the top of the part in a cycle block.

8.3.8 Output Format For STEP (Peck) Parameters

Enter the number that corresponds to the output format to use with *STEP* cycle parameters.

- 1 = The cycle *STEP* parameters will be output with the same sign (positive or negative) as they contained on the input *CYCLE* command.
- 2 = The cycle *STEP* parameters will always be output as positive values.
- 3 = The cycle *STEP* parameters will always be output as negative values.
- 4 = **PostWorks** will calculate the number of cuts required to satisfy the depth of each cut as specified by the *STEP* cycle parameter(s). The number of cuts required will be output in the cycle block.

8.3.9 Default Rapid Rate For Cycles

Enter the feed rate in Feed Per Minute to use when calculating the machining time for rapid moves during a cycle sequence. This feed will actually be used for rapid moves when **PostWorks** generated (manual) cycles are output. Specifying a value of 0 will cause the machine's rapid rate to be used.

8.4 Rapto And Retract Parameters

This section defines the processing of the rapto and retract planes for automatic cycles. It contains support for the output format for rapto/retract planes, the relationship of the rapto/retract planes to the top of the part, and when to retract the tool.

8.4.1 Mode For Rapto Plane Value

Enter the number that corresponds to the output format for the rapto plane value in a cycle block, as specified by the '*RAPTO,r*' cycle parameter.

1. Same as currently programmed mode.
2. Always absolute.
3. Always incremental.
4. Incremental distance from Top-of-part.

Some machines, though they support both absolute and incremental modes, require that the rapto plane in a cycle block be programmed in only one mode. When this is the case, it is usually the absolute mode.

8.4.2 Enter Calculation Type For Rapto Plane

Enter the number that corresponds to the calculation method for the cycle rapto plane when both primary and secondary axes are supported for any of the linear axes.

1. Rapto plane is based on input coordinate.
2. Rapto plane is based on active linear axis.

Calculation method 1 will base the rapto plane value solely on the input coordinate with no consideration for the active linear axis or the position of the stable axis.

Calculation method 2 bases the rapto plane value on the active linear axis taking into consideration the calculation method for the secondary axis and the position of the stable axis.

8.4.3 Enter Gage Height Difference Between Rapto Plane And Top Of Part

Some machines, such as Cincinnati, have an automatic rapto plane feature that will compensate for a clearance distance above the part when positioning above the hole in a canned cycle sequence.

Enter the automatic clearance value at this prompt. **PostWorks** will use this value when calculating the rapto plane output. The *RAPTO* cycle parameter should still be specified as an incremental distance from the top of the part, without any consideration given to the automatic clearance value.

Enter a value of 0 if the machine does not have an automatic rapto plane feature.

8.4.4 Mode For Retract Plane Value

Enter the number that corresponds to the output format for the retract plane value in a cycle block.

1. = Output the retract plane, as specified in the '*RTRCTO*' cycle parameter, as an absolute position.
2. = Output the retract plane as in incremental distance from the rapto plane.

- 3 = Output the retract plane as an incremental distance from the top of the part.

Some machines, though they support both absolute and incremental modes, require that the retract plane in a cycle block be programmed in only one mode.

8.4.5 Enter Calculation Type For Retract Plane

Enter the number that corresponds to the calculation method for the cycle retract plane when both primary and secondary axes are supported for any of the linear axes.

1. Retract plane is based on input coordinate.
2. Retract plane is based on active linear axis.

Calculation method 1 will base the retract plane value solely on the input coordinate with no consideration for the active linear axis or the position of the stable axis.

Calculation method 2 bases the retract plane value on the active linear axis taking into consideration the calculation method for the secondary axis and the position of the stable axis.

8.4.6 Enter The Final Retract Logic To Use

Enter the number that corresponds to the retract positioning move that will be performed at the end of a cycle sequence.

1. The *RETRCT* plane will be used.
2. Always retract to the clearance plane.
3. Always retract to the rapto plane.

PostWorks always needs to know where the tool is positioned. Since canned cycle motion is performed by the machine control and not by **PostWorks**, you must define the retract method that the machine will use.

8.4.7 Should The 1st RTRCTO Value Override Normal Retract Logic

Enter YES if the '*RTRCTO,ret*' cycle parameter specifies the actual position that the tool will retract to after completing a cycle sequence. In this case, **PostWorks** will use the 'ret' position as the final position in a cycle sequence.

8.4.8 Output A Retract Level Code ...

Some machines will always retract to the rapto plane at the end of a canned cycle sequence, but will accept a code, as specified by the *RTRCTO* cycle parameter, to be output when a plane other than the rapto plane is used to retract the tool. Other machines require a retract code on each cycle definition block.

Enter the number of the choice which conforms to your machine.

1. When *RTRCTO* is specified or *RETRCT/ON* is in effect.
2. Only when the *RTRCTO* parameter is specified.
3. Always.

A retract code will always be output whenever the *RTRCTO* parameter is specified on the *CYCLE* command. Selecting option 1 will also output a retract code whenever *RETRCT/ON* is in effect (the tool retracts to the clearance plane after each cycle operation). Option 2 will ignore the status of the *RETRCT* command when determining when to output a retract code. Option 3 will output a retract code on the cycle definition block when *RTRCTO* is specified, *RETRCT/ON* is in effect, or *RETRCT/OFF* is in effect.

Usage of option 3 allows machines which require retract codes to use source command and resulting cycle motion to be compatible with machine which use a modal setting for retraction to the clearance or rapto plane.

8.4.9 Output A Retract Block When The Rapto Plane Increases

Enter YES if the machine will not automatically adjust for a rapto plane level that is greater than the previous level. In this case, **PostWorks** will output a tool retract block whenever the rapto plane level increases.

8.4.10 Enter The Format For A Retract Block

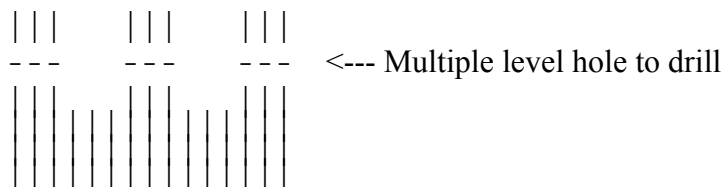
Enter the number that corresponds to the format of a tool retract block. A tool retract block will be output whenever the rapto plane level increases and it was enabled in the previous prompt.

1. The cycle will be cancelled and a rapid move will be output.
2. The cycle will be cancelled and a new retract plane will be output.

In both cases, the active cycle mode will be reinstated directly after the retract block.

8.4.11 Retract Tool At Each Level Of CYCLE/THRU

On some machines, the *CYCLE/THRU* canned cycle will generate motion to drill through multiple levels of a part, instead of performing a deep hole drilling cycle with multiple pecks. On a multiple level drill cycle, the part will look something like pictured below.



Enter NO if the *CYCLE/THRU* command will initiate a multiple level drill cycle. In this case, **PostWorks** will leave the tool at the hole bottom and not retract it after the cycle sequence.

8.4.12 Enter Motion Alteration For Cycle Generated Rapid Moves

Enter the value that corresponds to the way you want the cycle generated rapid positioning moves output to the control tape. The cycle positioning moves occur when transitioning between one cycle location and the next cycle location.

1. Use the default rapid alteration setting.
2. Alter rapid moves along the X-axis.
3. Alter rapid moves along the Y-axis.
4. Alter rapid moves along the Z-axis.
5. Alter rapid moves along major tool axis.

The default rapid alteration setting is used for all other rapid moves and is defined in the Rapid Set-up section of the Feedrate and Rapid section of **MPost**.

8.4.13 Retract/Plunge Rapid Moves Along Tool Axis

This prompt will only be display when cycle rapid moves are modified along the major tool axis. Enter YES if you want rapid motion to retract or plunge (depending on the movement of the major axis) along the tool axis, instead of moving straight along the major axis.

8.5 Cycle Block Output

This section defines the output format for cycle blocks, including when to output the cycle parameters.

8.5.1 Force Cycle Definition Code On The Initial Cycle Block

Enter YES if the first location in a canned cycle must contain the definition code for the active cycle sequence. **PostWorks** will force out the cycle definition code on the cycle block immediately following the *CYCLE* command.

If you enter NO at this prompt, then the normal output control for the cycle definition register will be used to determine when it will be output.

8.5.2 Force Active Linear Axes On The Initial Cycle Block

Enter YES if the first location in a canned cycle must contain both of the positioning axis locations, even if the position has not changed. **PostWorks** will force out both positioning axis registers on the cycle block immediately following the *CYCLE* command.

If you enter NO at this prompt, then the normal output control for the positioning axis registers will be used to determine when they will be output.

8.5.3 Force Linear Axes Output On Every Cycle Block

Enter YES if every location programmed during a cycle sequence must contain both of the positioning axis (XY) locations, even if the position has not been changed. **PostWorks** will force out both positioning axis registers on every positioning block during the cycle sequence.

If you enter NO at this prompt, then the normal output control for the positioning axis registers will be used to determine when they will be output.

8.5.4 Force Input Cycle Parameters On Every Cycle Block

Enter YES if every location programmed during a cycle sequence should be a full cycle definition block. A cycle definition block will contain the cycle definition code, axes position, final depth, feed rate, and the cycle registers that were programmed on the *CYCLE* command.

8.5.5 Output A Feedrate With Cycle Blocks

Enter YES if you can program a feed rate in a cycle sequence. If you enter NO to this prompt, then a feed rate code will not be output with the cycle sequence and the last programmed feed rate will be used for the cycle motion.

SEPARATE specifies that the feed rate should be output in a block prior to the actual cycle block. This setting is only valid when cycles are output as a [Macro call](#) since they work a bit differently than standard blocks. To force the feed rate out in a block prior to the cycle block with standard cycles define a [User Defined Block](#) with 'F,\$' specified at the beginning of the block.

8.5.6 Force Feedrate On The Initial Cycle Block

Enter YES if the first location in a canned cycle must contain a feed rate code. **PostWorks** will force out the feed rate code on the cycle block immediately following the *CYCLE* command.

If you enter NO at this prompt, then the normal output control for the feed rate register will be used to determine when it will be output.

8.5.7 Should TPI Value Be Output As A -ve Number With CYCLE/REVERS

Enter YES if the machine control requires that reverse tapping cycles (CYCLE/REVERS) contain a negative value for the [Threads Per Inch register](#). Most controls support a separate cycle code for reverse tapping. In this case enter NO.

8.5.8 Should A Full Cycle Block Be Output When Depth Changes

Enter YES if a full cycle definition block should be output whenever the absolute final depth position changes. A cycle definition block will contain the cycle definition code, axes position, final depth, feed rate, and the cycle registers that were programmed on the *CYCLE* command.

8.5.9 Automatically Lock Feedrate/Spindle Overrides With CYCLE/TAP

Enter YES if **PostWorks** should automatically output the disable feed rate and spindle override codes at the beginning of a manual thread tapping cycle. The overrides will be re-enabled after the tapping motion is completed.

This feature is used in order to prohibit the operator from changing the tool and spindle feeds at the machine.

8.5.10 Force Linear Interpolation Code After CYCLE/OFF

Enter YES if the linear interpolation code should be forced out on the first motion block following a *CYCLE/OFF* command.

If you enter NO at this prompt, then the normal output control for the linear interpolation register will be used to determine when it will be output.

8.6 Cycle User Defined Blocks

This section defines the *User Defined Block* to use for each supported automatic cycle sequence.

8.6.1 CYCLE/OFF User Defined Block

Enter the number of the *User Defined Block* you want to be output when the *CYCLE/OFF* command is programmed. Enter a value of 0 if the cycle cancellation block does not require a special format, other than having a *CYCLE/OFF* code.

8.6.2:14 CYCLE/Mode User Defined Block

This prompt will be displayed for every supported automatic cycle mode; *BORE*, *DRILL*, *TAP*, etc.

A) Standard Register

Enter the number of the User Defined Block that defines the output format for the *CYCLE/mode* definition block. User Defined Blocks that are used for *CYCLE* statements differ from their normal usage. The normal processing of a User Defined Block will force

out each code defined as part of the block, whereas *CYCLE* User Defined Blocks only define the order of the codes to be output in multiple blocks. The codes contained in a *CYCLE* User Defined Block will not necessarily be output.

For example, when a cycle block needs to be output in the following format, then you would specify a User Defined Block as described.

Cycle Block Format	User Defined Block description
G79 I1__ J1__ R__ \$	G(79), I1, J1, R, \$
G81 X__ Y__ Z__ F__ \$	

In the above example, a G79 cycle parameter definition block, containing, for example, both '*STEP*' cycle parameters and the '*RAPTO,r*' cycle parameter, is required prior to the actual cycle block. The User Defined Block for this example defines only the G79 block terminated with the End-of-Block character. All register that are not in the User Defined Block will be output in the following block. If the I1, J1 and R register are not output, then the entire G79 block will not be output.

The following register descriptors have special meaning when used in a *CYCLE* User Defined Block.

X	=	Both cycle positioning axis registers.
Z	=	Final depth register.
F	=	Feed rate register.

If the cycle definition block is not required to be output in more than a single block containing all of the cycle parameters, then enter a value of 0.

B) Macro Parameter

User defined blocks for Macro Parameters differ from other [User Defined Block](#), in that the order of the cycle [macro parameters](#) are entered in the User defined block rather than the registers. For example, a user defined block #4 for a cycle statement that is output as a macro call may read as follows.

```
#4 1,2,3,4,5,8,12,7
```

This will cause the cycle block to be output as follows.

```
CYCLE83 (parm-1,parm-2,parm-3,parm-4,parm-5,parm-8,parm-12,parm-7)
```

```
CYCLE83 (x,x,x,x,x,x,x,x,x,x,x,x,x,x,VRT)
```

8.7 Cycle Machining Times

This section defines the machining time calculation to use for each supported automatic cycle. You can choose a separate machining estimation based on **PostWorks** generated cycles for each supported automatic cycle.

8.7.1:13 Enter CYCLE/Mode Time Calculations

Enter the manual cycle mode that should be use for estimating the machining time for an automatic *CYCLE/mode* sequence. You can enter any of the following manual cycle modes.

BORE	BORE9	FACE	REVERS	THRU
BORE7	DEEP	MILL	SHIFT	
BORE8	DRILL	REAM	TAP	

See the **PWorks** manual cycle commands for a description of the simulated motion that will be used to estimate the canned cycle time.

This prompt will be displayed for every supported automatic cycle mode; *BORE*, *DRILL*, *TAP*, etc.

8.7.14 Use The First STEP Parameter In Time Calculations

Enter YES if the '*STEP,peck1*' parameter specifies the depth of the first cut for multiple pass cycles. If '*peck1*' specifies some other value, such as the number of cuts to make, then enter NO at this prompt, or else the machining time estimation for multiple pass cycles will be adversely affected.

8.7.15 Use The Second STEP Parameter In Time Calculations

Enter YES if the '*STEP,peck2*' parameter specifies the depth of the final cut for multiple pass cycles. If '*peck2*' specifies some other value, such as a retract value, then enter NO at this prompt, or else the machining time estimation for multiple pass cycles will be adversely affected.

8.7.16 Estimate Delta Distances During Automatic Cycles

Normally **PostWorks** will only use the output blocks during automatic cycles to calculate the machining distances (axis delta movements), which can be useful when comparing the [Slide Motion Recaps](#) from one run to another, since the delta summations will be different when comparing automatic cycle output to simulated cycle output. At times it is necessary to see a more exact estimate of the machining distances during automatic cycles, for example when estimating tool life.

Enter YES at this prompt if the total machining distances during automatic cycles should be included in the axis delta summations kept by **PostWorks**. Please note that only the positioning move, feed in, and feed out motions will be included in the machining distance calculations. Any peck and/or shift movement will not be included, since it is deemed only the cutting movement of the automatic cycle is important and **PostWorks** cannot accurately estimate how the machine will perform the peck and/or shift sequences.

8.8 Exit

Exits the Milling Cycles section and returns you to the previous section.

CHAPTER 9 Lathe Cycles

9.0 Walk Through Lathe Cycles

This selection will lead you through all of the sections used for defining lathe cycles, including cycle definition codes, cycle parameter codes, positioning options, User Defined Blocks for cycles, and machining time calculations.

9.1 Cycle Definition Codes

This section defines the cycle definition codes, for example, the codes output for *CYCLE/FACE*, *CYCLE/THREAD*, *CYCLE/TURN*, etc.

9.1.1 Does This Machine Support Automatic Cycles

Enter YES if the machine control will accept automatic (canned) cycle blocks and will perform the cycle simulation itself. An automatic cycle block usually contains a cycle definition code, cycle positions, and parameter codes, which will define the cycle to be performed.

Example:

```
G75 X3.5 Z.12 I.8 K1.4 D2.5 F7.5$
```

If you enter NO at this prompt, then **PostWorks** will output a series of linear moves that will simulate each cycle.

9.1.2 Enter Constant Lead Thread Cutting Code

Enter the register and its value that will enable a constant lead thread cutting sequence. This code will be output using the *THREAD* command and is usually G(33).

9.1.3 Enter Increasing Variable Lead Thread Cutting Code

Enter the register and its value that will enable increasing variable lead thread cutting at the machine. Increasing variable lead thread cutting will increase the lead of the thread by a specified amount across the length of the thread. This code will be output using the *THREAD/INCR* command.

9.1.4 Enter Decreasing Variable Lead Thread Cutting Code

Enter the register and its value that will enable decreasing variable lead thread cutting at the machine. Decreasing variable lead thread cutting will decrease the lead of the thread by a

specified amount across the length of the thread. This code will be output using the *THREAD/DECR* command.

9.1.5 Enter CYCLE/OFF Code

Enter the register and its value that will cancel an automatic cycle sequence. A cycle termination code is usually not required on lathes.

9.1.6:18 Enter CYCLE/Mode Code

Enter the register and its value that will enable an automatic cycle for this mode at the machine. This code will be output on *CYCLE/mode* sequences. If you do not enter a value here, then **PostWorks** will assume that the machine does not support this *CYCLE* mode and will simulate the cycle whenever a *CYCLE/mode* command is programmed.

Following is a list of valid *CYCLE* modes and the code that is normally associated with them.

DEEP	ROUGU	TURN / G(90)
DRILL	THREAD / G(2)	
FACE / G(94)	THRU	

9.1.7:19 Enter CYCLE/Mode With STEP Code

Enter the register and its value that will enable an automatic cycle for this mode at the machine. This code will be output on *CYCLE/mode* sequences which include the STEP parameter on the *CYCLE* command. If you do not enter a value here, then **PostWorks** will assume that the machine does not support this *CYCLE/STEP* mode and will simulate the cycle whenever a *CYCLE/mode,STEP* command is programmed.

Following is a list of valid *CYCLE/STEP* modes and the code that is normally associated with them.

DEEP / G(75)	ROUGH	TURN
DRILL	THREAD / G(76)	
FACE	THRU /G(74)	

9.2 Cycle Parameter Codes

This section defines the cycle parameter codes, for example, the codes output for '*RAPTO,r*', '*STEP,peck*', etc.

9.2.1 Enter Z-axis Lead Register

Enter the register that will contain the thread lead for the Z-axis as specified using the *THREAD* command and '*TPI,leadk*' cycle parameter. This register is usually 'F3'

9.2.2 Enter X-axis Lead Register

Enter the register that will contain the thread lead for the X-axis as specified using the *THREAD* command and '*TPI,leadl*' cycle parameter.

9.2.3 Enter Increasing Variable Lead Register

Enter the register that will contain the amount of increment for increasing variable lead thread cutting. Increasing variable lead thread cutting is enabled using the *THREAD/INCR* command.

9.2.4 Enter Decreasing Variable Lead Register

Enter the register that will contain the amount of decrement for decreasing variable lead thread cutting. Decreasing variable lead thread cutting is enabled using the *THREAD/DECR* command.

9.2.5 Enter Final Z-axis Position Register

Enter the register that will contain the final Z-axis position for an automatic cycle. Normally you would leave this prompt blank and **PostWorks** will use the appropriate Z-axis register for the final position value.

Some machines, though, require a separate register for the Z-axis when programming an automatic cycle. In this case enter a register other than an axis register that will contain the final Z-axis position.

9.2.6 Enter Final X-axis Position Register

Enter the register that will contain the final X-axis position for an automatic cycle. Normally you would leave this prompt blank and **PostWorks** will use the appropriate X-axis register for the final position value.

Some machines, though, require a separate register for the X-axis when programming an automatic cycle. In this case enter a register other than an axis register that will contain the final X-axis position.

9.2.7 Enter FEDTO Parameter Register

Enter the register that will contain the plunging axis offset value, as specified using the '*FEDTO,depth*', cycle parameter. This register is usually 'K2'.

9.2.8 Enter OFFSET Parameter 1 Register

Enter the register that will contain the Z-axis offset value as specified by the '*OFFSET,off1*' cycle parameter. This register is usually 'I2'.

9.2.9 Enter OFFSET Parameter 2 Register

Enter the register that will contain the X-axis offset value as specified by the '*OFFSET,off2*' cycle parameter,

9.2.10 Enter RAPTO Parameter Register

Enter the register that will contain the rapto distance, as specified using the '*RAPTO,r*' cycle parameter.

9.2.11 Enter REPEAT Parameter Register

Enter the register that will contain the final pass count, as specified using the '*REPEAT,rep*' cycle parameter.

9.2.12 Enter RTRCTO Parameter 1 Register

Enter the register that will contain the retract level, as specified using the '*RTRCTO,ret1*' cycle parameter.

9.2.13 Enter RTRCTO Parameter 2 Register

Enter the register that will contain the retract level, as specified using the '*RTRCTO,ret2*' cycle parameter. This register is usually 'R'.

9.2.14 Enter STEP Parameter 1 Register

Enter the register that will contain the initial depth of cut as specified by the '*STEP,peck1*' cycle parameter. '*peck1*' usually defines either the depth of the first cut or the depth of all cuts in a multiple pass cycle. The register is usually 'K2'.

9.2.15 Enter STEP Parameter 2 Register

Enter the register that will contain the final depth of cut as specified by the '*STEP,peck2*' cycle parameter. '*peck2*' usually defines the depth of the final cut in a multiple pass cycle. This register is usually 'I2'.

9.2.16 Enter TOOL Parameter 1 Register

Enter the register that will contain the angle of the cutting tool, as specified using the '*TOOL,t1*' cycle parameter. The angle of the cutting tool is usually used with thread cutting cycles. This register is usually 'D'.

9.2.17 Enter TOOL Parameter 2 Register

Enter the register that will contain the value specified by the '*TOOL,t2*' cycle parameter.

9.2.18 Enter Chamfer ON Code

Enter the register and its value that will enable automatic chamfering on thread cutting cycles. The *CYCLE/...,ON* command will enable chamfering. M(23) is usually used.

9.2.19 Enter Chamfer OFF Code

Enter the register and its value that will disable automatic chamfering on thread cutting cycles. The *CYCLE/...,OFF* command will disable chamfering. M(24) is usually used.

9.3 Cycle Positioning Parameters

This section contains support for the format for certain positioning parameters, such as the final position, step parameter output, and defines the default rapid rate for cycles.

9.3.1 Position To Taper Position Prior To Issuing CYCLE/THREAD,STEP Block

Some machines require that the tool already be positioned at the initial cut prior to programming a multiple pass tapered thread cutting cycle. If you enter YES at this prompt, then **PostWorks** will automatically cancel the cycle and position at the start of the tapered thread prior to outputting the *CYCLE/THREAD,STEP* parameter block.

Enter NO to this prompt if your machine does not have this requirement. Machines that require this feature usually use some of the axis registers to specify cycle parameters.

9.3.2 Position To Taper Position Prior To Issuing CYCLE/TURN,STEP Block

Some machines require that the tool already be positioned at the initial cut prior to programming a multiple pass tapered turning cycle. If you enter YES at this prompt, then **PostWorks** will automatically cancel the cycle and position at the start of the tapered cut prior to outputting the *CYCEL/TURN,STEP* parameter block.

Enter NO to this prompt if your machine does not have this requirement. Machines that require this feature usually use some of the axis registers to specify cycle parameters.

9.3.3 Mode For Automatic Cycles

Enter the number that corresponds to the output format for the automatic cycle block output.

1. Same as currently programmed mode.
2. Always absolute.
3. Always incremental.

Some machines, though they support both absolute and incremental modes, require that the automatic cycle position(s) be programmed in only one mode. When this is the case, it is usually the absolute mode.

9.3.4 Output Format For STEP (Peck) Parameters

Enter the number that corresponds to the output format to use with *STEP* cycle parameters.

- 1 = The cycle *STEP* parameters will be output with the same sign (positive or negative) as they contained on the input *CYCLE* command.
- 2 = The cycle *STEP* parameters will always be output as positive values.
- 3 = The cycle *STEP* parameters will always be output as negative values.

9.3.5 Default Rapid Rate For Cycles

Enter the feed rate in Feed Per Minute to use when calculating the machining time for rapid moves during a cycle sequence. This feed will actually be used for rapid moves when **PostWorks** generated (manual) cycles are output. Specifying a value of 0 will cause the machine's rapid rate to be used.

9.4 Cycle Block Output

This section defines the output format for cycle blocks, including when to output the cycle parameters.

9.4.1 Force Cycle Definition Code On The Initial Cycle Block

Enter YES if the first location in a canned cycle must contain the definition code for the active cycle sequence. **PostWorks** will force out the cycle definition code on the cycle block immediately following the *CYCLE* command.

If you enter NO at this prompt, then the normal output control for the cycle definition register will be used to determine when it will be output.

9.4.2 Force Active Linear Axes On The Initial Cycle Block

Enter YES if the first location in a canned cycle must contain both axis positions, even if the positions have not changed. **PostWorks** will force out both the Z and X axis registers on the cycle block immediately following the *CYCLE* command.

If you enter NO at this prompt, then the normal output control for the Z and X axis registers will be used to determine when they will be output.

9.4.3 Force Input Cycle Parameters On Every Cycle Block

Enter YES if every location programmed during a cycle sequence should be a full cycle definition block. A cycle definition block will contain the cycle definition code, axes position, feed rate, and the cycle registers that were programmed on the V command.

9.4.4 Force Feedrate On The Initial Cycle Block

Enter YES if the first location in a canned cycle must contain a feed rate code. **PostWorks** will force out the feed rate code on the cycle block immediately following the *CYCLE* command.

If you enter NO at this prompt, then the normal output control for the feed rate register will be used to determine when it will be output.

9.4.5 Force Linear Interpolation Code After CYCEL/OFF

Enter YES if the linear interpolation code should be forced out on the first motion block following a *CYCLE/OFF* command.

If you enter NO at this prompt, then the normal output control for the linear interpolation register will be used to determine when it will be output.

9.4.6 Automatically Cancel Cycle After First Move

Enter YES if cycles should be considered a one shot command and immediately cancelled after a single cycle block. If you enter NO, then the programmed cycle will remain active until explicitly cancelled by the *CYCLE/OFF* command.

9.5 Cycle User Defined Blocks

This section defines the User Defined Block to use for each supported automatic cycle sequence.

9.5.1 CYCLE/OFF User Defined Block

Enter the number of the *User Defined Block* you want to be output when the *CYCLE/OFF* command is programmed. Enter a value of 0 if the cycle cancellation block does not require any special format, other having a *CYCLE/OFF* code.

9.5.2:15 CYCLE/Mode User Defined Block

Enter the number of the User Defined Block that defines the output format for the *CYCLE/mode* definition block. User Defined Blocks that are used for *CYCLE* statements differ from their normal usage. The normal processing of a User Defined Block will force out each code defined as part of the block, whereas *CYCLE* User Defined Blocks only define the order of the codes to be output in multiple blocks. The codes contained in a *CYCLE* User Defined Block will not necessarily be output.

For example, when a cycle block needs to be output in the following format, then you would specify a User Defined Block as described.

Cycle Block Format	'User Defined Block description
-----	-----
G79 I1__ J1__ R__\$	G(79), I1, J1, R, \$
G90 X__ Z__ F__\$	

In the above example, a G79 cycle parameter definition block, containing, for example, both '*STEP*' cycle parameters and the '*RAPTO,r*' cycle parameters, is required prior to the actual cycle block. The User Defined Block for this example defines only the G79 block terminated with the End-of-Block character. All registers that are not in the User Defined Block will be output in the following block. If the I1, J1 and R registers are not output, then the entire G79 block will not be output.

The following register descriptors have special meaning when used in a *CYCLE* User Defined Block.

X = X-axis final position register.

Z = Z-axis final position register.
F = Feed rate register.

If the cycle definition block is not required to be output in more than a single block containing all of the cycle parameters, then enter a value of 0.

This prompt will be displayed for every supported automatic cycle mode; *DEEP*, *ROUGH*, *THREAD*, etc., including the 'STEP' versions of the cycle commands.

9.6 Cycle Machining Times

This section defines the machining time calculation to use for each supported automatic cycle. You can choose a separate machining estimation based on **PostWorks** generated cycles for each supported automatic cycle.

9.6.1:14 Enter CYCLE/Mode Time Calculations

Enter the manual cycle mode that should be used for estimating the machining time for an automatic *CYCLE/mode* sequence. You can enter any of the following manual cycle modes.

DEEP DRILL FACE ROUGH THREAD THRU TURN

See the **PWorks** manual cycle commands for a description of the simulated motion that will be used to estimate the canned cycle time.

This prompt will be displayed for every supported automatic cycle mode; *DEEP*, *ROUGH*, *THREAD*, etc., including the 'STEP' versions of the cycle commands.

9.7 Exit

Exits the Lathe Cycles section and returns you to the previous section.

CHAPTER 10 Sequence Numbers/Alignment Blocks

10.0 Sequence Numbers/Alignment Blocks

This section defines the output format for sequence numbers and alignment blocks.

10.1 Output Sequence Numbers By Default

Enter YES if you want sequence numbers output by default at the start of the part program. You can enable and disable sequence numbers during the part program by using the [SEQNO](#) command.

10.2 Register For Sequence Numbers

Enter the register descriptor for sequence numbers. This is usually 'N'.

10.3 Beginning Sequence Number

Enter the sequence number to output on the first block. This value can be changed using the [SEQNO/m](#) command.

10.4 Sequence Number Increment

Enter the value to increment sequence numbers by. This value will be added to the sequence number after each block. You can change this value during the part program by using the [SEQNO/m,INCR,i](#) command.

10.5 Output Sequence Numbers Every nth Block

Enter the value that defines the increment of output blocks that will contain sequence numbers. For example, if you enter 3, then every third block will contain a sequence number.

You can change this value during the part program by using the [SEQNO/m,INCR,i,n](#) command.

10.6 Maximum Sequence Number

Enter the maximum sequence number allowed. The sequence number will reset to 1 after this limit has been reached.

10.7 Register For Alignment Block Number

Enter the register to output as an alignment block sequence number. Alignment blocks are output using the [TMARK](#) command.

10.8 User Defined Block For Non-motion Alignment Blocks

Enter the number of the [User Defined Block](#) you want to be output for an alignment block that does not contain any motion. A value of 0 can be entered, disabling any extra codes from being output.

10.9 User Defined Block For Motion Alignment Blocks

Enter the number of the User Defined Block you want to be output for an alignment block that contains axes motion. A value of 0 can be entered, disabling any extra codes from being output.

10.10 User Defined Block For Automatic Cycle Blocks

Enter the number of the User Defined Block to output when an alignment block is output during a cycle sequence, because of the [CYCLE/START](#) command. If a value of 0 is entered, then only the alignment block register will be forced out in a cycle alignment block.

CHAPTER 11 Spindle & Coolant

11.0 Walk Through Spindle & Coolant Parameters

This selection will lead you through all of the aspects of defining spindle speeds and coolant output.

11.1 Spindle Set-up

This section defines the format for spindle speed output, surface speed per minute support, and the number of ranges that are supported.

11.1.1 Enter Spindle Speed Output Format

Enter the value that specifies the format for spindle speeds output to the Control Tape.

1. Use the spindle speed as reference only.
2. Output the spindle speed as a direct RPM/SFM value.
3. Output using a predefined spindle table.
4. Output as a percentage of the maximum spindle speed.

All values, except for number 1, will cause the spindle speed to be output. Number 3 uses a spindle table that is defined in a following section. Number 4 will output a percentage of the spindle speed, from 1 to 99 percent.

11.1.2 Does The Machine Controller Support SFM Spindle Control

Enter YES if the machine will accept codes that will control the spindle speed using Surface Feed per Minute speeds instead of Revolutions Per Minute speeds. You can switch between SFM and RPM control by using the *SPINDL* command.

If you enter NO, then **PostWorks** will calculate and output RPM spindle speeds when you request SFM speed control using the *SPINDL/SFM* command.

11.1.3 Enter The Number Of Spindle Ranges [1-3]

Enter the number of physical spindle speed ranges that the machine has. You can enter from 1 to 3 ranges. Following is a list of ranges supported, depending on the number entered.

- 1 = Medium
- 2 = Low, High
- 3 = Low, Medium, High

When multiple ranges are supported, you can select the range manually using the *SPINDL/...,range* command or **PostWorks** can automatically select the range when you use the *SPINDL/...,AUTO* command.

11.1.4 Low Range Spindle Speed Limits [Min, Max]

Enter the minimum and maximum spindle speeds allowed for the low range. This prompt will only be displayed when either 2 or 3 spindle ranges are supported.

11.1.5 Medium Range Spindle Speed Limits [Min, Max]

Enter the minimum and maximum spindle speeds allowed for the medium range. This prompt will only be displayed when either 1 or 3 spindle ranges are supported.

11.1.6 High Range Spindle Speed Limits [Min, Max]

Enter the minimum and maximum spindle speeds allowed for the high range. This prompt will only be displayed when either 2 or 3 spindle ranges are supported.

11.1.7 SFM Spindle Speed Limits [Min, Max]

Enter the minimum and maximum Surface Feed per Minute speeds allowed. This prompt will only be displayed when SFM control is supported by the machine.

11.1.8 Tolerance For RPM Output When In SFM Mode

This prompt will only be displayed when the machine does not support SFM spindle speed control and **PostWorks** converts SFM speeds to RPM speeds. Enter the tolerance that will be used to determine when a new RPM speed should be output. A new RPM speed will only be output when it changes by at least the tolerance entered.

11.1.9 Low Range Spindle Speed Limits For Mill Head [Min, Max]

Enter the minimum and maximum spindle speeds allowed for the low range of the milling attachment. This prompt will only be displayed when either 2 or 3 spindle ranges are supported and the machine type is a Mill/Turn.

11.1.10 Medium Range Spindle Speed Limits For Mill Head [Min, Max]

Enter the minimum and maximum spindle speeds allowed for the medium range of the milling attachment. This prompt will only be displayed when either 1 or 3 spindle ranges are supported and the machine type is a Mill/Turn.

11.1.11 High Range Spindle Speed Limits For Mill Head [Min, Max]

Enter the minimum and maximum spindle speeds allowed for the High range of the milling attachment. This prompt will only be displayed when either 2 or 3 spindle ranges are supported and the machine type is a Mill/Turn.

11.2 Spindle Registers

This section defines the registers that are required for spindle speed output, including speeds, direction, range selections, and mode selections.

11.2.1 Spindle RPM Speed Register

Enter the register descriptor for RPM spindle speeds. this is usually 'S'.

11.2.2 Spindle SFM Speed Register

Enter the register description for SFM spindle speeds. This prompt will only be displayed when the machine supports SFM speed control. The register format for SFM speeds is usually different than for RPM spindle speeds.

11.2.3 Spindle CLW Code

Enter the register and its value for turning on the spindle in a clockwise direction. This is usually M(03).

11.2.4 Spindle CCLW Code

Enter the register and its value for turning on the spindle in a counter-clockwise direction. This is usually M(04).

11.2.5 Spindle OFF Code

Enter the register and its value for turning the spindle off. This is usually M(05).

11.2.6 Spindle ORIENT Code

Enter the register and its value for stopping the spindle in its orientation location. This is a fixed location on the spindle that the tool will always stop at. Some machines do not have a spindle orientation position.

11.2.7 Spindle BOTH Code

Enter the register and its value for turning on the spindle when the *SPINDL/...,BOTH* command is encountered. This code usually consists of a spindle CLW code, and optional skip character, and a spindle CCLW code (M03/M04).

You can describe a register in this manner by using the text string 'M03/' as the beginning characters of the register and 'M04' as the ending characters. In this circumstance, enter only a register at this prompt, do not include a value.

11.2.8 Low Range Spindle Code

Enter the register and its value that will activate the low speed spindle range. This prompt will only be displayed when there are either 2 or 3 spindle ranges supported.

11.2.9 Medium Range Spindle Code

Enter the register and its value that will activate the medium speed spindle range. This prompt will only be displayed when there are either 1 or 3 spindle ranges supported.

11.2.10 High Range Spindle Code

Enter the register and its value that will activate the high speed spindle range. This prompt will only be displayed when there are either 2 or 3 spindle ranges supported.

11.2.11 RPM Spindle Speed Control Code

Enter the register and its value that will activate Revolutions per Minute speed control. usually, an RPM speed control code is only used when both RPM and SFM spindle speed controls are supported. The code normally used is G(97).

11.2.12 SFM Spindle Speed Control Code

Enter the register and its value that will activate Surface Feed per Minute speed control. This prompt will only be displayed when the machine supports SFM spindle speed control. The code normally used is G(96).

11.2.13 SFM Maximum Spindle Speed Control Code

Enter the register and its value that defines a block which contains the maximum spindle revolutions per minute when the machine is in SFM speed control mode. This prompt will only be displayed when the machine supports SFM spindle speed control.

The maximum spindle RPM block is output using the *SPINDL/MAXRPM* command and contains a limit on how fast the spindle can rotate. The code normally used is G(50).

11.2.14 SFM Spindle Radius Register

Enter the register descriptor that will be output with the *SPINDL/...,RADIUS* command. This command overrides the internally calculated radius used in SFM calculations. This prompt will only be displayed when the machine supports SFM spindle speed control.

11.2.15 Disable Spindle Speed Overrides Code

Enter the register and its value that will disable the machine operator from overriding spindle speeds. The disable spindle speed overrides code is output using the *SPINDL/LOCK,ON* command.

11.2.16 Enable Spindle Speed Overrides Code

Enter the register and its value that will enable the machine operator to override spindle speeds at the control. The enable spindle speed overrides code is output using the *SPINDL/LOCK,OFF* command.

11.2.17 User Defined Block For Spindle On Blocks

PostWorks normally outputs the spindle on codes (direction and speed) on the block following the *SPINDL/ON* command. If you require a special sequence for turning the spindle on or require that the spindle on codes be output in a block by themselves, then enter a predefined **User Defined Block** number here.

If you do enter a value other than 0 (no User Defined Block), then you must include the spindle codes in the block definition, or else they will not be output.

11.2.18 Output Spindle Off Codes In A Block By Themselves

Enter YES if you want the spindle off and orient codes output in a block by themselves. Otherwise, they will be output in the following block.

11.3:5 Spindle Speed Range Tables

These sections define the spindle speed tables for each of the supported ranges. The Spindle Table form allows you to define the available spindle speeds for each spindle range and the actual value each speed will output.

The value to the left of the equals, under the heading 'Code', is the value that will be output when the spindle speed matches the value to the right of the equals, under the 'Rpm' heading.

When using the **MPost** interface, there are no headings or equals character. The number in the left column of each section is the value that will be output when the spindle speed matches the value in the right column of each section.

The spindle speed table lookup will compare the programmed spindle speed to a value in the table, if the speeds match exactly or the programmed speed is greater than the table speed but less than the next speed in the table, then a match will occur.

The spindle speed values must be entered in ascending order (from lowest to highest). The table is structured so that it ascends from left to right and then from top to bottom (the same as a written page).

11.6 Coolant Set-up

Defines the codes required to turn the coolant on and off. Also, defines combination spindle on and coolant on codes.

11.6.1 Coolant Mist Code

Enter the register and its value that will turn on the mist coolant at the machine. The M(07) code is usually used.

11.6.2 Coolant Flood Code

Enter the register and its value that will turn on the flood coolant at the machine. The M(08) code is usually used.

11.6.3 Coolant Air Code

Enter the register and its value that will turn on the air only coolant at the machine.

11.6.4 Coolant OFF Code

Enter the register and its value for turning the coolant off. This is usually M(09).

11.6.5 Combination Spindle CLW And Coolant Mist Code

Enter the register and its value that will turn the mist coolant on and the spindle on in the clockwise direction. If your machine does not support this feature, then do not enter any code. For machines that support combination spindle/on and coolant/on codes, the M(13) code is normally used.

11.6.6 Combination Spindle CCLW And Coolant Mist Code

Enter the register and its value that will turn the mist coolant on and the spindle on in the counter-clockwise direction. If your machine does not support this feature, then do not enter any code. For machines that support combination spindle/on and coolant/on codes, the M(14) code is normally used.

11.6.7 Combination Spindle CLW And Coolant Flood Code

Enter the register and its value that will turn the flood coolant on and the spindle on in the clockwise direction. If your machine does not support this feature, then do not enter any code. For machines that support combination spindle/on and coolant/on codes, the M(17) code is normally used.

11.6.8 Combination Spindle CCLW And Coolant Flood Code

Enter the register and its value that will turn the flood coolant on and the spindle on in the counter-clockwise direction. If your machine does not support this feature, then do not enter any code. For machines that support combination spindle/on and coolant/on codes, the M(18) code is normally used.

11.6.9 Combination Spindle CLW And Coolant Air Code

Enter the register and its value that will turn the air coolant on and the spindle on in the clockwise direction. If your machine does not support this feature, then do not enter any code.

11.6.10 Combination Spindle CCLW And Coolant Air Code

Enter the register and its value that will turn the air coolant on and the spindle on in the counter-clockwise direction. If your machine does not support this feature, then do not enter any code.

11.6.11 Output Coolant On Codes In A Block By Themselves

Enter YES if you want coolant on (*MIST, FLOOD, AIR*) codes output in a block by themselves. Otherwise, they will be output in the following block.

11.6.12 Output Coolant Off Codes In A Block By Themselves

Enter YES if you want coolant off codes output in a block by themselves. Otherwise, they will be output in the block following a *COOLNT/OFF* command.

11.7 Exit

Exits the Spindle & Coolant section and returns you to the previous section.

CHAPTER 12 Feedrate & Rapid

12.0 Walk Through Feedrate & Rapid Parameters

This selection will lead you through all of the aspects of defining the supported feed rate and rapid formats, including FPM, FPR, 1/T and DPM feed rate modes, rapid motion adjustments, machine feed rate limits, etc.

12.1 Miscellaneous Feedrate Set-up

This section defines which feed rate modes are supported by the machine, how the *FEDRAT* command is handled and which feed rate mode to output with rotary motion.

12.1.1 Does The Machine Controller Support FPM Feedrates

Enter YES if the machine will accept *Feed per Minute feed rate* codes.

12.1.2 Does The Machine Controller Support FPR Feedrates

Enter YES if the machine will accept *Feed per Spindle Revolution feed rate* codes.

12.1.3 Does The Machine Controller Support DPM Feedrates

Enter YES if the machine will accept *Degrees per Minute feed rate* codes. DPM feed rates are only used with rotary axis movement.

12.1.4 Does The Machine Controller Support Inverse Time Feedrates

Enter YES if the machine will accept *1 / Time feed rate* codes.

12.1.5 Default Feedrate Mode

Enter the default input feed rate mode, you may choose one of the following modes.

FPM	=	Feed per minute.
FPR	=	Feed per spindle revolution.
INVERS	=	Inverse time.

12.1.6 Force Current Feed Output After FEDRAT Statement

Enter YES if you want to force the output of the current feed rate after a *FEDRAT* command is processed. Normally, **PostWorks** uses the Register Control Logic to determine if the feed rate code should be output.

12.1.7 Force Current Feed Output After Feedrate Mode Change

Enter YES if you want to force the output of the current feed rate after an output feed rate mode change, for example from FPM to Inverse time. Normally, **PostWorks** uses the Register Control Logic to determine if the feed rate code should be output.

12.1.8 Feedrate Mode For Rotary Axes Movement

Enter the value that corresponds to the feed rate mode that the machine expects to see when one or more rotary axes move.

1. *Feed per Minute*.
2. *Feed per Revolution*.
3. *Degrees per Minute*.
4. *Inverse time*.
5. Combination feed per Minute and Degrees per Minute.
6. Feed per Minute when linear motion is greater than rotary motion. Degrees per Minute when rotary motion is greater than linear motion.
7. Feed rate control point.

If you select FPM feed rates (1), then the output feed rates will be for the machine linear slides, not the programmed feed rate. **PostWorks** will calculate the linear axis feeds using the programmed feed rate and feed control point.

Combination FPM and DPM feedrates (5) require a *pulse weight* to be defined for the linear and rotary axes. The pulse weight is usually the minimum increment that each axis can move and is usually different for the linear and rotary axes. The pulse weights are defined in the Feed Per Minute section.

Control point feed rates (7) are usually the same as the input feed rate. Though, there are times when they will be lower than the programmed feed rate, such as when **PostWorks** slows the feed rate down due to one of the supported axes moving faster than its maximum limit.

This prompt will only be displayed when at least one rotary axis is supported.

12.1.9 Output Average Rotary Axes Radius With Feedrate

Enter YES is a rotary axis radius code is required when the feed rate control point is output with rotary axes movement. Machines which accept the actual programmed feed rate with rotary axes movement either support the programming of the rotary axes as linear axes (the circumference of the rotary axis is output instead of degrees), or they will require the average radius of the rotary axes which move to be output whenever it changes.

PostWorks will calculate the average radius of each rotary axis which moves, based on the radii at the start and end of the move.

12.1.10:13 Rotary Axis #n Radius Register

Enter the register that will contain the average rotary radius for this rotary axis, when the programmed feed rate is output with rotary axes movement and the radius is required for the machine control to calculate the actual feed rate at the machine.

12.1.14 Disable Feedrate Overrides Code

Enter the register and its value that will disable the machine operator from overriding programmed feed rates. The disable feed rate overrides code is output using the [*FEDRAT/LOCK,ON*](#) command and is usually M(49).

12.1.15 Enable Feedrate Overrides Code

Enter the register and its value that will enable the machine operator to override the programmed feed rate at the control. The enable feed rate overrides code is output using the [*FEDRAT/LOCK,OFF*](#) command and is usually M(48).

12.2 Feed Per Minute Feedrates

This section defines how FPM feedrates are output, including extended precision FPM feedrates, the [*pulse weight*](#) to use for linear and rotary axes when calculating degrees per minute feed rates.

12.2.1 Output FPM Feedrates As FPR/INVERS

This prompt will only be displayed when FPM feed rates are not supported by the machine. Enter the mode that you want input FPM feed rates output as, either [*FPR \(Feed per Revolution\)*](#) or [*INVERS \(Inverse time\)*](#).

12.2.2 Output FPM Feedrates Using A Predefined Table

Enter YES if the FPM feed rates are output as an entry in a table. The FPM feed rate table is defined in a following section. Enter NO if FPM feed rates are output as a direct value.

12.2.3 Output FPM Feedrates Based On The Control Point

FPM feedrates are calculated using a feedrate control point, which is normally the tool end point, but can be changed using the FEDRAT/LENGTH command. The output feedrate is usually based on the physical linear axes. When an output axis is adjusted separately from the tool end point, or it does not track along the standard axis line, such as a Z-axis that moves with the spindle, then the feedrate value output on the motion block can differ from the input values due to the difference in the machine axis slides distance travelled as compared to the tool end point.

Enter YES to this prompt if the above scenario matches your machine layout and the controller will compensated automatically for the difference in the tool end point feedrate as compared to the machine slides feedrate. Entering NO to this prompt will output the feedrate based on the physical linear axes.

12.2.4 FPM Register

Enter the register descriptor for FPM feed rate values, for example, F1.

12.2.5 Set FPM Mode Register And Value

Enter the register and its value that will activate Feed per Minute feed rate control. This code is usually only used when more than one feed rate mode is supported and is normally G(94).

12.2.6:7 Pulse Weight For 'Type' Axes

Feed rates for rotary axes moves require a pulse weight to be defined for the linear and rotary axes. The pulse weight is usually the minimum increment that each axis can move and is usually different for the linear and rotary axes. For example, if the minimum increment for this type of axis is .0001, then the pulse weight will usually be .0001.

12.2.8 Are Extended Precision FPM Feedrates Supported

Extended precision feed rates, when supported by a machine, are used to obtain greater accuracy to the right of the decimal point for the feed rate number. A predefined code is usually output with extended precision feed rates and the feed rate itself is multiplied by a constant. **PostWorks** will automatically switch between standard FPM mode and extended precision FPM mode, depending on the output feed rate.

Enter YES if your machine has this capability.

12.2.9 Multiplication Constant For Extended Precision FPM Feedrates

This prompt will only be displayed if extended precision FPM feed rates are supported. Enter the value to multiply extended precision feed rates by prior to their being output. This is usually a multiple of 10.

12.2.10 Extended Precision FPM Register

This prompt will only be displayed if extended precision FPM feed rates are supported. Enter the register description for extended precision feed rates. This register can usually be the same as the standard FPM register.

12.2.11 Set Extended Precision FPM Mode Register And Value

Enter the register and its value that will activate extended precision FPM feed rate control. This code is usually different and in a separate group than the set standard FPM feed rates code.

12.3 Feed Per Revolution Feedrates

This section defines how FPR feed rates are output, including automatically setting the FPR feed rate mode under certain circumstances.

12.3.1 Output FPR Feedrates As FPM/INVERS

This prompt will only be displayed when FPR feed rates are not supported by the machine. Enter the mode that you want input FPR feed rates output as, either **FPM (Feed per Minute)** or **INVERS (Inverse time)**.

12.3.2 Output FPR Feedrates Using A Predefined Table

Enter YES if the FPR feed rates are output as an entry in a table. The FPR feed rate table is defined in a following section. Enter NO if FPR feed rates are output as a direct value.

12.3.3 FPR Register

Enter the register descriptor for FPR feed rate values, for example, F2.

12.3.4 Set FPR Mode Register And Value

Enter the register and its value that will activate Feed per Revolution feed rate control. This code is usually only used when more than one feed rate mode is supported and is normally G(95).

12.3.5 Should FPR Feedrates Be Forced When The Z-axis Moves

Some older machines require that Feed per Revolution feed rates be in effect on every motion block where the Z-axis moves. Enter Yes if your machine has this requirement.

12.3.6 Force FPR Feedrate When FPM Feedrate Is Less Than

Some older machines, essentially lathes, require that Feed per Revolution feed rates be in effect on every motion block with a programmed feed rate in a certain cutting range. If your machine has this requirement, then enter the minimum FPR feed rate that can be output. All feed rates below this value will be output in the FRP mode.

If your machine does not have this requirement, then enter 0.

12.4 Degrees Per Minute Feedrates

This section defines how DPM feed rates are output. This section can only be entered when DPM feed rates are output with rotary motion.

12.4.1 DPM Register

Enter the register descriptor for DPM feed rate values, for example, F3.

12.4.2 Set DPM Mode Register And Value

Enter the register and its value that will activate Degrees per Minute feed rate control. This code is usually only used when more than one feed rate mode is supported.

12.4.3 Should DPM Mode Register Override Linear Register

Some machines use the Linear interpolation register to set Feed per Minute feed rates and the DPM mode register to set Degrees per Minute feed rates. In this case the DPM mode register belongs to the same group as the Linear mode register, not feed rate mode registers.

If you enter YES at this prompt, then when Degrees per Minute feed rates are in effect the DPM mode register will be output instead of the Linear mode register. If you enter NO, then the DPM mode register will be output along with the Linear mode register.

12.5 Inverse Time Feedrates

This section defines how 1/T feedrates are output, including extended precision 1/T feedrates.

12.5.1 How Should Inverse Time Feedrates Be Calculated

Enter the value that corresponds to the calculation that should be used when generating inverse time feed rates.

1. 1 / Time
2. (Constant * 2n * Feed) / (60 * Distance)
3. Machining time.
4. Vector Speed
5. (Linear + Rotary) / Time
6. Feed * ((Linear + Rotary) / Linear)

1/Time is the normal method for calculating inverse time feed rates. Calculation type (2) is usually used by older Bendix controls. Selection (3) will output the calculated machining time without any modifications.

The vector speed calculation (4) will base the feedrate number on the following calculation and is typically used with Toshiba machine controls.

$$F = \text{feed} \times \sqrt{\frac{\text{linear}^2 + \text{rotary}^2}{\text{linear}^2 + (\pi \times \text{radius} \times \text{rotary})^2}}$$

The combination linear and rotary delta over time calculation (5) uses the linear delta movement at the tool tip combined with the rotary axes delta movement (as if the rotary axes were extra linear axes) divided by the machining time for the move. This feed rate calculation is typically used with Okuma controllers and uses the following calculation.

$$F = \frac{\sqrt{\text{linear}^2 + \text{rotary}^2}}{\text{time}}$$

The option (6) calculation will base the feedrate number on the following calculation and is used by some Okuma controllers. The d- values depict the delta values (A and B are the rotary axes). It is possible that a scale factor needs to be applied to the rotary delta values for the Okuma controller. These [scale factors](#) are defined at the end of this section.

$$F = \text{Feed} / (\sqrt{dX^2 + dY^2 + dZ^2 + dA^2 + dB^2})$$

12.5.2 Inverse Time Feedrate Calculation For Circular Interpolation

Enter the value that corresponds to the calculation that should be used when inverse time feed rates are output on circular interpolation blocks.

1. 1 / Time
2. Feed /Radius
3. machining time.

This prompt will only be displayed when 1/Time inverse time calculations were selected for linear motion. Selection (3) will output the calculated machining time without any modifications.

12.5.3 Enter Constant For Inverse Time Calculations

This prompt will only be displayed when Bendix inverse time feed rate calculations are supported. Enter the value for the Constant variable in the calculation. This value is usually .2453.

12.5.4 Enter Command Pulse Weight For Inverse Time Calculations

This prompt will only be displayed when Bendix inverse time feed rate calculations are supported. The command pulse weight is used in the calculation to generate the 2n variable and is usually .0002.

12.5.5 Unit Of Time For Inverse Time Feedrates

Enter the unit of time, either MINUTES or SECONDS, to use when calculating inverse time feed rates. MINUTES is normally used.

12.5.6 Inverse Time Feedrate Register

Enter the register descriptor for inverse time feed rate values, for example, F3.

12.5.7 Set Inverse Time Mode Register And Value

Enter the register and its value that will activate inverse time feed rate control. This code is usually only used when more than one feed rate mode is supported and is normally G(93).

12.5.8 Are Extended Precision Inverse Time Feedrates Supported

Extended precision feed rates, when supported by a machine, are used to obtain greater accuracy to the right of the decimal point for the feed rate number. The feed rate values are usually multiplied by a constant prior to being output and are output with more digits to the right of the decimal point. **PostWorks** will automatically switch between standard inverse time mode and extended precision inverse time mode, depending on the output feed rate.

Enter YES if your machine has this capability.

12.5.9 Multiplication Constant For Extended Precision 1/T Feedrates

This prompt will only be displayed if extended precision inverse time feed rates are supported. Enter the value to multiple extended precision feed rated by prior to their being output. This is usually -1.024.

12.5.10 Extended Precision Inverse Time Register

This prompt will only be displayed if extended precision inverse time feed rates are supported. Enter the register description for extended precision feed rates. This register is usually different than the standard inverse time feed rate register.

12.5.11 Set Extended Precision Inverse Time Mode Register And Value

Enter the register and its value that will activate extended precision inverse time feed rate control. This code is usually the same as the code that sets standard inverse time feed rates.

12.5.12 Enter Constant For Rotary Delta Movement In Inches

Some controllers, such as the Okuma, require a scale factor to be applied to the rotary axis travel distance when calculating the feedrate number. When a scale factor is applied it is usually to convert millimeters to inches or vice versa. In the case of the Okuma controller, a value of 25.4 is typically used when a scale factor is required when the output is in Inch mode. Most controls do not require a scale factor, in which case 1. should be entered at this prompt.

12.5.13 Enter Constant For Rotary Delta Movement In Millimeters

Some controllers, such as the Okuma, require a scale factor to be applied to the rotary axis travel distance when calculating the feedrate number. When a scale factor is applied it is usually to convert millimeters to inches or vice versa. In the case of the Okuma controller, a value of .03937 could be used when a scale factor is required when the output is in Millimeter mode. Most controls do not require a scale factor, in which case 1. should be entered at this prompt.

12.6:7 Feed Per 'Unit' Feedrate Table

The Feed Rate Table form allows you to define the available feed rates and the actual value each feed will output.

The value to the left of the equals, under the heading 'Code', is the value that will be output when the feed rate matches the value to the right of the equals, under the 'Feed' heading.

When using the **MPost** interface, there are no headings or equals character. The number in the left column of each section is the value that will be output when the feed rate matches the value in the right column of each section.

The feed rate table lookup will compare the programmed feed rate to a value in the table, if the feeds match exactly or the programmed feed is greater than the table feed but less than the next feed in the table, then a match will occur.

The feed rate values must be entered in ascending order (from lowest to highest). The table is structured so that it ascends from left to right and then from top to bottom (the same as a written page).

12.8 Rapid Set-up

This section defines whether the machine supports rapid mode and how rapid motion should be modified prior to being output.

12.8.1 Does The Machine Controller Support Rapid Mode

Enter Yes if the machine will accept a code that will put it in rapid traverse positioning mode. If you enter NO, then **PostWorks** will output rapid moves with a feed rate equal to the rapid rate, as defined in the Machine Feedrate Limits section.

12.8.2 Enter Codes For Setting Rapid Mode

This prompt will only be displayed when the machine supports rapid traverse positioning mode. You can enter up to 5 codes, separated by commas, that will be output each time rapid mode is entered. Usually the G(00) code is used.

12.8.3 Does The Machine Controller Support Rapid Mode In **LMDP** Mode

Enter YES if the machine will accept a code that will put it in rapid traverse positioning mode while in polar interpolation mode on a Mill/Turn type machine. If you enter NO, then **PostWorks** will output rapid moves with a feed rate equal to the rapid rate, as defined in the Machine Feedrate Limits section.

12.8.4 Enter Code For Setting Rapid In LMDP Mode

Enter the rapid traverse code to use while in polar interpolation mode on a Mill/Turn type machine. Usually the D(10) code is used.

12.8.5 Does The Machine Controller Support Rapid Mode In LMDP Mode

Enter YES if the machine will accept a code that will put it in rapid traverse positioning mode while in cylindrical interpolation mode on a Mill/Turn type machine. If you enter NO, then **PostWorks** will output rapid moves with a feed rate equal to the rapid rate, as defined in the Machine Feedrate Limits section.

12.8.6 Enter Code For Setting Rapid In LMDP Mode

Enter the rapid traverse code to use while in cylindrical interpolation mode on a Mill/Turn type machine.

12.8.7 Output Linear Interpolation Code On Rapid Block

This prompt will only be displayed when the code for setting rapid traverse mode is not a G-code. Enter YES to this prompt if the rapid traverse code must be accompanied by a positioning code (either linear or circular).

If you enter NO, then **PWorks** will not attempt to output a positioning code when rapid is in effect.

12.8.8 Reset The Programmed Feedrate After A Rapid Move

Enter YES if you wish the last programmed feed rate output on the motion block following a rapid move, regardless if it is the same value or not.

12.8.9 Reset The Feedrate Mode After A Rapid Move

Enter YES if you require the currently active feedrate mode output on the motion block following a rapid move.

12.8.10 Enter Motion Alteration For Rapid Moves

Enter the value that corresponds to the way you want rapid moves output to the Control Tape.

1. Do not alter rapid moves.
2. Alter rapid moves along the X-axis.

3. Alter rapid moves along the Y-axis.
4. Alter rapid moves along the Z-axis.
5. Alter rapid moves along major tool axis.

If you select any of the options other than 1, then rapid motion will be altered as follows.

If the selected axis move is in the positive direction, then **PostWorks** will move in this axis first and then move in the other 2 axes.

If the selected axis move is in the negative direction, then **PostWorks** will move in the other 2 axes first and then in the selected axis.

PostWorks will determine the selected axis depending on the tool axis vector of each point, if option 5 is selected. For example, if the tool axis is 1,0,0, then the major axis for this move will be the X-axis.

The option selected here is only the default option, you can override it at any time during the part program using the **RAPID** command.

12.8.11 Retract/Plunge Rapid Moves Along Tool Axis

This prompt will only be displayed when rapid moves are modified along the major tool axis. Enter YES if you want rapid motion to retract or plunge (depending on the movement of the major axis) along the tool axis, instead of moving straight along the major axis. You can use the **RAPID/TOOL** command to accomplish this same feature anywhere within the part program.

12.8.12 Retract To Clearance Plane On Rapid Moves

Answering YES to this question causes the tool to retract to the defined clearance plane prior to positioning to the programmed location at the rapid rate. This feature can also be utilized by using the **RAPID/RTRCTO** command.

12.8.13 Mode For Rapid Moves

Enter the number that corresponds to the output format for rapid motion.

1. Same as currently programmed mode.
2. Always absolute.
3. Always incremental.

Some machines, though they support both absolute and incremental modes, require that rapid motion be output in only one mode. When this is the case, it is usually the absolute mode.

12.8.14 Modify Rapid Moves To Complete Fastest Axis First During Simulation

Enter YES if the machine enters positioning mode during rapid moves. This means that the machine will position each axis as fast as they move, with the axis with the shortest move time reaching its destination first, and each of the other axes reaching their final destination in the subsequent blocks. The move will not be in a straight line to the destination.

The rapid moves will only be modified during machine simulation output, this option will not affect the moves output to the MCD file.

12.9 Output Feedrate Limits

This section defines the minimum and maximum values that can be output in all feed rate modes.

12.9.1:6 'Mode' Feedrate Limits [Min, Max]

Enter the minimum and maximum values allowed for this feed rate mode. These limits affect only the output feed rate number and do not define the maximum feed rate allowed for each axis, which are defined in the Machine Feed Rate Limits section.

You will be prompted for each of the following feed rate modes, which are supported in this machine configuration.

1. Feed per Minute
2. Feed per Revolution
3. Degrees per Minute
4. Inverse time
5. Extended precision Feed per Minute
6. Extended precision Inverse time

12.9.7 Enter Unit Scale Factor For Feedrate Limit Values

Enter the scale factor to be applied to the output feedrate limit values when switching between Inches and Millimeters. This scale will normally be 10, since the feedrate registers typically differ by an accuracy of one digit between Inches and Millimeters. On some machines though, it is desirable to have this value be the standard units conversion factor of 25.4.

12.10 Machine Feedrate Limits

This section defines the maximum feed rate and rapid rate for each supported linear and rotary axis. It also defines the minimum amount of time each move is allowed to take.

12.10.1:10 Maximum 'Axis' Feedrate

Enter the maximum feed rate of the specified axis. **PostWorks** will check the actual feed rate of this axis against the maximum allowed and slow down the feed rate if it goes above this limit.

You will be prompted for each of the supported linear and rotary axes. The linear axis feed rates should be entered in Feed per Minute. The rotary axis feed rates should be entered in Degrees per minute.

12.10.11:20 'Axis' Rapid Rate

Enter the rapid traverse rate for the specified axis. This is usually, but not always, the same value as the maximum feed rate allowed. The rapid traverse rate is used for calculating machining time when in rapid mode.

You will be prompted for each of the supported linear and rotary axes. The linear axis feed rates should be entered in Feed per Minute. The rotary axis feed rates should be entered in Degrees per Minute.

12.10.21 Minimum Amount Of Time Allowed Per Move In Seconds

Enter the minimum amount of time, in seconds, that any given move is allowed to take. This feature is usually used on older machines with paper tape readers. It allows for the average time it takes to read a single block of information, thereby eliminating reader dwells during the cutting of the part.

12.10.22 Adjust Output Feedrates For Maximum Axes Feedrates

The post-processor will always calculate the actual feedrate based on the input feedrate, the maximum feedrate for each supported axis, the minimum move time allowed, and other factors. It is also standard procedure to output the feedrate adjusted for any of these limits that may be reached, so that the output feedrate exactly matches the feedrate on the machine and the machine is prevented from attempting to move an axis faster than it is capable of.

There are some controllers that will automatically slow down the feedrate based on the maximum axis speeds, but still require the programmed feedrate be output to the MCD block. Enter NO at this prompt if this describes your machine controller. The calculated machining times will still be adjusted for the limits of the machine.

Enter YES if the feedrate adjusted for the machine limits should be output. This is the actual feedrate that the machine will be moving.

12.11 Accel/Decl Time Adjustments

This section defines whether **PWorks** will adjust the machining time calculations for the acceleration and deceleration of each axis.

12.11.1 Adjust Machining Time For Acceleration/Deceleration Spans

The standard method of calculating the machining time of each move is to simply take the distance of the move and divide it by the programmed feed rate. The resulting time would be correct if the machine axes did not have to accelerate to obtain a higher speed or decelerate when moving into a corner.

Answering Yes to this prompt will cause **PostWorks** to take into account the acceleration rate of each axis and whether the axis has to decelerate when moving into a corner when calculating the machining time of each move. Allowing **PostWorks** to include acceleration/deceleration calculations in the machining time calculations should result in a much more accurate estimate of the time required to machine the part.

12.11.2:11 Maximum Velocity For 'Axis'

Enter the maximum acceleration velocity of the specified axis in units per second. This value will be used as both the acceleration and deceleration rate of this axis. If a value of 0 is entered at this prompt, then the average velocity of the move will be used for the entire move.

12.12 Exit

Exits the Feedrate & Rapid section and returns you to the previous section.

CHAPTER 13 Tool Change Sequence

13.0 Walk Through Tool Change Sequence

This selection will lead you through all of the subsections of defining the tool change sequence. Automatic/manual tool changes, loading and unloading of the tool, tool change codes, and tool length offsets are contained in these sections.

13.1 Tool Change Support

This section defines the ability of the machine to handle a tool change sequence, it defines the machine's support for automatic tool changes, preselecting the tool, size of tool grippers, number of spindles, number of turrets (for lathes), etc.

13.1.1 Maximum Tool Number Allowed

Enter the maximum tool number allowed for this machine. **PostWorks** will output a warning message whenever a tool number greater than this value is encountered. The tool number encountered will not be changed, only a warning message will be output.

13.1.2 Does The Machine Support 'Select Tool' Sequence

Enter Yes if the machine is capable of selecting the next tool to be loaded while it has a tool already loaded in this spindle. A tool selection sequence usually entails positioning the tool magazine to the selected tool and picking it up with the tool gripper. If you enter No to this prompt, then the *SELCTL* and *UNLOAD* commands will be disabled.

13.1.3 Does The Machine Have Both A SMALL And LARGE Gripper

Enter YES if the machine has two sizes of grippers that are used to load and unload the tool. The grippers are of different sizes to allow for tool holders of different sizes. Most machines usually only have one size gripper. The *SELCTL* and *LOADTL* commands reference the different size grippers.

13.1.4 Does The Machine Have Both A MAIN And AXIAL Spindle

Enter YES if the machine has a choice of two spindles to load the tool into. The spindles usually have two different RPM ranges with the axial spindle capable of higher speeds than the main spindle. Most machines usually only have a single spindle. The *SELCTL* and *LOADTL* commands reference the different speed range spindles.

13.1.5 Support SELECT/TOOL And LOAD/TOOL Commands

Enter YES if the SELECT/TOOL command is to be treated as a SELCTL command and the LOAD/TOOL command is to be treated as a LOADTL command. The syntax of these two commands after the TOOL parameter is identical to the SELCTL and LOADTL commands respectively.

Entering NO to this command will cause **PWorks** to report an error whenever either of these commands are encountered.

13.1.6 Number Of Turrets

Enter the number of turrets that this lathe has. You can enter either 1 or 2. If you enter 2, then the lathe supports both a Front and Rear turret. You can switch between the turrets using the **TURRET** command.

13.1.7 Distance Between Turrets

Enter the distance between the Front and Rear turret. This distance will be added to the X-axis radius whenever the Rear turret is activated. A value of other than 0 must be entered if the lathe is always oriented using the Front turret and **PostWorks** must compensate when using the Rear turret. This prompt will only be displayed when the machine type is a lathe and it supports 2 turrets.

13.1.8 Default Turret At Beginning Of Program

Enter either FRONT or REAR, depending on the turret you wish to be active at the start of the program. You can switch between the turrets using the **TURRET** command. This prompt will only be displayed when the machine type is a lathe and it supports 2 turrets.

13.2 Loading The Tool

This section defines the basic format of a tool change block. Whether the current tool needs to be unloaded first, if alignment blocks are required, amount of time to consider for a tool change, etc.

13.2.1 Should The Tool To Unload Be Output With A Tool Change

Enter YES if the tool number that is currently in the spindle should be output with a tool change code, instead of the tool number to place in the spindle. In this case, the tool to load (place in the spindle) will be output with the tool selection code and the tool to unload (place back in the tool magazine) will be output with the tool change sequence. This prompt will only be displayed when a tool selection sequence is supported.

Example

Input	Output
-----	-----
SELCTL/2	T02\$
LOADCTL/2	T01 M06\$ (assuming tool #1 is currently in the spindle)

13.2.2 Should An Unload Code Be Output For A Tool Change

Enter YES if the control does not support a single tool change code that will automatically perform the following sequence.

1. Pick up the next tool to be loaded.
2. Remove the current tool from the spindle.
3. Place the current tool back in the tool magazine.
4. Place the selected tool in the spindle.

Most machines accept a single code, such as M06, that will perform the above functions. In this case enter NO to this prompt. This prompt will only be displayed when a tool selection sequence is supported.

Example

SELCTL/2	G25 T02\$
LOADCTL/2	G27 T01\$ (assuming tool #1 is currently in the spindle)

13.2.3 Output Tool Number With A Tool Unload Block

Enter YES if your machine supports a code to unload the tool from the spindle without loading another tool and you are able to select the tool magazine location to place this tool. If your machine does not allow you to specify the tool magazine location when unloading a tool, then enter NO to this prompt. This prompt will only be displayed when a tool selection sequence is supported.

13.2.4 Output An Alignment Block On Tool Change Blocks

Enter YES if you want every tool change block to be an alignment block. The tool change block will be in the same format as a non-motion type alignment block.

13.2.5 Output An Alignment Block After A Tool Change Block

Enter YES if you want the Control Tape block directly following a tool change sequence to contain an alignment block. The alignment block number will follow the normal numbering scheme for sequence numbers.

13.2.6 Turn Off Tool Length Compensation Prior To Tool Change

Enter YES if you want **PostWorks** to automatically cancel tool length compensation prior to outputting a tool change block. All of the codes necessary to cancel tool length compensation will be output.

13.2.7 Turn On Tool Length Compensation After A Tool Change

Enter YES if you want **PostWorks** to automatically turn on tool length compensation after outputting a tool change block. All of the codes necessary to enable tool length compensation will be output.

13.2.8 Is A Tool Change Considered An End-of-Sequence

An End-of-Sequence will reset the sequence delta movements for each axis and the sequence machining time. It will also internally turn the coolant and spindle off. Enter YES if you want a tool change sequence to perform these functions.

13.2.9 Look Ahead For LOADTL When PPRINT Encountered

Enter YES if **PWorks** should look to see if the next command is a LOADTL or LOAD/TOOL command whenever a PPRINT command is processed. If the next command is a LOADTL command then the %Arg(2) variable passed to the PPRINT Macro will be set to 1, otherwise %Arg(2) will be set to 0.

From this description you can see that a PPRINT Macro is mandatory in order to take advantage of this feature. It is typically used to store the text of the PPRINT statement so it can be referenced and/or output in the LOADTL Macro.

Entering NO at this prompt will disallow **PWorks** from looking for a LOADTL command. The PPRINT Macro variable %Arg(2) will always be set to 0 in this case.

13.2.10 Calculate Separate Times For Each Instance The Same Tool Is Loaded

Typically **PWorks** will search the previous loaded tools each time a new tool is loaded and if there is a match it will treat this instance of the tool as a continuation of the previously loaded tool and the machining time will be added to the previously loaded tool.

If you would like duplicate tool numbers to maintain their own separate tool times, then enter YES at this prompt. In this case each time a tool is loaded the previously loaded tools will not be checked for a match and the machining times will be calculated separately.

You can use the TOOLNO/TIMES command to change this setting in the part program.

13.2.11 Amount Of Time For Tool Change Sequence In Seconds

Enter the amount of time to add to the total sequence time and machining time whenever a tool change sequence is output. Enter the average amount of time it takes the machine to change the tool only if you want this time taken into consideration for the total machining times.

13.3 Tool Change Codes

This section defines the codes that need to be output in order to perform a tool change sequence. The tool number register, tool change code, tool selection code, turret direction codes, etc. are contained in this section.

13.3.1 Pick Up Tool (With Small Gripper) Code

Enter the register and optional value that will pick up the tool from the tool storage (the small gripper will be used when 2 sizes of grippers are supported). This prompt will only be displayed when a "Select Tool" sequence is supported and should be left blank if the machine does not require a special code other than the tool number to select the next tool.

13.3.2 Pick Up Tool With Large Gripper Code

Enter the register and optional value that will pick up the tool from the tool storage using the large gripper. This prompt will only be displayed when a "Select Tool" sequence is supported and the machine has 2 sizes of grippers. Machines having 2 sizes of grippers usually require a unique code to pick up the tool with the specified gripper.

13.3.3 Select Tool Register

Enter the register for tool selection numbers. Leave this prompt blank if tool numbers should not be output to the Control Tape. The tool selection register is usually the same as the tool number register used for loading the tool. You can specify only a portion of a register, for example only the 1st and 2nd digits of the T register, by entering a real number with the register. The number to the left of the decimal point is the beginning digit within the register for the tool number and the number to the right of the decimal point is the ending digit.

Example

- T Use all digits of the T register.
- T(7,8) Use the 7th and 8th digits of the T register. The 1st through 6th digits will be used for other values, such as the tool length offset register and/or the cutter compensation register.

13.3.4 Tool Number Register

Enter the register for tool numbers. Leave this prompt blank if tool numbers should not be output to the Control Tape. You can specify only a portion of a register, for example only the 1st and 2nd digits of the T register, by entering a real number with the register. The number to the left of the decimal point is the beginning digit within the register for the tool number and the number to the right of the decimal point is the ending digit. The T register is usually used.

Example

- T Use all digits of the T register.
- T(7,8) Use the 7th and 8th digits of the T register. The 1st through 6th digits will be used for other values, such as the tool length offset register and/or the cutter compensation register.

13.3.5 Output The Tool Register On A Block By Itself

Enter YES if you want the tool number register output in a block by itself, without any other codes, not even the tool change code. Normally the tool number register and the tool change code can be in a block together.

13.3.6 Tool Change Code

Enter the register and optional value that will perform an automatic tool change. You can leave this prompt blank, in which case a code will not be output with an automatic tool change. M(06) is usually used for automatic tool changes.

13.3.7 Manual Tool Change Code

Enter the register and optional value that will perform a manual tool change. You can leave this prompt blank, in which case a code will not be output with a manual tool change. M(00) is usually used for manual tool changes.

13.3.8 Unload Tool (With Small Gripper) From Main Spindle Code

Enter the register and optional value that will unload the tool from the spindle and place it back in the tool magazine (the small gripper will be used when 2 sizes of grippers are supported, likewise the high speed spindle will be used when 2 spindles are supported). This prompt will only be displayed when a "Select Tool" sequence is supported and should be left blank if the machine does not have a special code, other than the tool change code, to unload a tool.

13.3.9 Unload Tool (With Small Gripper) From Axial Spindle Code

Enter the register and optional value that will unload the tool from the low speed spindle and place it back in the tool magazine (the small gripper will be used when 2 sizes of grippers are supported). This prompt will only be displayed when a "Select Tool" sequence is supported and the machine has 2 spindles. Machines with 2 spindles usually support a tool unload code.

13.3.10 Unload Tool With Large Gripper From Main Spindle Code

Enter the register and optional value that will unload the tool from the spindle using the large gripper and place it back in the tool magazine (the high speed spindle will be used when the machine supports 2 spindles). This prompt will only be displayed when a "Select Tool" sequence is supported and the machine has 2 sizes of grippers. Machines having 2 sizes of grippers usually require a unique code to unload the tool with the specified gripper.

13.3.11 Unload Tool With Large Gripper From Axial Spindle Code

Enter the register and optional value that will unload the tool from the low speed spindle using the large gripper and place it back in the tool magazine. This prompt will only be displayed when a "Select Tool" sequence is supported, the machine has 2 sizes of grippers and 2 spindles. Machines having 2 sizes of grippers usually require a unique code to unload the tool with the specified gripper.

13.3.12 Force Turret CLW Code

Enter the register and optional value that will force the turret to rotate clockwise when selecting the requested tool. This code will be output with the *TURRET/...,CLW* command.

13.3.13 Force Turret CCLW Code

Enter the register and optional value that will force the turret to rotate counter-clockwise when selecting the requested tool. This code will be output with the *TURRET/...,CCLW* command.

13.3.14 User Defined Block For Tool Changes

Enter the *User Defined Block* to use for tool change blocks. A User Defined Block is not normally needed for a tool change block, unless the machine requires codes other than the tool number and tool change code to perform the tool change.

Some machines, for example, may require a dwell after each tool change. In this case you would enter a User Defined Block number which would output the tool change codes in one block and machine dwell codes in the following block.

13.4 Tool Length Offsets

This section describes the support for tool length offsets. It includes the register description for tool length offset blocks and whether or not to add the tool length to the output axes.

13.4.1 Add Tool Lengths To Output Axes

Enter YES if you want the tool length programmed in the *LOADTL* command to be added to the output axes. The tool length will be added to the axes along the spindle vector of the machine. It is usually not necessary to add the tool length to the output axes, unless the machine has at least one rotary spindle (HEAD) axis. Most of today's machines will compensate for the tool length.

This can be dynamically changed by using the *PWorks* command *TOOLNO/OSETL*.

13.4.2 Register For Tool Length Offsets

Enter the register for tool length offsets. Leave this prompt blank if tool length offset registers are not supported by the machine control. You can specify only a portion of a register, for example only the 1st and 2nd digits of the T register, by entering a real number with the register. The number to the left of the decimal point is the beginning digit within the register for the tool offset register and the number to the right of the decimal point is ending digit. The H register is usually used.

Example

H Use all digits of the H register.

- T(1.4) Use the 1st through 4th digits of the T register. The 5th through nth digits will be used for other values, such as the tool number and/or the cutter compensation register.

13.4.3 Base Number For Tool Length Offsets

Enter a value that should be added to the tool length offset register number prior to output. A value other than 0 is usually entered only when the register used for tool length offsets is used for another offset, such as fixture offsets. For example, registers 0-99 may be used for fixture offsets, while 100-199 will be used for tool length offsets. In this case, enter 100 at this prompt and **PostWorks** will automatically add 100 to the input tool length offset register.

13.4.4 Code For Enabling Tool Length Offsets

Enter the register and its value that will activate the tool length offsets at the machine control. If the machine does not support tool length offsets, then leave this prompt blank.

13.4.5 Code For Disabling Tool Length Offsets

Enter the register and its value that will turn off the tool length offsets at the machine control. If the machine does not support tool length offsets, then leave this prompt blank.

13.4.6 Code For PLUS Tool Length Offsets

Enter the code that will enable tool length offsets in the positive direction at the machine control. This code can be output using the *TOOLNO/ADJUST,PLUS* command. If the machine does not support tool length offsets or does not allow a direction for offsets, then leave this prompt blank.

13.4.7 Code For MINUS Tool Length Offsets

Enter the code that will enable tool length offsets in the negative direction at the machine control. This code can be output using the *TOOLNO/ADJUST,MINUS* command. If the machine does not support tool length offsets or does not allow a direction for offsets, then leave this prompt blank.

13.4.8 Output MINUS Offsets As Negative Values

Enter YES if your machine requires the tool length offset register to be output as a negative value when tool length offsets are enabled in the negative direction. Negative direction tool length offsets can be enabled using the *TOOLNO/ADJUST,MINUS* command.

If the machine does not support tool length offsets in the negative direction or a specific code, such as G44, is used to enable negative offsets, then enter NO to this prompt.

13.4.9:11 'Axis' Tool Offset Register

Enter the register to use for output with the *TOOLNO/ADJUST,n,-AXIS* command. It is recommended that you do not use the same register that is used for the absolute axis, unless the axis value output in the tool length offset block is the actual axis position. **PostWorks** uses the absolute axis position register to calculate incremental moves, machining times, etc.

When the axis contained on the tool length offset block is an arbitrary number, such as always being zero, then specify a previously unused register at this prompt. For example, U1.

13.4.12 Output Plane Selection Code With Tool Length Offset Register

Enter YES if the machine control requires a plane selection code (G17, G18, G19, for example) on a tool length offset block. Most controls do not require the plane selection code to be output on a tool length offset block.

13.4.13 Output Tool Length Offset On Codes In A Block By Themselves

Enter YES if the machine control requires that the codes enabling tool length offsets be output in a block by themselves. If you enter NO to this prompt, then the *TOOLNO/ADJUST* command will determine how the enable tool length offset codes will be output.

13.4.14 Output Tool Length Offset Off Codes In A Block By Themselves

Enter YES if the machine control requires that the codes disabling tool length offsets be output in a block by themselves. If you enter NO to this prompt, then the *TOOLNO/ADJUST* command will determine how the disable tool length offset codes will be output.

13.4.15 User Defined Block For Tool Length Offsets

Enter the User Defined Block to use for tool length offset blocks. A *User Defined Block* is not normally needed for a tool length offset block, unless the machine requires codes other than the tool offset codes defined in this section on the tool length offset block.

Some machines, for example, may require a rapid code to be output in each tool length offset block. In this case you would enter a User Defined Block number that would output the tool length offset code(s) along with a rapid code.

This User Defined Block will be used for both enable and disable tool length offset block.

13.5 Exit

Exits the Tool Change Sequence section and returns you to the previous section.

CHAPTER 14 Cutter/Fixture Compensation

14.0 Walk Through Cutter/Fixture Compensation

This selection will lead you through all of the subsections of defining cutter compensation support and fixture compensation registers. Cutcom support at the control, including cutcom vectors and fixture compensation codes are contained in these sections.

14.1 Cutter Compensation Support

This section defines the ability of the machine to handle cutter compensation. It contains support for cutcom vectors and valid compensation planes.

14.1.1 Cutter Compensation Output Format

Enter the number corresponding to the type of cutter compensation which is supported by the machine.

- 0 = The machine will calculate the cutter compensation vector based on the previous position, current position and next position. It does not require an input cutcom vector to determine the offset direction.
- 1 = A cutter compensation vector in the direction of compensation will be output on each move between *CUTCOM/ON* and *CUTCOM/OFF*. This is the Cincinnati PQR cutcom vector format. Cutter compensation will usually be performed on every motion block that contains a cutcom vector, cutcom on and off codes are normally not required.
- 2 = A cutter compensation vector in the same direction as the next move will be output on each move between *CUTCOM/ON* and *CUTCOM/OFF*. This format usually contains the cutcom vector on a motion block as I J K's and does not support cutter compensation vectors on circular blocks.
- 3 = A cutter compensation vector in the same direction as the current move will be output on each move between *CUTCOM/ON* and *CUTCOM/OFF*. This format usually contains a cutcom vector in a block by itself with a cutcom vector definition code, such as G39. The cutcom vector will precede the motion block.
- 4 = The angle of the next move will be output on each motion block between *CUTCOM/ON* and *CUTCOM/OFF*. This format is usually used with machines that support polar coordinate programming, such as the SHARNOA control. Moves preceding a circular interpolation record will usually contain a code that states the direction of the circular move.

14.1.2 Unitize Cutcom Vector

Enter YES if the cutcom vector should be unitized to a length of 1.0 prior to being output. If you enter NO, then the cutcom vector components will be the delta distance of the current/next move. This prompt will only be displayed when the cutcom vector contains the direction of either the current or next move.

14.1.3 Is Cutter Compensation Allowed In All 3 Planes

Enter YES if cutter compensation is supported in all three machining planes, XY, ZX and YZ. If cutcom is only valid in the XY machining plane, then enter NO. An error message will be output when cutcom is only allowed in the XY plane and you specify either the *ZXPLAN* or the *YZPLAN* parameter on a *CUTCOM* command.

14.1.4 Is 3-dimensional Cutter Compensation Supported

3-dimensional cutter compensation is usually used to offset the tool with a defined corner radius along the normal of either the part or drive surface. Enter YES if the machine controller supports 3-dimensional cutter compensation.

14.1.5 Is 5-axis Cutter Compensation Allowed

Enter YES if 5-axis cutter compensation is supported. 5-axis cutter compensation will output a three dimensional offset vector in the direction of compensation whenever the tool axis is at an angle to any of the standard machining planes. A two dimensional vector will still be output when the tool axis is normal to a machining plane. This prompt will only be displayed when the cutter compensation output format is set to 1.

14.1.6 Are Approach/Departure Codes Supported

Enter YES if your machine allows you to specify an entry and/or exit method for cutter compensation. The approach/departure codes can be output using the *CUTCOM/...,MODIFY* command.

14.1.7 Maximum Angular Change When Cutcom Is Active

Enter the maximum angular change for axes motion that is allowable when cutter compensation is active. An error message will be output when a move changes direction by more than this amount. The error message is intended to let the programmer know when the tool reverses direction and the cutter compensation direction has not been changed. Enter 180 (degrees) to disable this checking.

14.1.8 Maximum Vector Component Value Allowed

Enter the maximum value that is allowed for each cutcom vector component. An error message will be output and the cutcom vector component value will be reduced to the maximum value allowed when this value is exceeded.

14.2 Cutter Compensation Control

This section describes how cutter compensation codes are output to the Control Tape.

14.2.1 Output Cutcom On Codes In A Block By Themselves

Enter YES if the enable cutter compensation codes should be output in a block by themselves. Usually, cutter compensation enable codes are output in the next motion block.

14.2.2 Output Cutcom Off Code In A Block By Itself

Enter YES if the disable cutter compensation codes should be output in a block by themselves. Usually, cutter compensation disable codes are output in the next motion block.

14.2.3 Output Directional Vector On Initial Cutcom On Block

Enter YES if the machine requires a cutcom vector on the initial cutcom on motion block. If you enter NO, then the initial motion block following the *CUTCOM* command will not contain a cutcom vector. This prompt will only be displayed when cutter compensation vectors in the direction of the current move are supported.

14.2.4 Output Directional Vector On Circular Block

Enter YES if a cutter compensation vector should be output on a circular interpolation block. Some machines require that you always enter tangent to a circle when cutter compensation is in effect and do not allow you to specify a direction for cutcom. This prompt will only be displayed when cutter compensation vectors in the direction of either the current or next move are supported.

14.2.5 Output Directional Vector Between Multiple Circular Blocks

Enter YES if a cutter compensation vector should be output on a circular interpolation block when transitioning to another circle. Some require that circular interpolation records be tangent when cutter compensation is in effect and do not allow you to specify a direction for cutcom. This prompt will only be displayed when cutter compensation vectors in the direction of either the

current or next move are supported and cutcom vectors are allowed on circular interpolation blocks.

14.2.6 Output An Offset Register With A Value Of '0' With Cutcom Off

Enter YES if the machine requires a cutter compensation register of 0 to be output when cutcom is disabled. Usually, no cutter compensation register is required when turning cutcom off.

14.2.7 Output An Error Message With Move In Non-Cutcom Axis

Enter YES if you want an error message output whenever a move is made in the non-cutter compensation axis, with no movement in either of the cutter compensation axes.

14.3 Cutter Compensation Codes

This section defines the codes that will be output with cutter compensation, including direction codes, vector registers, and departure codes.

14.3.1 Cutcom LEFT Code

Enter the code that will enable cutter compensation to the left of the tool path at the machine control. This code can be output using the *CUTCOM/LEFT* command. If the machine does not support cutter compensation or allow for a directional code (such as the Cincinnati PQR format), then leave this prompt blank.

14.3.2 Cutcom RIGHT Code

Enter the code that will enable cutter compensation to the right of the tool path at the machine control. This code can be output using the *CUTCOM/RIGHT* command. If the machine does not support cutter compensation or allow for a directional code (such as the Cincinnati PQR format), then leave this prompt blank.

14.3.3 Cutcom OFF Code

Enter the code that will disable cutter compensation at the machine control. This code can be output using the *CUTCOM/OFF* command. If the machine does not support cutter compensation or allow for a code to turn cutcom off, then leave this prompt blank.

14.3.4 Cutter Compensation Offset Register

Enter the register that contains the offset values for cutter compensation. Leave this prompt blank if the machine does not accept a user specified register for cutcom offset amounts. You can specify only a portion of a register, for example only the 1st and 2nd digits of the T register, by entering a real number with the register. The number to the left of the decimal point is the beginning digit within the register for the cutcom register and the number to the right of the decimal point is the ending digit. The D register is usually used.

Example

D	Use all digits of the D register.
T(7.8)	Use the 7th through the 8th digits of the T register. The 1st through the 6th digits will be used for other values, such as the tool number and/or the tool length offset register.

14.3.5 Base Number Of Cutter Compensation Register

Enter a value that should be added to the cutter compensation register number prior to output. A value other than 0 is usually entered only when the register used for cutter compensation is used for another offset, such as tool length offsets. For example, registers 0-99 may be used for tool length offsets, while 100-199 will be used for cutter compensation. In this case, enter 100 at this prompt and **PostWorks** will automatically add 100 to the input cutter compensation register.

14.3.6 Cutcom Vector Definition Code

Enter the register and optional value that defines a block containing a cutter compensation vector. This prompt will only be displayed when cutter compensation vectors in the direction of the current move are supported. The cutter compensation vector definition block will be output prior to the motion block. The code normally used is G(39).

14.3.7:9 'Axis' Vector Component Register

Enter the register to be used for the specified axis component of the cutcom vector. This prompt will only be displayed when cutter compensation vectors are supported. The X-axis register is usually P or I2, the Y-axis is Q or J2, and the Z-axis is R or K2.

14.3.10 Cutcom Angular Direction Register

Enter the register to be used for the angular direction of the next move while cutter compensation is in effect. This prompt will only be displayed when the machine requires the angular direction of the next move while in cutter compensation. The P register is usually used.

14.3.11 CLW Circular Cutcom Direction Code

Enter the register and optional value for specifying that the next move is a circular interpolation move in the clockwise direction. This prompt will only be displayed when the machine requires the angular direction of the next move while in cutter compensation. The E(2) code is usually used.

14.3.12 CCLW Circular Cutcom Direction Code

Enter the register and optional value for specifying that the next move is a circular interpolation move in the counter-clockwise direction. This prompt will only be displayed when the machine requires the angular direction of the next move while in cutter compensation. The E(3) code is usually used.

14.3.13:15 Approach/Departure Code #n

Enter the code that will specify the nth entry/exit method for cutter compensation. This code will be output with the *CUTCOM/...,MODIFY,n* command. This prompt will only be displayed when cutter compensation approach and departure codes are supported by the machine.

14.3.16 Approach/Departure Distance Register

Enter the register that will contain the distance/radius for the entry/exit method for cutter compensation. This register will be output with the *CUTCOM/...,MODIFY* command. This prompt will only be displayed when cutter compensation approach and departure codes are supported by the machine.

14.3.17 3-D Cutter Compensation Code

Enter the code that will enable 3-dimensional cutter compensation along the part/drive surface normal. This code can be output using the *CUTCOM/ENDPT* command.

14.3.18 Tool Radius Register

Enter the register that will contain the tool radius for 3-dimensional cutter compensation. If the machine control does not require the tool radius to be programmed with 3-D cutter compensation, then leave this prompt blank.

14.3.19 Tool Corner Radius Register

Enter the register that will contain the tool corner radius for 3-dimensional cutter compensation. If the machine control does not require the tool corner radius to be programmed with 3-D cutter compensation, then leave this prompt blank.

14.3.20 Rotary Axis Position Mode Register

Enter the register that contains the rotary axis movement method during 3-dimensional cutter compensation. If the machine does not support this rotary axis setting, then leave this prompt blank.

14.3.21 Maximum Feedrate Register

Enter the register that contains the feedrate limitation value during 3-dimensional cutter compensation. On highly curved parts, it is possible that the machine will accelerate quickly on certain moves while 3-D cutter compensation is in effect. The maximum feedrate register can be used on some machine controls to limit the feed rate to the value programmed into this register. If the machine control does not support a maximum feedrate setting during 3-D compensation, then leave this prompt blank.

14.4 Fixture Offsets

This section describes the support for fixture offsets. It includes the register description for fixture offsets and the codes to enable/disable fixture offsets.

14.4.1 Register For Fixture Offsets

Enter the register for controlling fixture offsets. Leave this prompt blank if fixture offset registers are not supported by the machine control. The H register is usually used. The *CUTCOM/ADJUST* command controls the output of fixture offsets.

14.4.2 Base Number For Fixture Offsets

Enter a value that should be added to the fixture offset register number prior to output. A value other than 0 is usually entered only when the register used for fixture offsets is used for another offset, such as tool length offsets. For example, registers 0-99 may be used for tool length offsets, while 100-199 will be used for fixture offsets. In this case, enter 100 at this prompt and **PostWorks** will automatically add 100 to the input fixture offset register.

14.4.3 Code For Enabling Fixture Offsets

Enter the register and its value that will activate fixture offsets at the machine control. If the machine does not support fixture offsets, then leave this prompt blank.

14.4.4 Code For Disabling Fixture Offsets

Enter the register and its value that will turn off fixture offsets at the machine control. If the machine does not support fixture offsets, then leave this prompt blank.

14.4.5 Code For PLUS Fixture Offsets

Enter the code that will enable fixture offsets in the positive direction at the machine control. The code can be output using the *CUTCOM/ADJUST,PLUS* command. If the machine does not support fixture offsets or not allow a direction for offsets, then leave this prompt blank.

14.4.6 Code For MINUS Fixture Offsets

Enter the code that will enable fixture offsets in the negative direction at the machine control. The code can be output using the *CUTCOM/ADJUST,MINUS* command. If the machine does not support fixture offsets or not allow a direction for offsets, then leave this prompt blank.

14.4.7 Output MINUS Offsets As Negative Values

Enter YES if your machine requires the fixture offset register to be output as a negative value when fixture offsets are enabled in the negative direction. Negative direction fixture offsets can be enabled using the *CUTCOM/ADJUST,MINUS* command.

If the machine does not support fixture offsets in the negative direction or a specific code, such as G46, is used to enable negative offsets, then enter NO to this prompt.

14.4.8:10 'Axis' Fixture Offset Register

Enter the register to use for output with the *CUTCOM/ADJUST,n,-AXIS* command. It is recommended that you do not use the same register that is used for the absolute axis, unless the axis value output in the fixture offset block is the actual axis position. **PostWorks** uses the absolute axis position register to calculate incremental moves, machining times, etc.

When the axis contained on the fixture offset block is an arbitrary number, such as always being zero, then specify a previously unused register at this prompt. For example, U1.

14.4.11 Output Fixture Offset ON Codes In A Block By Themselves

Enter YES if the machine control requires that the codes enabling fixture offsets be output in a block by themselves. If you enter NO to this prompt, then the *CUTCOM/ADJUST* command will determine how the enable fixture offset codes will be output.

14.4.12 Output Fixture Offset OFF Codes In A Block By Themselves

Enter YES if the machine control requires that the codes disabling fixture offsets be output in a block by themselves. If you enter NO to this prompt, then the *CUTCOM/ADJUST* command will determine how the disable fixture offset codes will be output.

14.5 Exit

Exits the Cutter/Fixture Compensation section and returns you to the previous section.

CHAPTER 15 Miscellaneous Post Commands

15.0 Walk Through Misc. Post Commands

This selection will lead you through all of the subsections of the miscellaneous post-process commands. The handling of *INSERT*, messages, *OPSKIP*, *OPSTOP*, *STOP*, *DELAY*, *GOHOME*, and *POSTN* is contained in these sections.

15.1 Messages And INSERT

This section defines how message and *INSERT* blocks should be output, including whether they contain sequence numbers and End-of-Block characters.

15.1.1 Add Sequence Numbers To Message Blocks

Enter YES if you want **PostWorks** to generate a sequence number for operator messages. If you enter NO, the message blocks will not contain a sequence number.

15.1.2 Add Sequence Numbers To INSERT Blocks

Enter YES if you want **PostWorks** to generate a sequence number for *INSERT* blocks. If you enter NO, then *INSERT* blocks will not contain a sequence number, unless the text of the *INSERT* statement contains one.

15.1.3 Output An End-of-Block On Message Blocks

Enter YES if you want message blocks terminated with the End-of-Block character(s). If you enter NO, then the message block will be added to the beginning of the next output block.

15.1.4 Output An End-of-Block On INSERT Blocks

Enter YES if you want *INSERT* blocks terminated with the End-of-Block character(s). If you enter NO, then the *INSERT* block will be added to the beginning of the next output block, unless *INSERT* text is terminated with the End-of-Block character(s).

15.1.5 Convert Message Spaces To Tabs

Normally, spaces in a message block will be output as spaces, which at the machine control is the same as leader. Some machines will remove these spaces and compress the text within the message. If you enter YES to this prompt, the spaces in the message block will be converted to the tab character, which will display as spaces on the machine control.

15.1.6 Align Message Blocks At Column

Enter a value between 1 and 66 to cause all output message blocks to be output at the same length. The value entered at this prompt defines the minimum length of the actual message (text between the message start and message end characters). Any messages less than this length will have spaces added to the end of them so that they are output using this length. Any messages longer than this length will not be modified.

Enter a value of 0 if the output message blocks should not be aligned. No spaces will be added to the message text.

The DISPLY/ALIGN command can be used to change this value in the part program.

15.1.7 Process PPRINT IPV Statements Normally

YES specifies *PPRINT IPV* statements should be processed as simple PPRINT statements (output to the print file and as message blocks). NO will cause **PWorks** to ignore *PPRINT IPV* statements, except when creating a simulation file.

PPRINT IPV statements are used to pass commands to **NCL/IPV**.

15.2 OPSKIP, OPSTOP, And STOP

This section defines how the *OPSKIP*, *OPSTOP* and *STOP* commands are handled, including the codes that are output with the *OPSTOP* and *STOP* commands, the [User Defined Block](#) for each command, and whether *OPSTOP* and *STOP* are considered an End-of-Sequence.

15.2.1 Enter Code For OPSTOP

Enter the register and optional value you want output with the *OPSTOP* command. You can leave this prompt blank, in which case a code will not be output with *OPSTOP*. M(01) is usually output as the optional stop code.

15.2.2 Enter Code For STOP

Enter the register and optional value you want output with the *STOP* command. You can leave this prompt blank, in which case a code will not be output with *STOP*. M(00) is usually output as the hard stop code.

15.2.3 User Defined Block After OPSKIP/OFF

Enter the number of the User Defined Block you want to be output after an *OPSKIP/OFF* command. The User Defined Block will usually contain the axes position and the current feed rate. A value of 0 will disable this feature.

15.2.4 User Defined Block After OPSTOP

Enter the number of the User Defined Block you want to be output after an *OPSTOP* command. The User Defined Block will usually contain the axes position and the current feed rate. A value of 0 will disable this feature.

15.2.5 User Defined Block After STOP

Enter the number of the User Defined Block you want to be output after an *STOP* command. The User Defined Block will usually contain the axes position and the current feed rate. A value of 0 will disable this feature.

15.2.6 Should OPSTOP Be Considered An End-of-Sequence

An End-of-Sequence will reset the sequence delta movements for each axis and the sequence machining time. Enter YES if you want the *OPSTOP* command to perform these functions.

15.2.7 Should STOP Be Considered An End-of-Sequence

An End-of-Sequence will reset the sequence delta movements for each axis and the sequence machining time. Enter YES if you want the *STOP* command to perform these functions.

15.3 DELAY

This section defines the output for machine dwell blocks, including the dwell block definition code and register, unit of time for dwell blocks, and how they should be output.

15.3.1 Enter Delay Definition Code

Enter the register and its value which defines a machine dwell block. This code is usually G(04). Machine dwell blocks can be output using the *DELAY* command.

15.3.2 Enter Delay Time Register

Enter the register that will contain the dwell value when machine dwells are output in units of time (seconds, minutes, etc.).

15.3.3 Enter Spindle Revolution Delay Register

Enter the register that will contain the dwell value when machine dwells are output in number of spindle revolutions. If the machine does not support dwell time in spindle revolutions, then leave this prompt blank.

15.3.4 Enter Output Format For Time Delay Blocks

Enter the number that corresponds to the format of dwell blocks that the machine expects to see.

- 1 = Dwell blocks will always be output in units of time.
- 2 = Dwell blocks will always be output as the number of spindle revolutions to dwell.
- 3 = Dwell blocks will be output in units of time when the feed rate mode is Feed per Minute, or in spindle revolutions when the feed rate mode is in Feed per Revolution.
- 4 = Dwell blocks will be output in units of time when specified as seconds on the *DELAY* command, or in spindle revolutions when the REV parameter is specified on the *DELAY* command.
- 5 = Multiple dwell blocks, containing only the delay definition code, will be output, causing the machine to dwell for the requested amount of time. Use this option only if the machine does not allow for a dwell time to be programmed, but will rather perform a fixed amount of dwell for each delay definition code programmed.
- 6 = Dwell blocks will always be output in inverse time format.

15.3.5 Unit To Output For Time Delays

Enter the number that corresponds to the unit of time that the machine expects to see in a dwell block. The *DELAY* command will always expect to see the delay time in seconds, regardless of the output format.

1. Milliseconds
2. Seconds
3. Minutes

15.3.6 Enter Time Interval In Seconds For Each Delay Block

Enter the amount of time, in seconds, that the machine will dwell per each delay definition code programmed. This time will be used by **PostWorks** to calculate the number of delay definition codes to output. This prompt will only be displayed when multiple dwell blocks are output to satisfy the programmed dwell time.

15.3.7 Does The X-axis Need To Be Output In A Delay Block

Some machines require dwell blocks to include at least one of the programmable axes. **PostWorks** will always include the X-axis in this case. Enter YES if the machine has this requirement.

15.3.8 Reset The Programmed Feed Rate After A Delay Block

Enter YES if the current feed rate should be reinstated after a dwell block. The feed rate should usually be reinstated on machines that use the same register for feed rates and dwell times.

15.4 GOHOME

This section defines the output codes for automatic return from/to the home position. The [home position](#) for **PostWorks** generated moves to home is defined in the "Define Machine Configuration" section.

15.4.1 Enter Code For [GOHOME/CHECK](#)

Enter the register and its value that will perform a return to home position and a verification by the control that the machine has actually reached this position. This code is usually G(27) on machines that support this feature.

15.4.2 Enter Code For [GOHOME/AUTO](#)

Enter the register and its value that will perform an automatic return to home position at the machine. This code is usually G(28) on machines that support this feature.

15.4.3 Enter Code For [GOHOME/FROM](#)

Enter the register and its value that will perform an automatic return from home position at the machine. This code is usually G(29) on machines that support this feature.

15.4.4 Enter Code For **GOHOME/NEXT**

Enter the register and its value that will perform an automatic return to the secondary reference position at the machine. This code is usually G(30) on machines that support this feature.

15.4.5:14 Enter Register For **GOHOME/'Axis',n**

Enter the register to use as output for the specified axis on the *GOHOME/mode* command. it is recommended that you do not use the same register that is used for the absolute axis, unless the axis value output in the automatic move to/from home position block is the actual axis position. **PostWorks** uses the absolute axis position register to calculate incremental moves, machining times, etc.

It is usually acceptable to use the axis incremental register, such as X2, since automatic move to/from home position blocks are normally output in the incremental mode.

You will be prompted for each of the supported axes.

15.5 POSTN

This section defines the output codes for the identifying program number and absolute/incremental preset axis registers blocks.

15.5.1 Enter Register For POSTN Program Number

Enter the register that will contain the identifying program number. The program number can be output using either the *POSTN/n* command or the *MACHIN/PWORKS,OPTION,4,n* command.

15.5.2 Enter Code For **POSTN/NORMAL**

Enter the register and its value that defines an absolute preset axis registers block. A preset axis registers block is usually used to enter offsets into registers at the control that will perform a linear compensation for the differences in the part program origin as compared to the origin on the machine. G(29) is usually used to define an absolute preset axis registers block.

15.5.3 Enter Code For **POSTN/INCR**

Enter the register and its value that defines an incremental preset axis registers block. A preset axis registers block is usually used to enter offsets into registers at the control that will perform a linear compensation for the differences in the part program origin as compared to the origin on the machine.

G(93) is usually used to define an incremental preset axis registers block on machines that support this feature.

15.5.4 Enter Code For *POSTN/OFF*

Enter the register and its value that will cancel any previously defined preset axis registers block. G(99) is usually used to cancel a preset axis registers block on machines that support this feature.

15.5.5 Output *POSTN/OFF* Code In A Block By Itself

Enter YES if the machine control requires that the code cancelling any previously defined preset axis registers block be output in a block by itself. If you enter NO to this prompt, then the code output with *POSTN/OFF* code will be output in the next block, along with any other codes that need to be output.

15.5.6:15 Enter Register For *POSTN/'Axis',n*

Enter the register to use as output for the specified axis in a preset axis registers block. Since the axis values output in an absolute preset block are used to store the actual positions on the machine, it is acceptable to use the same register that is used for the absolute axis output.

You will be prompted for each of the supported axes.

15.6 Exit

Exits the Miscellaneous post-processor command section and returns you to the previous section.

CHAPTER 16 Ultrasonic Blade Configuration

16.0 Walk Through Ultrasonic Blade Configuration

This selection will lead you through all of the subsections of the Ultrasonic Blade configuration menus. All aspects of defining the Ultrasonic Blade machine is covered in these sections, including Vacuum Pods.

16.1 Ultrasonic Blade Cutter Set-up

This section defines the configuration of an Ultrasonic Blade machine, mainly dealing with the attributes of the blade axis; registers, direction vector, retract logic, etc. This section is only valid when configuration an Ultrasonic Cutter type machine.

16.1.1 Enter Default Blade Direction Vector

Enter the vector (i, j, k) which defines the direction of the blade when the blade axis is at 0 degrees. **PostWorks** assumes that the blade is double edged, so a vector of 1,0,0 is equivalent to a vector of -1,0,0.

16.1.2 Register For Internal Blade Rotary Axis

Enter the register descriptor which will be used internally by **PostWorks** to keep track of the absolute position of the blade rotary axis. This register will not be output and is used by **PostWorks** to keep track of the blade axis on a linear scale of 360 degrees, no matter how the axis is actually output.

This register must be unique (not used as an output register for any other address). The A1, B1, C1, and C4 registers have initially been reserved as the internal rotary axis registers.

16.1.3 Register For Absolute Blade Rotary Axis

Enter the register descriptor for the absolute blade rotary axis, for example A2, B2, etc. You must enter a register even if the machine only supports incremental mode, as this register is used internally by **PostWorks** in calculating the incremental value for the blade axis. The A2, B2, C2 and C5 registers have initially been reserved as the absolute rotary axis registers.

16.1.4 Register For Incremental Blade Rotary Axis

Enter the register descriptor to use for incremental positioning of the blade rotary axis, for example A3, B3, etc. You do not need to enter a register if the machine does not support

incremental mode. The A3, B3, C3 and C6 registers have initially been reserved as the incremental rotary axis registers.

DO NOT enter the same register for absolute and incremental positions, as **PostWorks** uses the absolute positioning register to calculate the incremental move.

16.1.5 Blade Rotary Axis Is Output On A Linear/Rotary Scale

The blade rotary axis moves on a linear scale when a value of 720 degrees will move the axis 2 revolutions from 0 degrees. A rotary scale will always be output between 0 and 360 degrees, usually with the sign (+-) of the value determining the direction. Enter either LINEAR or ROTARY.

16.1.6 Current Position For Blade Rotary Axis

Enter the default position for the blade rotary axis at the start of the program. This location is input as the actual machine position.

16.1.7 Maximal Change Of Blade Rotary Axis

Enter the maximum angular change, in degrees, that the blade rotary axis can move while cutting. If the direction of the blade changes by more than this amount, the tool will be retracted, the blade will be rotated to the new direction, and the tool will be plunged back into the part. You can disable this feature by specifying a value of 90 degrees or more.

The Retract Tool prompts in the Motion Adjustments/[Rotary Axes Linearization Section](#) can be used to set the retract distances and feed rates. The [RETRACT/TOOL,ATANGL,a](#) command can be used to change this value anywhere in the part program.

16.1.8 Blade Rotary Axis Travel Limits [Min, Max]

Enter the minimum and maximum travel limits for the blade rotary axis. These limits can be change using the [CHECK/AXIS,4](#) command.

16.2 Ultrasonic Blade Alignment

Defines the controlling parameters for blade alignment during cutting motion. Automatic cutting alignment mode, alignment spans, and minimum angle change for blade alignment values can be set.

16.2.1 Automatic Blade Alignment Mode

Enter either ON or OFF to enable or disable automatic blade alignment mode at the beginning of the program. Automatic blade alignment mode is used to align the blade within a defined span when making long moves with directional changes. When this mode is enabled, directional change moves will be broken up so that the blade will be aligned with the move within a user definable distance. On multi-axis moves, the move may be broken up into multiple blocks containing directional changes, as the blade can change directions multiple times if there is a lot of rotary axis motion.

The *ALIGN/CUT* command can be used to enable/disable the automatic blade alignment mode at any time within the part program.

16.2.2 Minimum Span To Apply Align Alteration

Enter the minimum move distance allowed for the automatic blade alignment feature. When the distance of a move is less than the value entered here, then the move will be output as a single move and will not be broken up into multiple moves. This value can be changed in the part program by using the *ALIGN/CUT* command.

16.2.3 Minimum Blade Angle Which Will Have Align Alteration

Enter the minimum angular change allowed in the blade direction for the automatic blade alignment feature. When the directional change is less than the angle entered at this prompt, then the move will be output as a single move and will not be broken up into multiple moves. This value can be changed in the part program by using *ALIGN/CUT* command.

16.2.4 Span Distance At Which To Align Blade

Enter the span distance to be used when breaking up a move to align the blade direction. This distance is the length of the alignment span output at the beginning of the move and contains the blade direction for the entire move. This value can be changed in the part program by using the *ALIGN/CUT* command.

16.2.5 Blade Alignment Direction

Most Ultrasonic blades are V-shaped and will cut in either direction, but there are blades that have a flat edge on one side and an angled side on the other. When using this style of blade it is usually desirable to maintain the flat side of the blade against the part. This prompt defines the default leading edge of the blade.

‘FRONT’ specifies the front of the blade should always be pointed in the forward direction. ‘REAR’ specifies that the back of the blade should always be pointed in the forward direction. ‘BOTH’ specifies that either blade edge can be pointed in the forward direction.

The front of the blade is defined as the blade edge that points in the direction of the default blade direction vector when the blade axis is at 0 degrees, as defined in the previous section. This setting can be changed in the part program by using the *MODE/BLADE,ON,dir* command.

16.3 Ultrasonic Blade Positioning

Defines the controlling parameters for blade alignment for positioning moves, Blade alignment positioning mode, alignment position, and positioning feed rates are setup in this section.

16.3.1 Blade Alignment Positioning Mode

Enter either ON or OFF to enable or disable blade alignment positioning mode at the beginning of the program. Blade positioning mode is used to align the blade during a *RAPID* move according to the direction of the next cutting move, This places the blade in the proper orientation at the beginning of the move to make the next cut.

The *ALIGN/MOVE* command can be used to enable/disable the blade positioning alignment mode at any time within the part program.

16.3.2 Distance Above Location To Align Blade

Enter the distance above the next cutting location at which to align the blade during a *RAPID* move. The blade will be positioned to this location aligned to the angle required by the next cut. This value can be changed in the part program by using the *ALIGN/MOVE* command.

16.3.3 Enter Feed Rate To Position The Tool

Enter the feed rate to use when positioning the blade above the next cutting location when in blade alignment positioning mode. Entering a value of zero will cause the blade to be positioned in *RAPID* mode. This value can be changed in the part program by using the *ALIGN/MOVE* command.

16.3.4 Enter Feed Rate To Plunge The Tool

Enter the feed rate to use when plunging the blade from the alignment position location to the cutting position. Entering a value of zero will cause the blade to be positioned in *RAPID* mode. This value can be changed in the part program by using the *ALIGN/MOVE* command.

16.4 Smooth Cutting Command Set-up

This section defines the codes and parameters to be used with the curve blending command (*SMOOTH*).

16.4.1 Smooth ON Code

Enter the register and its value which will enable basic spline interpolation mode at the machine. This code is usually G(36).

16.4.2 Smooth OFF Code

Enter the register and its value which will terminate basic spline interpolation mode at the machine. The code is usually G(37).

16.4.3 Enable Curve Blending Codes

Enter the registers and their values that enable curve blending mode. Curve blending on the control will interpolate a smooth spline through the locations between the *SMOOTH/ATANGL* and *SMOOTH/ATANGL,OFF* commands. **PWorks** will output a directional move tangent to the beginning of the curve prior to initiating curve blending mode.

The curve blending enable code usually consists of a single register with no value which is set up using the following example:

<u>Begin</u>	<u>Format</u>	<u>Left</u>	<u>Right</u>	<u>Min</u>	<u>End</u>	<u>Format</u>
(SMOO	DECIMAL	0	0	0	TH-SP	SET

16.4.4 Disable Curve Blending Codes

Enter the registers and their values that terminate curve blending mode. Curve blending on the control will interpolate a smooth blending mode. Curve blending on the control will interpolate a smooth spline through the locations between the *SMOOTH/ATANGL* and *SMOOTH/ATANGL,OFF* command.

The curve blending off codes usually consist of two registers with no values which are set up using the following example:

<u>Begin</u>	<u>Format</u>	<u>Left</u>	<u>Right</u>	<u>Min</u>	<u>End</u>	<u>Format</u>
(SMOO	DECIMAL	0	0	0	TH_SP	SET
)	DECIMAL	0	0	0		SET

16.4.5 Curve Blending Limit Angle Register

Enter the register to use as the limiting angle for curve blending mode. This register sets the maximum angular change allowed during curve blending. If a move changes direction by more than this angle while in curve blending mode, then curve blending mode will be disabled for this set of locations.

This register is usually coupled tightly with the curve blending enable codes and is set up using the following example:

<u>Begin</u>	<u>Format</u>	<u>Left</u>	<u>Right</u>	<u>Min</u>	<u>End</u>	<u>Format</u>
:	FLOATING	3	3	1)	SET

16.4.6 Maximal Curve Blending Direction Change

During curve blending mode, the control will interpolate a spline through a series of locations which in essence generates a much smoother path than pure linear interpolation will achieve. Large direction changes during curve blending mode will generate greater deviations from the programmed tool path, sometimes creating undesirable results.

The maximum angular change of direction for curve blending mode can be controlled by either entering a default angle at this prompt, or by placing the maximal angular change in the *SMOOTH/ATANGL,ang* command. The value entered at this prompt will become the default angle for the *SMOOTH/ATANGL* command.

16.4.7 Initial Span For Initiating Curve Blending Mode

Curve blending mode requires a short span in the direction of the spline be output prior to the initiating codes for curve blending. Enter the default distance of this span at this prompt. **PWorks** will look ahead to the first few moves of the intended spline and output a span at the specified distance that is tangent to the calculated spline prior to outputting the curve blending enable codes. You can also specify this distance in the part program by using the *SMOOTH/ATANGL,ang,dis* command.

16.5 Vacuum Pods

This section defines the codes which control the Vacuum Pods on Ultrasonic Cutter type machines.

16.5.1 Pod Vacuum Pump OFF Code

Enter the register and its value that will turn the vacuum pump off. M(40) is usually used to turn off the vacuum pump on machines which support Vacuum Pods.

This code can be output using the *POD/AIR,OFF* command.

16.5.2 Pod Vacuum Pump ON Code

Enter the register and its value that will turn the vacuum pump on. M(41) is usually used to turn on the vacuum pump on machines which support Vacuum Pods.

This code can be output using the *POD/AIR,ON* command.

16.5.3 Pod Vacuum OFF Code

Enter the register and its value that will turn the vacuum to the pods off. M(140) is usually used to turn the vacuum off on machines which support Vacuum Pods.

This code can be output using the *POD/AIR,NOMORE* command.

16.5.4 Disable Pod Vacuum OFF Code

Enter the register and its value that will disable the operator from using the foot switch to turn off the vacuum to the pods. This code is usually M(141).

This code can be output using the *POD/LOCK,ON* command.

16.5.5 All Pods Down Code

Enter the register and its value that will cause all pods to unclamp, move down to their rest stop, clamp, and turn the vacuum to the pods of. This code is usually M(142).

This code can be output using the *POD/DOWN* command.

16.5.6 All Pods Low Pressure Code

Enter the register and its value that selects a low pressure vacuum to all pods. This code is usually M(148).

This code can be output using the *POD/AIR,LOW* command.

16.5.7 All Pods High Pressure Code

Enter the register and its value that selects a high pressure vacuum to all pods. This code is usually M(149).

This code can be output using the *POD/AIR,HIGH* command.

16.5.8:9 Pod Row/Column Address Registers

Ultrasonic Cutter type machines which support Vacuum Pods can usually address each pod individually to move them up, turn off vacuum pressure, etc. An individual pod is addressed by a row and column position.

Enter the register to use as the row and column position for individually addressed pods. Rows are normally counted along the X-axis and columns along the Y-axis.

16.5.10 Address Single Pod Code

Enter the register and its value which defines a block as addressing a single Vacuum Pod. this block will contain this code (usually G(140)), the row and column position of the addressed pod, and the operation to perform on the pod.

16.5.11 Addressed Pod Unclamp Code

Enter the register and its value which will unclamp the addressed Vacuum Pod. This code is usually M(143) and is output using the *POD/row,col,CLAMP,OFF* command.

16.5.12 Addressed Pod Clamp Code

Enter the register and its value which will clamp the addressed Vacuum Pod. This code is usually M(144) and is output using the *POD/row,col,CLAMP,ON* command.

16.5.13 Addressed Pod No Pressure Code

Enter the register and its value which will turn off the vacuum at the addressed pod. This code is usually M(145) and is output using the *POD/row,col,AIR,OFF* command.

16.5.14 Addressed Pop Up Code

Enter the register and its value which will raise the addressed Vacuum Pod. This code is usually M(146) and is output using the *POD/row,col,UP* command.

16.5.15 Addressed Pod Up/Clamp Code

Enter the register and its value which will unclamp the addressed Vacuum Pod, then raise it and clamp it. This code is usually M(147) and is output using the *POD/row,col,UP,CLAMP* command.

16.6 Exit

Exits the Ultrasonic Blade Configuration section and returns you to the previous section.

CHAPTER 17 Stringer Drilling Machine

17.0 Walk Through Ultrasonic Blade Configuration

This selection will lead you through all of the subsections of the Stringer Drilling Machine configuration menus. All aspects of defining this style of machine is covered in these sections.

17.1 Stringer Drill Set-up

This section defines the configuration of a Stringer Drilling machine, including the default head at the start of the program and various axis offset values.

17.1.1 Enter Active Head At Start Of Program

Define which Head will be active by default at the start of the program. This is typically Head 1, which is the standard milling head.

17.1.2 Enter Offset Distance For Head 2 Calculations

Heads 1 and 2 are on the same machine component and therefore move with one another. Enter the distance between Head 2 and Head 1 that will be used when calculating the Head 2 position from the input coordinates. This distance is typically 25.629 inches on a Torres machine.

17.1.3 Enter Pivot Offset Distance For Head 3 And 4 Calculations

Enter the pivot distance to add to the tool length when calculating the output axes positions for Heads 3 and 4.

17.2 Axes Register Description

Defines the output registers for each of the axes located on heads 2, 3, and 4. The head 1 axes registers are defined in the normal section for defining the machine axes registers.

17.2.1 Define registers for Head #

Enter the Head number that you want to define the axis registers for.

17.2.2 Register for X-axis

Enter the register descriptor for the absolute X-axis position for the corresponding head. Usually the *XI* register is used for Head 2, *UI* for Head 3, and *II* for Head 4.

17.2.3 Register for Y-axis

Enter the register descriptor for the absolute Y-axis position for the corresponding head. Usually the *YI* register is used for Head 2, *VI* for Head 3, and *JI* for Head 4.

17.2.4 Register for Z-axis

Enter the register descriptor for the absolute Z-axis position for the corresponding head. Usually the *ZI* register is used for Head 2, *WI* for Head 3, and *KI* for Head 4.

17.2.5 Register for Rotary axis #1

Enter the register descriptor for the absolute Rotary axis #1 located on Head 2. Usually the *AA* register is used.

17.2.6 Register for Rotary axis #2

Enter the register descriptor for the absolute Rotary axis #2 located on Head 2. Usually the *AB* register is used.

17.2.7 Register for Rotary axis #3

Enter the register descriptor for the absolute Rotary axis #3 located on Head 2. Usually the *AC* register is used.

17.3 Stringer Codes.

Defines the user defined blocks to output when activating the various heads and special codes relevant to the Stringer Drilling machine.

17.3.1 User Defined Block For Head #1 Activation

Enter the User defined block to output when Head #1 is activated. This block typically contains codes that are used to initialize and activate this head.

17.3.2 User Defined Block For Head #2 Activation

Enter the User defined block to output when Head #2 is activated. This block typically contains codes that are used to initialize and activate this head.

17.3.3 User Defined Block For Head #3 Activation

Enter the User defined block to output when Head #3 is activated. This block typically contains codes that are used to initialize and activate this head.

17.3.4 User Defined Block For Head #4 Activation

Enter the User defined block to output when Head #4 is activated. This block typically contains codes that are used to initialize and activate this head.

17.3.5 User Defined Block For Head #1 Home Move

Enter the User defined block to output when Head #1 is moved to its home position using the 'HEAD/1,HOME' command. This block is typically similar to the User defined block for Head #1 activation, but with an extra code or two added.

17.3.6 Code For Clip holder Slide Home Position

Enter the register and its value that positions the clip holder slide at its home position. This is typically AQ(7.2465).

17.3.7 Spindle ON Code For Drilling Head

Enter the register and its value for turning on the drilling heads' (3 & 4) spindles. This code is typically (M45).

17.3.8 Spindle OFF Code For Drilling Head

Enter the register and its value for turning off the drilling heads' (3 & 4) spindles. This code is typically (M46).

17.3.9 Coolant ON Code For Drilling Head

Enter the register and its value for turning on the drilling heads' (3 & 4) coolant. This code is typically (M10).

17.3.10 Coolant OFF Code For Drilling Head

Enter the register and its value for turning off the drilling heads' (3 & 4) coolant. This code is typically (M11).

17.3.11 Code For Activating The Milling Head

Enter the register and its value that will activate the milling head. This code is usually M(38).

17.3.12 Code For Activating The Drilling Heads

Enter the register and its value that will activate the drilling heads. This code is usually M(35).

17.4 Current Axes Position

This section defines the default position at the beginning of the program for each of the linear and rotary axes located on heads 2, 3, and 4.

17.4.1 Current Axes Positions For Head #

Enter the Head number that you want to define the current position for.

17.4.2 Current Position For X-axis

Enter the default position for the X-axis of the Head being defined at the start of the program. This location is input as the actual machine position and not the tool tip position.

17.4.3 Current Position For Y-axis

Enter the default position for the Y-axis of the Head being defined at the start of the program. This location is input as the actual machine position and not the tool tip position.

17.4.4 Current Position For Z-axis

Enter the default position for the Z-axis of the Head being defined at the start of the program. This location is input as the actual machine position and not the tool tip position.

17.4.5 Current Position For Rotary Axis #1

Enter the default position for Rotary axis #1 located on Head #2 at the start of the program.

17.4.6 Current Position For Rotary Axis #2

Enter the default position for Rotary axis #2 located on Head #2 at the start of the program.

17.4.7 Current Position For Rotary Axis #3

Enter the default position for Rotary axis #3 located on Head #2 at the start of the program.

17.5 Home Position

This section defines the machine Home position, for use by the '*HEAD/n,HOME*' command.

17.5.1 Home Positions For Head #

Enter the Head number that you want to define the Home position for.

17.5.2 X-axis Home Position

Enter the machine's X-axis home (reference point) position for the corresponding Head. The '*HEAD/n,HOME*' command can be used to position the Head to its home position.

17.5.3 Y-axis Home Position

Enter the machine's Y-axis home (reference point) position for the corresponding Head. The '*HEAD/n,HOME*' command can be used to position the Head to its home position.

17.5.4 Z-axis Home Position

Enter the machine's Z-axis home (reference point) position for the corresponding Head. The '*HEAD/n,HOME*' command can be used to position the Head to its home position.

17.5.5 Rotary Axis #1 Home Position

Enter the machine's Rotary axis #1 home (reference point) position for Head #2. The '*HEAD/2,HOME*' command can be used to position the Head to its home position.

17.5.6 Rotary Axis #2 Home Position

Enter the machine's Rotary axis #2 home (reference point) position for Head #2. The '*HEAD/2,HOME*' command can be used to position the Head to its home position.

17.5.7 Rotary Axis #3 Home Position

Enter the machine's Rotary axis #3 home (reference point) position for Head #2. The '*HEAD/2,HOME*' command can be used to position the Head to its home position.

17.6 Machine Limits

This section defines the axes travel limits of the axes located on heads 2, 3, and 4.

17.6.1 Machine limits for Head #

Enter the Head number that you want to define the machine limits for. The limits for Head #2 are checked a bit differently in that axis deltas are checked against the limit range (minimum/maximum) rather than the axis positions being checked against the absolute limit values.

17.6.2 X-axis Travel Limits [Min, Max]

Enter the minimum and maximum travel limits for the X-axis associated with the corresponding head.

17.6.3 Y-axis Travel Limits [Min, Max]

Enter the minimum and maximum travel limits for the Y-axis associated with the corresponding head.

17.6.4 Z-axis Travel Limits [Min, Max]

Enter the minimum and maximum travel limits for the Z-axis associated with the corresponding head.

17.6.5 Rotary Axis #1 Travel Limits [Min, Max]

Enter the minimum and maximum travel limits for Rotary axis #1 associated with Head #2.

17.6.6 Rotary Axis #2 Travel Limits [Min, Max]

Enter the minimum and maximum travel limits for Rotary axis #2 associated with Head #2.

17.6.7 Rotary Axis #3 Travel Limits [Min, Max]

Enter the minimum and maximum travel limits for Rotary axis #3 associated with Head #2.

17.7 Exit

Exits the Stringer Machine Configuration section and returns you to the previous section.

APPENDIX A Print Descriptor File

A.1 Introduction

The Print Descriptor file (*PDF*) is a user generated text file that contains information which defines the look of the **PostWorks** print file. Both text strings and variable data can be used to define the print file output.

The *PDF* file consists of up to 10 record definitions which describe the format of the print file output. Records can be defined for the print file page header, page trailer, motion record, non-motion record, *INSERT* records, *STOP*, *OPSTOP* & *LOADTL* messages, *PPRINT* commands, and for the *FINI* card.

A.2 Record Definitions

Format: **#RECORD n# [PAGE,BEFORE,AFTER]**

Print file record definitions consist of a *RECORD* statement, multiple *LINE* definitions within the record, and multiple text string and variable definitions within the *LINE* definition. The *RECORD* statement defines start of a new record.

'n' defines the record number and can be in the range of 1-10. 'PAGE', 'BEFORE' and 'AFTER' are optional qualifiers and must appear within square brackets ([]) when they are specified. 'PAGE' specifies that a new page will be output prior to this record. 'BEFORE' specifies that a blank line will be output prior to this record. 'AFTER' specifies that a blank line will be output at the end of this record.

Example

```
#RECORD 2# [ PAGE,AFTER ]
```

A.2.1 Line Definitions

Format: **#LINE n# [pstvar,PAGE,BEFORE,AFTER]**

The *LINE* statement begins a single line definition within a record. You can define a maximum of 6 lines per record. Each *LINE* definition will consist of a single text string and multiple variable definitions.

'n' defines the line number and can be in the range of 1-6. 'pstvar' is an optional *Post* Variable that defines the number of times this line should be output. It is used when you are outputting an array of values such as Tool data at the end of the program.

'PAGE', 'BEFORE' and 'AFTER' are optional qualifier and are the same on the *RECORD* statement, except they apply to the current line.

Example

```
#LINE 1# [ %NTOOL,BEFORE ]
```

A.2.2 Text Data

Format: **/TEXT/** ~text~

The *TEXT* statement defines the text string for the current *LINE* definition. You can have only one *TEXT* statement per *LINE* definition. 'text' is a text string that will be output on this line and must be delimited by tilders(~). If the text string is too long to fit in one line, you can continue it on the next line, but the text string on each line must be enclosed by tildes.

Example

```
/TEXT/ ~ Control Tape          Input~  
      ~ Clred~
```

A.2.3 Variable Data

Format: **/VARS/** [var,start,end,format,RIGHT ,>] , ...
 LEFT <
 CENTER *
 ~

The *VARS* statement defines variables that will be output on the current *LINE* definition and how they will be formatted. You can define a maximum of 15 variables per record. The variable format data must be enclosed in square brackets ([]). You can have more than 1 variable definition on a single line and continue them on multiple lines (though a single variable definition must be contained on a single line).

'var' defines a variable to be output on this line can be one of the following.

TAPEBLK	=	Output Control Tape Block.
LASTBLK	=	Last Control Tape Block output to the punch file.
DATE	=	Current date in the format '01-JAN-2000'
TIME	=	Current time in the format '13:12:32'.
PCHBRIEF	=	Name of output punch file which does not include the device and directory name.

PCHFILE	=	Name of output punch file which includes the device and directory name.
PCHNAME	=	Name of output punch file without the device, directory name and file extension.
COMPUTER	=	Computer name that PostWorks is being run on.
USERNAME	-	Account name PostWorks is being run under.
REVDATE	=	Revision date of PostWorks .
PARTNO	=	Text of the last encountered PARTNO card.
MPARTNO(n)	=	Text of each PARTNO card, with n specifying which PARTNO to access. A maximum of 100 PARTNO cards is allowed.
PPRINT	=	Text from last or current PPRINT command.
TLNAME(n)	=	Textual description of tools used in the program. This information comes from the ' PPRINT TLN ', tool description' command.
RECAP	=	Slide motion recap values, which contain the minimum and maximum travel limit, delta travel, and the delta movement for the specified axis. The recap data is a text string 50 characters in length. 'n' selects the axis which will generate the recap data and can be one of the following values. 1-3 = Input XYZ's. 4-6 = XYZ's after TRANS & ORIGIN . 7-9 = XYZ's after being adjusted for the rotary axes. 10-12 = XYZ's after TRANS/LAST . 13-18 = Primary and Secondary linear axes in the order X, U, Y, V, W, Z. 19-22 = Rotary axes on a linear scale. 23-32 = Output linear and rotary axes.
VERSION	=	Version number of PWorks .
MACHDESC	=	Text of the Machine Description Field in the corresponding MDF file.

The following print file variables are defined in the Clfile Header Record. This might be the first record in an APT source file and has the following format.

REMARK cname cldate cltime camnam camversn

Note that not all CAM systems are capable of creating this record.

CAMNAME = Name of CAM processor used to create the input clfile.

CAMVERSN	=	Version number of CAM processor.
CLNAME	=	Original name of clfile, as created by the CAM processor.
CLDATE	=	Creating date of the input clfile.
CLTIME	=	Creating time of the input clfile.
<i>Post Variable</i>	=	Any supported <i>Post</i> Variable in the form ' <i>%name</i> '. You may specify a single subscript with the variable name. A subscript of '*' specifies that the loop increment from the <i>LINE</i> statement will be used. See the <i>PostMacro</i> manual for a description of the available Post Variables .
Register	=	Any valid Register descriptor (without a value), for example, 'G1, X1, Y, F2, \$'. You may also specify 'G' or 'M' which will print a list of all G or M codes in the current Control Tape block.

'start' and 'end' specify the columnar starting and ending position within the current line to place the formatted 'var' text.

'format' specifies a Register descriptor to use when formatting a *Post* Variable. The parameters entered in ***MPost***'s [Tape Register Format](#) form for the specified register will be used. 'format' is only used when 'var' is a *Post* Variable or *RECAP*.

'RIGHT' specifies that the variable will be right justified in the slot defined by 'start' and 'end'. 'LEFT' specifies left justification. 'CENTER' centers the variable within the slot.

'>', '<', '*' and '~' are overflow control characters and describe what should happen if the variable text is too long to fit within the slot specified by 'start' and 'end'. '>' truncates the characters to the right and inserts the '>' character at the end. '<' truncates the characters to the left and inserts '<' character at the beginning. '*' fills the entire slot with asterisks. '~' wraps the overflow text to the next line.

Example

```
/VARS/ [ TAPBLK,1,51,C1,LEFT,~ ], [ %ISN,54,58,C1,RIGHT < ]
        [ %CLREC,61,65,C1,RIGHT,< ]
```

A.3 Example Print Descriptor File

In the following example, the *PDF* records have been assigned to the ***PostWorks*** output sequences as follows.

```
Page Header Record: 1
Page Trailer Record: (none)
Motion Block Record: 2
Non-motion Block Record: 3
INSERT Record: 4
```

STOP Record: 5
 OPSTOP Record: 5
 LOADTL Record: 5
 FINI Record: 6,7
 PPRINT Record: 8

```
=====

#RECORD 1# [PAGE,AFTER]
#LINE 1#
/TEXT/ ~NCCS default 5-axis machine center
~
/VARS/ [%PAGENO,129,132,N,RIGHT,*]

#LINE 2#
/TEXT/ ~PARTNO~
/VARS/ [PARTNO,7,72,C1,LEFT,>]

#LINE 3#
/TEXT/ ~Rundate: Punch File:~
/VARS/ [DATE,10,20,C1,LEFT,>], [TIME,23,31,C1,LEFT,>]
[PCHFILE,49,132,C1,LEFT,>]

#LINE 4# [BEFORE]
/TEXT/ ~ Control Tape Input Clrec ~
~X-axis Y-axis Z-axis A-axis B-axis IPM Time RPM~

#RECORD 2#
#LINE 1#
/VARS/ [TAPEBLK,1,51,C1,LEFT,~], [%ISN,54,58,N,RIGHT,<]
[%CLREC,61,65,N,RIGHT,<], [%XYZ(4),67,76,C2,RIGHT,>]
[%XYZ(5),78,87,C2,RIGHT,>], [%XYZ(6),89,98,C2,RIGHT,>]
[%AXIS(8),100,107,C3,RIGHT,>], [%AXIS(7),109,116,C3,RIGHT,>]
[%FEED(2),118,122,C4,RIGHT,>], [%MOVTIM,124,127,C5,RIGHT,>]
[%RPM,129,132,C1,RIGHT,>]

#RECORD 3#
#LINE 1#
/VARS/ [TAPEBLK,1,51,C1,LEFT,~], [%ISN,54,58,N,RIGHT,<]
[%CLREC,61,65,N,RIGHT,<]

#RECORD 4#
#LINE 1#
/TEXT/ ~
~ -- I N S E R T --~
/VARS/ [TAPEBLK,1,51,C1,LEFT,~], [%ISN,54,58,N,RIGHT,<]
[%CLREC,61,65,N,RIGHT,<]

#RECORD 5# [BEFORE,AFTER]
#LINE 1#
/TEXT/ ~ Tape Length = Total Sequence Time =~
~ Total Machine Time =~
/VARS/ [%TAPLEN,26,35,C4,LEFT,>], [%SEQTIM,64,73,C5,LEFT,>]
```

```

[%MCHTIM,102,111,C5,LEFT,>]

#LINE 2# [BEFORE]
/TEXT/ ~          Delta X =          Delta A =          ~
      ~Delta X =          Delta A =~
/VARS/ [%AXSDLS(1),25,34,C2,RIGHT,>], [%AXSDLS(8),51,60,C3,RIGHT,>]
      [%XYZDLS(4),83,92,C2,RIGHT,>], [%AXSDLS(8),109,118,C3,RIGHT,>]

#LINE 3#
/TEXT/ ~          Delta Y =          Delta B =          ~
      ~Delta Y =          Delta B =~
/VARS/ [%AXSDLS(3),25,34,C2,RIGHT,>], [%AXSDLS(7),51,60,C3,RIGHT,>]
      [%XYZDLS(5),83,92,C2,RIGHT,>], [%AXSDLS(7),109,118,C3,RIGHT,>]

#LINE 4#
/TEXT/ ~          Delta Z =          ~
      ~Delta Z =~
/VARS/ [%AXSDLS(5),25,34,C2,RIGHT,>], [%XYZDLS(6),83,92,C2,RIGHT,>]

#RECORD 6# [PAGE,BEFORE]
#LINE 1# [BEFORE,AFTER]
/TEXT/ ~          Ft of Tape          PP Fatals          PP Errors          PP Warnings~
      ~          Machine Time =~
/VARS/ [%TAPLEN,1,7,C4,RIGHT,>], [%NUMFAT,21,25,C1,RIGHT,>]
      [%NUMERR,37,41,C1,RIGHT,>], [%NUMWRN,53,57,C1,RIGHT,>]
      [%MCHTIM,91,100,C5,LEFT,>]

#LINE 2# [BEFORE]
/TEXT/ ~          ~
      ~Slide Motion Recap~

#LINE 3# [BEFORE,AFTER]
/TEXT/ ~          Min          Max          Travel          Delta          ~
      ~          Min          Max          Travel          Delta          ~

#LINE 4#
/TEXT/ ~          X-Part          ~
      ~          X-Part~
/VARS/ [%RECAP(23),13,63,C2,LEFT,>], [%RECAP(4),80,130,C2,LEFT,>]

#LINE 5#
/TEXT/ ~          Y-Part          ~
      ~          Y-Part~
/VARS/ [%RECAP(25),13,63,C2,LEFT,>], [%RECAP(5),80,130,C2,LEFT,>]

#LINE 6#
/TEXT/ ~          Z-Part          ~
      ~          Z-Part~
/VARS/ [%RECAP(27),13,63,C2,LEFT,>], [%RECAP(6),80,130,C2,LEFT,>]

#RECORD 7#
#LINE 1#
/TEXT/ ~          A-Part          ~
      ~          A-Part~

```

```

/VARS/   [RECAP(30),13,63,C3,LEFT,>], [RECAP(20),80,130,C3,LEFT,>]

#LINE 2# [AFTER]
/TEXT/   ~      B-Part                      ~
         ~      B-Part~
/VARS/   [RECAP(29),13,63,C3,LEFT,>], [RECAP(19),80,130,C3,LEFT,>]

#LINE 3# [BEFORE,AFTER]
/TEXT/   ~                                     TOOL      LENGTH~
         ~      TIME~

#LINE 4# [%NTOOL]
/VARS/   [%TLNO(*),48,55,C1,RIGHT,>], [%TLEN(*),58,69,C2,RIGHT,>]
         [%TLTIM(*),72,83,C5,RIGHT,>]

#RECORD 8#
#LINE 1#
/TEXT/   ~PPRINT~
/VARS/   [%PPRINT,7,72,C1,LEFT,>]

```

APPENDIX B Forms

B.1 Introduction

This Appendix contains a variety of forms that can assist you in the initial generation of a post-processor configuration. These forms cover the basic parameters required to generate a new post-processor using the **MPost** routine. It is recommended to make copies of the forms prior to using them and to leave the originals with the **MPost** reference manual.

The fields contained in the forms are described in the following sections.

B.2 Machine Description Form

Machine Type

'MILL', 'LATHE', 'Ultrasonic Cutter', or 'Mill/Turn'

Units

'INCH' or 'MM'

Spindle Axis

Contains the spindle axis vector of the machine, usually 0,0,1.

Example:

Machine Type: **MILL** Units: **INCH** Spindle Axis: 0,0,1

Spindle Limits

'Spindle Limits' defines the lower and upper limits for each of the supported spindle speed ranges. If the target machine contains only a single range, then use the 'Med' fields.

Example:

Low: **25 , 200** Med: **250 , 1200** High: **1000 , 4000**

Linear Axes

The 'Linear Axes' section defines the linear axes supported.

n-Reg

Enter the registers that will be used for the absolute and incremental positions for each of the supported axes.

Att/Det

'ATTACHED' or 'DETACHED' Should be filled out only when the major axis supports both a Primary and Secondary axis.

Dist

Enter the distance between the primary and secondary axis for this major axis when they are 'DETACHED'.

Example:

X-Reg: **X1 , X2** Reg: **U1 , U2** Att/Det: **ATTACHED** Dist:

Rotary Axis #n

The 'Rotary Axis' section defines the rotary axes supported. There is a separate section for each of the rotary axis.

Reg

Enter the registers that will be used for the internal, absolute, and incremental positions for this axis.

Table / Head

Mark either 'Rider' or 'Carrier' under the appropriate rotary type (Table or Head). The rotary axis types are listed in the order in which they must be defined.

1. Table riders.
2. Table carriers.
3. Head carriers
4. Head riders.

Rot. Axis

Enter the linear axis which the rotary axis rotates about, either 'XAXIS', 'YAXIS' or 'ZAXIS'.

Pos/Cont

'POSITION' or 'CONTOUR'

Scale

'LINEAR' or 'ROTARY'. Also enter the rotary axis circumference for 'LINEAR' scale axes.

Feed Control

'YES' or 'NO' specifies whether a positioning only rotary axis has feed rate control.

Positive Dir

'CLW' or 'CCLW'

Center

Enter the center coordinates of the rotary axis.

Example:

Reg: **A1,A2,A3**
Rot. Axis: **XAXIS**
Feed Control:
Center: **0,0,10**

Table (Rider,**Carrier**)
Pos/Cont: **CONTOUR**
Positive Dir: **CLW**

Head (Carrier,Rider)
Scale: **LINEAR,360**

B.3 Axis Considerations

Mutually Exclusive Axes

Exclusive Axes

Enter any axes that cannot be programmed to move simultaneously. You can enter up to 4 sets of mutually exclusive axes.

Con. Axis

Enter the controlling axis for each set of mutually exclusive axes. The order in which the axes are output is determined by this axis.

Negative Move

Enter the order in which to output the axes when the mutually exclusive axes move at the same time and the controlling axis moves in the negative direction. Use the '\$' character to signify an End-of-Block.

Positive Move

Enter the order in which to output the axes when the mutually exclusive axes move at the same time and the controlling axis moves in the positive direction.

Example:

	<u>Exclusive Axes</u>	<u>Con. Axis</u>	<u>Negative Move</u>	<u>Positive Move</u>
#1	X1,A	Z1	X1,Y1,Z1,A1,\$	Z1,\$

Axis Positions

Axis

The axis column should contain an entry for each of the supported linear and rotary axes.

Home

Enter the machine home "lights on" position for each axis.

Current

Enter the default starting position for each axis.

Increment

Enter the minimum distance that each axis can move in a single block.

TRANS

Enter the default *TRANS/-AXIS* value, which will be added to the output axis position, for each axis.

SCALE

Enter the default *TRANS/SCALE,-AXIS* value, which the output axis position will be multiplied by, for each axis.

Example:

<u>Axis</u>	<u>Home</u>	<u>Current</u>	<u>Increment</u>	<u>Trans</u>	<u>Scale</u>
X1	40.0	40.0	.0001	0.	1.

Machine Limits

The axis column should contain an entry for each of the supported linear and rotary axes.

Min

Enter the minimum travel limit for each axis.

Max

Enter the maximum travel limit for each axis.

Feed

Enter the maximum programmable feed rate for each axis.

Rapid

Enter the rapid traverse rate for each axis.

Example:

<u>Axis</u>	<u>Min</u>	<u>Max</u>	<u>Feed</u>	<u>Rapid</u>
X1	-40	40	100.0	200.0

B.4 Interference Zones

Axis

The Axis columns should contain only the axes that should be considered for this spindle/fixture collision (interference) zone. You can define up to 4 interference zones, each containing up to 10 axes.

Min

Enter the minimum travel limit to be considered as part of the interference zone for this axis.

Max

Enter the maximum travel limit to be considered as part of the interference zone.

Example:

<u>Axis</u>	<u>Min</u>	<u>Max</u>	<u>#1</u>	<u>Axis</u>	<u>Min</u>	<u>Max</u>
X1	-10000	-40		A1	90	90

B.5 Register Format

Begin

Enter the beginning identifying string for each register. You can enter up to 10 characters.

Format

Choose from one of the following format descriptors.

- DECIMAL - Contains a decimal point. Whole numbers will drop the decimal point.
- FLOATING - All numbers contain a decimal point.
- LEADING - Leading zero suppression.
- TRAILING - Trailing zero suppression.

Sign

Enter the signs that the formatted register value should contain. You can enter '+' for positive numbers and/or '-' for negative numbers. '-' can be changed to '*', signifying that a value of zero should contain a minus sign. Leave it blank for unsigned numbers. You can also enter 'PN' as the sign with which a positive numbers will contain 'P(' as the sign and a negative numbers will contain 'N(' as the sign. When 'PN' is specified, be sure to add a ')' as the ending character for this register.

Left

Enter the number of digits to the left of the (implied) decimal point for both Inch and Millimeter output.

Right

Enter the number of digits to the right of the (implied) decimal point for both Inch and Millimeter output.

End

Enter the ending identifying string for each register. You can enter up to 10 characters.

Control

Choose from one of the following output control parameters.

- NOT_USED - This register will never be output.
- ALWAYS - This register will be output on every block.
- SET - This register will be output whenever internally set by **PostWorks**.
- CHANGED - This register will be output whenever it changes.
- NONZERO - This register will be output whenever it has a value other than 0.
- MOTION - This register will only be output on motion blocks when its value changes.

Purpose

Enter the purpose of this register; absolute X-axis, plane selection codes, cycle parameter, etc. This field is used for reference only.

Example:

<u>Reg</u>	<u>Begin</u>	<u>Format</u>	<u>Sign</u>	<u>Left</u>		<u>Right</u>		<u>Min</u>	<u>Min</u>	<u>End</u>
				<u>IN</u>	<u>/MM</u>	<u>IN</u>	<u>/MM</u>	<u>Left</u>	<u>Right</u>	
X1	X	DECIMAL	-	3	4	4		3	0	1

Control Purpose

Changed Absolute X-axis

B.6 G/M-Codes

G/M-code

Enter the G or M-code.

Group

Enter the group that the G/M-code belongs to, in the range 0-A.

Function

Enter the function that the G/M-code will perform at the machine. This field is for reference only.

Command

Enter the command which will output this G/M-code. This field is for reference only.

Example:

<u>G-code</u>	<u>Group</u>	<u>Function</u>	<u>Command</u>
G04	0	Machine dwell	DELAY/n

B.7 Register Defaults

Preset Register Values

Enter the default value for each supported register. This will be the initial value stored in the register when **PostWorks** is started.

Example:

G3 17 G4 40 G5 80 G6 90

Register Order within Tape Block

Enter the order in which the supported registers should be output in a Control Tape block. You only need to enter the supported registers.

Example:

N O G0 G1 G2 G3 G4 X1 X2 Y1 Y2 Z1 Z2 F1 S

B.8 Spindle / Feedrate Tables

The Spindle/Feedrate Tables are used to define a table of values when either the spindle speed or feed rate is output using a predefined table. First, identify the purpose of the table, for example, 'Medium Range Table', 'IPM Feed Rates', etc. Then enter the values and speeds that make up the table.

The value to the left of the equals and under the heading 'Code' is the value that will be output when the programmed speed matches the value to the right of the equals and under the heading 'Speed'.

Example:

Table: **IPM Feed Rates**

Code	=	Speed	Code	=	Speed	Code	=	Speed	Code	=	Speed
01		1.2	02		2.4	03		3.8	04		5.6

B.9 User Defined Blocks

Purpose

Enter the function of this [User Defined Block](#), for example, 'Tool Change', 'Canned cycle w/dwell', 'Alignment block', etc.

Definition

Enter the format of this User Defined Block, including all registers and values that will be output. Use the '\$' character to signify an End-Of-Block.

Example;

Block #1 Purpose; **Alignment Block**

Definition: **O,G(0),G(90),X1,Y1,Z1,F,S,M1**

B.10 Cycles

Cycle Modes

'Cycle Modes' defines the canned cycle modes supported. There is a separate form for milling and lathe cycles.

Code

Enter the G-code for each of the supported canned cycle modes.

Block

Enter the number of the User Defined Block for each supported mode, if required.

Time

Enter the manual cycle mode to use for estimating the machining time for each supported canned cycle.

Example:

Code	Block	Time
86	BORE	BORE8

Cycle Parameters

Enter the register which will be output for each of the cycle command parameters.

Example

DWELL1	P	DWELL2	STEP1	I2	STEP2	J2
--------	----------	--------	-------	-----------	-------	-----------

Cycle Block

Enter any User Defined Block numbers and their descriptions which will be used for canned cycles.

Example:

Cycle Block # 1 : G(79),I2,J2,F,\$

Cycle Statements

Enter the syntax for each supported cycle command. You can enter multiple modes ('BORE', 'DRILL', etc.) for each command syntax. If a *Macro* has been defined to shorten the syntax of the cycle commands, then you should probably enter the *Macro* syntax here. Optional parameters should be enclosed in brackets.

Example

CYCLE/DRILL, FEDTO, depth, RAPTO, r ,IPM, feed, \$
REAM [REPEAT,rep]
TAP

B.11 Print Descriptor Records

A print descriptor record form is furnished for both a 132 column listing and an 80 column listing. Each form contains six line grids that can be used to define the format of the print file. The print descriptor records are contained in the [Print Descriptor File](#) (.pdf).

Record

Enter the PDF record number, i.e., #RECORD n#.

Type

Enter the output type for this record, for example, LOADTL, MOTION, etc. If this output type requires multiple print descriptor records, because of the number of lines output, then also enter 1 or 2 after the slash (each output type can contain up to 2 print descriptor records).

Registers

Enter any register descriptors and their format used in this print descriptor record. Registers used for formatting the print file are usually not used in the Control Tape output, i.e., they are marked as 'NOT_USED'.

Example:

Record # 3 Registers: C3 / nnnnn

Type: **Non-Motion**

----- **nnnnn nnnnn**

Machine Description

Machine Type: _____ Units: _____ Spindle Axis: _____

Spindle Limits

Low: _____, _____ Med: _____, _____ High: _____, _____

Linear Axes

Primary

Secondary

X-Reg: _____, _____ Reg: _____, _____ Att/Det: _____ Dist: _____

X-Reg: _____, _____ Reg: _____, _____ Att/Det: _____ Dist: _____

X-Reg: _____, _____ Reg: _____, _____ Att/Det: _____ Dist: _____

Rotary Axis # 1

Reg: _____, _____, _____ Table (Rider, Carrier) Head (Carrier, Rider)

Rot.Axis, _____ Pos/Cont: _____ Scale: _____, _____

Feed Control: _____ Positive Dir: _____

Center: _____, _____, _____

Rotary Axis # 2

Reg: _____, _____, _____ Table (Rider, Carrier) Head (Carrier, Rider)

Rot.Axis, _____ Pos/Cont: _____ Scale: _____, _____

Feed Control: _____ Positive Dir: _____

Center: _____, _____, _____

Rotary Axis # 3

Reg: _____, _____, _____ Table (Rider, Carrier) Head (Carrier, Rider)

Rot.Axis, _____ Pos/Cont: _____ Scale: _____, _____

Feed Control: _____ Positive Dir: _____

Center: _____, _____, _____

Machine Description

Rotary Axis # 4

Reg: _____, _____, _____ Table (Rider, Carrier) Head (Carrier, Rider)

Rot.Axis, _____ Pos/Cont: _____ Scale: _____, _____

Feed Control: _____ Positive Dir: _____

Center: _____, _____, _____

Axis Considerations

Mutually Exclusive Axes

Exclusive Axes	Con. Axis	Negative Move	Positive Move
#1 _____	_____	_____	_____
#2 _____	_____	_____	_____
#3 _____	_____	_____	_____
#4 _____	_____	_____	_____

Axis Positions

[illegible]

Machine Limits

[illegible]

Interference Zones

Interference Zones

Axis	Min	Max		Axis	Min	Max
_____	_____	_____	#1	_____	_____	_____
_____	_____	_____		_____	_____	_____
_____	_____	_____		_____	_____	_____
_____	_____	_____		_____	_____	_____
_____	_____	_____		_____	_____	_____
_____	_____	_____	2	_____	_____	_____
_____	_____	_____		_____	_____	_____
_____	_____	_____		_____	_____	_____
_____	_____	_____		_____	_____	_____
_____	_____	_____		_____	_____	_____
—			#3			
_____	_____	_____		_____	_____	_____
_____	_____	_____		_____	_____	_____
_____	_____	_____		_____	_____	_____
_____	_____	_____		_____	_____	_____
_____	_____	_____	#4	_____	_____	_____
_____	_____	_____		_____	_____	_____
_____	_____	_____		_____	_____	_____
_____	_____	_____		_____	_____	_____
_____	_____	_____		_____	_____	_____

Register Format

Reg	Begin	Format	Sign	Left IN / MM		Right IN / MM		Min Left	Min Right	End	Control	Purpose
A1	_____	_____	____	____	____	____	____	____	____	_____	_____	_____
A2	_____	_____	____	____	____	____	____	____	____	_____	_____	_____
A3	_____	_____	____	____	____	____	____	____	____	_____	_____	_____
B1	_____	_____	____	____	____	____	____	____	____	_____	_____	_____
B2	_____	_____	____	____	____	____	____	____	____	_____	_____	_____
B3	_____	_____	____	____	____	____	____	____	____	_____	_____	_____
C1	_____	_____	____	____	____	____	____	____	____	_____	_____	_____
C2	_____	_____	____	____	____	____	____	____	____	_____	_____	_____
C3	_____	_____	____	____	____	____	____	____	____	_____	_____	_____
C4	_____	_____	____	____	____	____	____	____	____	_____	_____	_____
C5	_____	_____	____	____	____	____	____	____	____	_____	_____	_____
C6	_____	_____	____	____	____	____	____	____	____	_____	_____	_____
D	_____	_____	____	____	____	____	____	____	____	_____	_____	_____
E	_____	_____	____	____	____	____	____	____	____	_____	_____	_____
F1	_____	_____	____	____	____	____	____	____	____	_____	_____	_____
F2	_____	_____	____	____	____	____	____	____	____	_____	_____	_____
F3	_____	_____	____	____	____	____	____	____	____	_____	_____	_____
G0	_____	_____	____	____	____	____	____	____	____	_____	_____	_____
G1	_____	_____	____	____	____	____	____	____	____	_____	_____	_____
G2	_____	_____	____	____	____	____	____	____	____	_____	_____	_____
G3	_____	_____	____	____	____	____	____	____	____	_____	_____	_____
G4	_____	_____	____	____	____	____	____	____	____	_____	_____	_____
G5	_____	_____	____	____	____	____	____	____	____	_____	_____	_____
G6	_____	_____	____	____	____	____	____	____	____	_____	_____	_____
G7	_____	_____	____	____	____	____	____	____	____	_____	_____	_____
G8	_____	_____	____	____	____	____	____	____	____	_____	_____	_____
G9	_____	_____	____	____	____	____	____	____	____	_____	_____	_____
GA	_____	_____	____	____	____	____	____	____	____	_____	_____	_____
H	_____	_____	____	____	____	____	____	____	____	_____	_____	_____
I1	_____	_____	____	____	____	____	____	____	____	_____	_____	_____
I2	_____	_____	____	____	____	____	____	____	____	_____	_____	_____

Register Format

Reg	Begin	Format	Sign	Left IN / MM		Right IN / MM		Min Left	Min Right	End	Control	Purpose
J1	_____	_____	____	____	____	____	____	____	____	_____	_____	_____
J2	_____	_____	____	____	____	____	____	____	____	_____	_____	_____
K1	_____	_____	____	____	____	____	____	____	____	_____	_____	_____
K2	_____	_____	____	____	____	____	____	____	____	_____	_____	_____
L	_____	_____	____	____	____	____	____	____	____	_____	_____	_____
M0	_____	_____	____	____	____	____	____	____	____	_____	_____	_____
M1	_____	_____	____	____	____	____	____	____	____	_____	_____	_____
M2	_____	_____	____	____	____	____	____	____	____	_____	_____	_____
M3	_____	_____	____	____	____	____	____	____	____	_____	_____	_____
M4	_____	_____	____	____	____	____	____	____	____	_____	_____	_____
M5	_____	_____	____	____	____	____	____	____	____	_____	_____	_____
M6	_____	_____	____	____	____	____	____	____	____	_____	_____	_____
M7	_____	_____	____	____	____	____	____	____	____	_____	_____	_____
M8	_____	_____	____	____	____	____	____	____	____	_____	_____	_____
M9	_____	_____	____	____	____	____	____	____	____	_____	_____	_____
MA	_____	_____	____	____	____	____	____	____	____	_____	_____	_____
N	_____	_____	____	____	____	____	____	____	____	_____	_____	_____
O	_____	_____	____	____	____	____	____	____	____	_____	_____	_____
P	_____	_____	____	____	____	____	____	____	____	_____	_____	_____
Q	_____	_____	____	____	____	____	____	____	____	_____	_____	_____
R	_____	_____	____	____	____	____	____	____	____	_____	_____	_____
S	_____	_____	____	____	____	____	____	____	____	_____	_____	_____
T	_____	_____	____	____	____	____	____	____	____	_____	_____	_____
U1	_____	_____	____	____	____	____	____	____	____	_____	_____	_____
U2	_____	_____	____	____	____	____	____	____	____	_____	_____	_____
V1	_____	_____	____	____	____	____	____	____	____	_____	_____	_____
V2	_____	_____	____	____	____	____	____	____	____	_____	_____	_____
W1	_____	_____	____	____	____	____	____	____	____	_____	_____	_____
W2	_____	_____	____	____	____	____	____	____	____	_____	_____	_____
X1	_____	_____	____	____	____	____	____	____	____	_____	_____	_____
X2	_____	_____	____	____	____	____	____	____	____	_____	_____	_____

Register Format

Reg	Begin	Format	Sign	Left IN / MM		Right IN / MM		Min Left	Min Right	End	Control	Purpose
Y1	_____	_____	___	___	___	___	___	___	___	_____	_____	_____
Y2	_____	_____	___	___	___	___	___	___	___	_____	_____	_____
Z1	_____	_____	___	___	___	___	___	___	___	_____	_____	_____
Z2	_____	_____	___	___	___	___	___	___	___	_____	_____	_____
AA	_____	_____	___	___	___	___	___	___	___	_____	_____	_____
AB	_____	_____	___	___	___	___	___	___	___	_____	_____	_____
AC	_____	_____	___	___	___	___	___	___	___	_____	_____	_____
AD	_____	_____	___	___	___	___	___	___	___	_____	_____	_____
AE	_____	_____	___	___	___	___	___	___	___	_____	_____	_____
AF	_____	_____	___	___	___	___	___	___	___	_____	_____	_____
AG	_____	_____	___	___	___	___	___	___	___	_____	_____	_____
AH	_____	_____	___	___	___	___	___	___	___	_____	_____	_____
AI	_____	_____	___	___	___	___	___	___	___	_____	_____	_____
AJ	_____	_____	___	___	___	___	___	___	___	_____	_____	_____
AK	_____	_____	___	___	___	___	___	___	___	_____	_____	_____
AL	_____	_____	___	___	___	___	___	___	___	_____	_____	_____
AM	_____	_____	___	___	___	___	___	___	___	_____	_____	_____
AN	_____	_____	___	___	___	___	___	___	___	_____	_____	_____
AO	_____	_____	___	___	___	___	___	___	___	_____	_____	_____
AP	_____	_____	___	___	___	___	___	___	___	_____	_____	_____
AQ	_____	_____	___	___	___	___	___	___	___	_____	_____	_____
AR	_____	_____	___	___	___	___	___	___	___	_____	_____	_____
AS	_____	_____	___	___	___	___	___	___	___	_____	_____	_____
AT	_____	_____	___	___	___	___	___	___	___	_____	_____	_____
AU	_____	_____	___	___	___	___	___	___	___	_____	_____	_____
AV	_____	_____	___	___	___	___	___	___	___	_____	_____	_____
AW	_____	_____	___	___	___	___	___	___	___	_____	_____	_____
AX	_____	_____	___	___	___	___	___	___	___	_____	_____	_____
AY	_____	_____	___	___	___	___	___	___	___	_____	_____	_____
AZ	_____	_____	___	___	___	___	___	___	___	_____	_____	_____

G-codes

G-code	Group	Function	Command
G0			
G1			
G2			
G3			
G4			
G5			
G6			
G7			
G8			
G9			
G10			
G11			
G12			
G13			
G14			
G15			
G16			
G17			
G18			
G19			
G20			
G21			
G22			
G23			
G24			
G25			
G26			
G27			
G28			
G29			
G30			
G31			
G32			
G33			
G34			
G35			
G36			
G37			
G38			
G39			
G40			
G41			
G42			
G43			
G44			
G45			
G46			
G47			
G48			
G49			
G50			
G51			
G52			
G53			
G54			
G55			
G56			
G57			
G58			
G59			
G60			
G61			
G62			
G63			
G64			
G65			
G66			
G67			
G68			
G69			
G70			
G71			
G72			
G73			
G74			
G75			
G76			
G77			
G78			
G79			
G80			
G81			
G82			
G83			
G84			
G85			
G86			
G87			
G88			
G89			
G90			
G91			
G92			
G93			
G94			
G95			
G96			
G97			
G98			
G99			

M-codes

[illegible]

Register Defaults

Preset Register Values

A1		A2		A3		B1	
B2		B3		C1		C2	
C3		C4		C5		C6	
D		E		F1		F2	
F3		G0		G1		G2	
G3		G4		G5		G6	
G7		G8		G9		GA	
H		I1		I2		J1	
J2		K1		K2		L	
M0		M1		M2		M3	
M4		M5		M6		M7	
M8		M9		MA		N	
O		P		Q		R	
S		T		U1		U2	
V1		V2		W1		W2	
X1		X2		Y1		Y2	
Z1		Z2		AA		AB	
AC		AD		AE		AF	
AG		AH		AI		AJ	
AK		AL		AM		AN	
AO		AP		AQ		AR	
AS		AT		AU		AV	
AW		AX		AY		AZ	

Register order within Tape Block

[illegible]

Spindle/Feedrate Tables

Table: _____

[illegible]

Table: _____

[illegible]

User Defined Blocks

Block #1 **Purpose:** _____

Definition: _____

Block #2 **Purpose:** _____

Definition: _____

Block #3 **Purpose:** _____

Definition: _____

Block #4 **Purpose:** _____

Definition: _____

Block #5 **Purpose:** _____

Definition: _____

Block #6 **Purpose:** _____

Definition: _____

Block #7 **Purpose:** _____

Definition: _____

Block #8 **Purpose:** _____

Definition: _____

Block #9 **Purpose:** _____

Definition: _____

Block #10 **Purpose:** _____

Definition: _____

User Defined Blocks

Block #11 **Purpose:** _____

Definition: _____

Block #12 **Purpose:** _____

Definition: _____

Block #13 **Purpose:** _____

Definition: _____

Block #14 **Purpose:** _____

Definition: _____

Block #15 **Purpose:** _____

Definition: _____

Block #16 **Purpose:** _____

Definition: _____

Block #17 **Purpose:** _____

Definition: _____

Block #18 **Purpose:** _____

Definition: _____

Block #19 **Purpose:** _____

Definition: _____

Block #20 **Purpose:** _____

Definition: _____

Milling Cycles

Cycle Modes: Code Block Time Code Block Time

_____	BORE	_____	_____	_____	BORE7	_____	_____
_____	BORE8	_____	_____	_____	BORE9	_____	_____
_____	DEEP	_____	_____	_____	DRILL	_____	_____
_____	FACE	_____	_____	_____	MILL	_____	_____
_____	REAM	_____	_____	_____	REVERS	_____	_____
_____	SHIFT	_____	_____	_____	TAP	_____	_____
_____	THRU	_____	_____	_____			

Cycle Parameters:

Depth _____	Top/part _____	RAPTO _____	TPI _____
DWELL1 _____	DWELL2 _____	STEP1 _____	STEP2 _____
OFFSET1 _____	OFFSETX _____	OFFSETY _____	OFFSETZ _____
RTRCTO1 _____	RTRCTO2 _____	REPEAT _____	

Cycle Block # _____: _____

Cycle Block # _____: _____

Cycle Block # _____: _____

Cycle Statements:

Mode	Parameters
CYCLE/ _____,	_____
_____,	_____
_____,	_____
_____,	_____
CYCLE/ _____,	_____
_____,	_____
_____,	_____
_____,	_____
CYCLE/ _____,	_____
_____,	_____
_____,	_____
_____,	_____

Lathe Cycles

Cycle Modes:	Code	Block	Time	Code	Block	Time
	DEEP			w/STEP		
	DRILL			w/STEP		
	FACE			w/STEP		
	ROUGH			w/STEP		
	THREAD			w/STEP		
	THRU			w/STEP		
	TURN			w/STEP		

Cycle Parameters:

LeadZ		LeadX		Lead/I		Lead/D	
FinalZ		FinalX		FEDTO		OFFSET1	
OFFSET2		RAPTO		REPEAT		RTRCTO1	
RTRCTO2		STEP1		STEP2		TOOL1	
TOOL2		Cham/ON		Cham/OFF			

Cycle Block # _____ : _____

Cycle Block # _____ : _____

Cycle Block # _____ : _____

Cycle Statements:

Mode	Parameters
CYCLE/	_____, _____
	_____, _____
	_____, _____
	_____, _____
CYCLE/	_____, _____
	_____, _____
	_____, _____
	_____, _____
CYCLE/	_____, _____
	_____, _____
	_____, _____
	_____, _____

Print Descriptor Records

Record # _____

Registers: _____ / _____ _____ / _____ _____ / _____

Type: _____ / _____

_____ / _____ _____ / _____ _____ / _____

Record # _____

Registers: _____ / _____ _____ / _____ _____ / _____

Type: _____ / _____

_____ / _____ _____ / _____ _____ / _____

Record # _____

Registers: _____ / _____ _____ / _____ _____ / _____

Type: _____ / _____

_____ / _____ _____ / _____ _____ / _____

Record # _____

Registers: _____ / _____ _____ / _____ _____ / _____

Type: _____ / _____

_____ / _____ _____ / _____ _____ / _____

A horizontal number line with a grid. The grid consists of 11 major columns and 100 minor columns. The major columns are labeled 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, 1, 2, 3. A vertical line is drawn at the 8th major column.

Print Descriptor Records

Record # _____

Registers: _____ / _____ _____ / _____ _____ / _____

Type: _____ / _____

_____ / _____ _____ / _____ _____ / _____

A blank 10x10 grid with columns numbered 1 to 10 at the bottom.

Record # _____

Registers: _____ / _____ _____ / _____ _____ / _____

Type: _____ / _____

_____ / _____ _____ / _____ _____ / _____

A blank 10x10 grid with columns numbered 1 to 10 at the bottom.

Record # _____

Registers: _____ / _____ _____ / _____ _____ / _____

Type: _____ / _____

_____ / _____ _____ / _____ _____ / _____

A blank grid for graphing, consisting of 8 columns and 4 rows. The columns are labeled 1 through 8 at the bottom.

APPENDIX C Automatic Documentation File

MACHINE DESCRIPTION

Page 1

MACHIN/PWORKS,8100132

CIN950 Emulation FOR: CINCINNATI 5-AXIS PROFILER WITH MODEL 950 CONTROLLER

Machine Type: Mill

Units: Inch

Axes Description:

X - Primary X-axis.

Y - Primary Y-axis.

Z - Primary Z-axis.

B - Carrier rotary Head.

Rotation vector: 0,1,0

Pivot Offsets: 0.,0.,0.

Contouring axis on a Linear scale.

A - Rider rotary Head.

Rotation vector: 1,0,0

Pivot Offsets: 0.,0.,5.25

Contouring axis on a Linear scale.

Spindle Axis: 0.,0.,1.

Machine Limits:

X-axis: 0.,120.

Y-axis: 0.,50.

Z-axis: 5.,29.

B -axis: -25.,25.

A -axis: -25.,25.

Current Position:

X-axis: 0.

Y-axis: 0.

Z-axis: 5.25

B -axis: 0.

A -axis: 0.

Low speed spindle range: 20.,4000.

High speed spindle range: 500.,4000.

MACHIN/PWORKS,8100132

CIN950 Emulation FOR: CINCINNATI 5-AXIS PROFILER WITH MODEL 950 CONTROLLER

Maximum feed rates:

X-axis: 200.
Y-axis: 200.
Z-axis: 100.
B -axis: 240.
A -axis: 240.

Rapid rates:

X-axis: 200.
Y-axis: 200.
Z-axis: 100.
B -axis: 240.
A -axis: 240.

MACHIN/PWORKS,8100132

CIN950 Emulation FOR: CINCINNATI 5-AXIS PROFILER WITH MODEL 950 CONTROLLER

Register Format	Definition	Reg / Default
=	Rewind stop	
/	Optional block skip	
(MSG,)	Message block	
\$	End-of-Block	
Nxxxxxx	Sequence number	N
:	Alignment block register	O
Gxx	G-code	G0
Gxx	G-code	G7 / 90.
Gxx	G-code	G8
Gxx	G-code	G1
Gxx	G-code	G3
Gxx	G-code	G4
Gxx	G-code	G5 / 70.
Gxx	G-code	G6
X-xxx.oooo	Absolute primary X-axis Primary X preset position	X1
X-xxx.oooo	Incremental primary X-axis	X2
Y-xxx.oooo	Absolute primary Y-axis Primary Y preset position	Y1
Y-xxx.oooo	Incremental primary Y-axis	Y2
Z-xxx.oooo	Absolute primary Z-axis Primary Z preset position	Z1
Z-xxx.oooo	Incremental primary Z-axis	Z2
Z-xxx.oooo	Cycle final depth	W2

MACHIN/PWORKS, 8100132

CIN950 Emulation FOR: CINCINNATI 5-AXIS PROFILER WITH MODEL 950 CONTROLLER

Register Format	Definition	Reg / Default
A-xxxx.000	Absolute rotary axis Rotary axis preset position	A2
A-xxxx.000	Incremental rotary axis	A3
B-xxxx.000	Absolute rotary axis Rotary axis preset position	B2
B-xxxx.000	Incremental rotary axis	B3
R-xxx.0000	Circle radius Cycle rapto plane	R
I-xxx.0000	X-axis circle center	I1
I-xxx.0000	Cycle step parameter	I2
J-xxx.0000	Y-axis circle center	J1
K-xxx.0000	Z-axis circle center	K1
K-xxx.0000	Z-axis circle intermediate point Cycle step parameter	K2
U-xxx.0000	Cycle X-offset parameter	U1
V-xxx.0000	Cycle Y-offset parameter	V1
W-xxx.0000	Cycle retract parameter	W1
Fxxx.o	FPM feed rate	F1
Fxx.000	Inverse time feed rate	F2
Fxx.00	Dwell amount	F3
Dxx	Cutter compensation offset	D
Hxx	Fixture offset register	H
Sxxxx	Spindle RPM value	S
Txxxxxxxx	Tool number	T

MACHIN/PWORKS, 8100132

CIN950 Emulation FOR: CINCINNATI 5-AXIS PROFILER WITH MODEL 950 CONTROLLER

Register Format	Definition	Reg / Default
Mxx	M-code	M0
Mxx	M-code	M1
Mxx	M-code	M2
Mxx	M-code	M3
Mxx	M-code	M4
A-xxxx.ooo	Internal rotary axis	A1
B-xxxx.ooo	Internal rotary axis	B1

MACHIN/PWORKS, 8100132

CIN950 Emulation FOR: CINCINNATI 5-AXIS PROFILER WITH MODEL 950 CONTROLLER

G-code	Definition	Group
G00	Rapid positioning	G1
G01	Linear interpolation	G1
G02	Clockwise circular interp CLW helical interp	G1
G03	Counter-clockwise circ interp	G1
G04	Dwell block	G0
G17	XY-plane selection	G2
G18	ZX-plane selection	G2
G19	YZ-plane selection	G2
G40	Cancel cutter compensation	G3
G41	Cutter compensation left	G3
G42	Cutter compensation right	G3
G60	Linear slowdown span CLW circular slowdown CCLW circular slowdown	G4
G61	Slowdown off	G4
G70	Units inches	G5
G71	Units millimeters	G5
G80	Cancel cycle	G6
G81	DRILL cycle block	G6
G82	FACE cycle block	G6
G83	DEEP cycle block THRU cycle block	G6
G84	TAP cycle block	G6

MACHIN/PWORKS, 8100132

CIN950 Emulation FOR: CINCINNATI 5-AXIS PROFILER WITH MODEL 950 CONTROLLER

G-code	Definition	Group
G85	REAM cycle block	G6
G86	BORE cycle block	G6
G87	BORE7 cycle block	G6
G88	BORE8 cycle block	G6
G89	BORE9 cycle block	G6
G90	Absolute positioning mode	G7
G91	Incremental positioning mode	G7
G92	Absolute preset axis block	G0
G93	Inverse time feed rate mode	G8
G94	FPM feed rate mode	G8

MACHIN/PWORKS, 8100132

CIN950 Emulation FOR: CINCINNATI 5-AXIS PROFILER WITH MODEL 950 CONTROLLER

M-code	Definition	Group
M00	Program stop Manual tool change	M0
M01	Optional stop	M0
M02	End of program	M0
M03	Spindle clockwise	M1
M04	Spindle counter-clockwise	M1
M05	Spindle off Spindle orientate	M1
M06	Tool change	M0
M07	Coolant flood	M2
M08	Coolant mist	M2
M09	Coolant off	M2
M13	Spindle CLW & Coolant flood	M1
M14	Spindle CCLW & Coolant flood	M1
M17	Spindle CLW & Coolant mist	M1
M18	Spindle CCLW & Coolant mist	M1
M30	Rewind	M0
M40	Spindle low range	M3
M41	Spindle high range	M3
M48	Enable feed rate overrides	M4
M49	Disable feed rate overrides	M4

MACHIN/PWORKS,8100132

CIN950 Emulation FOR: CINCINNATI 5-AXIS PROFILER WITH MODEL 950 CONTROLLER

Block	Format / Description
1	G1.. X.. Y.. Z.. A.. B.. F.. First motion block.
2	O.. G94 G00 X.. Y.. Z.. A.. B.. F1.. S.. M1.. Non-motion Alignment block. Motion Alignment block.
4	M3..\$ S.. M1.. Spindle on block.
5	J1 CYCLE/THRU definition block.

MACHIN/PWORKS,8100132

CIN950 Emulation FOR: CINCINNATI 5-AXIS PROFILER WITH MODEL 950 CONTROLLER

```

ALIGN/AXIS [,OFF ]
              AUTO  [,0.] [,NEXT ] [,TRANS ] [,DOWN ]
              RAPID          LIMIT      NOW      UP

ARCSLP/LINCIR,ON
              OFF

AUXFUN/n [,NOW ]
          NEXT

BREAK

BREAK/OFF

BREAK/ [ON ] [,n] [,RAPTO,0.] [,FEDTO,0.]
      AUTO
      TIMES
      DELTA

CHECK/LENGTH,OFF
              ON

CHECK/OUT,m [,XAXIS[,min,max]] [,YAXIS[,min,max]] [,ZAXIS[,min,max]] $
          [,AXIS[,n] [,min,max]]

CHECK/ [,XAXIS[,min,max]] [,YAXIS[,min,max]] [,ZAXIS[,min,max]] $
      [,AXIS[,n] [,min,max]]

CLRSRF/AVOID, [i,j,k,] d

CLRSRF/ [NORMAL,] [0.,0.,1.,] 0.

CLRSRF/TOOL,d

CLRSRF/START
      STOP
      NOMORE
      a,b,c,d, NEGX
      z,      NEGY
              NEGZ
              POSX
              POSY
              POSZ

```

MACHIN/PWORKS,8100132

CIN950 Emulation FOR: CINCINNATI 5-AXIS PROFILER WITH MODEL 950 CONTROLLER

```

COOLNT/ON
    FLOOD
    MIST
    OFF

COUPLE/OFF
    n, ATANGL, a

COUPLE/AUTO, ON
    OFF

CUTCOM/ADJUST, n

CUTCOM/LEFT    [, XYPLAN]  [, d]
    RIGHT      ZXPLAN
                XYZPLN

CUTCOM/ON
    OFF

CYCLE/mode, FEDTO, depth    [, IPM    , feed[, rap]]    [, RAPTO, r] $
                        dia, ang    IPR
                                MMPM
                                MMPR
                                TPI

    [, STEP, peck1[, peck2]]  [, OFFSET, off1[, off2]]  [, RTRCTO, ret1] $
    [, START[, seq[, inc[, frq]]]]

CYCLE/AUTO, ON
    OFF

CYCLE/AVOID

CYCLE/ON
    OFF

CYCLE/CIRCUL, DEPTH, z, RADIUS, d, ON    [, TOOL, tdia], STEP, z [, IPM    , f] $
                                IN                                IPR

    [, RAPTO, r]  [, ATANGL, a]  [, DOWN    ]  [, CCLW    ]
                                UP          CLW

DELAY/n  [, REV]

```

MACHIN/PWORKS,8100132

CIN950 Emulation FOR: CINCINNATI 5-AXIS PROFILER WITH MODEL 950 CONTROLLER

DISPLY/ON
OFF
AUTO

END

FEDRAT/[IPM ,] [n]
IPR
MMPM
MMPR
INVERS

FEDRAT/LENGTH,n
SCALE

FEDRAT/LOCK,ON
OFF

FEDRAT/MAXIPM,100000000.

FINI

FROM/x,y,z,i,j,k

GOHOME

INSERT text

LEADER/n

LINTOL/0.001
ON
OFF

LINTOL/ADJUST,OFF,0.001 [,ATANGL,1.] [,DELTA,30.] [,LENGTH,3.]
ON

LINTOL/AXIS,0.0001

LINTOL/LINEAR, delta
TOLER , tol
OFF

LINTOL/RAPID,OFF
ON

CIN950 Emulation FOR: CINCINNATI 5-AXIS PROFILER WITH MODEL 950 CONTROLLER

```
PREFUN/n  [,NOW      ]
              NEXT
```


MACHIN/PWORKS,8100132

CIN950 Emulation FOR: CINCINNATI 5-AXIS PROFILER WITH MODEL 950 CONTROLLER

```

-----
RAPID [/] [MODIFY] [,] [XAXIS ] [,] [RTRCTO] [,] [ON ]
                YAXIS                OFF
                ZAXIS
                TOOL

RAPID/FEDTO,LENGTH [,n]
        SCALE

RETRCT [/] [PAST] [,] [XAXIS ] [,] [feed]
                TO                YAXIS
                ZAXIS
                TOOL

RETRCT/ON
        OFF

RETRCT/TOOL [,OFF ] [,LENGTH[,ON ] [,0.]] [,OFFSET,UP ] [,0.]] $
                ON                OFF                LEFT
                                                RIGHT

        [,FEEDZ,0.[,10.[,10.]]]

REWIND

ROTHED/ [AXIS,n,] ATANGL,ang [,NEUTRL] [,IPM ,f] [,SAME ] [,NOW ]
                ORIENT      MMPM      ROTREF      NEXT
                CLW
                CCLW
                INCR

ROTHED/ADJUST,HEAD ,ON
                OFF

ROTHED/NEXT [,AXIS,n] ,CLW
                CCLW

ROTHED/ORIGIN [,AXIS,n] ,x,y,z

ROTHED/SHORT,COMBIN
                LARGE
                PLUS
                AXIS,n

SEQNO/ON
        OFF
        n [,INCR,i[,m]]

```

MACHIN/PWORKS,8100132

CIN950 Emulation FOR: CINCINNATI 5-AXIS PROFILER WITH MODEL 950 CONTROLLER

SET/LINEAR

LINCIR [, tol] [,ALL]
COMBIN

SLOWDN/OFF

ON [,NEXT]
AUTO

SPINDL/ON

OFF
ORIENT

SPINDL/MAXRPM,n

SPINDL/ [RPM] [,n] [,CLW] [,LOW] [,RADIUS,r]
SFM CCLW HIGH DIAMTR

STOP

TMARK [/] [n] [,] [NOW]

TMARK/AUTO

TRANS/ [SCALE,] n [,LAST]
x,y,z

TRANS/i1,j1,k1,d1, i2,j2,k2,d2, i3,j3,k3,d3

TRANS/ [TOOL] [,] [VECTOR [,i,j,k]]

TRANS/ [SCALE,] [,XAXIS,val] [,YAXIS,val] [,ZAXIS,val] [,AXIS[,n],val]

APPENDIX D Punch Header File

D.1 Introduction

The Punch Header File (*.phf*) is a user generated text file that contains information which defines an optional header and/or trailer punch file section. Both text strings and variable data can be used to define these sections.

The *PHF* file can consist of a header and/or trailer definition section. These sections can contain simple text strings combined with formatted post-processor and print file variables. Simple loops may also be defined to output array data, such as tool information.

See the [Print Descriptor File Appendix](#) in this manual, and the [Variable Definition Chapter](#) in the ***PostMacro*** Reference Manual for descriptions of available variables.

D.2 The Header Section

Format: **#HEADER#**

The **#HEADER#** record defines the start of the header section of the file. Each line after this and prior to either the **#TRAILER#** section or the end of the file will be included in the punch file header, prior to the Control Tape information generated by ***PWorks***. Any lines prior to the header section will be treated as comment lines.

D.3 The Trailer Section

Format: **#TRAILER#**

The **#TRAILER#** record defines the start of the trailer section of the file. Each line after this and prior to the end of the file will be included in the punch file trailer, after the Control Tape information generated by ***PWorks***.

D.4 Data Lines

Format: **text1 [var1,fmt] text2 [var2]**

The actual information for the header and trailer sections are stored in data lines. These lines contain simple textual data and optionally post-processor and print file variables. Blank data lines will be output as blank lines in the punch file.

Variables must be enclosed in square brackets ([]) and can have an optional format descriptor. With numeric variables, '*fmt*' is a Register descriptor to use when formatting the number for

output. With textual variables (*PARTNO*, *TLNAME*, etc.), '*fmt*' is the word position within the text to output. Words are separated by spaces, not punctuation in the text string.

A maximum of 512 characters is allowed in a data line (except between the LOOPST and LOOPND region, 256 characters maximum) and a maximum of 512 characters per line can be output to the punch file.

Example:

```
Part data: [PARTNO,1] Rundate: [DATE] [TIME]
Total Machining Time: [%MCHTIM,C5]
```

D.5 Looping Region

Format: /LOOPST/ [%var]

```
·
·
/LOOPND/
```

The looping capabilities of the Punch Header File enable data lines containing subscripted variables, such as tool data variables, to be output. '[%var]' contains a numeric variable which defines the number of times to output each of the data lines contained within the loop. Do not include a register format with this variable. A maximum of 256 characters is allowed for each data line between LOOPST and LOOPND, however a maximum of 512 characters can only be output to the punch file.

The data lines within the loop are defined normally, with the exception that subscripted variables can have an asterisk (*) as the subscript, which will use the current loop counter as the subscript.

Example:

```
/LOOPST/ [%NTOOL]
Tool Number: [%TLNO(*),T] Length: [%TLEN(*),Z1]
Description: [TLNAME(*)]
/LOOPND/
```

D.6 Example Punch Header File

```
#HEADER#
START OF TAPE HEADER
CAM SYSTEM: [CAMNAME] V[CAMVERSN]
POST-PROCESSOR: PostWorks [REVDATE]

INPUT FILE: [CLNAME]
```

CREATION DATE: [CLDATE] [CLTIME]
PART NUMBER: [PARTNO,1]
PART DESCRIPTION: [PARTNO,2]

OUTPUT FILE: [PCHFILE]
CREATION DATE: [DATE] [TIME]

LAST BLOCK: [LASTBLK]
MACHINING TIME: [%MCHTIM,C5]
TAPE LENGTH: [%TAPLEN,C5]

/LOOPST/ [%NTOOL]
TOOL NUMBER: [%TLNO(*),T] LENGTH: [%TLEN(*),Z1]
DESCRIPTION: [TLNAME(*)]
/LOOPND/

END OF TAPE HEADER
START OF CONTROL TAPE

#TRAILER#
END OF CONTROL TAPE
%%
END OF TAPE

INDEX

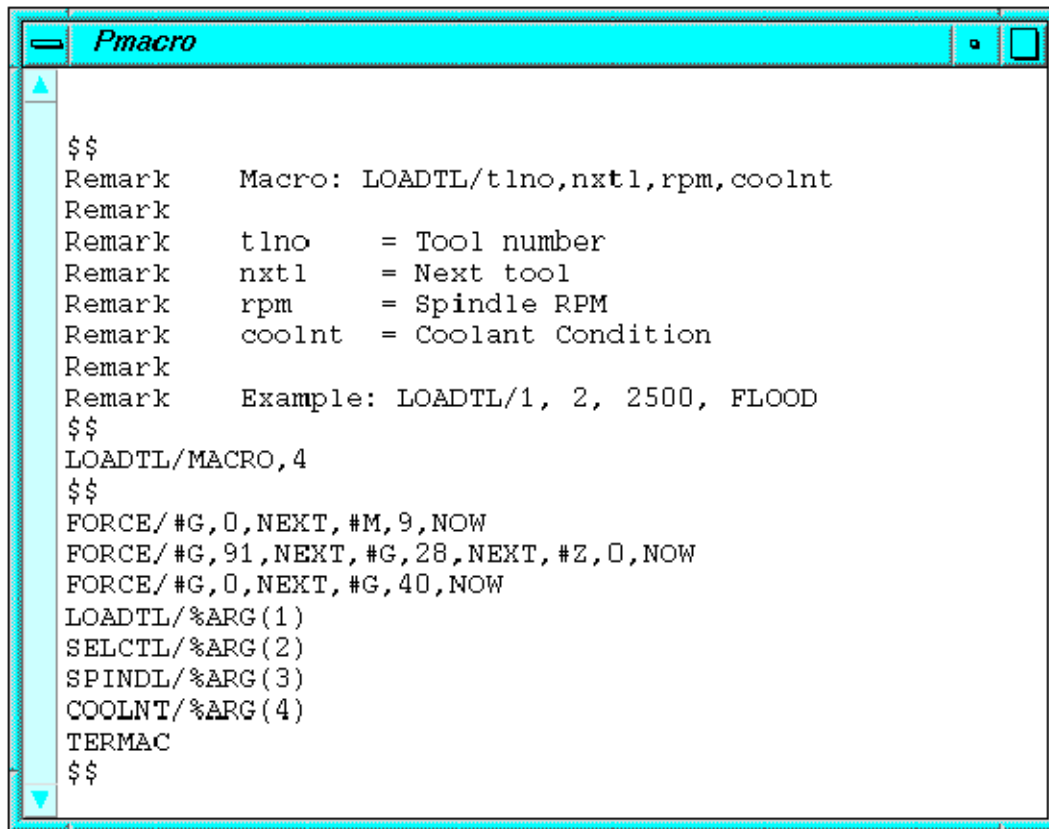
3-axis Circular Interpolation	6-3, 6-8
3-D Cutter Compensation	14-6
5-axis Cutter Compensation	14-2
Absolute	5-2, 5-3, 5-8
Acceleration Blocks	7-14
ALIGN	16-3, 16-4
Alignment Block	10-1, 13-3
Axis Designators	7-1
Blade Alignment	16-2, 16-4
BREAK	3-9
Carrier Axis	2-4
CHECK	2-13
Checksumming	3-6
Circular Interpolation	6-1, 6-2, 12-8, 14-3, 14-6
CLAMP	5-6
Clearance Plane	2-10, 2-11, 8-10, 8-13, 12-12
Control Tape	3-1, A-2
Coolant	11-6
Curve Blending	16-5
CUTCOM	14-1, 14-2, 14-4, 14-7
Cutter Compensation	14-1
CYCLE	9-1, 10-2, B-9
Cylindrical Interpolation	2-1, 2-9, 5-8, 5-9, 6-8, 6-10, 12-11
DATE	A-2
DELAY	15-3
Delta Distance	6-4
Direction Code	5-5
DISPLY/ON	3-5
End-of-block	3-5
End-of-Sequence	15-3
ENDPT	14-6
Extended Precision	12-9
FEDRAT	12-1
Feed Rate	12-1, 15-5, B-8
FINI	3-9, 4-2
Fixture Offsets	14-7
FROM	5-1
G-code	3-3, 3-5
GOHOME	2-10, 15-5
Head	2-4, B-2

Header	D-1
Helical Interpolation	6-6
Home Position	2-10
Inch	1-1
Incremental	5-2, 5-3, 5-8
INSERT	4-2, 15-1
Interference Zones	2-2, 2-12, B-5
LASTBLK	A-2
Lathe	2-1, 5-1, B-1
Lathe Cycles	9-1
Leader	3-6, 3-8
Linear Axes	2-2, 2-3, A-3, B-1
Linearization	7-3
LINTOL	7-4, 7-5
LMDP	6-10
LOADTL	4-2, 13-1, 13-3, 13-8
Look-Ahead Positioning	7-9
MACHIN	15-6
Machine Dwell	15-3
Machining Time	8-11, 8-19, 9-1, 9-6, 9-9, B-9
Macro Call	8-1
Maximum Axis Departure	7-13
MCHTOL	6-5
M-code	3-3, 3-5
Mill	2-1, B-1
Mill/Turn	2-1, 2-9, 5-8, 5-9, 6-2, 11-2, 12-10, B-1
Millimeter	1-2
Milling Cycles	8-1
MODE	5-2
Motion Block	4-1
Multi-head	2-5
Multi-table	2-5
Mutually Exclusive Axes	7-1, B-3
Non-motion Block	4-1
Operator Message	3-5, 15-1
OPSKIP	3-6, 15-2
OPSTOP	4-2, 15-2
ORIGIN	A-3
PARTNO	3-7, A-3
PCHBRIEF	A-2
PCHFILE	A-3
PCHNAME	A-3
Polar Coordinate	14-1
Polar Interpolation	2-1, 2-8, 5-8, 5-9, 6-8, 12-10
Post Variable	A-1, A-4
POSTN	15-6

PPRINT	3-5, 4-2, A-3
PPTOL	5-7
Print Descriptor File	4-1, A-1, A-4, B-10
Program Number	15-6
Pulse Weight	12-2, 12-3, 12-8
Punch File	3-7, A-2
Punch Header File	3-8, D-1
RAPID	8-2, 12-10, 12-11
Rapid Motion	12-12
RAPTO Plane	8-3, 8-4, 8-10, 8-13, 8-14
RECAP	A-3
Register Descriptor	A-4
Register Format	3-1, B-6
RETRACT	2-10, 2-11, 8-3, 8-13, 16-2
REVDAT	A-3
Revision Date	A-3
Rider Axis	2-4
ROTABL	5-3, 5-4, 7-9
Rotary Axes	2-2, 2-4, 5-2, 7-3, 12-2, A-3
Rotary Axes Look-Ahead	7-9
Rotary Axis	12-1, 16-1, B-2
RTRCTO	8-13
SELCTL	13-1, 13-3
SEQNO	10-1
Sequence Number	10-1
Shortest Route	7-8
SLOWDN	7-12
Slowdown Spans	7-11
SMOOTH	16-5, 16-6
SPINDL	11-1, 11-5
Spindle	B-1, B-8
Spline Interpolation	6-11
STOP	4-2, 15-2
Stringer	17-1
Surface Feed per Minute	11-1
Tab Sequential	3-5, 3-6
Table	2-4, B-2
Tape Break	3-9
TAPEBLK	A-2
THREAD	9-1
Thread Cutting	9-1
TIME	A-2
TLNAME	A-3
TMARK	3-10, 10-1
Tolerance	5-7, 6-5, 6-7, 7-4, 7-13, 11-2
Tool Axis	2-2

Tool Axis Adjustment	7-5
Tool Change	13-1
Tool Length Compensation	13-4
TOOLNO	13-9
Trailer	D-1
TRANS	5-7, A-3, B-4
Transformation Blocks	7-15
Travel Limits	2-11
TURRET	13-2
Ultrasonic Cutter	2-1, 7-8, 16-1, 16-6, B-1
Unidirectional Move	8-9
User Defined Blocks	B-9
User Defined Tape Blocks	3-4
Vacuum Pods	16-6
Word Address	3-5
XYPLAN	5-8
XY-plane	6-10
YZPLAN	5-9
YZ-plane	6-10
ZXPLAN	5-9
ZX-plane	6-10

PostMacro

A screenshot of a software window titled "Pmacro". The window has a light blue title bar with a minus sign, a maximize button, and a close button. The main area is white and contains text representing a macro. The text starts with "\$\$" and "Remark Macro: LOADTL/tlno,nxtl,rpm,coolnt", followed by definitions for tlno, nxtl, rpm, and coolnt. It then shows an example call "LOADTL/1, 2, 2500, FLOOD", another "\$\$" separator, and the command "LOADTL/MACRO,4". This is followed by another "\$\$" separator and a series of G-code commands: "FORCE/#G,0,NEXT,#M,9,NOW", "FORCE/#G,91,NEXT,#G,28,NEXT,#Z,0,NOW", "FORCE/#G,0,NEXT,#G,40,NOW", "LOADTL/%ARG(1)", "SELCTL/%ARG(2)", "SPINDL/%ARG(3)", "COOLNT/%ARG(4)", "TERMAC", and ends with "\$\$".

```
$$
Remark      Macro: LOADTL/tlno,nxtl,rpm,coolnt
Remark
Remark      tlno      = Tool number
Remark      nxtl      = Next tool
Remark      rpm       = Spindle RPM
Remark      coolnt    = Coolant Condition
Remark
Remark      Example: LOADTL/1, 2, 2500, FLOOD
$$
LOADTL/MACRO,4
$$
FORCE/#G,0,NEXT,#M,9,NOW
FORCE/#G,91,NEXT,#G,28,NEXT,#Z,0,NOW
FORCE/#G,0,NEXT,#G,40,NOW
LOADTL/%ARG(1)
SELCTL/%ARG(2)
SPINDL/%ARG(3)
COOLNT/%ARG(4)
TERMAC
$$
```

Post-processor Macros Reference Manual

Warning and Disclaimer

Every effort has been made to make this document complete and as accurate as possible.
However, no warranty or fitness is implied.

The information is provided on an "as is" basis. Numerical Control Computer Sciences shall have neither liability nor responsibility to any person or entity with respect to any loss or damages in connection with or rising from the information contained in this document.

Copyright 1983-2014 by Numerical Control Computer Sciences
Irvine, California. Printed in the United States of America.
All rights reserved. The contents of this publication may not
be reproduced in any form or by any means, electronic
or mechanical, including photocopying, recording, or
information storage and retrieval systems, for any
purpose other than the licensee's personal use,
without prior written consent from
Numerical Control Computer Sciences.

TABLE OF CONTENTS

CHAPTER 1 *PostMacro* Basics1-1

1.1	Introduction.....	1-1
1.2	Basic Statement Structure.....	1-1
1.2.1	Continuation Lines - \$	1-1
1.2.2	Multiple Statement Lines - ;.....	1-2
1.2.3	Comments - \$\$!	1-2
1.2.4	Major Words.....	1-2
1.2.5	Minor Words.....	1-3
1.2.6	Numbers	1-3
1.2.7	Text Strings.....	1-3
1.2.8	Variables.....	1-3
1.2.9	Post Variables.....	1-5
1.3	<i>PMacro</i> Versus <i>PWorks</i>	1-5

CHAPTER 2 Macro Definitions2-1

2.1	Introduction.....	2-1
2.2	The MACRO statement	2-1
2.3	The TERMAC Statement	2-2
2.4	Special Considerations	2-2
2.5	Special Macros	2-2
2.5.1	The G\$MAIN Macro	2-3
2.5.2	The FINI Macro	2-3
2.5.3	The GOTO Macro.....	2-3
2.5.4	The ERROR Macro	2-4
2.5.5	The OPTION Macro.....	2-4
2.5.6	The PPRINT Macro	2-5
2.5.7	The PSTERR Macro.....	2-5
2.5.8	The TAPBRK Macro	2-6
2.5.9	The CIRCLE Macro	2-7
2.5.10	The XFORM Macro	2-8

CHAPTER 3 Variable Definitions3-1

3.1	Introduction.....	3-1
3.2	Real Variable Declarations.....	3-1
3.3	Text Variable Declarations - CHAR.....	3-2
3.4	Post Variables.....	3-3
3.4.1	Macro Argument Variables Declarations - %ARG, %NARG	3-3
3.4.2	Reference Sequence Numbers: - %CLREC, %ISN.....	3-4

3.4.3	Number Of PARTNO Cards - %NPARTNO	3-4
3.4.4	Clfile Settings - %CUTTER, %MULTAX	3-4
3.4.5	PostWorks Common - %KPOSMP, %POSMAP, %USER	3-5
3.4.6	Control Tape Vars - %REG, %TAPLEN, %UNITS, %BRKPRM ...	3-5
3.4.7	Tape Registers Beginning/Ending Characters Vars - %REGST(92), %REGEN(92)	3-6
3.4.8	Print File Variables - %LINENO, %PAGENO	3-6
3.4.9	Error Messages - %ERROR, %ERRPC, %NUM---	3-6
3.4.10	Miscellaneous Variables - %PI, %SIMUL	3-7
3.4.11	Tool Data - %NTOOL, %TOOL, %SELCTL, %TLEN, %TLNO....	3-7
3.4.12	Feeds and Speeds - %FEED, %RPM, %SFM	3-8
3.4.13	Delta Move Distances - %MOVDIS	3-8
3.4.14	Machining Times - %---TIM, %TIMSCL	3-9
3.4.15	Clfile Points - %CLPT, %CLSAV, %TLVEC	3-9
3.4.16	Input Coordinates - %XYZ, %XYZ---	3-10
3.4.17	Linear Axes - %LINEAR, %LIN---	3-11
3.4.18	Rotary Axes - %ROTARY, %ROT---	3-11
3.4.19	Output Axes - %AXIS, %AXS---	3-12
3.4.20	PWorks Machine Number - %MACHIN	3-13
3.4.21	PWorks Machine Name - %PSTNAM	3-13
3.4.22	Home Position - %HOME	3-13

CHAPTER 4 Expressions4-1

4.1	Introduction.	4-1
4.2	Numeric Expressions	4-1
4.2.1	Number Operators	4-1
4.2.2	Use Of Parenthesis	4-2
4.2.3	Numeric Functions	4-2
4.2.4	Numeric Assignment Statements	4-4
4.3	Text Expressions.	4-5
4.3.1	Text Operators	4-5
4.3.2	Text Functions	4-6
4.3.3	Text Assignment Statements	4-9
4.4	Format Descriptors	4-10
4.4.1	Format Expressions	4-10
4.4.2	Register Labels	4-11

CHAPTER 5 Post-processor Commands.....5-1

5.1	Standard Post-Processor Commands	5-1
5.2	Text Post-Processor Commands	5-2
5.3	Special Post-Processor Commands	5-2
5.3.1	Appending Text To A Tape Block - APPEND	5-3
5.3.2	Multiple Control Tape Files - BREAK	5-3

5.3.3	Post Variable Assignments - DEFINE	5-4
5.3.4	Disabling Macros - DISABLE	5-4
5.3.5	Enabling Macros - ENABLE	5-4
5.3.6	Displaying Error Messages - ERROR	5-5
5.3.7	Forcing Output Of The Current Block - FORCE/NOW.....	5-5
5.3.8	Outputting User Defined Blocks - FORCE/BLOCK	5-5
5.3.9	Controlling The Output Of Registers - FORCE	5-6
5.3.10	Controlling The Output Order of Registers - REGORD	5-7
5.3.11	Motion Generating Statements - FROM, GOTO, GOTOC	5-7
5.3.12	Motion Register Setting Statement - POSCAL	5-8
5.3.13	Circular Motion Interpolation Statement - CIRCLE	5-9
5.3.14	Including An External File - INCLUD.....	5-9
5.3.15	Controlling The Listing File - LISTING	5-9
5.3.16	Referencing PMacro - MACHIN	5-10
5.3.17	Multax Control In The Output Clfile - MULTAX	5-10
5.3.18	Output To Listing File - PRINT	5-11
5.3.19	Controlling The PWorks Print File - PRINTF	5-11
5.3.20	Print File Output - PRINTF/RECORD.....	5-12
5.3.21	Displaying PWorks Error Messages - PSTERR.....	5-12
5.3.22	Automatic Documentation - REMARK	5-12
5.3.23	External Text File Commands	5-13
5.3.24	Clfile Read/Write Commands.....	5-15
5.3.25	Syntax Checking - SYNTAX.....	5-17

CHAPTER 6 Program Control Statements.....6-1

6.1	Introduction.....	6-1
6.2	Labels	6-1
6.3	The JUMPTO Statement.....	6-1
6.4	If Tolerancing (IFTOL)	6-2
6.5	Logical IF Statements.....	6-2
6.6	IF-THEN-ELSE Structures.....	6-3
6.7	The CONTINUE Statement	6-4
6.8	Repetitive DO Loops	6-4
6.9	The BREAKF Statement	6-5
6.10	The ENDDO Statement	6-5
6.11	The ENDMAC Statement.....	6-5

APPENDIX A Command/Variable Summary A-1

A.1	Post Variable Summary	A-1
A.2	Register Label Summary	A-1
A.3	Functions Summary	A-2
A.4	Special Macro Summary.....	A-5
A.5	Post-processor Command Summary.....	A-6

APPENDIX B Clfile Format B-1

B.1	Introduction.	B-1
B.2	ISN Record	B-2
B.3	Post-processor Command	B-2
B.4	Stock / Fixture Records.	B-3
B.5	Circular Record.	B-4
B.6	Motion Record	B-4
B.7	Expanded Motion Record	B-5
B.8	Motion Startup Record	B-6
B.9	Motion Check Surface Record	B-6
B.10	Cutter Record	B-7
B.11	Tracut Record	B-7
B.12	Cutter Display Record	B-8
B.13	Cutter Geometry Record.	B-10
B.13.1	NCL 9.6 And Below.	B-10
B.13.2	NCL 9.7 And Above.	B-10
B.14	SEQUNC Record	B-10
B.15	Units Record	B-11
B.16	Clfile Header Record	B-11
B.17	Multax Record	B-12

APPENDIX C Examples C-1

C.1	Introduction.	C-1
C.2	Converting Post-Processor Commands	C-1
C.3	Tool Change Macro	C-2
C.4	Creating a New Cycle Syntax.	C-4
C.5	Emulating PREPST 's HEAD Statement.	C-5
C.6	Emulating GEOPST's CHECK Statement.	C-8

INDEX. 1

CHAPTER 1 *PostMacro* Basics

1.1 Introduction

When working with the ***PostMacro*** language it is important to know the basics of command syntax and how it is represented in this manual. Although each type of command ([post-processor statements](#), [equations](#), [IF statements](#), etc.) have their own unique syntax, they are represented here using the same type of descriptors. This chapter covers the basic syntax of ***PostMacro*** commands.

1.2 Basic Statement Structure

This section describes the basic structure of statements contained in the [Macro definition](#) file. Each input line can contain up to 80 characters, longer lines will require a continuation character (\$) at the end of the line. The total length of the line can be up to 512 characters, including all continuation lines.

Most statements do not have fixed fields and can begin at any position in the line. The text post-processor commands, [APPEND](#), [INSERT](#), [LETTER](#), [PARTNO](#), and [PPRINT](#) are an exception, and must start in column 1 when a slash character (/) does not directly follow the major word.

Any command or qualifier may be entered in lower case, upper case, or any combination of each. Upper and lower case letters are treated the same by ***PostMacro***.

Optional command qualifiers will be enclosed in square brackets ([]). Optional qualifiers either do not have to be specified or have default values. The [...] symbol signifies that more qualifiers of the same type may appear within the command.

1.2.1 Continuation Lines - \$

Most statements are input on a single line, but it is possible for a single statement to be continued on one or more successive lines. The single dollar sign character (\$) indicates that the statement is continued on the following line.

A statement can have any number of continuation lines as long as the statement size is not greater than 512 characters. A dollar sign must be contained on every line that is continues on the following line. Any characters to the right of the dollar sign will be ignored, so comments may be placed there if desired.

The text post-processor commands [APPEND](#), [INSERT](#), [LETTER](#), [PARTNO](#), and [PPRINT](#) when not used with a following slash character (/) will use the dollar sign as a part of the text string and will not recognize it as a continuation character.


```
CHECK/XAXIS,-10,10,      $ Define Machine Limits
      YAXIS,0,32.5,      $
      ZAXIS,4.25,18.55
```

1.2.2 Multiple Statement Lines - ;

You may at times wish to place more than one statement on a single line. This can be done by separating each statement with the semicolon character (;). All statements contained on a single line will be processed exactly the same as if they were on separate lines, except the text post-processor commands, which must start in column 1.

Example:

```
TL(1) = 89302 ; TLLEN(1) = 10.5 ; TLREG(1) = 20
```

1.2.3 Comments - \$\$!

The double dollar sign (\$\$) and exclamation point (!) characters signify the start of a comment. You may use whichever of the two you wish, they are both treated the same. The comment character(s) mark the end of the executable statement; all characters to the left of the comment character(s) will be processed, while all characters to the right will be ignored. Comments may be included on a statement by themselves or at the end of an executable statement.

The text post-processor commands *APPEND*, *INSERT*, *LETTER*, *PARTNO*, and *PPRINT* when not used with a following slash character (/) will use the comment character(s) as a part of the text string and will not recognize them as comment characters.

Example:

```
$$ Move to HOME position
GOTO/HOME(1),HOME92),HOME(3)  ! CLERAP defines HOME
```

1.2.4 Major Words

A major word as the first parameter in a statement is considered a post-processor command and may be on a line by itself or with minor word/values. A slash character (/) separates the major word from its qualifiers. *LOADTL*, *STOP*, *FEDRAT* are major words.

Example:

```
STOP
LOADTL/1,LENGTH,10.5
```

1.2.5 Minor Words

A minor word is a recognized vocabulary word that is used as a parameter on a post-processor command. Minor words are also passed as arguments when a post-processor command calls a Macro and can then be referenced as a character string, containing the text representation of the minor word ("CLW", "OFF", etc.). Commas separate minor words on a post-processor command.

Example:

CUTCOM/LEFT,XYPLAN	\$\$	LEFT & XYPLAN are minor words
IF (%ARG(1) = "CLW") THEN	\$\$	Checks the 1st parameter of the
	\$\$	calling post-processor command for
	\$\$	the CLW minor word

1.2.6 Numbers

A string of digits that represent an integer or real number. Numbers can be signed (+ or -) and can contain a decimal point. Also, numeric variables and equations can be used most everywhere a number is allowed.

1.2.7 Text Strings

A text string is a group of characters that is enclosed in quotes. Text strings are normally used in text post-processor commands, listing file output and to represent minor words. Single or double quotes may be used to enclose the string, but the same type of quote that starts the string must also terminate it. The enclosing quotes will not be considered part of the string, and if a string contains either a single or double quote, then the other type of quote must be used to enclose the string. Text variables and equations can be used in most instances where a text string is allowed.

Example:

LMSG = "PUSH THE 'START' BUTTON"

1.2.8 Variables

PostMacro supports both numeric and text variables. A variable name can be considered a label to a storage location that contains either numeric or text data. Variable names can contain from 1 to 24 characters, which can be any combination of alphanumeric characters (A-Z, 0-9), but the first character must be a letter.

Variable names may not be the same as recognized **PostMacro** vocabulary words (major words, Minor words, statement identifiers, etc.). They must also be unique in the *Macro* in which they are defined, including global variables.

Variables that are defined in the *G\$MAIN Macro* are considered global and are accessible by all of the *Macros* in the program. Variables defined in all other *Macros* are local to the *Macro* in which they are defined. Therefore, variables in different *Macros* may have the same name, but they will reference different storage locations.

Any real or text variables used in the program must first be declared using with either the *REAL* or *CHAR* specification statement. Variables can be declared as an array up to three dimensions by associating them with a numeric subscript in the specification statement.

Example:

```
REAL A(1), B(2,3,5)
CHAR LTEXT(80), TEMTXT(3,20)
```

Each element of a real array holds one value. Text arrays are considered text strings, with one character stored in each array element. Therefore, you should make the array large enough to hold the largest string you would want stored in it.

Real variables that have been defined as arrays must have a subscript associated with them whenever they are referenced in the program. Subscripts are enclosed in parenthesis and directly follow the variable name. They may contain a number, variable or numeric expression.

Example:

```
A(1)
TLO(I)
PT1(INC*4-3)
```

Text variables are processed differently than real variables; while real variables are processed as a single value at a time, text variables will be processed as a string of characters. When a dimensioned text variable is specified without a subscript, then the entire text string is referenced. Text variable subscripts may contain a single numeric value of expression, or may contain a range specifier in the form (start:end). When a range subscript is used, then a substring of the text string is being referenced, starting with 'start' and ending with 'end'. Range subscripts may contain a number, variable or numeric expression.

Example:

```
TCOM
CDATE(13)
CTIME(1:NC-4)
```

In the following examples of valid and invalid references to variables, assume the variables have been declared using the specification statements below.

```
REAL A,TL(20),C(13)
CHAR TCOM(80),CDATE(20),CTIME(10)
```

Valid	Invalid	
-----	-----	
a	a(2)	\$\$ Undimensioned variables may \$\$ not have subscript.
TL(A)	TL	\$\$ Dimensioned real variables \$\$ must have s subscript.
C(3)	C(-1)	\$\$ The subscript value is out \$\$ of bounds
TCOM	TCOM(CDATE)	\$\$ Subscripts must be numeric.
CDATE(1:10)	CDATE(10:1)	\$\$ The starting value of a \$\$ range subscript must be less \$\$ than the ending value.
CTIME(4)	1CTIME	\$\$ Variable names must start \$\$ with a letter.

1.2.9 Post Variables

There are internal variables within **PostMacro** that may be accessed in the same manner as user defined variables. *Post variables* have the format '%name'.

Dimensioned *Post Variables* must have a subscript associated with them whenever they are referenced. Unlike real variable arrays, *Post variables* may contain a range subscript when referenced in a post-processor command. See the '[Variable Definitions](#)' chapter for a complete description of available *Post variables*.

Example:

```
IF (%ERROR == 1) THEN
LOADTL/%ARG(1:3),OFSETL
GOTO/%CLPT(1:%ARG(1)*NPT)
```

You can also change the values of *Post variables*, but unlike real and text variables, you use the *DEFINE* statement to store a value in them, not a standard assignment statement. See the '[Post-processor Commands](#)' chapter for a description of the *DEFINE* statement.

1.3 PMacro Versus PWorks

Post-processor *Macros* are supported in both the **PMacro** pre-processor and the **PWorks** post-processor. Because **PMacro** is a pre-processor and does not have any logic concerning the configuration of the machine tool, there are some features described in this manual that will not apply to **PMacro**, but will apply to **PWorks**. For example, certain *Post Variables* (*%MCHTIM*,

%TAPLEN, etc.). *Macros* (*PSTERR*, etc) and post-processor commands (*FORCE*, *PRINTF*, etc.). The features which only apply to **PWorks** will be noted as such in their description.

CHAPTER 2 Macro Definitions

2.1 Introduction

Macros can be defined for all post-processor commands (*LOADTL*, *STOP*, etc.) and to handle the initial startup of a program, the end of a program, motion cl points and error messages. All of the parameters in the calling post-processor command are passed to the *Macro* in the form of *Macro* arguments (*%ARG*).

A *Macro* definition consists of a *MACRO* statement, statements to be executed each time the *Macro* is called, and a *TERMAC* statement. *Macros* may contain the following:

1. Standard post-processor commands, such as *LOADTL*, *CUTCOM*, *PPRINT*, etc.
2. Text and real variables, which can be local to the current *Macro* or global throughout the program.
3. Numeric and text equations, including a full set of math and character string functions.
4. *GOTO/pt(s)* commands that will create new motion records in the output clfile.
5. *JUMPTO* branching.
6. Logical *IF* statements, including full *IF-THEN-ELSE* logic.
7. Repetitive *DO* loops.
8. References to internal *Post variables*, which can be retrieved and modified.
9. *SYNTAX* commands that perform the syntax of the post command that called the *Macro*.

2.2 The MACRO statement

Command Syntax: **major/MACRO,narg**

The *MACRO* statement begins the definition of *Macro*, where 'major' is a **major word** that defines which post-processor command will call this *Macro*.

'narg' is the maximum number (maximum 50) of words/values that can be passed to the *Macro* and must be specified, as **PostMacro** will allocate local storage for this number of arguments. Arguments are the parameters entered in the post-processor command that calls the *Macro*. An error will occur if too many arguments are passed to the *Macro*, and the argument list will be truncated to 'narg'.

Example:

```
LOADTL/MACRO,3
STOP/MACRO,0
GOTO/MACRO,2
PPRINT/MACRO,9
```

2.3 The TERMAC Statement

Command Syntax: **TERMAC**

The *TERMAC* statement defines the end of a *Macro* definition. When the *TERMAC* statement is reached during interpretation by the *Macro* processor, control will be passed to the statement following the post-processor command which called the *Macro*.

2.4 Special Considerations

Following is a list of circumstances that should be considered when defining and calling *Macros*.

1. Any post-processor command contained in a *Macro* will call its corresponding macro, if it has been defined, unless one of the following conditions is true.
 - a. The macro to be called is on the *Macro* call stack (currently active *Macros*).
 - b. There are 10 *Macros* currently on the *Macro* call stack.
 - c. The *Macro* has been disabled using the '*DISABLE*' command.
2. When a post-processor command calls a *Macro*, this command will not be output to the clfile. If you wish this command to be output to the clfile, then issue the following command within the *Macro* (*LOADTL* is used as an example).

LOADTL/%ARG(1:%NARG)

3. The definition of text command *Macros* (*INSERT*, *LETTER*, *PARTNO* & *PPRINT*) should contain a value of 9 to 16 as the number of arguments, since *PostMacro* allocates 8 characters for each argument and there can be 66 to 126 characters in these commands. Even so, *%ARG(1)* will contain all of the 126 characters of the text command.
4. The major word "*DISPLY*" cannot be used to define a macro.
5. The major word '*TPRINT*' cannot be used to define a macro or used inside a macro definition. Use the word '*PPRINT*' instead.

2.5 Special Macros

The following sections describe *Macros* which are not called by post-processor commands, but at specific points while processing the clfile. The *Macro* statement should be entered exactly as shown under command syntax, including the maximum number of arguments.

2.5.1 The G\$MAIN Macro

Command Syntax: **none**

The *G\$MAIN* Macro is automatically defined by **PostMacro** as '*G\$MAIN/MACRO,0*' and is called at the start of the clfile, before processing any of the clfile records. Do not put the *G\$MAIN Macro* definition in your *Macro* definition, as **PostMacro** will do it for you.

All statements prior to the first *Macro* definition will be placed in the *G\$MAIN Macro*. Also, any real and text variables defined in this *Macro* will be global to all of the *Macros* in the program, so define variables that must be shared between *Macros* here.

You may terminate the *G\$MAIN Macro* with a *TERMAC* statement, but it is not necessary. **PostMacro** will automatically terminate the *G\$MAIN Macro* when the first *Macro* definition statement is encountered.

2.5.2 The FINI Macro

Command Syntax: **FINI/MACRO, 0**

The *FINI Macro* is automatically called when the *FINI* record is encountered in the clfile and it has been defined. You should place any commands that you want executed just prior to the end of the program in this *Macro*. The *FINI* record will still be output to the clfile, after the *Macro* has been executed, and should not be included in the *FINI Macro*.

2.5.3 The GOTO Macro

Command Syntax: **GOTO/MACRO, 2**

The *GOTO Macro* is called whenever a motion record is encountered in the input clfile. *%ARG(1)* will be set to the number of points in the motion record and the *%CLPT Post variable* array will contain the actual point data.

%ARG(2) will contain the type of the statement that generated the motion record. 3 = FROM statement, 5 = Standard motion statement, 6 = Continuation record.

The motion record that calls the *GOTO Macro* will NOT be output to the clfile; you must use the following commands to output it exactly as is to the clfile.

```
NPT = 3
IF (%MULTAX == 1) NPT = 6
GOTO/%CLPT(1:%ARG(1)*NPT)
```


In the above example, notice how the variable `NPT` is changed depending on the value of the Post variable `%MULTAX` and is used in the `GOTO` statement. This is because the `%CLPT` array is a single dimension array with the X, Y, Z, I, J, K values of each cl point packed into it. When `MULTAX` is turned off, `%MULTAX = 0` and only the X, Y, Z values are in the `%CLPT` array, therefore `NPT = 3`. When `MULTAX` is turned on, `%MULTAX = 1` and the X, Y, Z, I, J, K values are stored in the array. `%ARG(1)*NPT` is used, because the actual number of points, not array elements, are passed as the first argument.

2.5.4 The ERROR Macro

Command Syntax: **ERROR/MACRO, 10**

It is sometimes desirable to trap a message that would normally be generated when an error occurs. The *ERROR Macro* is automatically called by **PostMacro** each time an error occurs in the *Macro* processor while processing the clfile. You can use this *Macro* to selectively disable some error messages and to perform certain operations when a specific error message is encountered.

The label of the error message is passed in `%ARG(1)` and is the only argument. The [error message label](#) is a text string containing 1 to 8 characters and can be referenced in the same way as a standard text argument variable. See the [PostWorks Users Guide for a description of error messages and their labels](#). See the [ERROR](#) command in the 'Post-processor Commands' chapter on how to output your own error messages.

In the following example, the 'Attempted divide by zero' error message has been disabled.

Example:

```
ERROR/MACRO,10
IF (%ARG(1) <> 'DIVZERO') ERROR/%ARG(1)I
TERMAC
```

2.5.5 The OPTION Macro

Command Syntax: **OPTION/MACRO, n**

The *OPTION Macro* is called whenever the first parameter of any post-processor command is the word '*OPTION*' and a *Macro* for this particular command has not been defined. The purpose of the *OPTION Macro* is to trap option commands which are in the program to set up options for a custom post-processor and are not supported by **PostWorks**.

'n' specifies the maximum number of arguments that can be passed to this *Macro*. Custom post-processors usually allow for a maximum of 20 parameters on a single command, though **PostWorks** allows for 50 parameters. The first argument will contain the major word of the

post-processor command which called the *OPTION Macro* instead of the minor word *OPTION*, such as *FEDRAT*, *LOADTL*, etc.

2.5.6 The PPRINT Macro

Command Syntax: **PPRINT/MACRO,10**

Normally a PPRINT macro is not required. However, a PPRINT macro is mandatory if the feature of “Look ahead for LOADTL when PPRINT encountered” is activated in the Menu item 13.2.9 of the corresponding MDF file.

%Arg(1) contains the characters string in the PPRINT statement.

%Arg(2) has a value of 1 if the next command is a LOADTL command. Otherwise, %Arg(2) has a value of 0.

Example:

```
PPRINT/MACRO,10
  IF (%ARG(2) == 0) THEN
    .... Next statement is not a LOADTL command ....
  ELSE
    .... Next statement is a LOADTL command ....
  ENDIF
TERMAC
```

2.5.7 The PSTERR Macro

Command Syntax: **PSTERR/MACRO,10**

The *PSTERR Macro* is called whenever an error is generated by **PWorks**, unlike the *ERROR Macro* which traps errors generated by the *Macro* processor. An example of a **PWorks** error would be 'Invalid minor word.' or 'X-axis exceeded positive limit.'. An example of a *Macro* processor error would be 'Attempted divide by 0.'.

The **PMacro** utility will ignore the *PSTERR Macro*, if it has been defined. **PWorks** will automatically call the *PSTERR Macro* whenever a post-processor error occurs, it will call the *ERROR Macro* whenever a **PostMacro** error occurs. You can use either or both of these *Macros* to selectively disable some error messages and to perform certain operations when a specific error message is encountered.

The [label of the error message](#) is passed in %ARG(1) and is the only argument. The error message label is a text string containing 1 to 8 characters and can be referenced in the same way as a standard text argument variable. See the [PostWorks Users Guide for a description of error](#)

[messages and their labels](#). See the [PSTERR](#) command in the 'Post-processor Commands' chapter on how to output trapped error messages.

Certain error messages can have a runtime substitution to the text of the error message. These messages are designated in the **PostWorks** Users Guide by having a variable enclosed in single quotes, such as '[EXCLAXS](#)', '[LMTUPR](#)', and '[TOOLMAX](#)'. In these cases the actual text of the error message will be passed to the *PSTERR Macro* instead of the label and a test for a portion of this error message text should be programmed, rather than a test for the label.

In the following example, the '*Circle radius over maximum.*' and '*Mutually exclusive axes 'a1' ...*' error messages have been disabled.

Example:

```
PSTERR/MACRO,10
  CHAR BUF(66)
  BUF = %ARG(1)
  IF (%ARG(1) == 'CIRMAXR' | BUF(1:4) == 'Mutu') THEN
  ELSE
    PSTERR
  ENDIF
TERMAC
```

2.5.8 The TAPBRK Macro

Command Syntax: **TAPBRK/MACRO, 0**

The *TAPBRK Macro* allows the programmer to define their own tape break sequences. Whenever the Control Tape is broken up into multiple sections due to a *BREAK* sequence, **PWorks** will call the *TAPBRK Macro*, if it has been defined. The sequence of commands contained in the *TAPBRK Macro* will override **PWorks**' normal processing of tape breaks.

This *Macro* will only be called when a tape break occurs. The following *TAPBRK Macro* example will perform the tape break sequence as described under the [BREAK/AUTO](#) command.

```
TAPBRK/MACRO,0
  RETRACT
  BREAK/PCHFILE
  TMARK
  PLUNGE
TERMAC
```

See the [BREAK](#) command in the **PWorks** Reference Manual for a complete description on tape breaks.

2.5.9 The CIRCLE Macro

Command Syntax: **CIRCLE/MACRO, 0**

This is a circular interpolation Macro that is called when a circular interpolation record is processed.

All parameters are passed to the CIRCLE Macro using the *%CIRFLG* and *%CIRPRM* post-processor variable arrays. These arrays contain the following circular interpolation data.

<i>%Cirflg</i> (1:3)	=	Pointers to axis components of circular record. 1 = X, 2 = Y, 3 = Z. A value of 0 defines this as a 3-axis circular interpolation record.
<i>%Cirflg</i> (4)	=	Starting quadrant for circular motion (1:4).
<i>%Cirflg</i> (5)	=	1 (Clockwise direction) 2 (Counter-clockwise)
<i>%Cirflg</i> (6)	=	0 (Standard circular motion) 1 (Machine generated helical motion) 2 (Post generated helical motion)
<i>%Cirflg</i> (7:8)	=	Not used.
<i>%Cirprm</i> (1:3)	=	Center of circle.
<i>%Cirprm</i> (4)	=	Radius of circle.
<i>%Cirprm</i> (5)	=	Beginning angle of circular motion.
<i>%Cirprm</i> (6)	=	Ending angle of circular motion.
<i>%Cirprm</i> (7)	=	Unsigned delta angle of circular motion.
<i>%Cirprm</i> (8:10)	=	Ending point of circular motion.
<i>%Cirprm</i> (11:13)	=	Starting point of circular motion. "XY, YZ, XZ" for standard 2-axis circular interpolation (XY,YZ,ZX), "XYZ" for 3-axis circular interpolation.
<i>%Cirprm</i> (14:16)	=	Starting vector for 3-axis circular interpolation.
<i>%Cirprm</i> (17:20)	=	Motion plane for 3-axis circular interpolation.
<i>%Cirprm</i> (21)	=	Linear distance per degree for 3rd axis during helical interpolation (-ve UP, +ve, DOWN)
<i>%Cirprm</i> (22:25)	=	Not used.

2.5.10 The XFORM Macro

Command Syntax: **XFORM/MACRO, 11**

The XFORM Macro is called whenever a transformation enable/disable block is output. If an XFORM Macro is defined, then **PostWorks** will call this Macro and will make no attempt to output the transformation codes, it is the user's responsibility to output the correct transformation blocks from within the XFORM Macro. In fact, the main purpose of the XFORM Macro is to output the transformation blocks.

The following Macro arguments will be passed into the XFORM Macro.

- %Arg(1) = 0 (Cancel Transformation block)
 - 1 (Only call to enable Transformation block)
 - 2 (First call "Pre Motion" to enable Transformation block)
 - 3 (Second call "Post_Motion" to enable Transformation block)
- %Arg(2) = X-translation
- %Arg(3) = Y-translation
- %Arg(4) = Z-translation
- %Arg(5) = Rotation about vector X-axis component
- %Arg(6) = Rotation about vector Y-axis component
- %Arg(7) = Rotation about vector Y-axis component
- %Arg(8) = Rotation about X-axis
- %Arg(9) = Rotation about Y-axis
- %Arg(10) = Rotation about Z-axis
- %Arg(11) = Rotation about vector angle

The rotation angle about a vector along with the rotations about the major axes will always be provided to the XFORM Macro. It is the user's responsibility to output the correct values. The major axes rotation angles will be calculated as specified by the following **MPost** prompt.

7.7.3 Enter transformation output logic:

Since the XFORM Macro is called from an output routine instead of being called from a post-processor command, there is a limitation on the post-processor commands that can be included in this Macro. The following list contains valid post-processor commands for the XFORM Macro.

APPEND	AUXFUN	FORCE	INSERT
OPSKIP	POSCAL	PPRINT	PRINT
PREFUN	REGORD	ROTABL/ADJUST	TRANS

CHAPTER 3 Variable Definitions

3.1 Introduction

PostMacro supports 4 types of variables; real, text, *Post variables* and *Macro* arguments. While *Post variables* and *Macro* arguments are defined by the **PostComp** compiler, real and text variables must first be declared using either the *REAL* or *CHAR* statement before being referenced.

Remember, any variables declared in the *GSMAIN Macro* are global and can be accessed by all of the *Macros* in the program. Variables allocated in all other *Macros* are local within that *Macro* and can only be referenced inside that *Macro*. See the '**PostMacro Basics**' chapter for a complete description on how variables can be referenced in your program.

3.2 Real Variable Declarations

Command Syntax: **REAL** **var1**[(**dim11**[,**dim12**[,**dim13**]])] [...]
varn[(**dimn1**[,**dimn2**[,**dimn3**]])]

The *REAL* statement is used to define real variables. Real variables may be declared as an array up to three dimensions by associating a numeric subscript, '(dim)', with them on the *REAL* statement. The dimension subscript must be a number, not a variable.

Example:

```
REAL A(10), I, SAVPT(6), T1(10,5), T2(3,4,5)
```

Note: Multiple dimension arrays can be referenced by using a single subscript. Multiple dimensions arrays are stored with the first dimension being incremented first, followed by the second dimension, and then by the third dimension if defined. For example, the following illustrates the mapping between referencing a multiple dimensioned array using 1 or 3 subscripts.

```
REAL A(4,10)
A(1)  = A(1,1)
A(2)  = A(2,1)
A(3)  = A(3,1)
A(4)  = A(4,1)
A(5)  = A(1,2)
A(6)  = A(2,2)
A(7)  = A(3,2)
A(8)  = A(4,2)
...
A(40) = A(4,10)
```

3.3 Text Variable Declarations - CHAR

Command Syntax: **CHAR var1[(dim11[,dim12[,dim13]])] [...]
varn[(dimn1[,dimn2[,dimn3]])]**

The *CHAR* statement is used to define text variables. Like real variables, text variables may be declared as an array up to three dimensions. Unlike real arrays, where each element of the array contains a single value, text arrays are considered text strings, with one character stored per element in the array. Therefore, you should make the array (32,767 maximum) large enough to hold all the text strings you would want stored in it.

Example:

CHAR MSG(80), DISBLK(20), ONOFF(10,3), TXT(30,3,2)

Note: 1. A multiple dimensioned text string can reference each individual string within the array by omitting the first subscript for three dimensional arrays only. For example

CHAR S(20,4,3)

S(1,1) = "String1 up to 20 cha"

S(2,1) = "String2 up to 20 cha"

S(3,1) = "String3 up to 20 cha"

S(4,1) = "String4 up to 20 cha"

.....

S(4,3) = "String12 up to 20 ch"

2. A two dimensional text array cannot be referenced in this manner. It can only be referenced as shown below.

CHAR S(20,4)

S(1:20,1) = "String1 up to 20 cha"

S(1:20,2) = "String2 up to 20 cha"

S(1:20,3) = "String3 up to 20 cha"

S(1:20,4) = "String4 up to 20 cha"

3. Partial string can be referenced by:

S(i:j,n) where "i" is the starting location. "j" is the ending location of the string and "n" is the string number for two dimensional text string

S(i:j,n,m) where "i" is the starting location, "j" is the ending location of the string, "n" and "m" defines the string number for three dimensional text string.

4. S = "" or S = ' ' defines an empty string.

5. The [STRLEN](#) funtion always report the maximum string length value for a multi-dimensioned string even the referenced string is partially filled or empty.

3.4 Post Variables

Post variables are **PostWorks** internal variables that may be accessed in the same manner as user defined variables. *Post variable* names are referenced using the form '*%name*'. See the '[PostMacro Basics](#)' chapter on how to reference *Post variables* in your program. See the '[Postprocessor commands](#)' chapter for a description on how to change the values of post variables. The following sections describe each of the available *Post variables*.

3.4.1 Macro Argument Variables Declarations - %ARG, %NARG

Command Syntax: **%ARG (n) , %NARG**

The parameters in a post-processor command that calls a *Macro* are passed to the *Macro* as arguments and are stored in the %ARG array. Unlike most of the other *Post variables*, each *Macro* will have its own copy of the %ARG array, and the values in the array elements will differ from *Macro* to *Macro*. The size of the %ARG array is defined in the [MACRO](#) statement as the maximum number of parameters to pass to the *Macro*.

The number of parameters in the calling post-processor command is stored in the %NARG variable. Like the %ARG array, each *Macro* will have its own copy of the %NARG variable and its value will differ from *Macro* to *Macro* and possibly from call to call.

PostMacro allocates both real and text storage for the %ARG variable. The manner in which %ARG is referenced determines whether it is treated as a real or text variable. Unlike user defined text variables, each element of the text portion of the %ARG array contains an 8 character text string, instead of a single character. The range subscript descriptor, when used with the %ARG array, points to multiple elements in the array and not to a substring of the text string.

When a [minor word](#) is passed to the *Macro*, %ARG will contain the text of the word ('CLW', 'ON', 'DRILL', etc.) and a numeric value of 0. When value is passed, %ARG will contain this value and a text string of a single space. You can test for a value or minor word using the following logic.

```
IF (%ARG(1) == ' ') THEN
    $$ %ARG(1) is a value.
ELSE
    $$ %ARG(1) is a minor word.
ENDIF
```

Assuming a *LOADTL Macro* has been defined, the following [LOADTL](#) command will set up the %ARG array as shown.

LOADTL/1, LENGTH, 10.5

%ARG(1) = 1 ; %ARG(2) = 'LENGTH' ; %ARG(3) = 10.5

Although you may not pass more parameters in a *Macro* call than the number of arguments allocated, you may pass fewer. When the calling postprocessor command contains fewer parameters than the number of arguments allocated, then the remaining elements of the %ARG array will contain the values they had in the previous call. If there was no previous call, then the array elements will have a default value of 0.

Note: The meanings of %ARG's are different for the GOTO and PPRINT macros. See the [GOTO](#) and [PPRINT](#) macro for details

3.4.2 Reference Sequence Numbers: - %CLREC, %ISN

Command Syntax: **%CLREC, %ISN**

The %CLREC variable contains the input cl record number. The cl record number will reflect any modifications made by post-processor *Macros* to the input clfile.

The %ISN variable contains the input sequence number, as it relates to the part program file, of the last record processed from the input clfile.

3.4.3 Number Of PARTNO Cards - %NPARTNO

Command Syntax: **%NPARTNO**

The %NPARTNO variable contains the number of [PARTNO](#) cards defined in the input clfile. A maximum of 100 PARTNO cards is allowed in the input clfile.

3.4.4 Clfile Settings - %CUTTER, %MULTAX

Command Syntax: **%CUTTER (7) , %MULTAX**

The parameters of the last *CUTTER* statement are stored in the %CUTTER array. The seven parameters that describe the *CUTTER* in most N/C languages are described below.

- 1 = Cutter diameter.
- 2 = Corner radius at bottom of cutter,
- 3 = Horizontal distance from center of corner radius (2) to center of cutter.
- 4 = Vertical distance from center of corner radius (2) to bottom of cutter.
- 5 = Angle in degrees that bottom of cutter makes with horizontal bottom line.
- 6 = Angle in degrees that side of cutter makes with a vertical line.
- 7 = Height of cutter

These parameters may differ depending on the N/C language used.

`%MULTAX` contains 1 when `MULTAX` is turned on in the input clfile and contains 0 when it is turned off. Please note that the `%MULTAX` variable reflects the status of `MULTAX` in the input clfile only, not the output clfile, which may be different because of the `MULTAX` command. See the 'Post-processor Commands' chapter for a description of the `MULTAX` command.

3.4.5 **PostWorks** Common - `%KPOSMP`, `%POSMAP`, `%USER`

Command Syntax: `%KPOSMP(4000), %POSMAP(4000), %USER(20)`

The `%KPOSMP` variable references the Integer portion of **PostWorks**' common area. The `%POSMAP` variable references the Real portion of common. You should be familiar with the internal storage of these commons before referencing either of these variables.

The `%USER` array is a dummy array that is not used by **PostWorks**. Its main purpose is to allow the programmer to store values here that can be output utilizing the [Print Descriptor File \(.pdf\)](#) in **PWorks**.

3.4.6 Control Tape Vars - `%REG`, `%TAPLEN`, `%UNITS`, `%BRKPRM`

Command Syntax: `%REG(92), %TAPLEN, %UNITS(2), %BRKPRM(3)`

The `%REG` array contains the current values for each of the [92 registers](#). `%REG(1)` contains the value for the A1 register, `%REG(2)` contains A2, etc. The register label with a beginning “#” character can be used instead of the corresponding integer such as `%REG(#C2)` contains C2.

The length of the output Control Tape is kept in the `%TAPLEN` variable. This length corresponds to the amount of tape that would be generated on a tape punch, which is approximately 120 characters per foot or 393.7 characters per meter. `%TAPLEN` will be in either feet or meters, depending on the output units.

The `%UNITS` array contains values that correspond to the input clfile (1) and output Control Tape (2) units. A value of 1 specifies Inch and value of 2 specifies Millimeters.

`%BRKPRM` stores the values associated with tape breaks. `%BRKPRM(1)` contains the tape length, in either feet or meters, since the last tape break, `%BRKPRM(2)` stores the machining time, in minutes, since the last tape break or End-of-sequence, and `%BRKPRM(3)` stores the machining distance since the last tape break or End-of-sequence.

PMacro does not set or change any of these variables.

3.4.7 Tape Registers Beginning/Ending Characters Vars - %REGST(92), %REGEN(92)

Command Syntax: **%REGST, %REGEN**

Each of these variables are arrays that have a dimensioned size of 92 entities. They can be used to access and define the beginning and ending characters of the tape registers.

The *%REGST* variable references the beginning characters of the tape registers. *%REGST(3)* or contains the beginning characters of register A1.

The *%REGEN* variable references the ending characters of the tape registers. *%REGEN(11)* contains the ending characters of register C5.

The register label with a beginning “#” character can be used instead of the corresponding integer such as *%REGST(#C2)* is same as *%REGST(8)*

3.4.8 Print File Variables - %LINENO, %PAGENO

Command Syntax: **%LINENO, %PAGENO**

The *%LINENO* variable contains the current line number of the output print file page. *%PAGENO* contains the current page number. These variables are used exclusively for **PWorks** print file and should only be used in the [Print Descriptor File \(.pdf\)](#). **PMacro** does not use either of these variables.

3.4.9 Error Messages - %ERROR, %ERRPC, %NUM---

Command Syntax: **%ERROR, %ERRPC, %NUMERR, %NUMFAT, %NUMIER, %NUMWRN**

The *%ERROR* variable contains the result of the last [SYNTAX](#) command executed. It contains 1 if the last *SYNTAX* command reported an error, otherwise it contains 0. See the 'Post-processor Commands' chapter for an explanation of the *SYNTAX* command.

The *%ERRPC* variable contains the PC location of the *Macro* instruction that generated the last error.

PMacro keeps a running count of the errors generated during the execution of the program. *%NUMERR* contains the number of errors that have occurred since the beginning of the program.

PWorks has four classes of error messages, Warning (*%NUMWRN*), Errors (*%NUMERR*), APT Source Input Syntax (*%NUMIER*) and FataIs (*%NUMFAT*). Warning messages are output when the error does not adversely affect the output, for example, too many parameters on a post-processor command. Error messages are output when it is likely that the error will affect the Control Tape output, for example, an ignored post-processor command. Errors occurred while processing a post-processor *Macro* will always generate an error message. APT Input Syntax messages will only be output if an invalid APT Input syntax is encountered. Fatal messages will only be output when an error occurred that will definitely cause the output to be incorrect, for example, if the tool axis cannot be satisfied with the active rotary axes.

%NUMWRN, *%NUMERR*, *%NUMIER* and *%NUMFAT* contain the number of each class of error that have occurred since the beginning of the program.

3.4.10 Miscellaneous Variables - *%PI*, *%SIMUL*

Command Syntax: ***%PI***, ***%SIMUL***

The *%PI* variable naturally contains the value of Pi; 3.1415927.

The *%SIMUL* variable is set to 1 when a simulation file is being generated. If both a punch file and simulation file are begin generated at the same time, then this variable will only be set to 1 during the simulation file generation. See [Simulation File](#) of the **PostWorks** reference manual for details of simulation file.

3.4.11 Tool Data - *%NTOOL*, *%TOOL*, *%SELCTL*, *%TLEN*, *%TLNO*

Command Syntax: ***%NTOOL***, ***%TOOL***, ***%SELCTL***, ***%TLEN(120)***, ***%TLNO(120)***

The Tool data variables contain information concerning the tools that have been used during the part program. Tool numbers, lengths, times, current tool and number of tools loaded since the beginning of the program are stored in these variables. The variables will actually contain the following values.

- %NTOOL* = Number of tools loaded up to the current location in the part program
- %TOOL* = A pointer to the Tool data arrays (*%TLEN*, *%TLNO* and *%TLTIM*).
%TOOL points to the tool that is currently loaded.
- %SELCTL*= The currently selected tool number.
- %TLEN* = Tool lengths for every tool used during the part program up to this point.
- %TLNO* = Tool numbers for every tool used during the part program up to this point.

See the '[Machining Times](#)' section in this chapter for a description of the `%TLTIM` (tool machining times) variable. **PMacro** does not use these variables, but they may be defined and referenced by the programmer for the purpose of writing a tool change *Macro*.

3.4.12 Feeds and Speeds - %FEED, %RPM, %SFM

Command Syntax: `%FEED(6), %RPM, %SFM`

The current feed rate for the tool tip, control point ([FEDRAT/LENGTH](#)), linear slides, and rotary axes (in Degrees per Minute) are stored in the `%FEED` array. The items of the `%FEED` array are described below.

- (1) = Tool tip feed rate.
- (2) = Control point ([FEDRAT/LENGTH](#)) feed rate.
- (3) = Machine linear axes feed rate.
- (4) = Machine rotary axes feed rate in Degrees per Minute.
- (5) = Combination degrees per minute and feed per minute rate (Pulses per minute).
- (6) = Feed per spindle revolution feed rate, at the feed rate control point.

PWorks keeps track of the current spindle speed in both Revolutions per Minute and Surface Feed per Minute. `%RPM` contains the current spindle revolutions per minute. `%SFM` contains the current surface feed per minute. The part radius is used for *SFM* calculations when programming lathes and the cutter diameter is used for mills.

`%RPM` and `%SFM` will be set to zero when the spindle is turned off.

These variable are only defined by **PWorks** and are never set or changed by **PMacro**.

3.4.13 Delta Move Distances - %MOVDIS

Command Syntax: `%MOVDIS(4)`

The current delta distances for the current move are stored in the `%MOVDIS` array. The items of the `%MOVDIS` array are described below.

- (1) = Delta movement for the tool tip.
- (2) = Delta movement for the feed rate control point ([FEDRAT/LENGTH](#)).
- (3) = Delta movement for the machine linear axes (tool top).
- (4) = Delta movement for the machine rotary axes in degrees.

The delta movement array is usually used in the [Print Descriptor File \(.pdf\)](#) in **PWorks** and is never set or change by **PMacro**.

3.4.14 Machining Times - %---TIM, %TIMSCL

Command Syntax: **%MOVTIM, %SEQTIM, %MCHTIM, %TLTIM(120), %TIMSCL**

PWorks calculates and stores the machining times for the current move, current sequence, each tool, and the entire part program. These values are stored in the following post variables.

- %MOVTIM = Amount of time that the current move takes.
- %SEQTIM = Total machining time for the current sequence.
- %MCHTIM = total machining time for the part program from the beginning to the current location.
- %TLTIM = Total machining time for each programmed tool. Also see the '[Tool Data](#)' section in this chapter.

The machining time variables are usually used in the [Print Descriptor File \(.pdf\)](#) in **PWorks** and are never set or changed by **PMacro**.

The %TIMSCL variable allows the programmer to modify the machining time that **PWorks** calculated for each move. The machining time will be multiplied by %TIMSCL. The value of %TIMSCL is set to 1.0 at the beginning of the program by **PWorks**, which generates machining times at 100 percent of calculations, but can be modified at any time during the program by using the [DEFINE/%TIMSCL](#) command in a *Macro*.

%TIMSCL is not used by **PMacro**.

3.4.15 Clfile Points - %CLPT, %CLSAV, %TLVEC

Command Syntax: **%CLPT(n), %CLSAV(6), %TLVEC(3)**

The cl point data of a motion record is stored in the %CLPT array. The %CLPT array can be a single dimensional or two dimensional array with the X, Y, Z and optionally the I, J, K tool axis vector packed in it. The size of the array depends on the number of points in the motion record and whether [MULTAX](#) is turned on or off. The %CLPT array is usually used in the [GOTO Macro](#), where the number of cl points in the motion record is passed in %ARG(1).

When [MULTAX](#) is turned off, only the X, Y, Z values are stored in the %CLPT array, therefore each cl point takes up 3 elements of the array. When [MULTAX](#) is turned on, the X, Y, Z, I, J, K values are stored in the array and each cl point takes up 6 elements.

See the description of the *GOTO Macro* in the '[Macro Definitions](#)' chapter for an example of how to reference the %CLPT array.

The `%CLSAV` array contains the X, Y, Z, I, J, K values of the last cl point output to the clfile. The tool axis components (I, J, K) are always stored in the `%CLSAV` array, unlike the `%CLPT` array, where they are only stored when `MULTAX` is turned on.

In the case where `MULTAX` is turned off, **PostMacro** will assume the last specified tool axis vector. A 0,0,1 tool axis vector is assumed at the beginning of the program.

The `%TLVEC` array contains the I, J, K components of the current tool axis vector. This array will be set by **PWorks** after the input cl points have been converted to output coordinates using the linear and rotary output axes specifications. Therefore, the `%TLVEC` array, if referenced in the `GOTO Macro`, will actually contain the previous tool axis vector. The `%CLPT` array, in this case, will contain the current tool axis vector.

PMacro does not set or change the `%TLVEC` array.

3.4.16 Input Coordinates - %XYZ, %XYZ---

Command Syntax: `%XYZ (12) , %XYZDLS (12) , %XYZDLT (12) , %XYZMAX (12) , %XYZMIN (12)`

The input coordinate arrays contain value corresponding to the input clfile XYZ position. The array item variables will contain the following axis positions.

- (1:3) = XYZ location from clfile without any modifications. This is similar to the `%CLPT` array but contains only a single location, and like the `%TLVEC` array, will not be set until all adjustments are made for the output linear and rotary axes.
- (4:6) = XYZ location after `TRANS` and `ORIGIN` have been applied.
- (7:9) = XYZ location after being adjusted for the rotary axes and tool length.
- (10:12) = XYZ location after `TRANS/LAST` has been applied.

The arrays themselves will contain the following values.

- `%XYZ` = Current input XYZ position.
- `%XYZDLS` = Input XYZ delta summations for the current sequence only.
- `%XYZDLT` = Input XYZ delta summations for the part program from the beginning to the current location.
- `%XYZMAX` = Maximum input XYZ travel limits reached for the part program from the beginning to the current location.
- `%XYZMIN` = Minimum input XYZ travel limits reached for the part program from the beginning to the current location.

The input XYZ location arrays are usually used in the [Print Descriptor File \(.pdf\)](#) in **PWorks** and are never set or changed by **PMacro**.

3.4.17 Linear Axes - %LINEAR, %LIN---

Command Syntax: **%LINEAR(6)** , **%LINDLS(6)** , **%LINDLT(6)** , **%LINMAX(6)** ,
%LINMIN(6)

The linear axis arrays contain values corresponding to the position of the linear (both primary and secondary) axes. The array item of these variables will contain the following axis positions.

- | | | | | | |
|-----|---|-----------------|-----|---|-------------------|
| (1) | = | Primary X-axis. | (2) | = | Secondary X-axis. |
| (3) | = | Primary Y-axis. | (4) | = | Secondary Y-axis. |
| (5) | = | Primary Z-axis. | (6) | = | Secondary Z-axis. |

These arrays are different than the output axis arrays (%*AXIS*), because they do not have the *TRANS/-AXIS* values applied to them. The arrays themselves will contain the following values.

- | | | |
|---------|---|------------------------------------------------------------------------------------------------------------|
| %LINEAR | = | Current primary and secondary linear axis positions. |
| %LINDIS | = | Linear axis delta summations for the current sequence only. |
| %LINDLT | = | Linear axis delta summations for the part program from the beginning to the current location. |
| %LINMAX | = | Maximum linear axis travel limits reached for the part program from the beginning to the current location. |
| %LINMIN | = | Minimum linear axis travel limits reached for the part program from the beginning to the current location. |

The linear axis arrays are usually used in the [Print Descriptor File \(.pdf\)](#) in **PWorks** and are never set or changed by **PMacro**.

3.4.18 Rotary Axes - %ROTARY, %ROT---

Command Syntax: **%ROTARY(4)** , **%ROTDLS(4)** , **%ROTDLT(4)** , **%ROTMAX(4)** ,
%ROTMIN(4)

The rotary axis arrays contain values corresponding to the position of the rotary axes on a [linear scale](#). The array item variables will contain the following axis positions.

- | | | | | | |
|-----|---|-----------------|-----|---|-----------------|
| (1) | = | Rotary axis #1. | (2) | = | Rotary axis #2. |
| (3) | = | Rotary axis #3. | (4) | = | Rotary axis #4. |

These arrays are different than the output axis arrays (%*AXIS*), because they do not have the *TRANS/-AXIS* values applied to them and are always on a [linear scale](#), no matter what the output format is. The arrays themselves will contain the following values.

- | | | |
|---------|---|--------------------------------------------------|
| %ROTARY | = | Current rotary axis positions on a linear scale. |
|---------|---|--------------------------------------------------|

- %ROTDLS = Rotary axis delta summations for the current sequence only.
- %ROTDLT = Rotary axis delta summations for the part program from the beginning to the current location.
- %ROTMAX = maximum rotary axis travel limits reached for the part program from the beginning to the current location.
- %ROTMIN = Minimum rotary axis travel limits reached for the part program from the beginning to the current location.

The rotary axis arrays are usually used in the [Print Descriptor File \(.pdf\)](#) in **PWorks** and are never set or changed by **PMacro**.

3.4.19 Output Axes - %AXIS, %AXS---

Command Syntax: **%AXIS (10) , %AXSDLS (10) , %AXSDLT (10) , %AXSMAX (10) , %AXSMIN (10)**

The output axis arrays contain values corresponding to the position of the linear (both primary and secondary) and rotary axes that are actually output to the Control Tape. The array items of these variable will contain the following axis positions.

- | | |
|-----------------------|-------------------------|
| (1) = Primary X-axis. | (2) = Secondary X-axis. |
| (3) = Primary Y-axis. | (4) = Secondary Y-axis. |
| (5) = Primary Z-axis. | (6) = Secondary Z-axis. |
| (7) = Rotary axis #1. | (8) = Rotary axis #2. |
| (9) = Rotary axis #3. | (10) = Rotary axis #4. |

The arrays themselves will contain the following values.

- %AXIS = Actual linear and rotary axis output positions, after all translations have been applied.
- %AXSDLS = Output linear and rotary axis delta summations for the current sequence only.
- %AXSDLT = Output linear and rotary axis delta summations for the part program from the beginning to the current location.
- %AXSMAX = maximum linear and rotary axis travel limits reached for the part program from the beginning to the current location.
- %AXSMIN = Minimum linear and rotary axis travel limits reached for the part program from the beginning to the current location.

The output axis arrays are usually used in the [Print Descriptor File \(.pdf\)](#) in **PWorks** and are never set or changed by **PMacro**.

3.4.20 *PWorks* Machine Number - %MACHIN

Command Syntax: **%MACHIN**

The %*MACHIN* variable contains the number of the [Machine Descriptor File \(MDF\)](#) current active. This number is specified on the [MACHIN/PWORKS](#) statement. When there is more than one machine number specified on the [MACHIN](#) statement, this variable will contain the machine number which is currently being processed.

This variable has a value of zero if a named MDF file is used.

3.4.21 *PWorks* Machine Name - %PSTNAM

Command Syntax: **%PSTNAM**

The %*PSTNAM* variable contains the textual string representation of the current active [Machine Descriptor File \(MDF\)](#) name or number.

3.4.22 Home Position - %HOME

Command Syntax: **%HOME (10)**

The %*HOME* array contains the machine's [home position](#) as defined in **MPost**. This position is moved to when the [GOHOME](#) statement is issued in the part program. The array items of the %*HOME* array contain the following axis positions.

- | | |
|-----------------------|-------------------------|
| (1) = Primary X-axis. | (2) = Secondary X-axis. |
| (3) = Primary Y-axis. | (4) = Secondary Y-axis. |
| (5) = Primary Z-axis. | (6) = Secondary Z-axis. |
| (7) = Rotary axis #1. | (8) = Rotary axis #2. |
| (9) = Rotary axis #3. | (10) = Rotary axis #4. |

The only way to change the home position during a part program is to use the [DEFINE](#) command to change the %*HOME* array.

CHAPTER 4 Expressions

4.1 Introduction

An expression consists of a basic single component (such as a variable, number, text string, etc.) or a combination of basic components with one or more operators or functions. Both numeric and text expressions are supported. Numeric expressions produce a single numeric value and text expressions produce a text string.

Also discussed in this chapter are [format descriptors](#). Format descriptors reference the [Register Formats](#) and are used when converting numbers to text strings and vice versa.

4.2 Numeric Expressions

Numeric expressions may be composed of arithmetic operators, real numbers, real variables, post variables and function designators. Parenthesis may be used to override the order of priority for operators. You can have up to 50 operators in an expression. Numeric expressions can appear most everywhere that a number can.

4.2.1 Number Operators

Number operators specify a computation that is to be performed on a set of numbers or real variables. Each operator in an expression is considered a single calculation and will produce a single value.

Calculations in an expression are done in order of priority; the highest priority functions are performed first, and when operators have the same priority, they will be performed from left to right. Following is a list of numeric operators and their meanings and priority.

Numeric Operator	Function	Priority	
-----	-----	-----	
()	Parenthesis	1	(highest)
Functions		2	
**	Exponentiation	3	
*	Multiplication	4	
/	Division	4	
+	Addition	5	
-	Subtraction	5	
==	Equivalent	6	
<>	No equivalent	6	
<	Less than	6	
>	Greater than	6	
<=	Less than or equal	6	
>=	Greater than or equal	6	

&	And	7	(lowest)
	Or	7	

Parenthesis and functions, although not operators, are contained in the preceding list to show their priority in an expression. The '== < > <= >= & |' operators are considered relational operators, not arithmetic operators. Each of these operators will return a value of zero if the relation is true or a value of one if the relation is false.

4.2.2 Use Of Parenthesis

Parenthesis may be used to signify a particular order of evaluation. When part of an expression is enclosed in parenthesis, this part is evaluated first, and then the remainder of the expression. When multiple levels of parenthesis are used in an expression, the innermost level will be calculated first. The following example illustrates the use of parenthesis and the effect they have on the outcome of an expression, The order in which the operators are evaluated is also shown.

Example:

$$\begin{array}{ccccccc} 8 & + & 6 & * & 3 & - & 4 & / & 2 & = & 24 \\ \wedge & & \wedge & & \wedge & & \wedge & & \wedge & & \\ 2 & & 1 & & 4 & & 3 & & & & \end{array}$$

$$\begin{array}{ccccccc} (8 & + & 6) & * & 3 & - & 4 & / & 2 & = & 40 \\ \wedge & & \wedge & & \wedge & & \wedge & & \wedge & & \\ 1 & & 2 & & 4 & & 3 & & & & \end{array}$$

$$\begin{array}{ccccccc} ((8 & + & 6) & * & 3 & - & 4) & / & 2 & = & 19 \\ \wedge & & \wedge & & \wedge & & \wedge & & \wedge & & \\ 1 & & 2 & & 3 & & 4 & & & & \end{array}$$

It is recommended to use parenthesis with complex equations when it is not clear what the order of operator evaluation will be.

4.2.3 Numeric Functions

PostMacro provides a set of arithmetic, trigonometric and miscellaneous functions to use in numeric expressions. Numeric functions are considered functions that return a number, even though their arguments may be numeric or textual in nature. Following is a list of functions and their purpose.

Arithmetic Function

ABS(n)	Absolute value
INT(n)	Integer value
SQRT(n)	Square root

Trigonometric Functions

COS(n)	Cosine
SIN(n)	Sine
TAN(n)	Tangent
ACOS(n)	Arc cosine
ASIN(n)	Arc sine
ATAN(n)	Arc tangent

Miscellaneous Functions

The miscellaneous functions will be described in detail, since their purpose is not as obvious as the arithmetic and trigonometric functions.

CTOR(t,#f)

The *CTOR* function converts a text string 't' to a number, using the format descriptor '#f'. The text string must contain valid numeric characters; an optional sign (+-), numeric digits (0-9) and optional decimal point (.). See the ['Format Descriptors'](#) section for information on format descriptors.

Example:

```
CREG = '40128'  
#C6 = L3.4, M0.1, S-  
CNUM = CTOR(CREG,#C6)    $$ CNUM = 4.0128
```

INDEX(t1,t2)

The *INDEX* function is case insensitive and searches for the first occurrence of the text string 't2' in the text string 't1'. *INDEX* will return the location, in 't1', of the start of the string 't2' or return zero if the string is not found.

Example

```
MSG = 'Load the next tool; Press the Start Button.'  
I=INDEX(MSG,';')    $$ I = 19
```

LOCATE(a(m),n)

The *LOCATE* function searches the real array 'a', starting at the subscript position 'm', for the value 'n' and returns the position in the array that contains this value. The position returned is based on the starting position 'm' within the actual array. *LOCATE* returns zero if the value is not found.

Example:

```
TNO(1) = 12345; TNO(2) = 45124; TNO(3) = 67893
TLEN(1) = 10.5; TLN(2) = 4.2; TLN(3) = 7.0
DREG(1) = 11; DREG(2) = 12; DREG(3) = 13
$$
I = LOCATE(TNO(1),%ARG(1))
$$
LOADTL/TNO(I),LENGTH,TLEN(I)
TOOLNO/ADJUST,DREG(I)
```

RINDEX(t1,t2)

The *RINDEX* function is case insensitive and searches for the last occurrence of the text string 't2' in the text string 't1' and returns the position in the text string 't1' of the start of the string 't2'. *RINDEX* will return zero if the string is not found.

Example:

```
BLK = 'N1G00X0.Y0.Z10.F3.0M08$'
I = RINDEX(BLK,'0')    $$ I = 21
```

STRLEN(text_var)

The *STRLEN* function returns the length of a text string.

Example:

```
ST1 = 'OPERATION 1: ROUGHING'
N = STRLEN(ST1)    $$ N = 21
```

4.2.4 Numeric Assignment Statements

A numeric assignment statement assigns a numeric value to a real variable or array element. The value stored in the variable or array element to the left of the equals (=) will be replaced with the value calculated from the numeric expression to the right of the equals.

The numeric assignment statement has the form:

$$v = e$$

Where:

- v = Real variable or array element to receive the value of the calculation. 'v' cannot be a number.
- e = Numeric expression, real variable, array element or number.

Example:

Valid

A = 1

C = C / A * (B(3) - 2)

Invalid

1 = A

\$\$ Number not allowed on left side of equal.

%PI = 3.14

\$\$ **Post variable** not allowed on left side of equal.

\$\$ Use the **DEFINE** command instead.

A = 3 + '2.5'

\$\$ Text string not allowed in numeric expression.

4.3 Text Expressions

Text expressions may be composed of text operators, text strings, text variables, *Macro arguments* and function designators. Parenthesis may be used to signify the order of priority of operators, but are usually not required, as text expressions are generally used to concatenate or compare strings. You can have up to 50 operators in an expression. Text expressions can appear most everywhere that a text string can.

Also, test variables can appear in a post-processor command and *Post Macro* will determine if the text variable is in actuality a minor word. If it is not, then an error will occur.

4.3.1 Text Operators

Text operators specify a computation that is to be preformed on a set of text strings or variables. Each operator in an expression is considered a single calculation and will produce a single text string.

Calculations in an expression are done in order of priority; the highest priority functions are performed first, and when operators have the same priority, they will be performed from left to right. Following is a list of text operators and their meanings and priority.

Text Operator	Function	Priority	
-----	-----	-----	
()	Parenthesis	1	(highest)
Functions		2	
+	Concatenate	3	
==	Equivalent	4	
<>	No equivalent	4	
&	And	5	(lowest)
	Or	5	

Parenthesis and functions, although not operators, are contained in the preceding list to show their priority in an expression. The '== <> & |' operators are considered relational operators, not text operators. Each of these operators will return a single space ' ' if the relation is true and a '1' if the relations is false.

4.3.2 Text Functions

PostMacro provides a set of text string functions to use in text expressions. Text functions are considered functions that return a text string, even though some of the arguments may be numeric in nature. Following is a list of functions and their purpose.

CLBRIEF

The *CLBRIEF* function returns the name of the input clfile that does not include the device and directory name.

Example:

```
MSG = 'Current input clfile name is ' + CLBRIEF
$$ MSG = 'Current input clfile name is input.cl'
```

CLEXT

The *CLEXT* function returns the extension of the input clfile only.

Example:

```
MSG = 'Current input clfile extension is ' + CLEXT
$$ MSG = 'Current input clfile extension is .cl'
```

CLNAME

The *CLNAME* function returns the name of the input clfile that includes the device and directory name.

Example:

```
MSG = 'Current input clfile is ' + CLNAME
$$ MSG = 'Current input clfile is C:/directory/file.cl'
```

COMMAND

The *COMMAND* function returns a text string representation of the post-processor command that called the current Macro. *COMMAND* uses the *%ARG* macro argument

array and the *%NARG* *Post variable* to build the command and does not require any arguments.

Example:

```
$$
$$    Assume the post-processor command:
$$      LOADTL/1,LENGTH,10.5
$$    Called the LOADTL Macro
$$
CHAR CMD(80)
CMD = COMMAND  $$ CMD = 'LOADTL/ 1., LENGTH, 10.5'
```

COMPUTER

The *COMPUTER* function returns the name of the computer the program is running on.

Example:

```
MSG = 'Computer Name is ' + COMPUTER
$$ MSG = 'Computer Name is My_Computer'
```

DATE

The *DATE* function returns the current date in the format 'dd-mmm-yyyy' and does not require any arguments.

Example:

```
INSERT/'(MSG,RUNDATE: ' + DATE + ')$'
$$ INSERT (MSG,RUNDATE: 12-SEP-2000)$
```

ERRTXT(t)

The *ERRTXT* function returns the text of the error message associated with the label 't'. If there is not an error message label 't', then *ERRTXT* will return 't'. See the **PostWorks Users Guide** for a list of error messages and their labels.

Example:

```
MSG = ERRTXT('DIVZERO')
$$ MSG = 'Attempted divide by zero.'
```

FMTCOD(n,#f)

The *FMTCOD* function converts a number or real variable 'n' to a text string, using a register [format descriptor](#) '#f'. the output text string will also contain the specified register's beginning and ending strings, as defined using the [Tape Register Format form](#) in the *MPost* utility.

FMTCOD is usually not used with *PMacro*, since the register beginning and ending strings cannot be defined. Use the *RTOC* function instead.

Example:

```
PNUM = 18
PREG = FMTCOD(PNUM,#P) $$ PREG = 'P0018'
```

PCHBRIEF

The *PCHBRIEF* function returns the name of the output punch file that does not include the device and directory name.

Example:

```
MSG = 'Current punch file name is ' + PCHBRIEF
$$ MSG = 'Current punch file name is file.ext'
```

PCHFILE

The *PCHFILE* function returns the name of the output punch file that includes the device and directory name.

Example:

```
MSG = 'Current punch file is ' + PCHFILE
$$ MSG = 'Current punch file is C:/directory/file.ext'
```

PCHNAME

The *PCHNAME* function returns the name of the output punch file without the device, directory name, and file extension.

Example:

```
MSG = 'Current punch file is ' + PCHNAME
$$ MSG = 'Current punch file is file'
```

RTOC(n,#f)

The *RTOC* function converts a number of real variable 'n' to a text string, using the format descriptor '#f'. See the [Format Descriptors](#) section for information on format descriptors.

Example:

```
XNUM = -3.45857
#U2 = F3.4, M1.0 S-
XREG = RTOC(XNUM,#U2)  $$ XREG = '-3.4586'
```

TIME

The *TIME* function returns the current time in the format 'hh:mm:ss' and does not require any arguments.

Example:

```
MSG = 'Current time is ' + TIME
$$ MSG = 'Current time is 10:35:48'
```

USERNAME

The *USERNAME* function returns the name of the user (computer log in name) who is running the program.

Example:

```
MSG = 'Programmer name is ' + USER
$$ MSG = 'Programmer Name is John Doe'
```

4.3.3 Text Assignment Statements

A text assignment statement assigns a text string to a text variable array or portion of an array. The string stored in the text array to the left of the equals (=) will be replaced with the string calculated from the text expression to the right of the equals.

The text assignment statement has the form:

$$v = e$$

Where:

v = Text variable or array descriptor to receive the string of the calculation. 'v' cannot be a text string. An array descriptor can be a single subscript or a range subscript

(start:end). If 'v' is a dimensioned variable declared without a subscript, then the entire array will receive the text expression.

e = Text expression, text variable, text array or text string.

Example:

Valid

LA = 'Message text.'

LB(6:10) = 'POINT'

LC95) = 'A'

LD = '(MSG,RUNDATE: ' + DATE + ' TIME: ' + TIME + ')

Invalid

'A' = LA \$\$ Text string not allowed on left side of equal.

LB = 3 + '2.5' \$\$ Number not allowed in text expression.

4.4 Format Descriptors

The functions [CTOR](#), [FMTCOD](#) and [RTOC](#) use a format descriptor that defines the format of the text string representation of a number. A format descriptor is recognized as a pound sign '#' followed by either a register label or a number, for example '#B1' or '#3'. Up to 92 descriptors may be defined, one for each Register Label.

When using **PWorks**, the format descriptors are usually defined using the [Tape Register Format form](#) in the **MPost** utility, not with a format expression within a *Macro*. A format expression, when used with **PWorks**, will overwrite the default format descriptor that was set using **MPost**.

In **PMacro**, it is not recommended to use a format descriptor until it has been previously defined in the *Macro* program.

4.4.1 Format Expressions

Command Syntax: **#f=Ll.r,Mm.n,S+**

T *****
F
D

A format expression consists of a format descriptor and parameters that define the attributes for this descriptor. Format expressions consist of the floating point format (leading/trailing zero suppression, floating point or decimal numbers), the minimum number of digits to the left and right of the (implied) decimal point, and whether the number should have a plus (+) and/or minus (-) sign.

Four floating point formats are supported by **PostWorks**. 'L' defines the number as not having a decimal point and leading zeros will be removed. 'T' also defines the number as not having a decimal point, but trailing zeros will be removed. 'F' specifies that the number will always contain a decimal point and leading and trailing zeros will be removed. 'D' specifies that the number will contain a decimal point, except for whole numbers (non-fractional) which will drop the decimal point. Leading zeros are before the first non-zero digit in the number (the sign '+, -' is not considered a digit) and trailing zeros are after the last non-zero digit or decimal point.

The number of digits to the left of the (implied) decimal point is defined by 'l' and the number of digits to the right of the (implied) decimal point is defined by 'r'. the actual decimal point, if specified by 'F' or 'D', is not considered a numeric digit and therefore should not be included in the values for 'l' or 'r'.

'M' specifies the minimum number of digits to output to the left of the (implied) decimal 'm', and to the right of the decimal point 'n'.

'S' specifies which signs, if any, are required in the number. '+' includes a plus sign on positive numbers, '-' includes a minus sign on negative numbers, and '*' includes a minus sign on negative numbers and the number zero. '-' and '*' are mutually exclusive; do not specify both in the same format expression.

You may also equivalence one format descriptor to another, using the following syntax.

#f1 = #f2

Example:

The following list shows how certain values will be converted using different format expressions.

Value	L3.4,M0.1,S-	T4.3,M2.0,S-	F4.2,M0.0,S+*	D3.1,M1.0,S+
-----	-----	-----	-----	-----
0.0	0	00	-.	+0
12.1234	121234	0012123	+12.12	+12.1
-5.01	-50100	-000501	-5.01	5
.3	3000	00003	+.3	+.3
10	-100000	0010	-10.	10

4.4.2 Register Labels

Logical register labels are used to specify a type of register, for example, a G-code when the group is not known, **PostWorks** will calculate which physical register to use depending on the value associated with the logical register or a current mode setting (absolute/incremental, etc.). See the **PostWorks** user Guide for a complete [description of the physical and logical registers](#) supported.

Following is a list of the register labels used by ***MPost*** and their corresponding format descriptor values. You may use either the Register label (#G7) or number value (#25) as a format descriptor.

Physical Registers

1	=	A1	17	=	F3	33	=	J2	48	=	N	63	=	Y1	78	=	AL
2	=	A2	18	=	G0	34	=	K1	49	=	O	64	=	Y2	79	=	AM
3	=	A3	19	=	G1	35	=	K2	50	=	P	65	=	Z1	80	=	AN
4	=	B1	20	=	G2	36	=	L	51	=	Q	66	=	Z2	81	=	AO
5	=	B2	21	=	G3	37	=	M0	52	=	R	67	=	AA	82	=	AP
6	=	B3	22	=	G4	38	=	M1	53	=	S	68	=	AB	83	=	AQ
7	=	C1	23	=	G5	39	=	M2	54	=	T	69	=	AC	84	=	AR
8	=	C2	24	=	G6	40	=	M3	55	=	U1	70	=	AD	85	=	AS
9	=	C3	25	=	G7	41	=	M4	56	=	U2	71	=	AE	86	=	AT
10	=	C4	26	=	G8	42	=	M5	57	=	V1	72	=	AF	87	=	AU
11	=	C5	27	=	G9	43	=	M6	58	=	V2	73	=	AG	88	=	AV
12	=	C6	28	=	GA	44	=	M7	59	=	W1	74	=	AH	89	=	AW
13	=	D	29	=	H	45	=	M8	60	=	W2	75	=	AI	90	=	AX
14	=	E	30	=	I1	46	=	M9	61	=	X1	76	=	AJ	91	=	AY
15	=	F1	31	=	I2	47	=	MA	62	=	X2	77	=	AK	92	=	AZ
16	=	F2	32	=	J1												

Logical Registers

-1	=	G	-2	=	M	-3	=	\$	-4	=	X	-5	=	U	-6	=	Y
-7	=	V	-8	=	Z	-9	=	W	-10	=	A	-11	=	B	-12	=	C
-13	=	@	-14	=	F												

CHAPTER 5 Post-processor Commands

5.1 Standard Post-Processor Commands

Post-processor commands are handled in a fashion similar to the way they are handled in the CAM processor. The basic syntax for post-processor commands is as follows.

major-word[/parameter-list]

The **major word** may be any of the recognized major words, such as *LOADTL*, *CUTCOM*, *PPRINT*, etc. The parameter list may contain numbers, variables, **minor words**, and equations, separated by commas.

Example

```
LOADTL/1,LENGTH,10.5
STOP
CUTCOM/LEFT,XYPLAN,dreg(I)
CHECK/XAXIS,minx+1,max-1
```

The standard parameter list contains one entity (variable, number, etc.) per parameter, you may also specify *Post variables*, including **macro arguments**, with a range subscript '(start:end)', allowing easier entry of multiple parameter commands. **PostMacro** will expand the parameter list to include all of the entities that are in the specified range.

Example:

```
TOOLNO/ADJUST,%ARG(1:3),NOW
```

Is the same as

```
TOOLNO/ADJUST,%ARG(1),%ARG(2),%ARG(3),NOW
```

Text variables and strings may also be used in the parameter list, but they must equate to a recognized minor word.

Example:

```
DIR = 'CLW'
SPINDL/300,DIR
```

5.2 Text Post-Processor Commands

The text post-processor commands (*INSERT*, *LETTER*, *PARTNO*, *PPRINT*, *APPEND*) may be input in the same way that the CAM processor accepts them; the **major word** followed by a string of characters. When they are input in this manner, then the major word must start in column 1.

In addition, you may put a slash (/) directly after the major word followed by a valid text expression.

Up to 512 characters can be specified for the text string or the output of the text expression when *APPEND*, *INSERT* and *PPRINT* commands are used inside **PostMacro**. This is different from the situation when they are used within the CAM processor, only up to 66 characters will be allowed for the **NCL** binary cl file or up to 80 characters for the APT Source textual cl file.

Although these commands allow up to 512 characters, the actual maximum characters that can be specified in these commands is shown in the following equation. This is due to the fact that there is a limit of characters per Tape Block that can be output to the Control Tape.

$$\text{"maximum number of characters allowed"} = L - N - M$$

Where:

- L Maximum number of characters per tape block can be output to the Control Tape. This is set in the **MPost** menu item 3.8.2: "Maximum length of a MCD block". The range is from 10 to 512.
- N Number of output character spaces utilized by the Sequence Number block at this moment if required.
- M Number of output character spaces utilized by the "Start of the Message Characters" and the "End of the Message Characters" for the *PPRINT* command if required.

Example:

```
PARTNOTAPE 1 POSITION 1
INSERTN1G00X0Y0Z10M05$
PPRINT/'LOAD TOOL #' + ITLNO(I)
INSERT/'(MSG,DATE: ' + DATE + ' TIME: ' + TIME + ')$'
```

5.3 Special Post-Processor Commands

The following sections describe post-processor commands that may appear in the *Macro* definition file, but since the CAM processor does not recognize them (or they are treated

differently), they cannot be placed in the part program. An exception is the *MACHIN* statement, which must be placed in the part program file and not in a *Macro* definition file.

5.3.1 Appending Text To A Tape Block - APPEND

Command Syntax: **APPENDtext**

The *APPEND* command is used to append text to the end of a Control Tape block. The text contained in the *APPEND* command will be output exactly as is at the end of the next block. **PWorks** does not alter or check the validity of the input text, so the *APPEND* command should not be used to output a code which is obtainable using another post-processor command.

Example:

```
APPENDM10M26$  
GOTO/1,2,3
```

Outputs the tape block 'G01X1.0Y2.0Z3.0M10M26\$' to the Control Tape.

5.3.2 Multiple Control Tape Files - BREAK

Command Syntax: **BREAK/PCHFILE**

The *BREAK/PCHFILE* command will break the Control Tape into multiple sections by performing the following functions.

1. Output an End-of-tape sequence (M02, End-of-tape characters, trailing leading leader, etc.).
2. Close the current punch file and open a new one. If the file extension of the punch file has number in it, then this number will be incremented by one. For example, if the current punch file extension is '.pu1', then the file extension of the new file will be '.pu2'.
3. If the *PARTNO* card contains the text 'REEL n', then 'n' will be incremented by one. For example, 'REEL 1' will be changed to 'REEL 2'.
4. If the program number was specified on the *MACHIN/PWORKS* command, then it will be incremented by one.
5. Output a Start-of-tape sequence (*PARTNO*, beginning leader, program number, rewind stop code, etc.).

The above sequence will also be performed whenever a tape break sequence is output that includes support for generating multiple punch files. *BREAK/PCHFILE* is usually used in the *TAPBRK macro*.

5.3.3 Post Variable Assignments - DEFINE

Command Syntax: **DEFINE/post_var1,val1 [...] post_varn,valn**

Post variables (*%ARG*, *%CLPT*, etc.) can be referenced anywhere a user declared variable can, except that you cannot assign them values using a standard variable assignment statement. You must use the *DEFINE* command when you want to change the value of a *Post variable*.

'post_var' is the name of the variable that you want to change and 'val' is the value you want stored in it. 'val' can be a number, variable, expression, minor word (for the *%ARG* variable only), or another *Post variable*.

Example:

```
DEFINE/%NARG,3, %ARG(1),1, %ARG(2),LENGTH, %ARG(3),10.5
```

5.3.4 Disabling Macros - DISABLE

Command Syntax: **DISABLE/macro**

There may be times when you do not want a Macro called when it normally would be. For example, you may define the *GOTO Macro* for use when a drilling cycle is in effect. You would disable the *GOTO Macro* from being called in the *G\$MAIN Macro* and enable it in the *CYCLE Macro*.

The *DISABLE* command disables a *Macro* from being called. 'macro' is the name of the *Macro* to disable.

5.3.5 Enabling Macros - ENABLE

Command Syntax: **ENABLE/macro**

The *ENABLE* command allows a *Macro* to be called. 'macro' is the name of the *Macro* to enable. All defined macros are enabled at the start of the program by default.

Example:

```
ENABLE/PPRINT
```

5.3.6 Displaying Error Messages - ERROR

Command Syntax: **ERROR/text-expression**

The *ERROR* command enables you to output your own error message (up to 80 characters) at any place in the program. The error message will be output in the same format as error messages generated by **PostWorks** and will increment the error count.

'text-expression' can either be the [label of an internal error message](#) or the text of the message (up to 80 characters) you wish to display. **PostMacro** will search the error message labels for a match and if one is found, the text of the error message for this label will be used. Otherwise, the text expression entered with the command will be used. See the **PostWorks User Guide** for a complete description of error messages and their labels.

Example:

```
ERROR/'DIVZERO'  
ERROR/'Invalid command: ' + COMMAND
```

5.3.7 Forcing Output Of The Current Block - FORCE/NOW

FORCE/NOW

Some registers (G-codes, M-codes Spindle codes, etc.) are not immediately output, but wait for the next motion block before being output to the Control Tape. The *FORCE/NOW* command forces the output of any registers awaiting output in a single block.

FORCE/NOW is only valid with the **PWorks** post-processor, it will be ignored by **PMacro**.

5.3.8 Outputting User Defined Blocks - FORCE/BLOCK

Command Syntax: **FORCE/BLOCK, n**

The *FORCE/BLOCK* command outputs the [User Defined Block](#) specified by 'n' on the next output Control Tape block. User Defined Blocks are defined in the **MPost** utility and are usually output with certain sequences (*LOADTLs*, first motion block, sequence start block, etc.). 'n' can be in the range of 1-20.

FORCE./BLOCK is only valid with the **PWorks** post-processor, it will be ignored by **PMacro**.

5.3.9 Controlling The Output Of Registers - FORCE

Command Syntax: **FORCE/reg1[,val1],NOW[...] regn[,val],NOW**

NEXT	NEXT
MOTION	MOTION
ON	ON
OMIT	OMIT
OFF	OFF
NOMORE	NOMORE

The *FORCE* command forces or suppresses the output of selected registers. 'reg' specifies which register to output/suppress and can be either a numeric or symbolic representation of a physical or logical register. 'val' optionally defines the value for the register and should be specified only when forcing the output of the register. If 'val' is not specified, then the last output value for the register will be used.

PWorks will not check the validity of any of the codes output using the *FORCE* command, except for storing this value in the register. Therefore, any codes output that will change a mode or position on the machine, will not change that mode or position within **PWorks**. For example, if a G91 (incremental programming code) is output using the *FORCE* command, then a G90 (absolute programming code) will be reinstated on the next motion block, incremental mode will not be activated.

'*NOW*' will force the output of the Control Tape block after this register has been output. If you specify multiple registers on the *FORCE* command and want them to be output in a single block, then specify '*NOW*' only on the last register and '*NEXT*' on all of the preceding registers.

'*NEXT*' specifies that the register will be output with next block. '*MOTION*' outputs the register on the next motion block. '*ON*' forces the output of the register on all succeeding blocks until '*FORCE/reg,NOMORE*' is encountered.

'*OMIT*' suppresses the output of the register on the next block only. It will be output normally following the next output block. '*OFF*' suppresses the output of the register on all succeeding blocks until cancelled by '*FORCE/reg,NOMORE*'.

'*NOMORE*' cancels both '*ON*' and '*OFF*'. If a register value is specified with '*NOMORE*', then this value will be stored as the current value for the specified register.

When forcing the output of registers, *FORCE* acts similar to a runtime [User Defined Block](#). The *FORCE* command is only valid with the **PWorks** post-processor, it will be ignored by **PMacro**.

Example:

```
FORCE/-1,0,NEXT, 20,91,NEXT, 21,NEXT, 66,0,NOW
FORCE/#G,0,NEXT, #G2,91,NEXT, #Z2,0,NOW
```

5.3.10 Controlling The Output Order of Registers - REGORD

Command Syntax: **REGORD** / [**START**,] **r1**, [...] **rn**
NEXT

This command allows the user to change the output register order from within a Macro.

“*START*” specifies that the beginning of the register order is being defined, while “*NEXT*” will append the registers specified in this command will be appended to the registers specified in previous commands. “*START*” and “*NEXT*” are used since there can be a maximum of 50 parameters on a post-processor command, but there are 92 registers. If neither of these words is specified, then “*NEXT*” is assumed.

“r1, [...] rn” specifies the list of registers in the order in which they are to be placed in an output block. Only physical registers can be specified (#G0, #X1, etc.), logical registers are not allowed.

Example:

```
REGORD/START,#N,#G0,#G1,#G2,#G3,#G4,#G5
REGORD/NEXT,#X1,#X2,#Y1,#Y2#Z1,#Z2,#A2,#A3,#B2,#B3,#F1
REGORD/#R,#D,#H,#S,#T,#M0,#M1,#M2,#M3,#M4
```

PMacro will ignore this command.

5.3.11 Motion Generating Statements - FROM, GOTO, GOTOC

Command Syntax: **FROM** /**x1,y1,z1** [, **i1,j1,k1**] [...] **\$**
GOTO
GOTOC

xn,yn,zn [, **in,jn,kn**]

The *FROM* command outputs a *FROM* motion type record to the clfile. The *GOTO* command outputs a standard motion type record to the cl file. The *GOTOC* command output a continuation motion type record to the cl file.

‘x, y, z’ specify the cl point(s) to output and ‘i, j, k’ specify the tool axis vector(s) to associate with the cl point(s). The tool axis vector (i, j, k) can only be present when *MULTAX* is turned on in the input clfile.

When *MULTAX* is turned off, you must have a multiple of 3 values in the command (x, y, z). When *MULTAX* is turned on, you must have a multiple of 6 values (x, y, z, i, j, k). An error will occur and the command will be ignored if either of the previous conditions is not met.

You can specify up to 48 values with this command, by specifying a single value for each of the parameters. This equates to 16 cl points with *MULTAX* turned off and 8 cl points with multax turned on. You may also use a *Post variable* with a range subscript in combination with single values or in place of them. This allows for a maximum of 240 values to be specified with the command. The most common post variable to use is the *%CLPT* array.

Example:

```
FROM/X,Y,Z
GOTO/%CLSAV(1:3),I,J,K
GOTOC/%CLPT(1:%NARG*NPT)
```

5.3.12 Motion Register Setting Statement - POSCAL

Command Syntax: **POSCAL/x1,y1,z1[,i1,j1,k1]**

The *POSCAL* command calculates the output axes position based on the input tool end point and tool axis position. The calculated axis positions will not be output to the cl file or the [simulation file](#). However, all the corresponding motion output registers will be updated and can be referenced using the *%AXIS*, *%LINEAR*, *%ROTARY*, *%TLVEC*, *%XYZ* and *%REG* post-processor variables. The *POSCAL* command is valid only in a post-processor Macro.

The output of this command is very similar to the *FROM* command when ***MPost Menu 5.1.1: "Specify handling of FROM statement"*** is set to "2", i.e. "FROM is not output and will set the current values for each axis". However, when the MDF file is configured in this manner, the *FROM* will output a motion record to the [simulation file](#) where as the *POSCAL* will never output to the simulation file.

Caveat: There is no difference between the POSCAL and FROM command calculated results if the input unit is same as the output unit. However, the %Clpt data corresponding to the linear axes must be modified with the following logic before applying the %Clpt data to the POSCAL command..

If(%Units(1) == 1 & %Units(2) == 2) Then \$\$ Input Unit is Inches. Output Unit is MM.

 DEFINE/%Clpt(1),%Clpt(1)/25.4

 DEFINE/%Clpt(2),%Clpt(2)/25.4

 DEFINE/%Clpt(3),%Clpt(3)/25.4

Else if (%Units(1) == 2 & %Units(2) == 1) Then \$\$ Input Unit is MM. Output Unit is Inches.

 DEFINE/%Clpt(1),%Clpt(1)*25.4

 DEFINE/%Clpt(2),%Clpt(2)*25.4

 DEFINE/%Clpt(3),%Clpt(3)*25.4

Endif

5.3.13 Circular Motion Interpolation Statement - CIRCLE

Command Syntax: **CIRCLE/parms**

The CIRCLE command allows the user to output circular interpolation records from a Macro.

“parms” is a list of 27 values that correspond to the first 6 values of the *%CIRFLG* array and the first 21 values of the *%CIRPRM* array. The parameters can be entered individually on the command or as an array range. For example, the following Macro will output the circular interpolation record exactly as it is input.

```
CIRCLE/MACRO,0
      CIRCLE/%Cirflg(1:6),%Cirprm(1:21)
TERMAC
```

Notes:

Circular interpolation is always calculated internally as being in the XY-plane. The XYZ pointers defined in the *%CIRFLG*(1:3) array are used to place the circular interpolation on the programmed plane. For example, for XY-plane circular the values should be 1,2,3, for ZX-plane circular interpolation the values should be 3,1,2, and for YZ-plane circular the values should be 2,3,1. A value of 0 in *%CIRFLG*(3) states that this is a 3-axis circular interpolation record.

If an error is generated during the circular interpolation being calculated, then the move will be output in linear motion and the CIRCLE Macro will not be called.

No attempt to validate the parameters provided on the CIRCLE command is made. It is the programmer's responsibility to the integrity of the data input on the CIRCLE command.

5.3.14 Including An External File - INCLUDE

Command Syntax: **INCLUDE/file-name**

The *INCLUDE* command reads in all of the lines from 'file-name' into the current Macro definition file. The lines will be included at the location of the *INCLUDE* command. You may have nested *INCLUDE*s within included files up to ten levels deep.

5.3.15 Controlling The Listing File - LISTING

Command Syntax: **LISTING/ON**
OFF
PAGE

The *LISTING* command controls the output to the listing file. The listing file must have been enabled at runtime, using the *LISTING* option, for this command to have any effect.

'*ON*' enables all output to the listing file, including page headings, a text representation of the output clfile, *PRINT* statements, and error messages. '*OFF*' disables the output to the listing file, except for *PRINT* statements and error messages. '*PAGE*' causes the listing file to skip to a new page.

5.3.16 Referencing **PMacro** - MACHIN

Command Syntax: **MACHIN/PMACRO,n1[...]n10**

The *MACHIN/PMACRO* statement specifies which *Macro object file(s)* will be used to process the input clfile. This statement must be contained in the part program file and not in the *Macro* definition file, since the *Macro* object file will not be called until a *MACHIN* statement is processed.

The *MACHIN/PMACRO* statement should appear only once in the part program file, as **PMacro** will only process the first one, all others will be ignored and not output to the clfile. The *MACHIN/PMACRO* statement must be present when running the **PMacro** pre-processor standalone or an error will occur and the input clfile will not be processed.

MACHIN/PMACRO is not required when running the **PWorks** post-processor, as **PWorks** also contains the **PostMacro** processor. Use the *MACHIN/PWORKS* statement, as defined in the **PWorks** Reference Manual, instead.

'*n*' is used to specify the *Macro* object file, in the form *pmacro_n.OBJ*, and can be any positive number. You must specify at least one machine number and may specify up to ten. **PMacro** will process the input clfile and create a separate output file for each of the machine numbers specified. The *MACHIN runtime option* can be used to override the *MACHIN/PMACRO* statement.

Example:

```
MACHIN/PMACRO,4032,3023,3011
```

In the above example, the input clfile will be processed for the *Macro* object files *pmacro_4032.OBJ*, *pmacro_3023.OBJ*, and *pmacro_3011.OBJ*.

5.3.17 Multax Control In The Output Clfile - MULTAX

Command Syntax: **MULTAX/ON**
OFF

The *MULTAX* command, when contained in *Macro*, enables or disables the output of tool axis vectors to the output clfile. The *MULTAX* command contained in the part program file controls the inclusion of tool axis vectors in the input clfile.

By default, **PMacro** will use the *MULTAX* command in the part program file to control *MULTAX* in the output clfile. When a *MULTAX* command is encountered in a Macro, it will override any *MULTAX* command in the part program file in relation to the output clfile, but will have no bearing on controlling the tool axis vectors contained in the input clfile.

'ON' enables the output of tool axis vectors, 'OFF' disables the output of tool axis vectors. *MULTAX* is only valid with **PMacro** and has no effect on **PWorks**.

5.3.18 Output To Listing File - PRINT

Command Syntax: **PRINT/text-expression**

The *PRINT* command allows you to output text strings to the listing file. 'text-expression' may be any valid text expression, such as a text string, variable or equation. Up to 132 characters can be output to the listing file with this command.

Example:

```
PRINT/'$$ Define the new curve points.'
PRINT/'PT(' + RTOC(SUB,#1) + ') = POINT/' + PTDATA
```

5.3.19 Controlling The **PWorks** Print File - PRINTF

Command Syntax: **PRINTF/ON**
OFF
PAGE

The *PRINTF* command controls the output to the **PWorks** print file. The print file must have been enabled at runtime, using the *PRINT* option, for this command to have any effect.

'ON' enables all output to the print file, including all records defined in the [Print Descriptor File \(.pdf\)](#). 'OFF' disables all output to the print file. 'PAGE' causes the print file to skip to a new page and print the header record.

The *PRINTF* command is only valid with the **PWorks** post-processor, it will be ignored by **PMacro**.

5.3.20 Print File Output - PRINTF/RECORD

Command Syntax: **PRINTF/RECORD, n**

The *PRINTF/RECORD* command allows you to output a record defined in the [Print Descriptor File \(.pdf\)](#). 'n' specifies the record number to print can be in the range of 1-10. The record will only be output to the print file, not the listing file.

Use *PPRINT* command to output a text string to the print file. The *PRINTF* command is only valid with the **PWorks** post-processor, it will be ignored by **PMacro**.

5.3.21 Displaying PWorks Error Messages - PSTERR

Command Syntax: **PSTERR**

The *PSTERR* command is only valid in the [PSTERR Macro](#) and is used to output the post-processor error which called the *Macro*. If the *PSTERR Macro* has been defined and the *PSTERR* command is not encountered, then an [error message](#) will NOT be output.

In the following example, the 'Circle radius over maximum.' error message has been disabled. All other **PWorks** generated errors will be output.

Example:

```
PSTERR/MACRO,10
  IF (%ARG(1) == 'CIRMAXR') THEN
  ELSE
    PSTERR
  ENDIF
TERMAC
```

5.3.22 Automatic Documentation - REMARK

Command Syntax: **REMARK text**

The *REMARK* command is used by **PostComp** to create a documentation file for the input *Macro* file. When the [DOCUMENT](#) runtime option has been enabled, **PostComp** will output the text of the *REMARK* statement to the documentation file. 'text' is purely a text string, text expressions may not be specified.

5.3.23 External Text File Commands

Command Syntax: **FOPEN**/*fn*,*mode*,*filename*[,*ierr*]
FCLOSE/*fn*
FREW/*fn*
FREAD/*fn*,*data*,*nc*[,*ierr*]
FWRITE/*fn*,*data*[,*ierr*]

“*fn*” specifies the file number associated with the opened file. This value can be in the range of 0-5, allowing for 5 simultaneously opened files. If a value of 0 is specified, then **PostWorks** will automatically assign the next available file number to the opened file and if 0 is specified using a variable, then this variable will be updated with the assigned file number.

“*ierr*” can be specified on most file commands and will be set to non-zero if an error occurred during the file operation. If “*ierr*” is not specified, then a post-processor error will be generated. The values that *ierr* can return are as follows.

- 0 = File operation was successful.
- 1 = No such file.
- 2 = File is already open.
- 3 = File I/O error.
- 4 = Buffer too small. Data is truncated.
- 5 = File is not open.
- 6 = End of file encountered during read operation.
- 7 = Invalid file mode.
- 8 = Invalid file number.
- 9 = File already exists.

FOPEN opens the file specified by ‘filename’. If ‘filename’ consists of an asterisk followed by a file extension (i.e. *.txt), then the base filename of the punch file with the specified file extension will be used. The file can be opened using one of the following access modes as specified by the text string/variable mode.

- READ** = Open file in read only mode.
- WRITE** = Open file in write mode. An existing file can be open.
- APPEND** = Open an existing file in write mode and position at the end of the file.
- NEW** = Open file in write mode. The file cannot already exist.
- TEMP** = Open a temporary file for writing. This file will be automatically deleted when it is closed.

FCLOSE closes the specified file. **FREW** positions the file pointer at the beginning of the specified file.

FREAD reads a text record from the file. ‘data’ will receive the record text and ‘nc’ will return the number of characters read.

FWRITE writes out the text specified by 'data' to the file.

The following example will copy the contents of an input file to an output file in its entirety.

```
Real fn,err,opnd(2),n,nc
Char lbuf(80),sbuf(80),acs(20),file(20)
$$
$$...Initialize routine
$$
    opnd(1) = 0
    opnd(2) = 0
$$
$$...Open input file
$$
    acs = "READ"
    file = "input.txt"
    FOPEN/1,acs,file,err
    If (err == 6) JUMPTO/endit
    If (err <> 0) Then
        lbuf = "#Error " + rtoc(err,#N) + " opening input file."
        PRINT/lbuf
        JUMPTO/endit
    Endif
    opnd(1) = 1
$$
$$...Open output file
$$
    fn = 0
    FOPEN/fn,"WRITE","output.txt",err
    If (err <> 0) Then
        lbuf = "#Error " + rtoc(err,#N) + " opening output file."
        PRINT/lbuf
        JUMPTO/endit
    Endif
    opnd(2) = 1
    FWRITE/fn,"-- Start of Output File --"
$$
$$...Loop to read/write files
$$
    n = 0
id1:
    FREAD/1,lbuf,nc,err
    If (err == 6) Jumpsto/endit
    If (err <> 0) Then
        lbuf = "#Error " + rtoc(err,#N) + " reading input file."
        PRINT/lbuf
        JUMPTO/endit
    Endif
    n = n + 1
    sbuf = "Line " + rtoc(n,#N) + ": " + lbuf
    FWRITE/fn,sbuf,err
    If (err <> 0) Then
```

```

        lbuf = "#Error " + rtoc(err,#N) + " writing output file."
        PRINT/lbuf
        JumpTo/endit
    Endif
    JUMPTO/id1
$$
$$...End of Macro
$$
endit:
    If (opnd(1) == 1) FCLOSE/1
    If (opnd(2) == 1) FCLOSE/fn
TERMAC

```

5.3.24 Clfile Read/Write Commands

Command Syntax: **CLREAD/irec,ipt[,ierr]**
CLFIND/irec,ipt,word_list[,ierr]
CLWRIT[/ierr]

These commands use the *%CLDATA* array to communicate with the calling Macro. This array is similar to the *%ARG* Macro Argument array in that both real values and text strings are stored in the array. The *%NCLD* post-processor variable contains the number of values stored in the *%CLDATA* array.

The first four values of the *%CLDATA* array contain the following information.

```

%Cldata(1) = ISN
%Cldata(2) = Clfile Record Number
%Cldata(3) = Record Type
%Cldata(4) = Record Subtype

```

Text strings that represent the clfile record type (or major word for post-processor commands) are stored in the Record Type field. The Record Subtype field also contains the major word text string for type 2000 records. Refer to the Clfile Format document for an in-depth description of the various clfile record types.

CLREAD will read the specified clfile record and load the *%CLDATA* array with the clfile record values. *'irec'* is the clfile record number to read and *'ipt'* is the pointer within the physical clfile record where the logical record begins. *'irec'* and *'ipt'* will be updated to point to the next clfile record after the **CLREAD** command is processed. *'ierr'* will be set to non-zero if an error occurred while reading the clfile. If *'ierr'* is not specified, then a post-processor error will be generated.

The current clfile record number can be referenced using the *%FIREC* post-processor variable. The logical record position can be referenced using the *%FIPT* variable. When using the clfile commands to read the clfile it is not recommended to use the *%FIREC* and *%FIPT* variables

directly in the command, because they will be updated to point to the next clfile record, meaning that when normal processing of the clfile resumes any records processed by the **CLREAD** and **CLFIND** routines will be skipped and the clfile record following these records will be the next record processed.

Following is a typical method for performing a look-ahead sequence for the next *LOADTL* command.

```

        frc = %FIREC
        fpt = %FIPT
id1:
    CLREAD/frc,fpt,ierr
    If (ierr == 0 & %Cldata(3) <> 14000) Then
        If (%Cldata(3) == "LOADTL") Then
            sbuf = "Next Tool = " + rtoc(%Cldata(5),#T)
            PPRINT/sbuf
        Else
            Jump to/id1
        Endif
    Else
        PPRINT/"Error Could not find LOADTL."
    Endif

```

CLFIND will continue to read from the clfile until one of the entity specified in the word_list is found. A maximum of 20 entities is allowed in the specified word_list with each one separated by a comma character. They can be specified as either the clfile record type to find (1000, 2000, 5000, etc.) or the post-processor major word to find (1055, 1041, etc.). It can be a numeric value (5000), a vocabulary word (GOTO), or a text string/variable ("LOADTL"). The "irec", "ipt", and "ierr" variables all behave the same as in the **CLREAD** command.

Following is sample code using the **CLFIND** command that mimics the sample code given above for the **CLREAD** command.

```

        frc = %FIREC
        fpt = %FIPT
    CLFIND/frc,fpt,LOADTL,ierr
    If (ierr == 0) Then
        sbuf = "Next Tool = " + rtoc(%Cldata(5),#T)
        PPRINT/sbuf
    Else
        PPRINT/"Error Could not find LOADTL."
    Endif

```

The **CLWRIT** command is for use in **PMacro** and writes out the contents of the %CLDATA array to the output clfile. **CLWRIT** is ignored by **PWorks**. The %NCLD variable must be set to the number of values to write out. It is the user's responsibility to make sure that the %CLDATA array contains valid data prior to writing it out to the clfile as no check by **PMacro** will be made.

5.3.25 Syntax Checking - SYNTAX

Command Syntax: **SYNTAX/minor-word[, [...]minor-word]**

number	number
range	range

The *SYNTAX* command is used to verify that the post-processor command that called the current *Macro* is syntactically correct. The **minor words**/values in the calling command will be checked against the minor words/values in the *SYNTAX* command. If there is a conflict, the **%ERROR** *Post variable* will be set to one, otherwise it will be set to zero. An error message will not be output by **PostMacro**; you must use the **ERROR** command if you want to output an error message.

The *SYNTAX* command parameters must be in the same position within the command as the parameters in the post-processor command that called the *Macro* are expected to be in. The first parameter in the *SYNTAX* command will be checked against the first parameter in the post-processor command, the second against the second, etc.

You can specify minor words, numbers and ranges. Variables and expressions may not be used. A range has the format 'min-max', where 'min' is the minimum numeric value acceptable and 'max' is the maximum acceptable value. The minus character (-) defines the parameter as being a range.

Post-processor commands will usually accept different minor words for any one parameter. For example, **COOLNT** may accept *ON*, *OFF*, *FLOOD* and *MIST* for the first parameter. In this situation, all of the words/values that are valid for this parameter should be enclosed in parenthesis.

When there are more parameters in the post-processor command than in the *SYNTAX* command, then the parameters that are present will be checked against the *SYNTAX* parameters. If the command syntax requires a fixed number of parameters, then the **%NARG** variable should be checked in addition to the *SYNTAX* command.

Example:

SYNTAX/(ON,OFF,LEFT,RIGHT),(XYPLAN,YZPLAN,ZXPLAN),0-64

Accepts the command syntax:

```
CUTCOM/ON      ,XYPLAN,N    $$ N can be in the range 0 to 64
              OFF  YZPLAN
              LEFT  ZXPLAN
              RIGHT
```

The *SYNTAX* command is assumed to be in the *CUTCOM* macro in the above example. The following examples show valid and invalid commands when checked with the *SYNTAX* command:

SYNTAX/(ON,OFF),8,1,1-9999

Valid: CHECK/ON,8,1,4500

Invalid: CHECK/OFF,7,1,40 \$\$ The 2nd parameter must be 8.

SYNTAX/25-4000,(CLW,CCLW),(HIGH,LOW)

Valid: SPINDL/300,CLW,LOW

Invalid: SPINDL/4200,CLW,HIGH \$\$ The 1st parameter must be in
 \$\$ the range of 25-4000.

SYNTAX/0-99999,LENGTH,0-20

Valid: LOADTL/10,LENGTH,4.6

Invalid: LOADTL/6,LENGTH,4.5,OFSETL \$\$ There are more
 \$\$ parameters in the
 \$\$ command than in the
 \$\$ SYNTAX statement.

CHAPTER 6 Program Control Statements

6.1 Introduction

Statements within a *Macro* are usually executed in the order in which they were written. However, you may use control statements to transfer execution to another statement in the same *Macro*. Control statements supported are the *JUMPTO/label* command, logical *IF* statements (including *IF-THEN-ELSE* structures) and repetitive *DO* loops.

This chapter also discusses labels and the *CONTINUE* statement. Although these are not control statements in themselves, they are usually used in conjunction with control statements.

6.2 Labels

Labels can precede any statement in a *Macro*, except for the *Macro definition statement*, and are used as program markers for the *JUMPTO* command and *DO* loop. Labels are recognized as a text string, up to 24 characters in length and terminated with a colon (:). The characters can be any alphanumeric character, but the first character must be a letter. Labels that are entered in a command (*JUMPTO*, *DO*) must not contain a colon.

Labels are local to a *Macro*, therefore different *Macros* can have the same labels.

Example:

Valid	Invalid	
-----	-----	
ID1:	ID1	\$\$ Missing colon.
I1234567:	123:	\$\$ The 1st character must be a letter.
JUMPTO/ID1	JUMPTO/ID1:	\$\$ Colon not allowed.

6.3 The JUMPTO Statement

Command Syntax: **JUMPTO/label**

The *JUMPTO* command unconditionally transfers program control to the statement located at the specified label. The label in the *JUMPTO* command must not contain a colon.

Example:

```
JUMPTO/ID1
.
.
.
ID1: SPINDL/%ARG(1),CLW
```

In the above example, all of the statements between the *JUMPTO* command and the label ID1 will be skipped. The *SPINDL* command will be executed directly after the *JUMPTO* command.

6.4 If Tolerancing (IFTOL)

Command Syntax: **IFTOL/tol**

Real numbers stored in computers, though accurate to more than ten places, constantly have rounding problems. A number stored as 15.0 and another as 14.9999999999 would in most cases be considered equal, but computer comparisons must be exact and these numbers are considered different.

The *IFTOL* statement defines a tolerance to use when determining if two numbers are equal in an IF statement. It allows the programmer to loosen up the constraints on the equality between two numbers. 'tol' specifies the maximum difference between two numbers that determines if they are equal or not.

Example:

```
IFTOL/.0
IF (.0001 == 0.) ...    $$ Not Equal
IFTOL/.0001
IF (.0001 == 0.) ...    $$ Equal
```

6.5 Logical IF Statements

Command Syntax: **If (e) command**

The logical *IF* statement evaluates an expression and conditionally executes a single command based on this evaluation. 'e' is a valid numeric or text expression to be evaluated. If 'e' is true, then 'command' will be executed and program control will continue with the next statement. If 'e' is false, then program control will bypass 'command' and skip to the next statement.

Example:

```
IF (%ARG(1) == 'CLW') COOLNT/FLOOD
IF (NUM >= 3 & NUM <= 6) JUMPTO/ID1
```

6.6 IF-THEN-ELSE Structures

Command Syntax: **IF (e1) Then**

```
    .  
    .  
    .  
    ELSE IF (e2) THEN  
    .  
    .  
    .  
    ELSE  
    .  
    .  
    .  
    ENDIF
```

The *IF-THEN_ELSE* structure conditionally executes groups of commands based on one or more expression evaluations. The only required commands in a block *IF* structure is the beginning *IF* statement and the *ENDIF* statement. The *ELSE statements* are optional.

The statements contained between *IF/ELSE* and the next *ELSE/ENDIF* statements are considered a block, and will be executed when the expression contained in the preceding conditional is true. No more than 1 block per *IF* structure will be executed on each entry.

The *IF THEN* statement begins a block *IF* construct. The block following it is executed if the *IF THEN* conditional is true.

The *ELSE IF THEN* statement is an optional statement within a block *IF* construct. The block following it is executed only when all of the previous *IF THEN* and *ELSE IF THEN* conditionals are false, and its conditional is true. You can have an unlimited number of *ELSE IF THEN* statements in a block *IF* structure.

The *ELSE* statement is also an optional statement. The block following it will be executed only when all of the previous *IF THEN* and *ELSE IF THEN* conditionals are false. There can be only one *ELSE* statement per block *IF* structure.

The *ENDIF* statement terminates the block *IF* construct. You may nest up to ten block *IF* structures and [DO loops](#), but *DO* loops must be wholly contained within a single block of the *IF* structure. They may not cross *ELSE* and *ENDIF* boundaries.

Example:

```
    IF (%ARG(1) == 'CLW') THEN  
        COOLNT/FLOOD
```

```

        SPINDL/ON,CLW
    ELSE IF (%ARG(1) == 'CCLW') THEN
        COOLNT/MIST
        SPINDL/ON,CCLW
    ELSE IF (%ARG(1) == 'OFF') THEN
        COOLNE/OFF
        SPINDLE/OFF
    ELSE
        ERROR/'INVSYN'
    ENDIF

```

6.7 The CONTINUE Statement

Command Syntax: **CONTINUE**

The *CONTINUE* statement is a 'no operation' command and is usually used as the terminator for a *DO* loop.

6.8 Repetitive DO Loops

Command Syntax: **DO label i=st,en,inc**

The *DO* statement defines the start of a repetitive *DO* loop. A *DO* loop executes a group of statements repeatedly, until the condition specified with the parameters on the *DO* statement is met.

'label' is the label of a statement which is used as the end of the *DO* loop. This statement will be included as the last statement in the *DO* loop. 'label' cannot contain a colon ":".

'i' is a real variable which will be used to control the *DO* loop. 'i' will be set to 'st' at the beginning of the loop and will be incremented by 'inc' at the end of the loop. The *DO* loop will continue until the value of 'i' is greater than 'en'.

'st' is the starting value for 'i' and can be a real variable, *Post variable* or number.

'en' is the ending value of the *DO* loop control variable 'i'. 'i' will actually be greater than 'en' at the end of the loop. 'en' can be a real variable, *Post variable* or number.

'inc' specifies the increment for the *DO* loop control variable 'i' and can be a real variable, *Post variable* or number.

Although not necessary, it is recommended that you terminate a *DO* loop with a *CONTINUE* statement. You can also terminate the execution of a *DO* loop prematurely by using a *JUMPTO*

command within the *DO* loop. It is strongly recommended that you do not pass control into a *DO* loop from outside the loop, as the results will be unpredictable.

You may nest *DO* loops within other *DO* loops and block *IF* structures, but the nested *DO* loop must be wholly contained in the outer *DO* loop or block *IF* structure. A nested *DO* loop may also have the same terminating label as its outer *DO* loop. You may nest up to ten block of *DO* loops and *IF* structures.

Example:

```
DO ID1 i=1,10,1
    GOTO/X(I),Y(I),Z(I)
ID1: CONTINUE
```

6.9 The BREAKF Statement

Command Syntax: **BREAKF**

The BREAKSF statement is used to skip the remainder of the current IF block and passes control to the statement following its ENDIF statement. If multiple levels of IF blocks are being defined it only breaks out of the innermost block.

6.10 The ENDDO Statement

Command Syntax: **ENDDO**

The ENDDO statement is used to skip the remainder of the current DO loop and passes control to the statement following the loop's ending label. If nested DO loops are being defined it only terminates the innermost loop

6.11 The ENDMAC Statement

Command Syntax: **ENDMAC**

The ENDMAC statement is used to complete the processing of the active Macro prematurely as if the TERMAC statement was reached.

APPENDIX A Command/Variable Summary

A.1 Post Variable Summary

%ARG(n)	%AXIS(10)	%AXSDLS(10)	%AXSDLT(10)
%AXSMAX(10)	%AXSMIN(10)	%BRKPRM(3)	%CIRFLG(8)
%CIRPRM(25)	%CLDATA(n)	%CLPT(n)	%CLREC
%CLSAV(6)	%CUTTER(7)	%ERROR	%ERRPC
%FEED(6)	%HOME(10)	%ISN	%KPOSMP(4000)
%LINDLS(6)	%LINDLT(6)	%LINEAR(6)	%LINENO
%LINMAX(6)	%LINMIN(6)	%MACHIN	%MCHTIM
%MOVDIS(4)	%MOVTIM	%MULTAX	%NARG
%NCLD	%NTOOL	%NUMERR	%NUMIER
%NUMFAT	%NUMWRN	%PAGENO	%PARTNO
%PI	%POSMAP(4000)	%PSTNAM	%REG(92)
%REGST(92)	%REGEN(92)	%ROTARY(4)	%ROTDLS(4)
%ROTDLT(4)	%ROTMAX(4)	%ROTMIN(4)	%RPM
%SELCTL	%SEQTIM	%SFM	%SIMUL
%TAPLEN	%TIMSCL	%TLEN(120)	%TLNO(120)
%TLTIM(120)	%TLVEC(3)	%TOOL	%UNITS(2)
%USER(20)	%XYZ(12)	%XYZDLS(12)	%XYZDLT(12)
%XYZMAX(12)	%XYZMIN(12)		

A.2 Register Label Summary

Physical Registers

A1	A2	A3	B1	B2	B3	C1	C2	C3	C4	C5	C
D	E	F1	F2	F3	G0	G1	G2	G3	G4	G5	G6
G7	G8	G9	GA	H	I1	I2	J1	J2	K1	K2	L
M0	M1	M2	M3	M4	M5	M6	M7	M8	M9	MA	N
O	P	Q	R	S	T	U1	U2	V1	V2	W1	W2
X1	X2	Y1	Y2	Z	Z2	AA	AB	AC	AD	AE	AF
AG	AH	AI	AJ	AK	AL	AM	AN	AO	AP	AQ	AR
AS	AT	AU	AV	AW	AX	AY	AZ				

Logical Registers

A	B	C	@	F	G	M	U	V	W	X	Y
Z	\$										

A.3 Functions Summary

ABS (n)

Returns the absolute value of the real number 'n'.

ACOS (n)

Returns the angle, in degrees, whose cosine is 'n'.

ASIN (n)

Returns the angle, in degrees, whose sine is 'n'.

ATAN (n)

Returns the angle, in degrees, whose tangent is 'n'.

CLBRIEF

Returns the name of the input clfile that does not include the device and directory name.

CLEXT

Returns the extension of the input clfile only.

CLNAME

Returns the name of the input clfile that includes the device and directory name.

COMMAND

Returns a text string representation of the post-processor command that called the current Macro.

COMPUTER

Returns the name of the computer the program is running on.

COS (n)

Returns the cosine of the angle 'n'. 'n' is in degrees.

CTOR (t, #f)

Converts a text string 't' to a number, using the format descriptor '#f'.

DATE

Returns a text string representation of the current date, in the format 'dd-mmm-yyyy'.

ERRTXT (t)

Returns the text of the error message whose label is 't'.

FMTCOD (n, #f)

Converts a real value 'n' to a text string, using the format descriptor '#f'. The output text string will also contain the specified register's beginning and ending strings.

INDEX (t1, t2)

Returns the location of the first occurrence of the text string 't2' in the text string 't1'.

INT (n)

Returns the integer portion of the real number 'n'.

LOCATE (a (m) , n)

Returns the position in the real array 'a' of the first occurrence of the value 'n' starting from position m.

PCHBRIEF

Returns the name of the output punch file that does not include the device and directory name.

PCHFILE

Returns the name of the output punch file that includes the device and directory name.

PCHNAME

Returns the name of the output punch file without the device, directory name, and file extension.

RINDEX (t1, t2)

Returns the location of the last occurrence of the text string 't2' in the text string 't1'.

RTOC (n, #f)

Converts a real value 'n' to a text string, using the format descriptor '#f'.

SIN (n)

Returns the sine of the angle 'n'. 'n' is in degrees.

SQRT (n)

Returns the square root of 'n'.

STRLEN (text_var)

Returns the length of a text string.

TAN (n)

Returns the tangent of the angle 'n'. 'n' is in degrees.

TIME

Returns a text string representation of the current time, in the format 'hh:mm:ss'.

USERNAME

Returns the name of the user (computer log in name) who is running the program.

A.4 Special Macro Summary

ERROR/MACRO, 10

The *ERROR Macro* is automatically called each time an error occurs in the *Macro Processor*. *%ARG(1)* will contain the label of the error message which caused the *Macro* to be called.

CIRCLE/MACRO, 0

This is a circular interpolation Macro that is called when a circular interpolation record is processed.

FINI/MACRO, 0

The *FINI Macro* is automatically called when the *FINI* record is encountered in the clfile.

G\$MAIN

All statements prior to the first *Macro* definition will be placed in the *G\$MAIN Macro*, which is automatically defined by **PostMacro**. All user defined variables contained in this *Macro* are considered global and can be accessed by all other *Macros* defined. **PostMacro** will call the *G\$MAIN Macro* at the beginning of the clfile, prior to all other processing.

GOTO/MACRO, 2

The *GOTO Macro* is called whenever a motion record is encountered in the input clfile. *%ARG(1)* contains the number of points in the motion record, *%ARG(2)* will contain the motion type, and the *%CLPT Post variable* array will contain the actual point data.

OPTION/MACRO, n

The main purpose of the *OPTION Macro* is to trap any 'major/OPTION' commands in the part program which were meant for a custom post-processor. *%ARG(1)* will contain the major word of the command which called the *Macro*.

PSTERR/MACRO, 10

The *PSTERR Macro* is automatically called each time an error is generated by **PWorks**. *%ARG(1)* will contain the label of the error message which caused the *Macro* to be called.

TAPBRK/MACRO, 0

The *TAPBRK Macro* allows the programmer to define their own tape break sequences and will be called whenever the Control Tape is broken up into multiple sections due to a *BREAK* sequence. The sequence of commands contained in this *Macro* will override the normal processing of a tape break.

XFORM/MACRO, 11

The XFORM Macro is called whenever a transformation enable/disable block is output. If an XFORM Macro is defined, then **PostWorks** will call this Macro and will make no attempt to output the transformation codes, it is the user's responsibility to output the correct transformation blocks from within the XFORM Macro. In fact, the main purpose of the XFORM Macro is to output the transformation blocks.

A.5 Post-processor Command Summary

APPEND text

Appends a text string to the end of the following block. Similar to the *INSERT* command.

BREAK/PCHFILE

Performs an End-of-Tape sequence, closes the active punch file, opens a new file, and outputs a Start-of-Tape sequence. *BREAK/PCHFIL* is usually used in the *TAPBRK Macro*.

BREAKF

Skips the remainder of the current IF block and passes control to the statement following its ENDIF statement. If multiple levels of IF blocks are being defined it only breaks out of the innermost block.

CHAR var1 [(dim11 [, dim12 [, dim13]])] [...]

Declares text variables which will be used in the program. All variables used in the program have to be declared first. Text variables can be defined as a single/double/third dimensional array.

CLFIND/irec, ipt, word_list [, ierr]

Read the clfile until the one of the record types specified by word_list (20 maximum) is found.

CLREAD/irec,ipt[,ierr]

Read the specified clfile record and load the %CLDATA array with the clfile record values.

CLWRIT[/ierr]

Writes out the contents of the %CLDATA array to the output clfile.

CONTINUE

A 'no operation' command that is usually used as the terminator for a *DO* loop.

DEFINE/post_var1,val1[...]post_varn,valn

Assigns a value to a *Post variable*. 'post_var' is the *Post variable* that you want to change and 'val1' is the value you want stored in it.

DISABLE/macro

Disables a macro from being called when it normally would be.

DO label i=st,en,inc

Defines the start of a repetitive *DO* loop. 'label' is the label of a statement that is used as the end of the loop. 'i' is the *DO* loop control variable, 'st' is the beginning value for 'i', 'en' is the ending value, and 'inc' is the value that 'i' will be incremented by at the end of each pass through the loop.

ELSE

Defines the default block of an *IF-THEN-ELSE* structure, which is executed when all other conditions in the structure are false.

ELSE IF (e) THEN

An optional statement in an *IF-THEN-ELSE* structure that defines a secondary conditional in relation to the *IF* statement and previous *ELSE* statements. The block following the *ELSE* statement will be executed only when all of the previous conditionals are false and this conditional is true.

ENABLE/macro

Allows a *Macro* to be called. This is the default condition.

ENDIF

Defines the end of an *IF-THEN-ELSE* structure.

ENDDO

Skips the remainder of the current DO loop and passes control to the statement following the loop's ending label. If nested DO loops are being defined it only terminates the innermost loop.

ENDMAC

Prematurely completes the processing of the active Macro as if the **TERMAC** statement was reached.

ERROR/text-expression

Outputs an error message in the same format as error messages generated by **PostMacro** and increments the error count.

#f = #m

Defines a format descriptor using the attributes of another format descriptor.

```
#f = Ll.r,Mm.n,S+-  
      T          *  
      F  
      D
```

Defines a format descriptor using a parameter expression. The floating point format, number of digits to the left of the (implied) decimal point, the number of digits to the right, the minimum number of digits on either side of the decimal point, and whether or not a sign is included in the output of a number are specified.

FCLOSE/fn

Closes the specified external file referenced by 'fn'.

FOPEN/fn,mode,filename[,ierr]

Open the external file “filename” with the access mode specified by “mode” and assign the file reference number “fn”.

FORCE/NOW

Forces the output of any registers awaiting output along with an End-of-block code.

FORCE/BLOCK,n

Forces out the User Defined Block specified by ‘n’.

FORCE/reg1[,val1],NOW	[...] regn[,valn],NOW
NEXT	NEXT
MOTION	MOTION
ON	ON
OMIT	OMIT
OFF	OFF
NOMORE	NOMORE

Forces or suppresses the output of selected registers. ‘NOW’, ‘NEXT’, ‘MOTION’, and ‘ON’ force the output of the register. ‘OMIT’ and ‘OFF’ suppress the output of the register. ‘NOMORE’ cancels both ‘ON’ and ‘OFF’.

FREW/fn

Rewind the external file referenced by “fn” to the beginning of the file.

FREAD/fn,data,nc[,ierr]

Read the external file referenced by “fn” and put into “data” variable.

FWRITE/fn,data[,ierr]

Write the content of the “data” variable to the external file referenced by “fn”.

FROM/x1,y1,z1[i1,j1,k1]

Outputs a *FROM* statement to the clfile.

GOTO/x1,y1,z1[,i1,j1,k1][...]xn,yn,zn,[in,jn,kn]

Outputs a motion record to the clfile.

GOTOC/x1,y1,z1[,i1,j1,k1][...]xn,yn,zn,[in,jn,kn]

Outputs a motion continuation record to the clfile.

IF (e) command

Evaluates a numeric or test expression and executes the 'command', if the expression is true.

IF (e) THEN

Defines the beginning of an *IF-THEN-ELSE* structure. The block following the *IF* statement is executed if the condition 'e' is true. Otherwise, control is passed to the next *ELSE* or *ENDIF* statement.

IFTOL/tol

Defines the tolerancing factor to be used in *IF* statements when comparing two numbers for equality.

INCLUDE/file-name

Includes the specified file into the current *Macro* definition file.

JUMPTO/label

Unconditionally transfers program control to the statement located at the specified label. 'label' must not contain a colon.

label:

A label can be 1 to 24 characters in length and must be terminated by a colon ":" when used as a program marker. Labels can contain any alphanumeric character, but the first character must be a letter.

LISTING/ON
OFF
PAGE

Controls the output to the listing file. 'ON' enables all output to the listing file, 'OFF' disables the output to the listing file, except for *PRINT* statements and error messages, and 'PAGE' causes the listing file to skip to a new page.

MACHIN/PMACRO,n1 [...] n10

Defines the machine number(s) of the macro object files to process the input clfile for. The macro object file(s) have the format: pmacro_n.OBJ.

major/MACRO,narg

Begins the definition of a *Macro*. 'major' is a **major word** that defines which post-processor command will call this *Macro*. 'narg' is the maximum number of parameters that can be passed to the *Macro*.

major-word [/parameter-list]

Standard post-processor command, with a major word and an optional list of parameters separated by commas.

major-word text

Text post-processor command (*INSERT*, *LETTER*, *PARTNO*, *PPRINT*) with 132 characters of text.

major-word/text-expression

Text post-processor command with a text expression describing the text string.

MULTAX/ON
OFF

Controls the output of tool axis vectors with motion records to the output clfile.

POSCAL/x1,y1,z1[,i1,j1,k1]

The *POSCAL* command calculates the output axes position based on the input tool end point and tool axis position. The calculated axis positions will not be output and can be

referenced using the *%AXIS*, *%LINEAR*, *%ROTARY*, *%TLVEC*, *%XYZ* and *%REG* post-processor variables. The *POSCAL* command is valid only in a post-processor Macro.

PRINT/text-expression

Outputs a text string to the listing file.

PRINTF/ON

OFF

PAGE

Control the output to the **PWorks** print file. 'ON' enables all output to the print file, 'OFF' disables all output to the print file, except for error messages, and 'PAGE' causes the print file to skip to a new page.

PRINTF/RECORD,n

Prints the [Print Descriptor File](#) (.pdf) record number 'n'.

PSTERR

PSTERR is only valid in the *PSTERR Macro* and is used to output the **PWorks** error which called the *Macro*.

REAL var1 [(dim11[,dim12[,dim13]])][...]

Declares real variables which will be used in the program. All variables used in the program have to be declared first. Real variables can be defined as a single/double/third dimensional array.

REGORD/[START,]r1,[...]rn NEXT

Allows the user to change the output register order from within a Macro.

REMARK text

The text of the *REMARK* statement is output to a documentation file when the *DOCUMENT* runtime option has been specified with **PostComp**.

SYNTAX/minor-word[, [...]minor-word]

number	number
range	range

Verifies that the post-processor command that called the current *Macro* is syntactically correct and sets the Post variable *%ERROR* to one if there is a conflict. You can specify minor words, numbers, and ranges as valid parameters. You may also group multiple values as a single parameter by enclosing them within parenthesis.

TERMAC

Defines the end of a *Macro* definition.

v = e

Assigns the result of an expression to a real or text variable.

APPENDIX B Clfile Format

B.1 Introduction

Clfile records have a common format whether they are a post-processor command, motion record, cutter statement, etc. The first four values of each clfile record contain the following information

%Cldata(1) = ISN
%Cldata(2) = Clfile Record Number
%Cldata(3) = Record Type
%Cldata(4) = Record Subtype

The ISN is the Input Sequence Number and is the line number from the CAM system that generated the clfile record. The Clfile Record Number is typically a sequential value with the first record assigned 1, the second 2, etc. The Record Type is the type of clfile record. Following is a list of the types of records that can be found in a clfile.

1000	=	ISN Record
2000	=	Post-processor Command
2600	=	Stock Record
2601	=	Fixture Record
3000	=	Circle Record
5000	=	Motion Record
5200	=	Expanded Motion Record
5210	=	Motion Startup Record
5220	=	Motion Check Surface Record
6000	=	Cutter Command
7000	=	Tracut Record
7100	=	Cutter Display Record
7110	=	Cutter Geometry Record (9.6)
7120	=	Cutter Geometry Record
7200	=	SEQUNC Record
7300	=	Units Record
7400	=	Clfile Header Record
9000	=	Multax Record
14000	=	Fini Record

Each clfile record type is explained in the following sections. The Record Subtype value is dependent on the Record Type value and is described separately for each Record Type.

The clfile record data is stored in the *%CLDATA* array. This array is similar to the *%ARG* Macro Argument array in that both real values and text strings are stored in the array. The *%NCLD* post-processor variable contains the number of values stored in the *%CLDATA* array. Text strings can be stored in the Record Type and Record Subtype fields as well as for post-processor words in the type 2000 record. Refer to the section for each clfile record type to see which fields support a text string along with the real variable. If a text string is supported it will be designated in parenthesis after the variable. For example,

%Cldata(3) = 5000 (GOTO)

specifies that the word GOTO will be stored in *%Cldata(3)* along with the value 5000.

Please note that sometimes a single *%CLDATA* array item will internally contain multiple short and/or long integer values. In this case the short integer values will be marked with the **(In)** designator and the long integer values with the **(Jn)** designator, where n is the integer position within the *%CLDATA* value that holds the specified value. These integer values are not individually accessible to the user.

There are also some clfile records that have text embedded within the *%CLDATA* array, such as the Cutter Display Record, MACHIN card, etc. These text strings are also not accessible to the user and are marked with the **(T)** designator. The only text strings that are accessible are word values stored in *%Cldata(3)*, *%Cldata(4)*, and as minor words in a Post-processor Command type record. The text of a textual Post-processor command is also accessible.

B.2 ISN Record

Format:

<i>%Cldata(1)</i>	=	ISN
<i>%Cldata(2)</i>	=	Clfile Record Number
<i>%Cldata(3)</i>	=	1000 (ISN)
<i>%Cldata(4)</i>	=	0
<i>%Cldata(5:%Ncld)</i>	=	Call stack

Description: The ISN record contains the line number from the source file that generated the clfile record as well as the call and loop stack that was active when the clfile record was generated.

B.3 Post-processor Command

Format:

<i>%Cldata(1)</i>	=	ISN
<i>%Cldata(2)</i>	=	Clfile Record Number
<i>%Cldata(3)</i>	=	2000 (Major word)
<i>%Cldata(4)</i>	=	Major Word Value (Major word)
<i>%Cldata(5:n)</i>	=	Minor words/values

Description: Post-processor commands store the text representation of the major word in both the %Cldata(3) and %Cldata(4) fields. The minor words/values are stored in the %Cldata(5:%Ncld) fields and are referenced similarly to the %Arg(1:%Narg) array as far as the real and text storage is handled.

Textual post-processor commands (PARTNO, PPRINT, etc.) will store the entire text of the command in the %Cldata(5) field.

B.4 Stock / Fixture Records

Format:	%Cldata(1)	=	SN
	%Cldata(2)	=	Clfile Record Number
	%Cldata(3)	=	2600 (Stock) 2601 (Fixture)
	%Cldata(4)	=	30 (STL file) 331 (Remove chips) 340 (Box) 576 (Clone solid) 577 (Move solid) 620 (Cylinder) 627 (Torus) 631 (Sphere) 632 (Cone) 732 (Modify) 843 (Remove solid) 1075 (LOAD file)
	%Cldata(5) (J1)	=	Solid definition type
	%Cldata(5) (J2)	=	Solid ID
	%Cldata(6:14)	=	Solid parameters
	%Cldata(5) (J1)	=	Solid ID (Load)
	%Cldata(5) (J2)	=	Number of chars in filename (Load)
	%Cldata(6:n)	=	Load filename
	%Cldata(5) (J1)	=	Solid ID (STL)
	%Cldata(5) (J2)	=	Number of chars in filename (STL)
	%Cldata(6) (J1)	=	STL Units
	%Cldata(7:14)	=	STL filename
	%Cldata(5) (J1)	=	Solid ID (CLONE)
	%Cldata(5) (J2)	=	Solid ID to be cloned
	%Cldata(6) (J1)	=	Number of copies
	%Cldata(5) (J1)	=	Number of entities to move (MOVE)
	%Cldata(6:17)	=	Matrix to move solids through
	%Cldata(18) (T)	=	Matrix name
	%Cldata(19:n)(J)	=	Solid Ids to move
	%Cldata(5) (J1)	=	Number of entities to remove (REMOVE solid)
	%Cldata(5:n) (J)	=	Solid Ids to remove
	%Cldata(5) (J1)	=	Number of entities to modify (MODIFY)

%Cldata(5) (J2)	=	Color
%Cldata(6) (J1)	=	Visibility
%Cldata(6) (J2)	=	Translucency
%Cldata(7) (J1)	=	Active
%Cldata(8)	=	Tolerance
%Cldata(9:n)(J)	=	Solid Ids to modify

Description: Contains **NCL/IPV** Stock (2600) and Fixture (2601) commands. The data stored in the clfile record differs depending on the Stock or Fixture command processed as shown above.

B.5 Circular Record

Format:	%Cldata(1)	=	ISN
	%Cldata(2)	=	Clfile Record Number
	%Cldata(3)	=	3000 (CIRCLE)
	%Cldata(4)	=	2
	%Cldata(5:7)	=	Circle center point
	%Cldata(8:10)	=	Circle axis
	%Cldata(11)	=	Circle radius

Description: The circular record contains a circle definition and precedes motion that was created using a circle or cylinder as the drive surface. The motion record (Type 5000/5200, Subtype 5) directly following and all continuation motion records (Subtype 6) belong to the circular record. The next non-motion record or non-continuation motion record (Type 5000/5200, Subtype 3 or 5) terminates the circular record.

B.6 Motion Record

Format:	%Cldata(1)	=	ISN
	%Cldata(2)	=	Clfile Record Number
	%Cldata(3)	=	5000 (GOTO)
	%Cldata(4)	=	3 (FROM record)
			5 (Standard motion)
			6 (Continuation record)
	%Cldata(5:%Ncld)	=	Tool centerline data

Description: This motion record is created by the simple and automated machining routines, such as GOTO, PROFIL, POCKET, etc., which do not calculate contact points and forward vectors. If an expanded clfile is not used as input, then all Motion Records will be of type 5000. Each record will contain at least one tool end point and optionally a tool axis vector if MULTAX is enabled. A maximum of twenty cl points can be in a single record. The actual number of points in the record can be

determined by the equation '(%NCLD-4) divided by 3 (MULTAX/OFF) or 6 (MULTAX/ON)'.

B.7 Expanded Motion Record

Format:	%Cldata(1)	=	ISN
	%Cldata(2)	=	Clfile Record Number
	%Cldata(3)	=	5200 (GOTO)
	%Cldata(4)	=	1 (Positioning move) 2 (Departure move) 4 (Approach move) 5 (Standard motion) 6 (Continuation record)
	%Cldata(5:7)	=	Tool end point
	%Cldata(8:10)	=	Tool axis vector
	%Cldata(11:13)	=	Forward direction
	%Cldata(14:16)	=	Part surface contact point
	%Cldata(17:19)	=	Part surface normal vector
	%Cldata(20:22)	=	Drive surface contact point
	%Cldata(23:25)	=	Drive surface normal vector

Description: This motion record is created by the standard GOFWD style **NCL** motion commands. Each record will contain at least one tool end point, tool axis vector, forward vector, part surface contact point and normal vector, and the drive surface contact point and normal vector. A maximum of twenty cl points can be in a single record. The actual number of points in the record can be determined by the equation '(%NCLD-4) divided by 21'. The second cl point will start in array item 26, the third in array item 47, etc.

The 'subtype' parameter contains the motion type for this record. It describes the generated motion as Positioning (1), Departure (2), Approach (4), Cutting (5), or a Continuation of the previous motion record (6). Following is a list of motion types and the commands that can generate each type.

<u>Motion Type</u>	<u>Command</u>
(1) Positioning	GO Auto-Start
(2) Departure	FEDRAT/OUT
(4) Approach	FEDRAT/AT
(5) Cutting	GOFWD GOLFT GORGT
(6) Continuation	(All)

The FROM and GOTO commands generate type 5000 motion records. The Advanced Machining routines create a variety of motion records.

The part surface and drive surface normal vectors will always point away from the respective surface towards the cutter.

B.8 Motion Startup Record

Format:

Format:%Cldata(1)	=	ISN
%Cldata(2)	=	Clfile Record Number
%Cldata(3)	=	5210 (FROM)
%Cldata(4)	=	3
%Cldata(5:7)	=	Tool end point
%Cldata(8:10)	=	Tool axis vector
%Cldata(11:13)	=	Forward direction
%Cldata(14:16)	=	Part surface contact point
%Cldata(17:19)	=	Part surface normal vector
%Cldata(20:22)	=	Drive surface contact point
%Cldata(23:25)	=	Drive surface normal vector

Description: This motion record is created by the GOFWD type motion commands. It contains the startup position for the current move. The startup position will contain the current tool position, tool axis vector, the initial forward vector for this motion, the part surface contact point and normal vector, and the drive surface contact point and normal vector. The tool startup position is usually the final position of the previous motion, but the forward vector will reflect the initial direction for this motion, not the ending direction from the previous motion. When the two motions are tangent, then these directions will be similar.

The part surface and drive surface normal vectors will always point away from the respective surface towards the cutter. The contact points and normal vectors should be the same as the ending position from the previous motion.

B.9 Motion Check Surface Record

Format:

%Cldata(1)	=	ISN
%Cldata(2)	=	Clfile Record Number
%Cldata(3)	=	5220 (FROM)
%Cldata(4)	=	3
%Cldata(5:7)	=	Tool end point
%Cldata(8:10)	=	Tool axis vector
%Cldata(11:13)	=	Check surface contact point
%Cldata(14:16)	=	Check surface normal vector

Description: This motion record is created by the GOFWD type motion commands. It follows the expanded motion records and will duplicate the final tool position calculated for this motion, plus contain the check surface contact point and normal vector. The typical output from the GOFWD type motion commands begins with a Motion Startup Record (5210), continues with Expanded Motion Record(s) (5200), and ends with a Motion Check Surface Record (5220).

The check surface normal vector will always point away from the check surface towards the cutter.

B.10 Cutter Record

Format:

%Cldata(1)	=	ISN
%Cldata(2)	=	Clfile Record Number
%Cldata(3)	=	6000 (CUTTER)
%Cldata(4)	=	0
%Cldata(5)	=	Cutter Diameter
%Cldata(6)	=	Corner Radius
%Cldata(7)	=	Side Angle
%Cldata(8)	=	Side Radius for Barrel Cutter
%Cldata(9)	=	Z-Height for Barrel Cutter
%Cldata(10)	=	Flat Angle for Barrel Cutter

Description: The Cutter record contains the parameters specified on the CUTTER command. The parameters describe above are for an **NCL** style cutter. It is possible that not all of the parameters described will be in the command, the %NCLD variable can be checked for the actual number of parameters in the command.

An APT style Cutter record will typically contain the initial four fields along with seven parameters for the cutter definition.

B.11 Tracut Record

Format:

%Cldata(1)	=	ISN
%Cldata(2)	=	Clfile Record Number
%Cldata(3)	=	7000
%Cldata(4)	=	1 (TRACUT/ON) 2 (TRACUT/OFF)
%Cldata(5:16)	=	Tracut matrix when TRACUT/ON

Description: The Tracut record specifies whether the following clfile records are altered by a matrix and if so contains the actual matrix used to alter them.

B.12 Cutter Display Record

Format:

%Cldata(1)	=	ISN
%Cldata(2)	=	Clfile Record Number
%Cldata(3)	=	7100 (CUTTER)
%Cldata(4)	=	1 (CUTTER/DISPLY,parms 2 (CUTTER/DISPLY,symbol “pre- NCL V9.6 ”) 3 (CUTTER/DISPLY,PART-ALL 4 (CUTTER/DISPLY,shank “pre- NCL V9.6 ”) 5 (CUTTER/DISPLY,symbol “ NCL V9.6 ”) 6 (CUTTER/DISPLY,SHANK/HOLDER “ NCL V9.6 ”) 7 (CUTTER/DISPLY,symbol “ NCL V9.7+ ”) 8 (CUTTER/DISPLY,symbol “ NCL V9.7+ ”)
%Cldata(5:%Ncld)	=	Contains the CUTTER/DISPLY parameters.

Description: The CUTTER/DISPLY parameters vary depending on the subtype of the command (%Cldata(4)). These various subtype parameter definitions are described below.

CUTTER/DISPLY,parms

%Cldata(4) = 1
%Cldata(5:%Ncld) = Cutter parameters (same as the Cutter Record).

CUTTER/DISPLY,symbol (pre 9.6)

%Cldata(4) = 2
%Cldata(5:7) (T) = Cutter Symbol)
%Cldata(8) = 2 (Symbol Cutter)
3 (Symbol Holder w-CUTTER)
4 (Symbol Holder w-CUTTER/DISPLY)
%Cldata(9) = Attach-Z, Lathe X-offset
%Cldata(10) = Lathe Y-offset
%Cldata(11) = Lathe Z-attach
%Cldata(12) = Lathe Z-depth

CUTTER/DISPLY,PART-ALL

%Cldata(4) = 3
%Cldata(5) (I4) = 0 (PART)
1 (ALL)

CUTTER/DISPLY,SHANK (pre 9.6)

%Cldata(4) = 4
%Cldata(5) = Diameter/Width
%Cldata(6) = Height/Length
%Cldata(7) = Lathe depth
%Cldata(8) = Lathe Y-offset
%Cldata(9) = 716-1000 (CUTTER)
157-10000 (HOLDER)

CUTTER/DISPLY,symbol (9.6)

%Cldata(4) = 5
%Cldata(5:7) (T) = Symbol
%Cldata(7) (I3) = 2 (Symbol)
3 (Tool file)
4 (Pt-list)
%Cldata(8) = Lathe Z-min
%Cldata(9) = Lathe Z-max

CUTTER/DISPLY,SHANK-HOLDER (9.6)

%Cldata(4) = 6
%Cldata(5:7) (T) = Symbol
%Cldata(7) (I3) = 1 (Parameters)
2 (Symbol)
3 (Tool file)
4 (Pt-list)
%Cldata(7) (I4) = 0 (Shank as Cutter)
1 (Shank as Holder)
2 (Holder)
%Cldata(8) = Diameter/Lathe Width/Symbol Offset
%Cldata(9) = Height/Lathe Length/Lathe Y-offset
%Cldata(10) = Z-offset/Lathe depth/Lathe Z-min
%Cldata(11) = Y-offset/Lathe Z-max

CUTTER/DISPLY,symbols (9.7+)

%Cldata(4) = 7
%Cldata(5) (I1) = Number of chars in Symbol
%Cldata(5) (I2) = 2 (Symbol)
3 (Tool file)
4 (Pt-list)
%Cldata(6) = Z-min for Lathe tools
%Cldata(7) = Z-max for Lathe tools
%Cldata(8:%Ncld) = Symbol (T)

CUTTER/DISPLY,SHANK-HOLDER (9.7+)

%Cldata(4) = 8
%Cldata(5) (I1) = Number of chars in Symbol
%Cldata(5) (I2) = 1 (Parameters)
2 (Symbol)
3 (Tool File)
4 (Pt-list)
%Cldata(5) (I3) = 0 (Shank as Cutter)
1 (Shank as Holder)
2 (Holder definition)
%Cldata(6) = Diameter/Lathe Width/Symbol Offset
%Cldata(7) = Height/Lathe Length/Lathe Y-offset

%Cldata(8)	=	Z-offset/Lathe depth/Lathe Z-min
%Cldata(9)	=	Y-offset/Lathe Z-max
%Cldata(10:%Ncld)	=	Symbol (T)

B.13 Cutter Geometry Record

B.13.1 NCL 9.6 And Below

Format:	%Cldata(1)	=	ISN
	%Cldata(2)	=	Clfile Record Number
	%Cldata(3)	=	7110
	%Cldata(4)	=	1 (Last record) 2 (Continuation record follows)
	%Cldata(5:7)	=	Symbol
	%Cldata(7) (I4)	=	Number of points in record
	%Cldata(8) (I1)	=	1 (Mill geometry) 2 (Lathe/Blade Geometry)
	%Cldata(9:%Ncld)	=	XY-IJ point-vector list

Description: Contains the 2-D point and vector list describing the profile of the cutter.

B.13.2 NCL 9.7 And Above

Format:	%Cldata(1)	=	ISN
	%Cldata(2)	=	Clfile Record Number
	%Cldata(3)	=	7120
	%Cldata(4)	=	1 (Last record) 2 (Continuation record follows)
	%Cldata(5) (I1)	=	Number of chars in Symbol
	%Cldata(5) (I2)	=	1 (Mill Geometry) 2 (Lathe/Blade Geometry)
	%Cldata(5) (I1)	=	Number of points in record
	%Cldata(6:n) (T)	=	Symbol
	%Cldata(n:%Ncld)	=	XY-IJ point-vector list

Description: Contains the 2-D point and vector list describing the profile of the cutter.

B.14 SEQUNC Record

Format:	%Cldata(1)	=	ISN
	%Cldata(2)	=	Clfile Record Number
	%Cldata(3)	=	7200
	%Cldata(4)	=	1 (SEQUNC/label) 2 (SEQUNC/END)

%Cldata(5:7) (T)	=	SEQUNC label
%Cldata(8:13)	=	Tool end point (XYZIJK)
%Cldata(14:19)	=	Cutter parameters
%Cldata(20:29)	=	Cutter display parameters (Part 1)
%Cldata(30) (I1)	=	Number of chars in Cutter symbol
%Cldata(30) (I2)	=	Number of chars in Shank symbol
%Cldata(30) (I3)	=	Number of chars in Holder symbol
%Cldata(31)	=	(Not used)
%Cldata(32:34)(I)	=	Cutter integer parameters
%Cldata(35) (I1)	=	Rapid setting
%Cldata(35) (I2)	=	Multax setting
%Cldata(35) (I3)	=	Feed rate type
%Cldata(35) (I4)	=	Spindle direction
%Cldata(36)	=	IPM Feed rate
%Cldata(37:48)	=	Tracut matrix
%Cldata(49)	=	IPR Feed rate
%Cldata(50)	=	Spindle RPM
%Cldata(51:53)(I)	=	Cycle integer parameters
%Cldata(54:63)	=	Cycle real parameters
%Cldata(64:73)	=	Cutter display parameters (Part 2)
%Cldata(74:n) (T)	=	Cutter, Shank, Holder symbols

Description: Contains the information required to start processing or simulation from this point in the clfile.

B.15 Units Record

Format:	%Cldata(1)	=	ISN
	%Cldata(2)	=	Clfile Record Number
	%Cldata(3)	=	7300
	%Cldata(4)	=	1 (Inches) 2 (Millimeters)

Description: Contains the units setting for the input clfile.

B.16 Clfile Header Record

Format:	%Cldata(1)	=	ISN
	%Cldata(2)	=	Clfile Record Number
	%Cldata(3)	=	7400
	%Cldata(4)	=	(Not used)
	%Cldata(5:14)(T)	=	Clfile name
	%Cldata(15:16)(T)	=	Clfile creation date
	%Cldata(17)(T)	=	Clfile creation time
	%Cldata(18)(T)	=	CAM system that created clfile
	%Cldata(19)(T)	=	CAM system version number

Description: Contains information concerning the creation of the clfile, including the clfile name, creation date and time, and CAM system name and version number.

B.17 Multax Record

Format:

%Cldata(1)	=	ISN
%Cldata(2)	=	Clfile Record Number
%Cldata(3)	=	9000 (MULTAX)
%Cldata(4)	=	0 (MULTAX/OFF) 1 (MULTAX/ON)

Description: Contains the MULTAX setting. When MULTAX/OFF is in effect the motion records (5000) will contain the tool end point only (XYZ). When MULTAX/ON is in effect the motion records will contain the tool end points and tool axis vectors (XYZIJK).

APPENDIX C Examples

C.1 Introduction

This chapter gives examples of *Macros* and what some of their uses may be. These examples are meant to show the basic construct of *Macros* in general and how they may be used to improve development and run time.

C.2 Converting Post-Processor Commands

The following example shows how you might convert a command that was recognized by a custom post-processor into command that are recognized by **PostWorks** and will produce the same output.

In this case a one, three, or four parameter *ORIGIN* statement is converted into a three parameter *ORIGIN* statement and/or a *TOOLNO/ADJUST* statement.

```
ORIGIN/MACRO,4
REMARK Command:   ORIGIN/ [ x,y,z ] [ ,h ]
REMARK
REMARK Description: The ORIGIN command accepts as input the part origin
REMARK               and/or a tool offset register. You may specify 1, 3
REMARK               or 4 parameters. A single parameter ORIGIN command
REMARK               specifies a tool offset register, a three parameter
REMARK               ORIGIN command defines the part origin and a four
REMARK               parameter command defines both the part origin and
REMARK               a tool offset register.
REMARK               The ORIGIN Macro will convert the above syntax into
REMARK               an ORIGIN and/or TOOLNO/ADJUST command that is
REMARK               acceptable to the GEN4AX post-processor.
$$
$$...Check validity of command
$$
    Syntax/-9999-9999,-9999-9999,-9999-9999,-9999-9999
    If (%Error == 1 | %Narg == 2) Then
        Print
        Print/Command
        Error/'NUMVAREX'
        Jumpto/endit
    Endif
$$
$$...Convert input command to command(s) acceptable to PostWorks
$$
    If (%Narg >= 3) ORIGIN/%Arg(1:3)
    If (%Narg <> 3) TOOLNO/ADJUST,%Arg(%Narg)
$$
$$...End of routine
$$
endit: Termac
```

C.3 Tool Change Macro

The following example shows a basic tool change *Macro*. The sample *LOADTL Macro* will load the next tool, turn the spindle and coolant on, position the rotary table, and select the next tool.

Remember, this tool change *Macro* is an example only, and is not meant to run any specific machine.

```
$$
$$...Initialize tool length compensation register
$$
Real pnum
    pnum = 0
Termac

LOADTL/Macro,12
REMRK
REMARK Command: LOADTL/tn,LENGTH,tl,NEXT,ntn,RPM,rv      $
REMARK
REMARK          [,FLOOD] [,STOP] [,TAP] [,TABLE,pos]
REMARK          MIST
REMARK          AIR
REMARK
REMARK Description: The LOADTL command will output all required codes to
REMARK                perform a tool change sequence at the machine.
REMARK
REMARK          'tn' specifies the tool number to load. "LENGTH" is
REMARK                required for syntax. 'tl' specifies the tool length.
REMARK
REMARK          'NEXT,ntn' specifies the next tool to select.
REMARK          'RPM,rv' specifies the spindle speed.
REMARK
REMARK          'FLOOD', 'MIST' and 'AIR' specify the type of
REMARK                coolant to turn on. The default is 'FLOOD'.
REMARK
REMARK          If 'STOP' is specified, then a STOP command will be
REMARK                issued directly following the tool change. 'TAP'
REMARK                enables the spindle tapping range.
REMARK
REMARK          'TABLE,pos' specifies the table rotation for this
REMARK                sequence. The default is 0.
REMARK
Real icol,itap,istp,tpos,i
$$
$$...Assume default settings
$$
    icol = 0
    istp = 0
    itap = 0
    tpos = 0
$$
$$...Parse command
```



```

$$
If (%Narg < 7 | %Error == 1) JumpTo/errlab
If (%Arg(1) <> ' ' | %Arg(2) <> 'LENGTH' | $
    %Arg(3) <> ' ' | %Arg(4) <> 'NEXT' | $
    %Arg(5) <> ' ' | %Arg(6) <> 'RPM' | $
    %Arg(7) <> ' ') JumpTo/errlab
If (%Narg > 7) Then
    Do id1 i=1,%Narg,1
        If (%Arg(i) == 'FLOOD' | %Arg(i) == 'MIST' | $
            %Arg(i) == 'AIR') Then
            icol = i
        Else If (%Arg(i) == 'STOP' | $
            %Arg(i) == 'CENTER') Then
            istp = 1
        Else If (%Arg(i) == 'TAP') Then
            itap = 1
        Else If (%Arg(i) == 'TABLE') Then
            If (i == %Narg) JumpTo/errlab
            tpos = %Arg(i+1)
            i = i + 1
        Endif
    id1: Continue
    Endif
$$
$$...Cancel coolant & spindle
$$
    COOLNT/OFF
    SPINDL/OFF
    LEADER/6
    FORCE/#G,90,NOW, #G,94,NOW, #G,1,NOW
$$
$$...Move to tool change position & change tools
$$
    POSTIN/XAXIS,10,YAXIS,15,ZAXIS,18
    FORCE/#M,25,NEXT
    POSITN/ZAXIS,20
    SELCTL/%Arg(1),LENGTH,%Arg(3)-5,MODIFY
    LOADTL/%Arg(1),LENGTH,%Arg(3)-5
    If (istp == 1) STOP
$$
$$...Position rotary table
$$
    FORCE/#A,NEXT
    ROTABLE/ATANGL,tpos
$$
$$...Reinstate coolant & spindle
$$...Select next tool
$$
    If (itap == 1) FORCE/#M,40,NEXT
    SPINDL/%Arg(7),CLW
    If (icol == 0) Then
        COOLNT/FLOOD
    Else
        COOLNT/%Arg(icol)

```

```

        Endif
        FORCE/#T,%Arg(5),NOW
$$
$$...Turn on tool length compensation
$$
        FORCE/NOW
        pnum = pnum + 1
        TOOLNO/ADJUST,pnum
        JUMPTO/endit
$$
$$...Syntax error
$$
errlab:
        Error/'INVSYN'
        PPRINT/Command
endit:
Termac

```

C.4 Creating a New Cycle Syntax

Cycle statements, because of the universal nature of **PostWorks**, contain more parameters that they would in a custom post-processor. This sample *CYCLE Macro* will accept a shorter version of the *CYCLE* command and expand it to the syntax which **PostWorks** expects.

```

CYCLE/Macro,20
REMARK
REMARK  The CYCLE Macro is used to convert a short cycle command to the
REMARK  syntax required by PostWorks. The following commands will be
REMARK  converted.
REMARK
REMARK      Input: CYCLE/BORE ,depth,feed,IPM,r
REMARK                      BORE8
REMARK                      BORE9
REMARK                      DRILL
REMARK                      FACE
REMARK                      REAM
REMARK                      TAP
REMARK
REMARK      Output: CYCLE/mode,FEDTO,depth,IPM,feed,RAPTO,r
REMARK
REMARK      Input: CYCLE/CUTANG,dia,ang,feed(,DWELL),IPM,r
REMARK
REMARK      Output: CYCLE/DRILL,FEDTO,dia,ang,IPM,feed,RAPTO,r
REMARK                      FACE
REMARK
REMARK      Input: CYCLE/DEEP,depth,peck,feed,IPM,r
REMARK                      THRU
REMARK
REMARK      Output: CYCLE/mode,FEDTO,depth,STEP,peck,IPM,feed,RAPTO,r
REMARK
$$
$$...CYCLE/BORE

```

```

$$...      BORE8
$$...      BORE9
$$...      DRILL
$$...      FACE
$$...      REAM
$$...      TAP
$$
    If (%Arg(1) == 'BORE' | %Arg(1) == 'BORE8' |
        %Arg(1) == 'BORE9' | %Arg(1) == 'DRILL' |
        %Arg(1) == 'FACE' | %Arg(1) == 'REAM' |
        %Arg(1) == 'TAP') Then
        CYCLE/%Arg(1), FEDTO, %Arg(2), %Arg(4), %Arg(3), $
        RAPTO, %Arg(5)
$$
$$...CYCLE/CUTANG
$$
    Else if (%Arg(1) == 'CUTANG') Then
        If (%Narg < 7) Then
            CYCLE/DRILL, FEDTO, %Arg(2), %Arg(3), %Arg(5), $
            %Arg(4), RAPTO, %Arg(6)
        Else
            CYCLE/FACE, FEDTO, %Arg(2), %Arg(3), %Arg(6), $
            %Arg(5), RAPTO, %Arg(7)
$$
$$...CYCLE/DEEP
$$...      THRU
$$
    Else If (%Arg(1) == 'DEEP' | %Arg(1) == 'THRU') Then
        CYCLE/%Arg(1), FEDTO, %Arg(2), STEP, %Arg(3), $
        %Arg(5), %Arg(4), RAPTO, %Arg(6)
$$
$$...Unknown cycle command
$$
    Else
        CYCLE/%Arg(1:%Narg)
    Endif
Termac

```

C.5 Emulating *PREPST*'s HEAD Statement

The following example show how you might modify cl points before final processing by a machine specific post-processor. In this case, one of the features of the *HEAD* statement, from *PREPST*, is emulated. The functionality of these *Macros* is exactly the same as in *PREPST*.

```

$$
$$...Declare global variables and
$$...initialize routine
$$
Real dia,rad
    MULTAX/OFF
    Disable/GOTO
Termac

```

```

HEAD/Macro,3
$$
REMARK Command: HEAD/OFF
REMARK [ON] [,dia,rad]
REMARK
REMARK Description: The HEAD command is used to alter c1 points using
REMARK the given point and a tool axis vector. The
REMARK programmer drives a surface with a zero diameter
REMARK cutter and TLAXIS/NORMAL,PS in effect. The HEAD
REMARK statement will then output the points as if the
REMARK surface was driven with the specified cutter
REMARK diameter and corner radius, in a 3 axis mode.
REMARK
REMARK 'OFF' turns off the HEAD command.
REMARK 'ON' turns on the HEAD command.
REMARK
REMARK 'dia' specifies the cutter diameter. The default
REMARK is 1.0. 'rad' defines the corner radius. The
REMARK default is .5.
$$
$$...Check command syntax
$$
If (%Narg == 0) JumpTo/synerr
$$
$$...HEAD/OFF
$$
If (%Arg(1) == 'OFF') Then
    If (%Narg <> 1) JUMPTO/synerr
    Disable/GOTO
$$
$$...HEAD/ON
$$
Else if (%Arg(1) == 'ON') Then
    If (%Narg == 1) Then
        Enable/GOTO
        dia = 1.0
        rad = .5
    Else
        If (%Narg <> 3) JumpTo/synerr
        If (%Arg(2) <> ' ' | %Arg(3) <> ' ') $
            JumpTo/synerr
        If (%Arg(2) <= 0. | %Arg(3) <= 0. | $
            %Arg(3)*2 > %Arg(2)) JumpTo/synerr
        Enable/GOTO
        dia = %Arg(2)
        rad = %Arg(3)
    Endif
$$
$$...HEAD/dia,rad
$$
Else
    If (%Narg <> 2) JumpTo/synerr
    If (%Arg(1) <> ' ' | %Arg(2) <> ' ') $

```

```

        Jumpto/synerr
    If (%Arg(1) <=0. | %Arg(2) <=0. | $
        %Arg(2)*2 > %Arg(1)) Jumpto/synerr
    Enable/GOTO
    dia = %Arg(1)
    rad = %Arg(2)
Endif
Jumpto/endit
$$
$$... Syntax error
$$
synerr:
    Print
    Print/Command
    Error/' INVSYN'
$$
$$...Endof routine
$$
endit: Termac

GOTO/Macro,2
    Real pt(6),i,j
    Char la(80)
$$
$$...Multax must be in effect
$$
    If (%Multax <> 1) Then
        ERROR/'Multax must be in effect.'
        Disable/GOTO
        Jumpto/endit
    Endif
$$
$$...Loop through points
$$
    Do id20 i=1,%Arg(1)*6,6
        Do id2 j=1,6,1
            pt(j) = %Clpt(i+j-1)
id2:    Continue
$$
$$...Adjust points
$$
    pt(1) = (pt(1) + rad*pt(4)) + ((dia/2-rad) * $
        (pt(4)/sqrt(pt(4)**2+pt(5)**2)))
    pt(2) = (pt(2) + rad*pt(5)) + ((dia/2-rad) * $
        (pt(5)/sqrt(pt(4)**2+pt(5)**2)))
    pt(3) = pt(3) + rad*pt(6) - rad
    Define/%Clpt(i),pt(1), %Clpt(i+1),pt(2), $
        %Clpt(i+2),pt(3)
id20: Continue
$$
$$...Go to points
$$
    GOTO/%Clpt(1:%Arg(1)*6)
$$

```

```

$$...End of routine
$$
endit: Termac

```

C.6 Emulating GEOPST's CHECK Statement

The following example demonstrates a geometry building set of *Macros*. These *Macros* emulate the *CHECK/ON,8,6* statement from the GEOPST utility. Using the *cl* point motion, a point will be defined each time a move crossed a defined plane. The output point will lie on the intersection of the move and the plane.

Since the geometry building program is not used to modify the *clfile*, but to create a file of point definitions, the listing file should be enabled (*/LISTING*) and the *clfile* should be disabled (*/NOOBJECT*).

```

$$
$$...Declare global variables and
$$...Initialize routine
$$
    Real ckpl(4),isub
    Char name(6)
    #1 = F4.5,M1.0,S-
    #2 = D5.0,M1.0,S-
    Listing/OFF
    Disable/GOTO
    Disable/PPRINT
Termac
CHECK/Macro,8
$$
REMARK  Command:  CHECK/OFF
REMARK                      ON,8,6,s,i,j,k,d
REMARK
REMARK  Description: The Check command takes a clfile as input and
REMARK                  defines points, from cl point motion, that lie
REMARK                  on a specified plane. The point defined will be
REMARK                  the tool end point at the intersection of cutter
REMARK                  cutter path and the plane.
REMARK
REMARK                  'OFF' disables the output of points.
REMARK                  'ON' turns on the CHECK command, enabling the
REMARK                  output of plane intersection points.
REMARK
REMARK                  's' is the beginning subscript number for the
REMARK                  point array that will be defined.
REMARK                  'i,j,k,d' defines the canonical form of the
REMARK                  intersection plane.
REMARK
REMARK                  A PPRINT statement containing the name of the
REMARK                  point array to define must follow the CHECK/ON
REMARK                  command.
$$

```

```

        Real sum
$$
$$...Check command syntax
$$
        Syntax/ (ON,OFF) , 8,6,1-99999,-1-1,-1-1,-1-1,$
            -99999-99999
        If (%Error == 1) Jump to/inv syn
$$
$$...CHECK/OFF
$$
        If (%Arg(1) == 'OFF') Then
            If (%Narg <> 1) Jump to/inv syn
            Disable/GOTO
$$
$$...CHECK/ON
$$
        Else
            If (%Narg <> 8) Jump to/inv syn
            isub = %Arg(4)
            name = ' '
$$
$$.....Unitize plane
$$
        sum = sqrt(%Arg(5)**2 + %Arg(6)**2 + %Arg(7)**2)
        If (sum == 0) Jump to/inv plan
        ckpl(1) = %Arg(5) / sum
        ckpl(2) = %Arg(6) / sum
        ckpl(3) = %Arg(7) / sum
        ckpl(4) = %Arg(8)
$$
$$.....Turn on the CHECK statement
$$
        Enable/PPRINT
        Enable/GOTO
    Endif
    Jump to/endit
$$
$$...Invalid comand syntax
$$
inv syn:
    Print
    Print/Command
    Error/' INVSYN'
    Disable/PPRINT
    Diable/GOTO
    Jump to/endit
$$
$$...Zero length vector
$$
inv plan:
    Print
    Print/Command
    Error/' Zero length vector'
    Disable/PPRINT

```

```

        Diab!e/GOTO
$$
$$...End of routine
$$
endit: Termac

PPRINT/MACRO,9
        name = %Arg(1)
        Disable/PPRINT
TERMAC

GOTO/Macro,2
        Real npt,stocl(6),i,j,pt(3),vei,vej,vek,t1,dist1, $
            dist2,is,ptio(3)
        Char tcmd(80)
$$
$$...Initialize routine
$$
        npt =3 ; if (%Multax == 1) npt = 6
$$
$$...Check to see if a name was specified
$$
        If (name == ' ') Then
            Print
            Error/'You must define the Point name using PPRINT.'
            Disable/PPRINT
            Disable/GOTO
            Jump!o/endit
        Endif
$$
$$...Save last Goto point
$$
        Do id0 j=1,3,1
            stocl(j) = %Clsav(j)
id0: Continue
        If (%Arg(2) == 3) Jump!o/endit
$$
$$...Loop through points
$$
        Do id20 i=1,%Arg(1)*npt,npt
            Do id2 j=1,3,1
                pt(j) = %Clpt(i+j-1)
id2: Continue
$$
$$.....Get what side of the plane
$$.....the point are on
$$
        dist1 = (stocl(1)*ckpl(1) + stocl(2)*ckpl(2) + $
            stocl(3)*ckpl(3)) - ckpl(4)
        dist2 = (pt(1)*ckpl(1) + pt(2)*ckpl(2) + $
            pt(3)*ckpl(3)) - ckpl(4)
$$
$$...See if tool path crosses plane
$$

```



```

        If ((dist1 <=0 & dist2 <=0) | (dist1>=0 & $
            dist2 >=0)) Jumppto/savpt
$$
$$...Calculate plane intersection point
$$
    vei= pt(1) - stock(1)
    vej= pt(2) - stock(2)
    vek= pt(3) - stock(3)
    t1 = sqrt(vei**2 + vej**2 + vek**2)
    vei = vei / t1
    vej = vej / t1
    vek = vek / t1
    t1 = dist / (vei*ckpl(1) + vej*clpl(2) + $
                vek*ckpl(3))
$$
    ptio(1) = stocl(1) - t1*vei
    ptio(2) = stocl(2) - t1*vej
    ptio(3) = stocl(3) - t1*vek
$$
$$...Output point definition
$$
    tcmd = name + '(' + rtoc(isub,#2) + ') = POINT/' $
          + rtoc(ptio(1),#1) + ',' + $
          + rtoc(ptio(2),#1) + ',' + $
          + rtoc(ptio(3),#1)

    print/tcmd
    isub = isub + 1
$$
$$...Save point
$$
savpt:
    Do id10 j=1,3,1
        stocl(j) = pt(j)
id10: Continue
id20: Continue
$$
$$...Save last point in Goto record
$$
endit:
    is = (%Arg(1)-1) * npt + 1
    GOTO/%Clpt(is:is+npt-1)
Termac

```

INDEX

!	1-2
\$	1-1
\$\$	1-2
:	6-1
ABS	4-2, A-2
ACOS	4-3, A-2
APPEND	1-1, 1-2, 5-2, 5-3
APPEND text	A-6
ARG	3-3
Arithmetic Function	4-2
Array	1-4, 3-3, 4-3
ASIN	4-3, A-2
Assignment Statements	4-4
ATAN	4-3, A-2
AXIS	3-12
AXSDLS	3-12
AXSDLT	3-12
AXSMAX	3-12
AXSMIN	3-12
BLOCK	A-9
BREAK	2-6, 5-3, A-6
BREAKF	6-5
BRKPRM	3-5
CHAR	1-4, 3-2, A-6
CIRCLE	2-7
CIRFLG	2-7
CIRPRM	2-7
CL Point	3-9
CL Record Number	3-4
CLBRIEF	4-6
CLDATA	5-15
CLEXT	4-6
Clfile	5-10
CLFIND	5-15
CLNAME	4-6
CLPT	2-3, 3-9, 5-8
CLREAD	5-15
CLREC	3-4
CLSAV	3-9
CLWRIT	5-15

COMMAND	4-6, A-2, C-4
Comments	1-2
COMPUTER	4-7
Continuation Lines	1-1
CONTINUE	6-4, A-6, A-7
Control Point	3-8
Control Tape	3-5
COS	4-3, A-2
CTOR	4-3, 4-10, A-3
CUTTER	3-4
CYCLE	C-4
DATE	4-7, A-3
DEFINE	1-5, 5-4, A-7
Delta Summations	3-10, 3-11
DISABLE	2-2, 5-4, A-7, C-8
DO	6-4, A-7
Documentation	5-12
ELSE	6-3, A-7
ENABLE	5-4, A-8, C-9
ENDDO	6-5
ENDIF	6-3, A-8
ENDMAC	6-5
End-of-Sequence	3-5
ERROR	2-4, 3-6, 5-5, 5-17, A-5, A-8
Error Message	2-4, 4-7, 5-5
ERRPC	3-6
ERRTXT	4-7, A-3
Expressions	4-1, 4-5
FCLOSE	5-13
FEDRAT	3-8
FEED	3-8
Feed Rate	3-8
FINI	2-3, A-5
FMTCOD	4-8, 4-10, A-3
FOPEN	5-13
FORCE	5-5, 5-6, A-9, C-3
Format Descriptor	A-3
FREAD	5-13
FREW	5-13
FROM	5-7, A-9
FWRITE	5-13
G\$MAIN	1-4, 2-3, A-5
GEOPST	C-8
GOHOME	3-13
GOTO	2-3, 3-10, 5-7, A-5, A-10, C-6, C-7
GOTOC	5-7, A-10

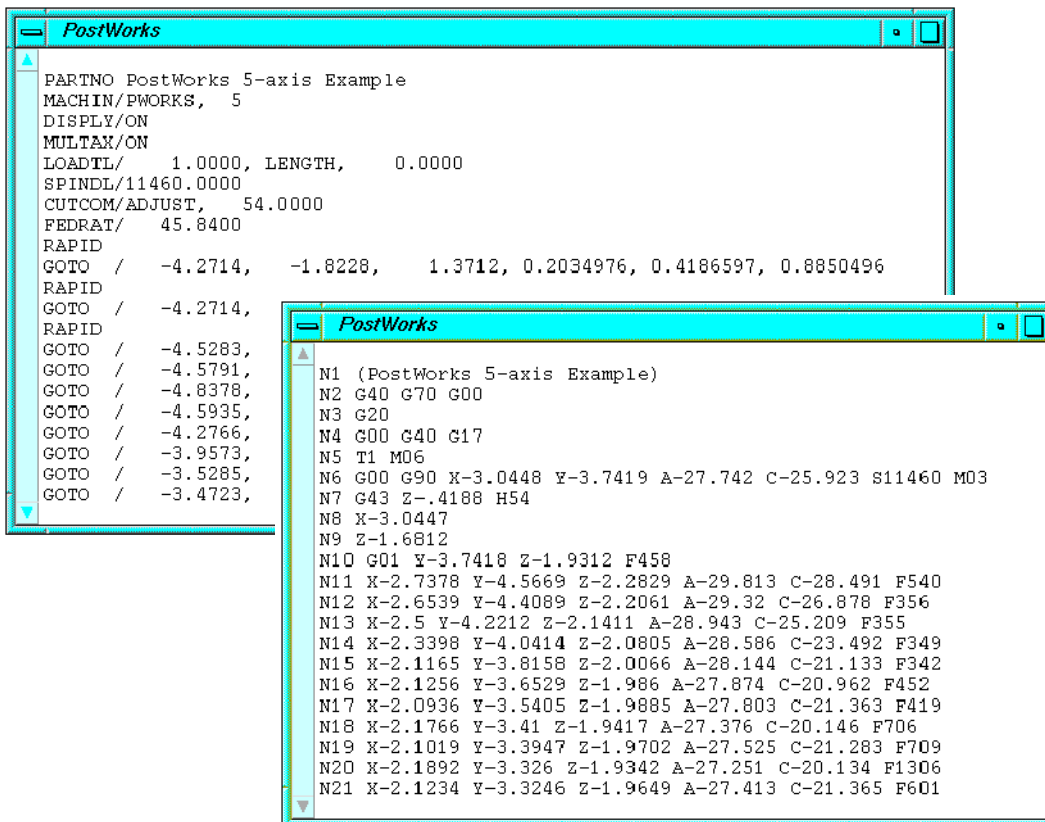
HOME	3-13
IF	A-10
IF-THEN-ELSE	6-3
IFTOL	6-2, A-10
INCLUD	5-9, A-10
INDEX	4-3, A-3
Input Sequence Number	3-4
INSERT	1-1, 1-2, 2-2, 5-2, A-11
INT	4-2, A-3
ISN	3-4
JUMPTO	6-1, 6-4, A-10
KPOSMP	3-5
Label	6-1, A-10
Lathe	3-8
LETTER	1-1, 1-2, 2-2, 5-2, A-11
LINDLS	3-11
LINDLT	3-11
LINEAR	3-11
Linear Axis	3-11
LINENO	3-6
LINMAX	3-11
LINMIN	3-11
LISTING	5-9, A-11, C-8
Listing File	5-9, 5-11
LOCATE	4-3, A-3
Logical IF	6-2
Logical Register	4-12, A-1
Loop	6-4
MACHIN	3-13, 5-3, 5-10, A-11
Machine Descriptor File	3-13
Machining Times	3-9
MACRO	2-1
Macro	3-3, 5-4
Macro Argument	3-1
Major Word	1-2, 5-1
MCHTIM	3-9
Mill	3-8
Minor Word	1-3, 3-3, 5-17
MOTION	A-9
MOVDIS	3-8
MOVTIM	3-9
MPost	5-5
MULTAX	2-4, 3-5, 5-7, 5-10, A-11
Multiple Statement Lines	1-2
NARG	3-3
NCLD	5-15

NEXT	A-9
NOMORE	A-9
NOW	A-9
NTOOL	3-7
Number	1-3
NUMERR	3-6
NUMFAT	3-6
NUMIER	3-6
NUMWRN	3-6
OFF	A-9
OMIT	A-9
ON	A-9
Operators	4-1, 4-5
OPTION	2-4, A-5
ORIGIN	3-10, C-1
PAGE	A-11
PAGENO	3-6
Parameter List	5-1
Parenthesis	4-2
PARTNO	1-1, 1-2, 2-2, 5-2, A-11
PC	3-6
PCHBRIEF	4-8
PCHFILE	4-8, A-6
PCHNAME	4-8
Physical Register	4-12, A-1
PI	3-7
PMacro	1-5
POSCAL	5-8
POSMAP	3-5
Post Variables	1-5
Post-processor Command	1-3, 2-1, 4-6, 5-2, 5-17
PPRINT	1-1, 1-2, 2-2, 2-5, 5-2, A-11
PREPST	C-5
PRINT	5-10, 5-11, A-12
Print Descriptor File	3-6
Print File	5-11
PRINTF	5-11, A-12
PSTERR	2-5, 5-12, A-5
PSTNAM	3-13
PWorks	1-5
Range Subscript	4-9, 5-1, 5-8
REAL	1-4, 3-1, A-12
Real Variable	1-4, 3-1, 4-4
RECORD	A-12
REG	3-5
REGEN	3-6

Register	3-5, 4-8, 5-6, A-1
Register Label	4-11
REGST	3-6
REMARK	5-12, A-12
RINDEX	4-4
ROTARY	3-11
ROTDLS	3-11
ROTDLT	3-11
ROTMAX	3-11
ROTMIN	3-11
RPM	3-8
RTOC	4-9, 4-10, A-4
SEQTIM	3-9
SFM	3-8
SIMUL	3-7
SIN	4-3, A-4
Spindle Speed	3-8
SQRT	4-2, A-4
Square Brackets	1-1
STRLEN	4-4
Subscript	1-4
SYNTAX	3-6, 5-17, C-9
TAN	4-3, A-4
TAPBRK	2-6, A-6
Tape Break	2-6, 5-3
TAPLEN	3-5
TERMAC	A-13
Text String	1-3, 3-3
Text Variable	1-4, 3-2
THEN	6-3, A-7, A-10
TIME	4-9, A-4
TIMSCL	3-9
TLEN	3-7
TLNO	3-7
TLTIM	3-9
TLVEC	3-9
TOOL	3-7
Tool Axis Vector	3-10, 5-7, 5-11
Tool Change	C-2
Tool Data	3-7, 3-9
TOOLNO	C-1
TRANS	3-10
Travel Limits	3-10, 3-12
Trigonometric Functions	4-3
UNITS	3-5
USER	3-5

User Defined Block	5-5
USERNAME	4-9
Variable.	1-3, 3-1
XFORM	2-8
XYZ	3-10
XYZDLS.	3-10
XYZDLT	3-10
XYZMAX.	3-10
XYZMIN	3-10

PWorks



```
PARTNO PostWorks 5-axis Example
MACHIN/PWORKS, 5
DISPLY/ON
MULTAX/ON
LOADTL/ 1.0000, LENGTH, 0.0000
SPINDL/11460.0000
CUTCOM/ADJUST, 54.0000
FEDRAT/ 45.8400
RAPID
GOTO / -4.2714, -1.8228, 1.3712, 0.2034976, 0.4186597, 0.8850496
RAPID
GOTO / -4.2714,
RAPID
GOTO / -4.5283,
GOTO / -4.5791,
GOTO / -4.8378,
GOTO / -4.5935,
GOTO / -4.2766,
GOTO / -3.9573,
GOTO / -3.5285,
GOTO / -3.4723,
```

```
N1 (PostWorks 5-axis Example)
N2 G40 G70 G00
N3 G20
N4 G00 G40 G17
N5 T1 M06
N6 G00 G90 X-3.0448 Y-3.7419 A-27.742 C-25.923 S11460 M03
N7 G43 Z-.4188 H54
N8 X-3.0447
N9 Z-1.6812
N10 G01 Y-3.7418 Z-1.9312 F458
N11 X-2.7378 Y-4.5669 Z-2.2829 A-29.813 C-28.491 F540
N12 X-2.6539 Y-4.4089 Z-2.2061 A-29.32 C-26.878 F356
N13 X-2.5 Y-4.2212 Z-2.1411 A-28.943 C-25.209 F355
N14 X-2.3398 Y-4.0414 Z-2.0805 A-28.586 C-23.492 F349
N15 X-2.1165 Y-3.8158 Z-2.0066 A-28.144 C-21.133 F342
N16 X-2.1256 Y-3.6529 Z-1.986 A-27.874 C-20.962 F452
N17 X-2.0936 Y-3.5405 Z-1.9885 A-27.803 C-21.363 F419
N18 X-2.1766 Y-3.41 Z-1.9417 A-27.376 C-20.146 F706
N19 X-2.1019 Y-3.3947 Z-1.9702 A-27.525 C-21.283 F709
N20 X-2.1892 Y-3.326 Z-1.9342 A-27.251 C-20.134 F1306
N21 X-2.1234 Y-3.3246 Z-1.9649 A-27.413 C-21.365 F601
```

Post-processor Commands Reference Manual

Warning and Disclaimer

Every effort has been made to make this document complete and as accurate as possible.
However, no warranty or fitness is implied.

The information is provided on an "as is" basis. Numerical Control Computer Sciences shall have neither liability nor responsibility to any person or entity with respect to any loss or damages in connection with or rising from the information contained in this document.

Copyright 1983-2014 by Numerical Control Computer Sciences
Irvine, California. Printed in the United States of America.
All rights reserved. The contents of this publication may not
be reproduced in any form or by any means, electronic
or mechanical, including photocopying, recording, or
information storage and retrieval systems, for any
purpose other than the licensee's personal use,
without prior written consent from
Numerical Control Computer Sciences.

TABLE OF CONTENTS

CHAPTER 1 Beginning and Ending Sequences1-1

1.1	Part Description - PARTNO.....	1-1
1.2	The MACHIN Card - MACHIN.....	1-1
1.3	Include External File - PPRINT INCLUD.....	1-3
1.4	Rewind Stop - TMARK/AUTO.....	1-4
1.5	Outputting Leader - LEADER.....	1-4
1.6	Part Program Number - POSTN.....	1-4
1.7	Rewinding The Control Tape - REWIND.....	1-5
1.8	Physical End-of-Program - FINI.....	1-5
1.9	Logical End-of-Program - END.....	1-5

CHAPTER 2 Control Tape Commands.....2-1

2.1	Sequence Numbers - SEQNO.....	2-1
2.2	Alignment Blocks - TMARK.....	2-1
2.3	Optional Skip Blocks - OPSKIP.....	2-2
2.4	Tape Block Checksumming - CHECK/LENGTH.....	2-3
2.5	Inserting Tape Blocks - INSERT.....	2-3
2.6	Miscellaneous (M) Codes - AUXFUN.....	2-4
2.7	Preparatory (G) Codes - PREFUN.....	2-4
2.8	Operator Messages - DISPLY.....	2-5
2.9	Print File Messages - PPRINT, TPRINT.....	2-5
2.10	User Specified Tape Break - BREAK.....	2-6
2.11	Disabling Tape Breaks - BREAK/OFF.....	2-6
2.12	Tape Break Sequences - BREAK/mode.....	2-7

CHAPTER 3 Tool Change Sequences3-1

3.1	Selecting A Mill Tool - SELCTL or SELECT.....	3-1
3.2	Loading A Mill Tool - LOADTL or LOAD.....	3-2
3.3	Loading A Special Head - LOADTL/SET.....	3-3
3.4	Unloading A Mill Tool - UNLOAD.....	3-4
3.5	Selecting A Lathe Tool - TURRET.....	3-4
3.6	Tool Description - PPRINT TLN.....	3-6
3.7	Program Stop - STOP.....	3-6
3.8	Optional Stop - OPSTOP.....	3-6
3.9	Coolant Control - COOLNT.....	3-6
3.10	Tool Length Offsets - TOOLNO/ADJUST.....	3-7
3.11	Expanded Tool Length Offset Control - TOOLNO/ADJUST.....	3-7
3.12	Whether To Add Tool Length To The Output Or Not.....	3-9

3.13	Tool Machine Time Calculation - TOOLNO/TIMES	3-9
3.14	Fixture Offsets - CUTCOM/ADJUST	3-9
3.15	Expanded Fixture Offset Control - CUTCOM/ADJUST	3-10
CHAPTER 4 Feeds and Speeds		4-1
4.1	Feed Rates - FEDRAT	4-1
4.2	Feed Rate Control Point - FEDRAT/LENGTH	4-1
4.3	Rapid Moves - RAPID	4-2
4.4	Partial Rapid Moves - RAPID/FEDTO	4-3
4.5	Maximum Feed Rate - FEDRAT/MAXIPM	4-4
4.6	Maximum Degrees Per Minute - MAXDPM	4-4
4.7	Feed Rate Overrides - FEDRAT/LOCK	4-5
4.8	Simple Spindle Control - SPINDL	4-5
4.9	Spindle Speed Control - SPINDL	4-5
4.10	Maximum Spindle Speed - SPINDL/MAXRPM	4-6
4.11	Spindle Speed Overrides - SPINDL/LOCK	4-7
4.12	Machine Dwells - DELAY	4-7
CHAPTER 5 Positioning Commands		5-1
5.1	The FROM Statement - FROM	5-1
5.2	Machine Axes Positioning - POSITN	5-1
5.3	Position The Rotary Axes - ROTABL, ROTHED	5-2
5.4	Clearance Plane - CLRSRF	5-3
5.5	Clearance Distance - CLRSRF/TOOL	5-4
5.6	Retracting The Tool - RETRACT	5-4
5.7	Repositioning The Tool - PLUNGE	5-5
5.8	Move To Home Position - GOHOME	5-6
5.9	Move To Reference Point - GOHOME/mode	5-6
CHAPTER 6 Motion Control		6-1
6.1	Axis Travel Limits - CHECK	6-1
6.2	Interference Zones - CHECK/OUT	6-1
6.3	Output Axes Tolerances - PPTOL	6-2
6.4	Rotary Axes Direction - ROTABL/NEXT	6-3
6.5	Shortest Route Determination - ROTABL/SHORT	6-4
6.6	Rotary Axes Clamping - CLAMP	6-5
6.7	Automatic Clamping Of Rotary Axes - CLAMP/AUTO	6-5
CHAPTER 7 Machining Modes		7-1

7.1	Absolute/Incremental Programming - MODE/INCR.....	7-1
7.2	Tool Axis Vectors - MULTAX	7-1
7.3	Primary Spindle Axis - MODE/TOOL.....	7-2
7.4	Linear Axes Selection - MODE/-AXIS	7-3
7.5	Rotary Axes Selection - MODE/AXIS.....	7-3
7.6	Multi Heads/Tables Selection - ROTABL/ON.....	7-4
7.7	Change Rotary Axes Configuration	7-4
7.8	Interchange Of Linear And Rotary Axes - MODE/ROTATE	7-5
7.9	Rotary Axis Linearization - LINTOL.....	7-6
7.10	Rapid Move Rotary Axis Linearization - LINTOL/RAPID.....	7-7
7.11	Rotary Axis Positional Tolerance - LINTOL/AXIS.....	7-8
7.12	Simple Cutter Compensation Control - CUTCOM	7-8
7.13	Cutter Compensation Direction Control - CUTCOM.....	7-9
7.14	3-D Cutter Compensation Control - CUTCOM/ENDPT	7-10
7.15	Slowdown Blocks - SLOWDN/ON	7-11
7.16	Acceleration Blocks - SLOWDN/RAMP.....	7-12
7.17	Transformation Blocks - TRANS.....	7-13
7.18	Enable/Disable Transformation of Coordinates - TRANS/TOOL	7-13
7.19	Automatic Tool Retraction - RETRCT/TOOL.....	7-14
7.20	Mill/Turn Turning Mode - MODE/LATHE.....	7-15
7.21	Mill/Turn Milling Mode - MODE/MILL,AUTO.....	7-15
7.22	Mill/Turn Cylindrical Interpolation - MODE/MILL,DIAMTR.....	7-16
7.23	Mill/Turn Polar Interpolation -- MODE/MILL,FACE	7-17

CHAPTER 8 Motion Adjustments.....8-1

8.1	Preset Axis Registers - POSTN	8-1
8.2	FROM Preset Axis Registers - POSTN/XYPLAN.....	8-2
8.3	Cancel Preset Axis Registers - POSTN/OFF	8-2
8.4	Part Origin - ORIGIN.....	8-2
8.5	CI Point Modifications - TRANS	8-3
8.6	CI Point Transformations - TRANS/mx	8-3
8.7	CL Point Translate Up Tool (Spindle) Axis - TRANS/UP	8-4
8.8	Output Axis Modifications - TRANS/-AXIS.....	8-4
8.9	Limit Planes - CLRSRF/START	8-5
8.10	Creating Intermediate Moves - LINTOL/LINEAR	8-7
8.11	Rotary Axis Center - ROTABL/ORIGIN.....	8-8
8.12	Rotary Axis Circumference - ROTABL/SIZE	8-8
8.13	Rotary Axis Coordinate Rotations - ROTABL/ADJUST.....	8-9
8.14	Tolerance Truncating Tool Axis Vector Components - PPTOL/VECTOR..	8-9

CHAPTER 9 Circular/Spline Interpolation9-1

9.1	Circular Interpolation Output - ARCSLP.....	9-1
9.2	Circular Tolerances - MCHTOL.....	9-1

9.3	Circular Interpolation Analyzation - SET/LINCIR	9-2
9.4	Helical Interpolation	9-3
9.4.1	Helical Interpolation Control - COUPLE/AUTO	9-3
9.4.2	Helical Interpolation - COUPLE	9-3
9.4.3	Helical Interpolation - CYCLE/CIRCUL	9-4
9.5	Spline Interpolation - ARCSLP/SPLINE	9-5

CHAPTER 10 Canned Cycles.....10-1

10.1	Cycle Control Commands	10-1
10.1.1	Automatic/Canned cycles - CYCLE/AUTO	10-1
10.1.2	Cycle Output Control - CYCLE	10-1
10.2	Automatic Mill Cycles - CYCLE/mode	10-3
10.3	Automatic Mill Cycles For Siemens 840D - CYCLE/...,PARAMS,m1,n1,.....	10-5
10.4	Manual Mill Cycles	10-7
10.4.1	Boring Cycle - CYCLE/BORE	10-7
10.4.2	Boring Cycle - CYCLE/BORE7	10-8
10.4.3	Boring Cycle - CYCLE/BORE8	10-9
10.4.4	Back Boring Cycle - CYCLE/BORE9	10-10
10.4.5	Chip Breaking Drill Cycle - CYCLE/DEEP	10-11
10.4.6	Drill Cycle - CYCLE/DRILL	10-12
10.4.7	Positioning Cycle - CYCLE/MILL	10-13
10.4.8	Reaming Cycle - CYCLE/REAM	10-14
10.4.9	Fine Boring Cycle - CYCLE/SHIFT	10-15
10.4.10	Hole Tapping Cycle - CYCLE/TAP,REVERS	10-16
10.4.11	Deep Hole Drilling Cycle - CYCLE/THRU	10-17
10.5	Miscellaneous Mill Cycle Commands	10-18
10.5.1	Cycle Retract Plane - RETRACT	10-18
10.5.2	Cycle Avoidance Plane - CLRSRF/AVOID	10-18
10.5.3	Part/Fixture Avoidance - CYCLE/AVOID	10-19
10.6	Single Pass Lathe Thread Cutting	10-20
10.6.1	Enabling Thread Cutting - THREAD	10-20
10.6.2	Disabling Thread Cutting - THREAD/OFF	10-21
10.7	Automatic Lathe Cycles - CYCLE/mode	10-22
10.8	Manual Lathe Cycles	10-23
10.8.1	Grooving/Drilling - CYCLE/DEEP,DRILL,THRU	10-24
10.8.2	Facing Cycle - CYCLE/FACE	10-25
10.8.3	Turning Cycle - CYCLE/ROUGH,TURN	10-26
10.8.4	Thread Cutting Cycle - CYCLE/THREAD	10-27

CHAPTER 11 Ultrasonic Cutter11-1

11.1	Knife Blade Tracking Control - MODE/BLADE	11-1
11.2	Z-axis Variations Motion - HEAD/ZIGZAG	11-2

11.3	Automatic Ultrasonic Blade Alignment - ALIGN/CUT.....	11-3
11.4	Ultrasonic Blade Positioning - ALIGN/MOVE	11-4
11.5	Spline Interpolation - SMOOTH	11-5
11.6	Curve Blending - SMOOTH/ATANGL.....	11-5
11.7	Vacuum Pods	11-6
11.7.1	Vacuum Control - POD/AIR	11-6
11.7.2	Disable Operator Vacuum Control - POD/LOCK	11-6
11.7.3	Move All Pods Down - POD/DOWN.....	11-7
11.7.4	Addressed POD Vacuum Off - POD/row,col,AIR.....	11-7
11.7.5	Clamp Addressed Pod - POD/row,col,CLAMP	11-7
11.7.6	Addressed Pod Up - POD/row,col,UP	11-7
 APPENDIX A Post-processor Command Summary		A-1
 APPENDIX B Special Considerations		B-1
B.1	Introduction.....	B-1
B.2	AC Style Rotary Head	B-1
B.2.1	Enable/Disable The Automatic Look Ahead Feature - ALIGN/AXIS	B-5
B.3	Look Ahead Feature For Machine With a C-axis Table.....	B-6
B.4	Tool Axis Adjustment Near Apex Point Of AC-style Rotary Axes Machine - LINTOL/ADJUST.....	B-6
 INDEX		1

CHAPTER 1 Beginning and Ending Sequences

1.1 Part Description - PARTNO

Syntax: **PARTNO***text*

The *PARTNO* card is used to describe the part being programmed. It is usually the first block to the Control Tape.

The text string of this command cannot be more than 66 characters if the clfile is of the file type **NCL** binary. The text string of this command cannot be more than 80 characters if the clfile is of the file type APT Source textual.

A maximum of 100 PARTNO is allowed in a single input cl file.

Example:

```
PARTNO 138876-01 POSITION 1 OF 3 REEL 3 OF 7
```

Identifies the part program and optionally outputs a *PARTNO* card to the Control Tape.

1.2 The MACHIN Card - MACHIN

Syntax: **MACHIN/PWORKS**,*n1* [...] *n10* [, **OPTION**, *m1*, *v1* [...]]
psname

or

```
PPRINT MACHIN/PWORKS,n1 [...] n10 [, OPTION, m1, v1 [...] ]  
psname
```

The *MACHIN* command specifies which [Machine Descriptor File\(s\)](#) (*MDF*) will be used to process the input clfile. This command must be contained in the part program file and not in a **PostMacro** definition file, since the Macro object file will not be loaded until a *MACHIN* command is encountered.

The *MACHIN* command should appear only once in the part program file, as **PWorks** will only process the first one, all others will be ignored. The *MACHIN* command must be present or the **PWorks** run time machine option must be used, otherwise an error will occur and the input clfile will not be processed.

‘*n*’ defines which *MDF* file, in the form *PWORKS_n.MDF*, to use and can be any positive number. You must specify at least one machine number and may specify up to ten. ‘*psname*’

defines which *MDF* file, in the form *PWORKS_pstname.MDF*, to use. **PWorks** will process the input clfile and create separate output files for each of the machine specified. The *MACHIN* run time option can be used to override the *MACHIN/PWORKS* command.

‘*OPTION*’ specifies that the remaining parameters will be used to define machine options. These options override the default settings defined in the **MPost** utility. Following is a list of available options.

The “*PPRINT MACHIN*” statement is to allow CAM systems that do not support the *MACHIN* card to still specify the post-processor name in the clfile

m	v	Description
-	-	-----
1	1	Input is in inches
	2	Input is in millimeters
2	1	Output is in inches
	2	Output is in millimeters
3	n	Print Descriptor File (PDF) file to use when formatting the print file.
4	n	Specifies the program number to output to the Control Tape. If the program number is entered here instead of in the <i>POSTN</i> command, then it will be output prior to the <i>PARTNO</i> card. It will also be incremented by 1 and output whenever the Control Tape file is broken up into multiple files due to the BREAK command.
5	n	Specifies the maximum allowable length of an MCD block. A value in the range of 10 to 512 is supported.
6	n	Specifies whether linearized points should be generated in the machine simulation file. These extra points are usually not necessary when running the machine simulator.

A value of “1” breaks up a simulation motion so that each of the machine axes moves simultaneously towards their final position. The move is broken up so that simulating the motion will show how the move should look on the machine. This option is typically used when playing back the motion or performing material removal simulation.

A value of “2” does not break up simulation motion. The motion is output as it is to the MCD file. This option is typically used as input to machine simulation. The

movement of the simulation machine axes will mimic the actual movement at the machine.

A value of “3” breaks up a simulation motion so that the tool tip will move in a straight line while the rotary axes move towards their final position. This option is typically used for machine controls that support tool tip programming and can be used for simple simulation and machine simulation.

7	n	Specifies whether to suppress the output of the punch file when a warning, error, or fatal is encountered. A value of “0” specifies not to suppress the punch file. A value of “1” specifies to suppress the punch file on a warning, error, or fatal. A value of “2” specifies to suppress the punch file on an error or fatal. A value of “3” specifies to suppress the punch file on a fatal.
8	n	Specifies whether to apply the TRANS/mx transformation matrix to the tool end points and tool axis vectors output to the simulation file. A value of “1” will apply the transformation matrix to the tool end points and tool axis vectors output to the simulation file. A value of “0” will not apply the transformation. The default is “0”.

Example:

MACHIN/PWORKS,312,315,OPTION,3,10,4,1028

PWorks will process the input part program for both *PWORKS_312.MDF* and *PWORKS_315.MDF*. Separate output files will be created for each of the Machine Descriptor Files. The Print Descriptor File ‘PWORKS_10.pdf’ will be use as the print file format, overriding the *PDF* file specified using the **MPost** routine. The program number register will be output as the first block of the Control Tape and will contain a value of 1028.

1.3 Include External File - PPRINT INCLUD

Syntax: **PPRINT INCLUD/file**

PostWorks has the ability to include an external cl file or APT source file into the input clfile or APT source file.

Any format of input file (**NCL**, *MasterCam*, APT Source, *Catia*, etc.) can be included into an input file of any type. This feature allows the user to program a portion of the part in a different system than the main CAM system and to combine these multiple programs into a single program.

If the filename is too long to fit on a single PPRINT command it can be continued onto a second command by using the tilde (~) continuation character. For example:

```
PPRINT INCLUD/C:\users\horatio\part_programs\810019_3\~  
PPRINT 810019_setup_1.as
```

Included files can be nested inside of other included files.

1.4 Rewind Stop - TMARK/AUTO

Syntax: **TMARK/AUTO**

The *TMARK/AUTO* command outputs a rewind stop code. The rewind stop code will usually be output by **PWorks** at the beginning of the program.

1.5 Outputting Leader - LEADER

Syntax: **LEADER/n**

The *LEADER* command outputs 'n' inches/mm of leader. Inches or millimeter will be selected based on the default input units. **PWorks** can be set to automatically output the beginning and trailing leader, using **MPost** routine.

Example:

```
LEADER/24
```

Outputs 24 inches of leader (assuming the input units are set to INCH).

1.6 Part Program Number - POSTN

Syntax: **POSTN/n**

This version of the *POSTN* command will output an identifying program number to the Control Tape. the program number is usually output at the very beginning of the program on machines that support this feature. The *MACHIN/PWORKS,OPTION,4* command can also be used to output the program number.

Example:

```
POSTN/88
```

Outputs the identifying program number register with a value of 88.

1.7 Rewinding The Control Tape - REWIND

Syntax: **REWIND**

The *REWIND* command outputs a tape rewind code, usually M30. This code rewinds the Control Tape to the beginning of the program. *REWIND* is usually placed just prior to the *FINI* command.

1.8 Physical End-of-Program - FINI

Syntax: **FINI**

The *FINI* command signals the end of the clfile and is the last clfile record to be processed. An End-of-Tape sequence will be performed by **PWorks** after reading the *FINI* command.

1.9 Logical End-of-Program - END

Syntax: **END**

The *END* command processes exactly the same as the *FINI* command, except **PWorks** is not terminated and will continue to process the clfile.

CHAPTER 2 Control Tape Commands

2.1 Sequence Numbers - SEQNO

Syntax: **SEQNO/ON**
OFF
n [, INCR, i [, m]]

The *SEQNO* command controls the output of sequence numbers. ‘*ON*’ resumes the sequencing of Control Tape blocks. ‘*OFF*’ disables the sequencing of blocks.

‘*n*’ specifies the starting sequence number. If ‘*n*’ is the only parameter specified, then only the next block will contain a sequence number. ‘*INCR, i*’ specifies the value to increment the sequence numbers by. ‘*m*’ defines the frequency of blocks which will contain sequence numbers.

Example:

SEQNO/ON

Resumes/initiates the output of sequence numbers. The parameters (n, i, m) specified in the last *SEQNO* command or in the ***MPost*** utility will be reinstated.

SEQNO/10,INCR,5,2

Enables the output of sequence numbers. The tape block following the *SEQNO* command will contain a sequence number of 10. Every other block will contain a sequence number, with the value being incremented by 5.

```
N10 G01 X1.3 Y2.588 F10.0$  
X1.34 Y2.632$  
N15 G00 Z8.2$
```

2.2 Alignment Blocks - TMARK

Syntax: **TMARK [/] [n] [[,] NOW]**

The *TMARK* command outputs an alignment block. The format of the alignment block is defined in the ***MPost*** utility and usually contains enough codes to allow the machine operator to start the machine at this block.

‘*n*’ specifies the sequence number value for the alignment block. This value will not interrupt the normal sequencing of Control Tape blocks. If ‘*n*’ is not specified, then the next sequence number will be used.

‘*NOW*’ specifies that the alignment block will be output immediately. If ‘*NOW*’ is not specified, then the alignment block will be output with the next Control Tape block.

Example:

TMARK/100,NOW

Outputs an alignment block immediately. The sequence number of the alignment block will be 100. The codes contained in this block are specified in the ‘[Non-motion alignment block](#)’ User Defined Block.

2.3 Optional Skip Blocks - OPSKIP

Syntax: **OPSKIP/ON[,n1[...]n9]**
OFF

The *OPSKIP* command enables the programmer to specify blocks which may or may not be executed at the machine

‘*ON*’ specifies that the Control Tape blocks following this command will be preceded by the optional skip character(s), usually a slash (/). These blocks will be ignored when the optional skip feature at the machine has been turned on.

‘*OFF*’ disables the optional skip output.

Some machines allow for multiple levels of optional skip codes. For these machines, ***PostWorks*** supports 9 levels of skip codes. By entering one or more numbers after the mode parameter (*ON*, *OFF*), the programmer can enable or disable the various levels of optional skip blocks. The output optional skip codes will contain the opskip character(s) and a single digit number, ‘/I’. for example.

An *OPSKIP/OFF* command without any level values will disable all levels of optional skip blocks.

Example:

OPSKIP/OFF

Disables the output of the optional skip character(s) on output blocks.

OPSKIP/ON,2,4

Enables the output of levels 2 and 4 optional skip codes until an *OPSKIP/OFF* or *OPSKIP/OFF,2,4* command is encountered. The output will be similar to the following.

/2 /4 N... X... Y...\$

2.4 Tape Block Checksumming - CHECK/LENGTH

Syntax: **CHECK/LENGTH, ON**
OFF

The *CHECK/LENGTH* command controls the output of a [checksum code](#) on each tape block. When checksumming is enabled, **PWorks** will total the decimal values of each character in the tape block and will add a code to the block which contains the result of this summation.

Checksumming is primarily used to verify the contents of the Control Tape when transmitting it to the machine control. Verify that the machine supports checksumming before using this command.

‘*LENGTH*’ is required for proper syntax.

‘*ON*’ enables the output of checksumming codes. ‘*OFF*’ disables checksumming.

Example:

```
CHECK/LENGTH,ON
```

Enables checksumming and the output of summation codes on each tape block.

2.5 Inserting Tape Blocks - INSERT

Syntax: **INSERTtext**

The *INSERT* command is used to output a Control Tape block. The text contained in the *INSERT* command will be output exactly as is to the Control Tape. **PWorks** does not alter or check the validity of the input text, so the *INSERT* command should not be used to output a code which is obtainable using another post-processor command.

The text string of this command cannot be more than 66 characters if the clfile is of the file type **NCL** binary. The text string of this command cannot be more than 80 characters if the clfile is of the file type APT Source textual.

Example:

```
INSERTG00G94G90X0Y0Z10.0$
```

Outputs the tape block ‘G00G94G90X0Y0Z10.0\$’ to the control Tape.

2.6 Miscellaneous (M) Codes - AUXFUN

Syntax: **AUXFUN/n [[,] NOW]**
NEXT

The *AUXFUN* command causes the miscellaneous code ‘n’ to be output. **PWorks** will not check the validity of the M-code, except for the value of ‘n’, therefore it is recommended that an M-code which can be output using another post command, not be output with the *AUXFUN* command.

‘*NOW*’ specifies that the miscellaneous code be output in a block by itself. ‘*NEXT*’ specifies that the miscellaneous code be output in the next block. ‘*NEXT*’ is the default.

Also see the *FORCE* command in the **PostMacro** Reference Manual.

Example:

```
AUXFUN/6,NOW
```

Outputs the miscellaneous code M06 in a block by itself.

2.7 Preparatory (G) Codes - PREFUN

Syntax: **PREFUN/n [[,] NOW]**
NEXT

The *PREFUN* command causes the preparatory code ‘n’ to be output. **PWorks** will not check the validity of the G-code, except for the value of ‘n’, therefore it is recommended that a G-code which can be output using another post command, not be output with the *PREFUN* command.

‘*NOW*’ specifies that the preparatory code be output in a block by itself. ‘*NEXT*’ specifies that the preparatory code be output in the next block. ‘*NEXT*’ is the default.

Also see the *FORCE* command in the **PostMacro** Reference Manual.

Example:

```
PREFUN/73,NEXT
```

Outputs the preparatory code G73 in the next tape block.

2.8 Operator Messages - DISPLY

Syntax: **DISPLY/ON**
OFF
AUTO
ALIGN, n
text_string

The *DISPLY* command enables the programmer to output operator messages to the Control Tape using the *PPRINT* statement. The *DISPLY* command cannot be used to define a *Post Macro*.

‘ON’ enables the output of *PPRINT* statements as operator messages. This statement is modal and will stay in effect until *DISPLY/OFF* is encountered. ‘*AUTO*’ outputs the current date and time in an operator message block.

‘ALIGN, n’ allows the user to specify the minimum length “n” of the actual message (text between the start message and end message characters) that will be output. A value of between 1 and 66 will line up the output message blocks at this column, plus the message and end characters. A value of 0 will disable message block alignment.

The ‘DISPLY/text_string’ command is the same as the ‘PPRINTtext_string’ command. This command cannot be used inside any *Post Macro*.

Example:

DISPLY/AUTO

Outputs the current date and time in an operator message block. For example, ‘N1(MSG,RUNDATE: 28-MAR-2000 10:14:38)\$’.

2.9 Print File Messages - PPRINT, TPRINT

Syntax: **PPRINTtext**
or
TPRINTtext

The *PPRINT* or *TPRINT* command allows the programmer to insert comments into the print file. Any characters following *PPRINT* or *TPRINT* will be listed in the print file.

The text of the *PPRINT* or *TPRINT* command will also be output as an operator message if *DISPLY/ON* is in effect.

The *PPRINT* and *TPRINT* commands are virtually the same except *TPRINT* cannot be used to define a *Post Macro* or used inside any *Post Macro*. If a *PPRINT Post Macro* exists, it will apply to the *TPRINT* command also.

The text string of this command cannot be more than 66 characters if the clfile is of the file type **NCL** binary. The text string of this command cannot be more than 80 characters if the clfile is of the file type APT Source textual.

Example:

```
PPRINT ** SEQUENCE 10 TOOL #1 1,0 X .5 END MILL **
```

2.10 User Specified Tape Break - **BREAK**

Syntax: **BREAK**

The *BREAK* command, without any parameters specified, will output a tape break sequence whenever the tape length exceeds the maximum length specified in the *BREAK/ON,n* command.

This version of the *BREAK* command is only valid when *BREAK/ON* is in effect and will be ignored if tape breaks have not been enabled or if the tape break mode is other than *ON* (*AUTO*, *TIMES*, *DELTA*). See the *BREAK/mode* command syntax for a description of the *BREAK/ON* command and tape break sequences.

Example:

```
BREAK
```

Outputs a tape break sequence at this point in the program only when *BREAK/ON,n* is in effect and the current tape length exceeds the maximum specified by 'n'.

2.11 Disabling Tape Breaks - **BREAK/OFF**

Syntax: **BREAK/OFF**

The *BREAK/OFF* command disables the tape break feature. No checking will be done for exceeding tape length, machining time or machining distances. See the *BREAK/mode* command for a description of tape break sequences.

Example:

```
BREAK/OFF
```

Disable all checking for the automatic output of tape break sequences.

2.12 Tape Break Sequences - BREAK/mode

Syntax: **BREAK/ON** [*,n*] [*,RAPTO,r*] [*,FEDTO,f*]
 AUTO
 TIMES
 DELTA

The *BREAK/MODE* command is used to control the output of tape break sequences. Tape breaks are usually used for machines that have a limited amount of memory for Control Tape storage.

‘*ON*’ specifies that a tape break will occur whenever the tape length exceeds the maximum allowed (specified by ‘*n*’) and a *BREAK* command without any parameters is encountered. The *BREAK/ON* command in conjunction with the *BREAK* command allows the programmer to specify the exact location within the part program where a tape break can occur. The following tape break sequence will usually take place when *BREAK/ON* is in effect.

- 1) Output an End-of-Tape mark and open a new punch file.
- 2) Issue a *TMARK* command.

‘*AUTO*’ specifies that a tape break will occur whenever the tape length exceeds the maximum allowed (specified by ‘*n*’). This mode will cause **PWorks** to automatically check the tape length after each block is output and to output a tape break whenever the tape length exceeds the maximum allowed. The *BREAK/AUTO* command does not require a *BREAK* command with no parameters to perform a tape break sequence. The following tape break sequence will usually take place when *BREAK/AUTO* is in effect.

- 1) Issue *RETRCT* command.
- 2) Output an End-of-Tape mark and open a new punch file.
- 3) Issue a *TMARK* command.
- 4) Issue a *PLUNGE* command.

‘*TIMES*’ specifies that a tape break will occur whenever the machining time exceeds the maximum allowed (specified by ‘*n*’). The *BREAK/TIMES* command follows the same rules as the *BREAK/AUTO* command, except that the machining time is used for comparison, instead of the tape length.

‘*DELTA*’ breaks the tape whenever the machining distance of the linear axes exceeds the maximum allowed, as specified by ‘*n*’. The *BREAK/DELTA* command will also follow the same rules as the *BREAK/AUTO* command, except that machining distance will be used. Both machining distance and time will only be considered for motion blocks which move at the programmed feed rate, moves made in rapid mode will not be considered. Also, both of these will be reset whenever End-of-sequence (*STOP*, *LOADTL*, etc.) is encountered.

All modes stay in effect until explicitly changed in another *BREAK/mode* command or until *BREAK/OFF* is encountered.

‘*n*’ defines the maximum tape length before a tape break will occur, when either ‘*ON*’ or ‘*AUTO*’ is active. When ‘*TIMES*’ is in effect, ‘*n*’ defines the maximum machining time allowed. When ‘*DELTA*’ is in effect, ‘*n*’ defines the maximum linear axes delta movement allowed. If ‘*n*’ is omitted from this command, then the last specified value for the active mode will be used.

‘*RAPID,r*’ specifies the feed rate at which to retract the tool when a *RETRACT* command is issued in a tape break sequence. A value of 0 will retract the tool at the rapid traverse rate.

‘*FEDTO,f*’ specifies the feed rate at which to reposition the tool when a *PLUNGE* command is issued in a tape break sequence. A value of 0 will reposition the tool at the rapid traverse rate.

The example tape break sequences shown above can be modified using the ***MPost*** utility. If a *TAPBRK Macro* is defined, then ***PWorks*** will automatically call this *Macro* at a tape break sequence and will not perform any of the functions as set up in the ***MPost*** utility for tape breaks.

The following commands will have an effect on tape breaks. See the appropriate sections for a description of these commands.

<i>MPost</i>	<i>PostMacro</i>	<i>PWorks</i>
-----	-----	-----
Tape Break Sequences	TAPBRK/Macro BREAK/PCHFILE	BREAK BREAK/OFF CLRSRF

Example:

BREAK/ON,100

Breaks the tape into multiple sections whenever the tape length exceeds 100 (ft/m) and a *BREAK* command with no parameters is encountered.

BREAK/TIMES,45,RPATO,0,FEDTO,10

Breaks the tape into multiple sections whenever the machining time (not including rapid moves) exceeds 45 minutes. The tool will retract to the clearance level as defined by the *CLRSRF* command at the rapid rate, a tool break sequence will be output, and the tool will plunge back to its original position at a feed rate of 10 FPM. It is assumed that the retract feature for tape break sequences has been enabled.

BREAK/AUTO

Breaks the tape into multiple sections whenever the tape length exceeds the value specified in either the last *BREAK/AUTO* command or in the ***MPost*** utility.

CHAPTER 3 Tool Change Sequences

3.1 Selecting A Mill Tool - SELCTL or SELECT

Syntax: **SELCTL**/*tn* [[, **LENGTH**], *tl*] [[, **LARGE**] [[, **HIGH**] \$
SELECT **AUTO** [, *lastn*] **SMALL** **LOW**

[[, **MODIFY**]
NOW
NEXT, *tm*

This command causes the machine to select the next tool to be loaded and is only valid with the Mills which have an automatic tool changer and are able to preselect the next tool. Either SELCTL or SELECT can be used. However, **MPost** Menu item 13.1.5 “[Support SELECT/TOOL and LOAD/TOOL commands](#)” must be set to **YES** before SELECT can be used.

If the machine does have tool selection support and you do not issue a *SELCTL* prior to a *LOADTL*, then **PWorks** will internally issue the *SELCTL* and output a warning message.

‘*tn*’ specifies the tool number to select. ‘*AUTO*’ will search the input clfile for the next *LOADTL* statement and automatically select this tool. ‘*lastn*’ when specified with the ‘*AUTO*’ parameter defines the tool number to select as the last tool of the program. When a *LOADTL* command is found during the search, then ‘*lastn*’ is ignored. If *FINI* is reached without locating a *LOADTL* command, then ‘*lastn*’ will be used as the selected tool. An error message will be output if *FINI* is encountered and ‘*lastn*’ is not specified.

‘*LENGTH,tl*’ specifies the tool length for this tool. The word ‘*LENGTH*’ is optional. The tool length ‘*tl*’, if selected in the **MPost** utility, will be applied to the output coordinates.

‘*LARGE*’ specifies that the large size gripper be used to pick up the tool. ‘*SMALL*’ selects the small size gripper. This parameter is only valid with [machines which support two sizes of tool grippers](#) and will be ignored if the machine does not. If this parameter is not specified, then the last used gripper size will be selected.

‘*HIGH*’ selects the high speed (main) spindle to load the tool into. ‘*LOW*’ selects the low speed (axial) spindle. This parameter is only valid with [machines which have two spindles](#) and will be ignored if the machine does not. If this parameter is not specified, then the last spindle selection will be used.

‘*MODIFY*’ specifies that the input values are for reference only and will not output a tool selection, though, all parameters will be stored internally within **PWorks**. ‘*NOW*’ outputs a tool selection in a block by itself. The tool selection codes will normally be output with the following block. ‘*NEXT,tm*’ specifies that the tool selection codes will be output in the next motion block consisting of at least ‘*tm*’ seconds of machining time. If a *LOADTL* statement is encountered prior

to the tool selection codes being output, then the tool selection codes will be forced out immediately before the tool change sequence.

Example:

SELECTL/1,LENGTH,5.2,LARGE,HIGH

Picks up tool #1 with a programmed length of 5.2 utilizing the large gripper. This tool will be loaded into the high speed spindle at the next tool change sequence.

SELCTL/3,4.3,NOW

Select tool #3 as the next tool to be loaded, with a programmed length of 4.3. Outputs the tool selection codes in a block by themselves.

SELCTL/AUTO,NEXT,15

Select the tool specified on the next *LOADTL* statement, but will not output the tool selection codes until a move that takes at least 15 seconds is programmed.

3.2 Loading A Mill Tool - **LOADTL** or **LOAD**

Syntax: **LOADTL**/*tn* [[, **LENGTH**] , *tl*] [[, **OFFSET** , *h* [, *d*]] [[, **LARGE**] **\$**
LOAD **SMALL**

[[, **HIGH**] [[, **AUTO**]

LOW **MODIFY**

MANUAL

The *LOADTL* command causes a [tool change sequence](#) to be output to the Control Tape. Either *LOADTL* or *LOAD* can be used. However, **MPost** Menu item 13.1.5 “[Support SELECT/TOOL and LOAD/TOOL commands](#)” must be set to **YES** before *LOAD* can be used.

LOADTL is only valid with Mills, use the [TURRET](#) command for a tool change sequence on Lathes.

‘*tn*’ specifies the tool number to load. ‘*LENGTH,tl*’ specifies the tool length for this tool. The word ‘*LENGTH*’ is optional. The tool length ‘*tl*’, if selected in the **MPost utility**, will be applied to the output coordinates.

‘*OFFSET*’ specifies [compensation registers](#) to use with the newly loaded tool. ‘*h*’ specifies a tool length offset register and ‘*d*’ specifies a cutter compensation register. ‘*OFFSET*’ is usually only used when tool compensation is automatically enabled at a tool change with the tool number and the compensation register as part of the same code. For example, ‘*Ttthh*’, where ‘*tt*’ is the tool number and ‘*hh*’ is the compensation register.

‘*LARGE*’ specifies that the large size gripper be used to load the tool. ‘*SMALL*’ selects the small size gripper. This parameter is only valid with machines which support two sizes of tool grippers and will be ignored if the machine does not. If this parameter is not specified, then the last used gripper size will be selected.

‘*HIGH*’ selects the high speed (main) spindle to load the tool into. ‘*LOW*’ selects the low speed (axial) spindle. This parameter is only valid with machines which have two spindles and will be ignored if the machine does not. If this parameter is not specified, then the last spindle selection will be used.

‘*AUTO*’ is the default modifier and outputs an automatic tool change, ‘*MODIFY*’ specifies that the input values are for reference only and will not output a tool change sequence, though, all parameters will be stored internally within **PWorks**. ‘*MANUAL*’ outputs a manual tool change sequence.

Example:

```
LOADTL/1,LENGTH,5.2,LARGE,HIGH
```

Loads tool #1 with a programmed length of 5.2 into the high speed spindle utilizing the large size gripper. An automatic tool change will be output.

```
LOADTL/3,4.3,OFFSET,103,MODIFY
```

Internally selects tool #3 with a programmed tool length of 4.3 as the current tool. Also selects the tool length offset register 103. A tool change sequence will not be output.

```
LOADTL/8,MANUAL
```

Outputs a manual tool change sequence and internally stores tool number 8 as the current tool.

3.3 Loading A Special Head - **LOADTL/SET**

Syntax: **LOADTL/SET,parm1,tool1[...][,parm5,tool5]**

This *LOADTL* style command informs the operator to load a special head into the spindle and is typically placed in the G\$MAIN macro.

The *LOADTL/SET* command allows the user to define up to 5 post processor words allowed as the tool number on the *LOADTL* statement designating the head type to load. “*parm?*” can be any valid post processor vocabulary word and “*tool?*” specifies the corresponding tool number to output when this head is loaded.

The standard *LOADTL* statement will be updated to accept the post processor words defined in this command as a valid tool number and any parameters after the post processor word will be ignored.

Example:

LOADTL/SET,MILL,0,DRILL,99

The command “*LOADTL/DRILL,90*” will output T99 M06 to the punch file and the numerical parameter value “90” will be ignored.

3.4 Unloading A Mill Tool - *UNLOAD*

Syntax: ***UNLOAD/TOOL [, tn]***

The *UNLOAD* command causes the machine to unload the tool currently loaded in the spindle and is only valid with Mills which have an automatic tool changer and are able to preselect the tool. If the machine requires an unload sequence to be output prior to loading a new tool, then *LOADTL* will automatically output the codes to unload the tool. Therefore, *UNLOAD* is rarely used, and usually only at the end of the program.

‘*TOOL*’ is required for proper syntax. ‘*tn*’ specifies the tool number to unload. If ‘*tn*’ is not specified, then the tool number currently loaded in the spindle will be used.

Example:

UNLOAD/TOOL

Unloads the tool currently in the spindle.

UNLOAD/TOOL,5

Unloads the tool currently in the spindle and places it in tool holder #5.

3.5 Selecting A Lathe Tool - *TURRET*

Syntax: ***TURRET/tn [[, RADIUS] , r] [[, OFFSET , h [, d]] [[, ADJUST , z , x] \$***

***[, FRONT] [, CLW] [, AUTO]
REAR CCLW MODIFY
MANUAL***

The *TURRET* command causes a [tool change sequence](#) to be output to the Control Tape. *TURRET* is only valid with Lathes, use the [LOADTL](#) command for a tool change sequences in Mills.

'*tn*' specifies the tool number to load. '*RADIUS,r*' specifies the tool nose radius. The word '*RADIUS*' is optional and the tool nose radius '*r*' will only be used for reference.

'*OFFSET*' specifies [compensation registers](#) to use with the newly loaded tool. '*h*' specifies a tool compensation register and '*d*' specifies a cutter compensation register. '*OFFSET*' is usually only used when tool compensation is automatically enabled at a tool change or the tool number and compensation register are part of the same code. For example, '*Ttthh*', where '*tt*' is the tool number and '*hh*' is the compensation register.

'*ADJUST*' allows you to apply translation values for both the Z and X axes with this tool only. '*z*' specifies the amount to add to the Z-axis and '*x*' is the amount to add to the X-axis. '*x*' is always given as a radial value.

'*FRONT*' specifies that the selected tool is in the front turret. '*REAR*' selects the rear turret and is only valid on lathes with two turrets. An automatic translation, as specified in the [MPost utility](#), will be applied to the X-axis when '*REAR*' is selected.

'*CLW*' specifies that the turret should rotate in the clockwise direction when loading the selected tool. '*CCLW*' specifies a counter-clockwise direction.

'*AUTO*' is the default modifier and outputs an automatic tool change. '*MODIFY*' specifies that the input values are for reference only and will not output a tool change sequence, though, all parameters will be stored internally within **PWorks**. '*MANUAL*' outputs a manual tool change sequence.

Example:

TURRET/1,RADIUS,.05,FRONT,CLW

Selects tool #1 with a tool nose radius of .05 as the current tool. This tool is located in the front turret and the turret will rotate in the clockwise direction to select this tool. An automatic tool change sequence will be output.

TURRET/3,.08,OFFSET,103,REAR,MODIFY

Internally selects tool #3 with a tool nose radius of .08 as the current tool. This tool is located in the rear turret. A tool change sequence will not be output.

TURRET/8,ADJUST,.5,0,MANUAL

Selects tool #8 with an unknown tool radius as the current tool. This tool is located in the last specified turret. .5 will be added to the Z-axis until another *TURRET* command cancels this adjustment. A manual change sequence will be output.

3.6 Tool Description - PPRINT TLN

Syntax: **PPRINT TLN,tool-description**

A description of the tool currently loaded is specified using the '*PPRINT TLN*' format of the *PPRINT* statement. 'tool-description' is a text string which describes the current tool and can be referenced in both the [Print Descriptor File \(PDF\)](#) and [Punch Header File \(PHF\)](#) by using the *TLNAME* variable.

Example:

```
PPRINT TLN,.75 x .15" or End Mill
```

3.7 Program Stop - STOP

Syntax: **STOP**

The *STOP* command outputs a [program stop code](#), usually M00, to the Control Tape.

3.8 Optional Stop - OPSTOP

Syntax: **OPSTOP**

The *OPSTOP* command outputs an [optional stop code](#), usually M01, to the Control Tape.

3.9 Coolant Control - COOLNT

Syntax: **COOLNT/ON**

OFF
FLOOD
MIST
AIR

The *COOLNT* command outputs the [proper codes](#) to turn the coolant on and off. '*ON*' turns the last specified coolant type on. '*OFF*' turns off the coolant. '*FLOOD*' turns the flood coolant on. '*MIST*' turns the mist coolant on. '*AIR*' turns the air coolant on.

Example:

```
COOLNT/FLOOD
```

Turns the flood coolant on at the machine.

3.10 Tool Length Offsets - TOOLNO/ADJUST

Syntax: **TOOLNO/ADJUST,ON**
OFF

The *TOOLNO/ADJUST* command is used to output tool [length offset blocks](#) to the Control Tape. The proper coding will be output to allow the programmer to enable/disable tool length offset at the control. Tool length offsets allow the control to compensate for variations in the programmed tool length.

This command syntax should only be used for machines which support tool length offsets and do not let you specify a register number or direction for tool length offsets. Use the *TOOL/ADJUST,n* expanded syntax for machines which allow for more control in specifying tool length offsets.

‘ON’ enables tool length offset compensation at the machine control. ‘OFF’ disables tool length offsets.

Example:

TOOL/ADJUST,ON

Enable tool length offset compensation at the machine control.

3.11 Expanded Tool Length Offset Control - TOOLNO/ADJUST

Syntax: **TOOLNO/ADJUST,n[,d][,PLUS][,XAXIS[,pos]]\$**
MINUS
[,YAXIS[,pos]][,ZAXIS[,pos]][,XYPLAN][,NOW]
YZPLAN NEXT
ZXPLAN

The *TOOLNO/ADJUST* command is used to output tool length offset blocks to the Control Tape. The [proper coding](#) will be output to allow the programmer to enable tool length offsets at the control. This is the expanded syntax for *TOOLNO/ADJUST* and allows you to specify a register number and direction for tool length offsets.

‘n’ specifies the register number which contains the tool length offset. ‘d’ optionally specifies a register number which contains the offsets for cutter compensation. ‘d’ is usually only specified when the tool length offset and cutter compensation registers are part of the same code. For example, ‘*Httcc*’, where ‘*tt*’ is the tool length offset register and ‘*cc*’ is the cutter compensation register.

‘*PLUS*’ selects a positive direction for compensation. ‘*MINUS*’ selects a negative direction for compensation. This parameter may not be supported by your particular machine and will be ignored in this case.

‘*XAXIS*’, ‘*YAXIS*’, and ‘*ZAXIS*’ outputs the specified axis code in the offset block. One, two or all three of these words can appear in this command. If none of these words are present in the command, then either no axes will be output with the tool offset block or the next programmed axis positions will be used, depending on whether or not ‘*NOW*’ or ‘*NEXT*’ have been specified.

‘*pos*’ specifies the value to be output for the corresponding axis. If this value is not specified then a default value of 0 will be used.

‘*XYPLAN*’ selects the XY machining plane for tool length compensation, ‘*ZXPLAN*’ selects the ZX machining plane and ‘*YZPLAN*’ selects the YZ machining plane. This parameter should only be specified when the machining plane code (usually G17, G18 and G19) needs to be output with the tool length offsets block. If this parameter is not specified, then the default machining plane as defined by the *MODE/TOOL* command will be used.

‘*NOW*’ outputs the tool length offset codes in a block by themselves. If any of the ‘*-AXIS*’ parameters are specified, then the tool length offset codes will automatically be output in a block by themselves. ‘*NEXT*’ outputs the tool length offset codes in the next motion block which contains movement in the tool length compensation axis (usually the Z-axis). If neither ‘*NOW*’, ‘*NEXT*’ nor any of the ‘*-AXIS*’ parameters are specified, then the tool length offset codes will be output in the following block.

Example:

TOOLNO/ADJUST,10,PLUS,ZAXIS

Enables tool length offsets using register #10 in the positive direction. The tool length codes will be output in a block by themselves with a Z-axis value of 0.

TOOLNO/ADJUST,15,NEXT

Enables tool length offsets using register #15. The tool length offset codes will be output in the next motion block containing movement in the tool length compensation axis.

TOOLNO/ADJUST,13,XAXIS,1,YZPLAN

Enables tool length offsets in the YZ plane (along the X-axis) using register #13. The tool length offset codes will be output in a block by themselves with a X-axis value of 1.

3.12 Whether To Add Tool Length To The Output Or Not

Syntax: **TOOLNO/OFFSETL, ON**
OFF

The *TOOLNO/OFFSETL* command is used to control whether the tool length is added to the output coordinates. 'ON' will add the programmed tool lengths to the output coordinates, 'OFF' will not.

3.13 Tool Machine Time Calculation - TOOLNO/TIMES

Syntax: **TOOLNO/TIMES, ONCE**
ALL

The *TOOLNO/TIMES* command is used to control whether to count each tool usage as a separate tool even if the tool numbers are the same. Typically **PWorks** will search the previously loaded tools in an attempt to match the current tool.

'ALL' will treat each tool as a separate tool whether or not it matches a previously loaded tool. Machine times will be accumulated separately each time the same tool is loaded.

'ONCE' will not treat each tool as a separate tool if it matches a previously loaded tool. Machine times will be accumulated together each time the same tool is loaded.

3.14 Fixture Offsets - CUTCOM/ADJUST

Syntax: **CUTCOM/ADJUST, ON**
OFF

The *CUTCOM/ADJUST* command is used to output fixture offset blocks to the Control Tape. The [proper coding](#) will be output to allow the programmer to enable/disable fixture offsets at the control. Fixture offsets allow the control to compensate for deviations in loading the part onto the fixture or for loading multiple parts onto the fixture.

This command syntax should only be used for machines that support fixture offsets and do not let you specify a register number or direction for fixture offsets. Use the [CUTCOM/ADJUST,n](#) expanded syntax for machines which allow for more control in specifying fixture offsets.

'ON' enables fixture offset compensation at the machine control. 'OFF' disables fixture offsets.

Example:

CUTCOM/ADJUST,ON

Enable fixture offset compensation at the machine control.

3.15 Expanded Fixture Offset Control - CUTCOM/ADJUST

Syntax: **CUTCOM/ADJUST**,*n* [, **PLUS**] [, **XAXIS** [, *pos*]] [, **YAXIS** [, *pos*]]
MINUS

[, **ZAXIS** [*pos*]]

The *CUTCOM/ADJUST* command is used to output fixture offset blocks to the Control Tape. The [proper coding](#) will be output to allow the programmer to enable fixture offsets at the control. This is the expanded syntax for *CUTCOM/ADJUST* and allows you to specify a register number and direction for fixture offsets.

‘*n*’ specifies the register number which contains the fixture offset.

‘*PLUS*’ selects a positive direction for compensation. ‘*MINUS*’ selects a negative direction for compensation. This parameter may not be supported by your particular machine and will be ignored in this case.

‘*XAXIS*’, ‘*YAXIS*’, and ‘*ZAXIS*’ outputs the specified axis code in the offset block. One, two or all three of these words can appear in this command. If none of these words are present in the command, then either no axes will be output with the fixture offset block or the next programmed axis positions will be used, depending on options selected in the [MPost utility](#).

‘*pos*’ specifies the value to be output for corresponding axis. If this value is not specified then a default value of 0 will be used.

Example:

CUTCOM/ADJUST,10,PLUS,ZAXIS

Enables fixture offsets using register #10 in the positive direction. The fixture offset codes will be output in a block by themselves with a Z-axis value of 0.

CUTCOM/ADJUST,15

Enables fixture offsets using register #15.

CHAPTER 4 Feeds and Speeds

4.1 Feed Rates - FEDRAT

Syntax: **FEDRAT/ [IPM ,] [n]**
IPR
MMPM
MMPR
INVERS

The *FEDRAT* command defines the feed rate at which the Feed Rate Control Point moves. ‘n’ specifies the desired feed rate.

‘*IPM*’ and ‘*MMPM*’ select the Feed per Minute feed rate input mode. ‘*IPR*’ and ‘*MMPR*’ select the Feed per Revolution feed rate input mode. A new *FPM* feed rate will be calculated when *FPR* feed rates are in effect and the spindle speed changes. ‘*INVERS*’ selects the *FPM* feed rate input mode and Inverse Time feed rate output.

All specifiers stay in effect until explicitly cancelled by another specifier.

Example:

FEDRAT/IPM,40

All cutting (non-rapid) moves following this command will move at a rate of 40 inches/millimeters per minute.

FEDRAT/.05,IPR

All cutting (non-rapid) moves following this command will move at a rate of .05 times the current spindle revolutions per minute. For example, if the current RPM is 300, then the feed rate will be 15 FPM.

4.2 Feed Rate Control Point - FEDRAT/LENGTH

Syntax: **FEDRAT/LENGTH, n**
SCALE

The *FEDRAT/LENGTH* command allows the programmer to define the cutting point on the tool. This point is used to maintain the programmed feed rate when angular tool axis changes occur.

‘*LENGTH*’ specifies that the cutting point is maintained at a constant distance from the tool end point. ‘n’ specifies the length up the tool, from the tool end point, which is used to maintain the

programmed feed rate (this point, referred to as the Feed Rate Control Point, will move at the programmed feed rate). Specifying a length of 0 will result in the tool tip being used.

‘*SCALE*’ specifies that the cutting point is maintained at a percentage of the programmed tool length from the tool end point. The programmed tool length is obtained from the *LOADTL/tl,LENGTH,tl* command. ‘*n*’ specifies the percentage of the tool length up the tool, from the tool end point, which is to be used to maintain the programmed feed rate (this point, referred to as the Feed Rate Control Point, will move at the programmed feed rate). Specifying a scale of 0 will result in the tool tip being used.

Example:

FEDRAT/LENGTH,1

Move the Feed Rate Control Point from the tool end point to a distance of 1 inch/mm up the tool axis.

FEDRAT/SCALE,.5

Move the Feed Rate Control Point from the tool end point to a point half way up the programmed tool length.

4.3 Rapid Moves - RAPID

Syntax: **RAPID** [/] [**MODIFY**] [[,] **X***AXIS*] [[,] **R***TRCTO*] [[,] **ON**]
Y*AXIS* **OFF**
Z*AXIS*
T*OOL*

RAPID with no parameters caused the next motion to move at the rapid traverse rate. The motion will be altered as selected in the *MPost utility*. ‘*MODIFY*’ alters the move as specified in the *MPost utility*, but will make the move(s) at the programmed feed rate.

‘*-AXIS*’ overrides the default rapid logic for modifying the motion and for the next move only, sets the major tool axis as ‘*X**AXIS*’, ‘*Y**AXIS*’ or ‘*Z**AXIS*’. ‘*TOOL*’ causes the next move to retract/plunge along the tool axis vector to the clearance plane instead of moving along a major axis.

‘*RTRCTO*’ will retract the tool to the clearance plane as defined by the *CLRSRF* command prior to moving to the programmed location at the rapid rate. The tool will retract at the default feed rate specified for the *RETRCT* command.

‘*ON*’ causes *RAPID* to be modal until the next post-processor command is encountered. The default ‘*OFF*’ parameter specifies a one-shot *RAPID* command.

Example:

RAPID

The next move will be made in rapid traverse mode. After this move is completed, then the programmed feed rate will be reinstated. The move may be altered, as specified in the ***MPost*** utility.

RAPID/MODIFY,ZAXIS

The next move will be output using the programmed feed rate and may be altered, depending on the movement of the Z-axis. If the Z-axis moves in the positive direction, then the Z-axis will be output first, then the X and Y axes. If the Z-axis moves in the negative direction, then the X and Y axes will be output first, then the Z-axis in the following block.

4.4 Partial Rapid Moves - RAPID/FEDTO

Syntax: **RAPID/FEDTO, LENGTH [, n]**
SCALE

The *RAPID/FEDTO* command makes a portion of the programmed move at the rapid traverse rate and the remainder of the move at the programmed feed rate. Both moves will be output in a straight line, the normal *RAPID* logic will not apply.

'*LENGTH*' specifies that '*n*' defines a distance at the end of the move at which the tool is to feed at the programmed feed rate. '*SCALE*' specifies that '*n*' is a percentage of the move at which to feed at the programmed feed rate, with 1.0 equal to 100%.

You can force the next *RAPID* move not to be modified as it normally would be by not specifying '*n*' on the command. The entire move will be made at the rapid rate and the normal rapid alteration logic will not be used.

Example:

RAPID/FEDTO,LENGTH,.

The next move will move at the rapid traverse rate to within .5 inches/millimeters of the final destination. The last .5 inches/millimeters will be made at the programmed feed rate. If the move is shorter than .5, then the entire move will be made at the programmed feed rate.

RAPID/FEDTO,LENGTH

The next move will move in a straight line, at the rapid traverse rate. The normal *RAPID* alteration logic will not apply to this move.

4.5 Maximum Feed Rate - FEDRAT/MAXIPM

Syntax: **FEDRAT/MAXIPM,n**

The *FEDRAT/MAXIPM* command allows the programmer to define the maximum feed rate at which the tool tip or linear axis slides may move.

'*MAXIPM*' is required for correct syntax.

'*n*' specifies the maximum feed rate value. **PWorks** will slow down the programmed feed rate whenever the tool tip or linear axis slides exceeds this value. A value of 0 will disable this feature.

Example:

FEDRAT/MAXIPM,80

PWorks will slow down any output feed rates so that neither the tool end point nor the machine's linear axis slides move faster than 80 inches/millimeters per minute.

4.6 Maximum Degrees Per Minute - MAXDPM

Syntax: **MAXDPM/n**

The *MAXDPM* command sets the maximum feed rate in degrees per minute for the rotary axes. If the programmed feed rate caused the rotary axes to move at rate in excess of that specified by '*n*', then **PWorks** will reduce the programmed linear feed rate so that the degrees per minute feed rate is the maximum allowed

Example:

MAXDPM/240

PWorks will slow down any output feed rates which exceed 240 DPM so that the combined feed rate of the rotary axes does not move faster than 240 degrees per minute.

4.7 Feed Rate Overrides - FEDRAT/LOCK

Syntax: **FEDRAT/LOCK, ON**
OFF

The *FEDRAT/LOCK* command enables/disables the operator from overriding the programmed feed rates at the machine control. ‘*ON*’ disables feed rate overrides, ‘*OFF*’ enables feed rate overrides. The proper code as specified in the *MPost utility* will be output with this command.

Example:

FEDRAT/LOCK,ON

Disable the machine operator from overriding the programmed feed rate.

4.8 Simple Spindle Control - SPINDL

Syntax: **SPINDL/ON**
OFF
ORIENT

The *SPINDL* command turns on and off the spindle. ‘*ON*’ turns on the spindle and uses the last programmed spindle speed, range, and direction. ‘*OFF*’ turns the spindle off. ‘*ORIENT*’ stops the spindle in its locked (orientated) position.

Example:

SPINDL/ON

Turn spindle on using the last programmed spindle speed, range, and direction.

4.9 Spindle Speed Control - SPINDL

Syntax: **SPINDL/[RPM] [[,]n] [[,]CLW] [[,]LOW] [[,]RADIUS, r]**
SFM CCLW MEDIUM DIAMTR, D
BOTH HIGH
AUTO

The *SPINDL* command controls the speed and direction of the [spindle](#). ‘*RPM*’ selects Revolutions per Minute spindle speed mode. ‘*SFM*’ selects Surface Feed per Minute spindle speed mode. If the machine does not support *SFM* spindle control, then *PWorks* will calculate the *RPM* speeds based on either the part radius for lathes or the cutter diameter for mills. *RPM* and *SFM* stay in

effect until explicitly cancelled by the other mode. 'n' specifies the spindle speed in the current mode.

'*CLW*' turns the spindle on in the clockwise direction. '*CCLW*' turns the spindle on in the counter-clockwise direction. '*BOTH*' outputs the spindle direction using the '*SPINDL/BOTH*' code from the **MPost utility**, which is usually used to output both a clockwise and counter-clockwise code, allowing the operator to determine which direction to move the spindle in.

'*LOW*' selects the low spindle speed range. '*MEDIUM*' selects the middle range, and '*HIGH*' selects the high range. '*AUTO*' specifies that the spindle speed range will automatically be selected by **PWorks**, depending on the input spindle speed.

'*RADIUS*' and '*DAIMTR*' override the internally calculated radius/diameter for *SFM* calculations. The part radius is used when programming lathes and the cutter diameter is used for mills when neither of these parameters are specified. If either of these parameters is specified, then the part radius or diameter will be increased by 'r' or 'd' when programming a lathe or the cutter diameter will be replaced by 'd' or two times 'I' when programming mills.

Example:

```
SPINDL/RPM,850,MEDIUM,CLW
```

Turns the spindle on in the clockwise direction, at a speed of 850 revolutions per minute and using the middle spindle speed range.

```
SPINDL/SFM,1200,CCLW,DIAMTR,1.0
```

Turns the spindle on in the counter-clockwise direction. The spindle revolutions per minute speed will be calculated using 1200 surface feet/meters per minute based on a 1.0 inch/mm diameter cutter.

4.10 Maximum Spindle Speed - SPINDL/MAXRPM

Syntax: **SPINDL/MAXRPM, n**

The *SPINDL/MAXRPM* command defines the maximum spindle *RPM* allowed while the spindle is in Surface Feed per Minute (*SFM*) mode. The spindle *RPM* will be maintained by **PWorks** and by the machine control if it supports this feature.

Example:

```
SPINDL/MAXRMP,1000
```

The spindle speed will be slowed down whenever it is programmed in *SFM* mode and the speed exceeds 1000 RPM.

4.11 Spindle Speed Overrides - SPINDL/LOCK

Syntax: **SPINDL/LOCK, ON**
OFF

The *SPINDL/LOCK* command enables/disables the operator from overriding the programmed spindle speeds at the machine control. 'ON' disables spindle speed overrides. 'OFF' enables [spindle speed overrides](#).

Example:

SPINDL/LOCK,ON

Disable the machine operator from overriding the programmed spindle speed.

4.12 Machine Dwells - DELAY

Syntax: **DELAY/n [, REV]**

The *DELAY* command causes the machine to dwell for the specified amount of time or spindle revolutions as specified by the *MPost utility*.

When 'n' is the only parameter on the *DELAY* command, it specifies the amount of time in seconds to dwell. 'REV' specifies that 'n' is given in number of spindle revolutions to dwell.

Example:

DELAY/2.5

The machine will dwell for a period of two and half seconds.

DELAY/10,REV

The machine will dwell for 10 spindle revolutions. The time of the dwell is dependent on the current spindle speed.

CHAPTER 5 Positioning Commands

5.1 The FROM Statement - FROM

Syntax: **FROM**/**x,y,z,i,j,k**

The *FROM* command is usually used to orient the tool end point position at the beginning of the part program and can be handled in a variety of different ways. See the **MPost** [utility](#) for a description on the handling of the *FROM* command.

Example:

```
FROM/0, 0, 0, 0, 0, 1
```

Sets the current tool end point position to X=0, Y=0, Z=0, and the tool axis vector to 0, 0, 1. This position will usually not be output to the Control Tape.

5.2 Machine Axes Positioning - POSITN

Syntax: **POSITN**/ [[**X**AXIS [, **n**] , **pos**] [[,] **Y**AXIS [, **n**] , **pos**] \$
[[,] **Z**AXIS [, **n**] , **pos**] [[,] **A**xis [, **n**] , **pos**]] [. . .]

The *POSITN* command positions the selected axes at the specified location. This location is input as the actual machine position instead of the tool end point position (which is programmed using the *GOTO* and motion commands). **PWorks** will calculate a new tool end point and related positions from this location.

‘-*AXIS*’, used in conjunction with ‘*n*’, selects which axes to position. ‘*n*’ can be 1 (Primary) or 2 (Secondary) with the linear axes (*X*AXIS, *Y*AXIS, *Z*AXIS). ‘*n*’ can be 1 through 4 with the rotary axes (*AXIS*). If ‘*n*’ is not specified it will default to 1. ‘*pos*’ defines the location for the specified axis to position at.

Example:

```
POSITN/XAXIS,10, YAXIS,2,7.5, AXIS,25, AXIS,2,0
```

Moves the machine to the position of X=10, Y Secondary=7.5, Rotary axis #1=25 degrees, and Rotary #2=0 degrees. This move will be made at the rapid rate.

5.3 Position The Rotary Axes - ROTABL, ROTHED

```
Syntax: ROTABL/[AXIS,n,]ATANGL,ang[,NEUTRL][,IPM ,f] $
        ROTHED                                ORIENT    MMPM
                                           CLW
                                           CCLW
                                           INCR

        [,SAME  ][,NOW ][POSITN]
        ROTREF    NEXT    CUT
```

The *ROTABL* and *ROTHED* commands are exactly the same, you can use either one depending on preference. These commands are used to position a rotary axis without using the tool axis vector of an input cl point.

‘*AXIS,n*’ selects the rotary axis to position. ‘n’ can be in the range of 1-4, with the default being 1. ‘*ATANGL,ang*’ selects the angle to position the rotary axis at. ‘ang’ will be entered as an absolute position, incremental distance, or on a scale of 0-360 degrees, depending on the next positioning qualifier.

‘NEUTRL’ is the default positioning qualifier and specifies that ‘ang’ is input on a rotary scale (0-360 degrees). **PWorks** will move the rotary axis in the direction which will take the shortest route. ‘ORIENT’ specifies ‘ang’ is input on a linear scale, the rotary axis will move to this exact position. ‘CLW’ specifies that ‘ang’ is input on a rotary scale and the rotary axis will move in the clockwise direction. ‘CCLW’ specifies that ‘ang’ is input on a rotary scale and the rotary axis will move in a counter-clockwise direction. ‘INCR’ specifies that ‘ang’ is an incremental value and the rotary axis will be rotated ‘ang’ degrees from its current position.

'IPM' and *'MMPM'* specify the feed rate at which the Feed Rate control Point should move. **PWorks** will calculate the degrees per minute feed rate using this value and the distance of the Feed Rate Control Point from the center of the rotary axis. A value of 0 specifies the rotary axis will move at the rapid rate.

‘*SAME*’ does not move the linear machine axes with the rotary axis. The tool end point will actually move in relationship to the part. ‘*ROTREF*’ causes the linear machine axes to move in order to keep the tool end point at the same position in reference to the part.

‘*NOW*’ will move the rotary axis in a block by itself. ‘*NEXT*’ will adjust the tool axis vector to reflect the new rotary position and the rotary axis will not move until the next motion block. ‘*NEXT*’ is only valid when *MULTAX* is turn off.

‘*POSITN*’ will rotate the axis without linearization. This is the default mode. ‘*CUT*’ specifies the rotational move will be linearized.

Example:

ROTABL/ATANGL,35,CLW,IPM,20

Moves Rotary axis #1, in a clockwise direction, to a position of 35 degrees. The Feed Rate Control Point position on the tool will move at a rate of 20 FPM as compared to the part.

ROTABL/AXIS,2,ATANGL,720,ORIENT,SAME,IPM,0

Moves Rotary axis #2 to an absolute position of 720 degrees, assuming this axis moves on a linear scale. The linear machine slides will not move and the rotary axis will move at the rapid rate.

ROTABL/AXIS,1,ATANGL,10,ROTREF

Moves Rotary axis #1 to a position of 10 degrees. **PWorks** will determine the direction of the move based on the shortest route. The linear machine slides will move with the rotary axis, keeping the tool end point at the same location in reference to the part. This move will not be linearized. The last programmed feed rate will be in effect for this move.

5.4 Clearance Plane - CLRSRF

Syntax: **CLRSRF**/[**NORMAL**,] [**i**,**j**,**k**,]**d**

The *CLRSRF* command defines the clearance plane at which the tool will retract to when the *RETRACT* command is encountered. The clearance plane must be defined in reference to the machine's axis positions, not the input tool end point coordinates.

'*NORMAL*' differentiates this command from the *CLRSRF/TOOL* command. If neither '*NORMAL*' nor '*TOOL*' is specified, then '*NORMAL*' is assumed. The *CLRSRF/NORMAL* command will effectively cancel the *CLRSRF/TOOL* command and vice versa.

'*i, j, k*' is the clearance plane vector definition. If this parameter is not specified, then the primary tool axis vector as defined in the *MODE/TOOL* command will be used.

'*d*' is the distance along the defined vector for the clearance plane. This parameter must be defined.

Example:

CLRSRF/8.5

Defines the clearance plane for the *RETRACT* command at a distance of 8.5 along the primary tool axis vector (usually 0, 0, 1).

CLRSRF/NORMAL, 1, 0, 0, 3.2

Defines the clearance plane for the *RETRACT* command as being 3.2 in the positive X-axis.

5.5 Clearance Distance - CLRSRF/TOOL

Syntax: **CLRSRF/TOOL, d**

The *CLRSRF/TOOL* command defines the clearance distance that the tool will retract when the *RETRACT* command is encountered. The tool will actually retract to a distance of ‘d’ plus the current tool length.

‘*TOOL*’ differentiates this command from the *CLRSRF/NORMAL* command and must be specified. The *CLRSRF/TOOL* command will effectively cancel the *CLRSRF/NORMAL* command and vice versa.

‘d’ is the distance along the current tool axis vector, in addition to the programmed tool length, which will be used for the retract distance.

Example:

```
LOADTL/1, LENGTH, 5.5  
CLRSRF/TOOL, 1.5
```

Defines the clearance distance for the *RETRACT* command to be 7.0 (5.5 + 1.5).

5.6 Retracting The Tool - RETRACT

Syntax: **RETRACT [/] [PAST] [[,] XAXIS] [[,] feed]
 TO YAXIS
 ZAXIS
 TOOL**

The *RETRACT* command retracts the tool to the current clearance plane if *CLRSRF/NORMAL* is in effect or to the clearance distance if *CLRSRF/TOOL* is in effect.

‘*PAST*’ retracts the tool end point to the clearance plane. ‘*TO*’ will retract the pivot point or tool stop, whichever applies, to the clearance plane.

Specifying an ‘-*AXIS*’ parameter will retract the tool in the direction of this primary axis until it reaches the clearance plane. ‘*TOOL*’ will retract the tool along the current spindle vector. The tool will always retract in reference to the actual machine’s axes, not the input coordinate system.

'*feed*' specifies an optional feed rate at which to retract the tool. If '*feed*' is not specified or is specified as 0, then the tool will be retracted in rapid mode. '*feed*' must be specified in Feed Per Minute. The programmed feed rate will be reinstated after the *RETRACT* command.

Example:

RETRACT

Retracts the tool to the programmed clearance level at the rapid rate.

RETRACT/PAST,ZAXIS,30

Retracts the tool tip straight along the machine's Z-axis to the programmed clearance level at a feed rate of 30.

RETRACT/TO,TOOL

Retracts the tool stop position along the current spindle axis to the programmed clearance level.

5.7 Repositioning The Tool - PLUNGE

Syntax: **PLUNGE** [/**feed**]

The *PLUNGE* command repositions the tool to the location it was at prior to the last programmed *RETRACT* command. An error message will be output and this command will be ignored if a *RETRACT* command has not been programmed.

'*feed*' specifies an optional feed rate at which to reposition the tool. If '*feed*' is not specified or is specified as 0, then the tool will be repositioned using rapid mode. '*feed*' must be specified in Feed Per Minute. The programmed feed rate will be reinstated after the *PLUNGE* command.

Example:

PLUNGE

Positions the tool to the location it was at prior to the last *RETRACT* command at the rapid rate.

PLUNGE/30

Positions the tool to the location it was at prior to the last *RETRACT* command at a feed rate of 30.

5.8 Move To Home Position - GOHOME

Syntax: **GOHOME**

The *GOHOME* command, without any parameters, move all supported axes to their machine home position. The move to home will be done in rapid mode and all rules of rapid motion will apply (the move may be broken up into multiple moves, depending on the default setting for rapid motion).

The machine home position is defined in the *MPost* utility.

Example:

GOHOME

Moves all supported axes to their home position at the rapid rate.

5.9 Move To Reference Point - GOHOME/mode

Syntax: **GOHOME/CHECK** [[, *X* **AXIS** [, *n*] [, *pos*]] [, *Y* **AXIS** [, *n*] [, *pos*]] \$
AUTO
FROM
NEXT

[, *Z* **AXIS** [, *n*] [, *pos*]] [, *AXIS* [, *n*] [, *pos*]] [. . .]

This version of the *GOHOME* command outputs the proper coding to cause the machine to move from/to its home (reference point) position. When using this version of the *GOHOME* command, it is assumed that the machine control has the capability of automatically moving to/from the home position. The machine will first move to an intermediate position before moving to its home position.

‘*CHECK*’ specifies a return to home position with the machine confirming that the tool has actually reached this position. ‘*AUTO*’ specifies an automatic return to the home position and is the most commonly used mode. ‘*FROM*’ specifies an automatic return from home position and is usually used after the machine was moved to home using the ‘*AUTO*’ or ‘*NEXT*’ qualifier. ‘*NEXT*’ specifies a move to the 2nd home position on the machine.

‘*-AXIS*’ specifies which linear axes will move to their home position. ‘*AXIS*’ selects a rotary axis. Only the axes included in this command will move to their home position.

‘*n*’ is an optional number and selects which of the specific axes to use. On the linear axis specifiers (*X***AXIS**, *Y***AXIS**, *Z***AXIS**), ‘*n*’ can be either 1 (Primary) or 2 (Secondary). On the rotary axis parameter, ‘*n*’ can be in the range of 1-4.

'pos' specifies the values for the intermediate point. If this parameter is not specified with its corresponding axis designator, then a default value of 0 will be used.

Example:

GOHOME/AUTO,ZAXIS

Moves the Z-axis to its home position. If incremental mode is in effect, then the Z-axis will not move to an intermediate point prior to moving to its home position. If absolute positioning is in effect, then the Z-axis will move to 0 prior to moving to its home position.

GOHOME/CHECK,XAXIS,10,YAXIS,2,13,AXIS,2,0

Moves the primary X-axis, secondary Y-axis, and Rotary axis #2 to their home position. The machine will first move to an intermediate position of X=10, Y2=13, and Rotary axis #2=0, prior to moving to its home position. The machine will verify that it has reached its home position in these axes after the move has been completed.

CHAPTER 6 Motion Control

6.1 Axis Travel Limits - CHECK

Syntax: **CHECK/** [**X**AXIS [,n] ,min,max] [[,] **Y**AXIS [,n] ,min,max] \$
[[,] **Z**AXIS [,n] ,min,max] [[,] **A**XIS [,n] ,min,max]] \$
[. . .]

The *CHECK* command defines the axis limits of the machine. ‘-*AXIS*’ specifies which linear axis to set the limits for. ‘*AXIS*’ selects a rotary axis. ‘n’ is an optional number and selects which of the specified axis to set. On the linear axis parameters (*X*AXIS, *Y*AXIS, *Z*AXIS), ‘n’, can be either 1 (Primary) or 2 (Secondary). On the rotary axes parameter, ‘n’ can be in the range of 1 - 4. If ‘n’ is not specified it will default to 1.

‘min’ sets the lower limit for the axis and ‘max’ sets the upper limit. To disable limit checks by **PWorks** define unreachable limits.

Example:

CHECK/XAXIS, -15, 40, YAXIS, 2, 0, 25, AXIS, 2, -10, 90

Sets the X-axis travel limits to -15 through 40, the secondary Y-axis limits to 0 through 25, and Rotary axis #2 limits to -10 degrees through 90 degrees.

6.2 Interference Zones - CHECK/OUT

Syntax: **CHECK/OUT**,m [[,] **X**AXIS [,n] ,min,max] \$
[[,] **Y**AXIS [,n] ,min,max] \$
[[,] **Z**AXIS [,n] ,min,max] \$
[[,] **A**XIS [,n] ,min,max]] [. . .]

The *CHECK/OUT* command defines machine interference zones, which are used by **PWorks** to determine when the spindle and machine will collide. When an interference zone is entered, **PWorks** will output a warning message, but will not alter the programmed machine position. It will be determined that the machine has entered an interference zone when all of the axes contained in the *CHECK/OUT* command are within the range specified. You can define up to 4 interference zones and all defined interference zones will be actively checked for each tool location.

‘*OUT*’ is required for proper syntax. ‘*m*’ specifies the interference zone being defined and can be in the range of 1-4.

‘*-AXIS*’ specifies which axes to include for checking as part of the interference zone currently being defined. Only the axes included on this command will be checked for spindle and machine collisions. Any axes omitted from this command will not be considered as part of this interference zone.

‘*n*’ is an optional number and selects which of the specific axes to use. On the linear axis specifiers (*XAXIS*, *YAXIS*, *ZAXIS*), ‘*n*’ can be either 1 (Primary) or 2 (Secondary). On the rotary axes parameter, ‘*n*’ can be in the range of 1-4. If ‘*n*’ is not specified it will default to 1.

‘*min*’ specifies the lower limit of the interference zone for the specified axis. ‘*max*’ specifies the upper limit for this axis. The axis will be considered inside the interference zone when its location is equal to or greater than the lower limit and less than or equal to the upper limit. All axes defined as part of this command must be within their limits for **PWorks** to output a warning message.

Example:

```
CHECK/OUT,1,XAXIS,-10,5,ZAXIS,0,5,AXIS,1,90,90
```

Defines the first interference zone as being dependent on the X-axis, Z-axis, and Rotary axis #1. This interference zone will be breached whenever the X-axis is between -10 and 5, the Z-axis is between 0 and 5, and Rotary axis is #1 is at 90 degrees.

6.3 Output Axes Tolerances - PPTOL

Syntax: **PPTOL/tol**

```
[ [XAXIS[,n],xtol] [[,]YAXIS[,n],ytol] $  
[[,]ZAXIS[,n],ztol] [[,]AXIS[,n],atol]] [...]
```

The *PPTOL* command defines the minimum axis movement that will be output to the Control Tape for each axis. Prior to motion being output, **PWorks** will calculate the distance from the last position to the current position for each axis. If this distance is less than the tolerance specified, then the move for the axis being checked will not be output.

‘*tol*’ specifies the same tolerance for each axis. ‘*-AXIS*’, used in conjunction with ‘*n*’, selects which axis to define the tolerance for. ‘*n*’ can be 1 (Primary) or 2 (Secondary) with the linear axis parameters (*XAXIS*, *YAXIS*, *ZAXIS*). ‘*n*’ can be 1 through 4 with the rotary axes parameter (*AXIS*). If ‘*n*’ is not specified it will default to 1. ‘*tol*’ defines the tolerance for the specified axis.

Example:

PPTOL/.0002

Sets the output tolerance to .0002 for each of the active axes. Output will be disabled for the separate axes until a move of at least .0002 occurs. The tolerance will be checked independently for each axis.

PPTOL/AXIS,.002, AXIS,2,.002

Sets the output tolerance for both of the rotary axes to .002 degrees. Neither of the rotary axes will be output until they move at least .002 degrees.

6.4 Rotary Axes Direction - ROTABL/NEXT

Syntax: **ROTABL/NEXT** [, **AXIS** , **n**] , **CLW**
ROTHED **CCLW**

When 2 or more rotary axes are active there is a possibility that a tool axis vector can be satisfied using two different axis rotations. Normally **PWorks** will choose which of the rotary axis positions to use base on the following criteria.

If one set of rotations is within machine travel limits and the other set is outside of limits, then the rotations within the limits will be used.

If both sets of rotations are within or outside of the machine limits, then the set with the least rotary movement will be used. If both sets require the same amount of rotation, then it is a toss up on which set will be used.

The **ROTABLE/NEXT** command allows the programmer to override **PWorks**' internal logic and force a specific rotary axis to move a certain direction in the next motion block only. Though, at no time will **PWorks** move the axis out of limits if one set of rotations will remain within limits.

'**AXIS,n**' selects the rotary axis to force the direction on. '**n**' can be in the range of 1-4, with the default being 1.

'**CLW**' requests the specified rotary axis to move in the clockwise direction. '**CCLW**' requests the specified rotary axis to move in the counter-clockwise direction.

Example:

ROTABL/NEXT,AXIS,2,CCLW

Forces the direction of Rotary axis #2 to be counterclockwise for the next motion block. If a counterclockwise direction forces any of the active axes to move out of limits, then this command will be ignored.

6.5 Shortest Route Determination - ROTABL/SHORT

Syntax: **ROTABL/SHORT, COMBIN [, ON]**
ROTHED LARGE NEXT
PLUS
AXIS, n

When two or more rotary axes are active there is a possibility that a tool axis vector can be satisfied using two or more different axis rotations. **PostWorks** will usually select the position which causes the least amount of movement. This command selects the method of determining the shortest route which **PostWorks** will use. The default method is defined using the **MPost** utility. See the [Motion Adjustments Chapter](#) in the **MPost** Reference Manual

‘**COMBIN**’ will combine and average the movements of all active rotary axis and use this as the delta movement. ‘**LARGE**’ will compare the single largest movements from the different sets of rotary axis calculations. ‘**PLUS**’ will total all rotary axis movements in each set. ‘**AXIS**’ will base the total rotary movement bases on a single user defined axis ‘**n**’.

‘**ON**’ is the default condition and will change the shortest route determination logic until another **ROTABL/SHORT** command is encountered. ‘**NEXT**’ causes this command to apply to the next single move only. The ‘**NEXT**’ qualifier specifies that this logic is only valid for the next single move and the previous shortest route determination logic will be reinstated after this move.

Example:

ROTABL/SHORT,AXIS,2

Specifies the shortest route logic bases on the second rotary axis until another ROTABL/SHORT command is encountered.

ROTABL/SHORT,AXIS,1,NEXT

Specifies the shortest route logic for the next single move bases on the first rotary axis. The previous shortest route logic will be reinstated after this move.

This feature is useful for specifying which axis should move when both the rotary axes move the same distance in a single command. For example, the following command sequence will cause the A-axis to move to fulfill the tool axis vector when an AC-style head configuration is active.

FROM/0,0,5, 0,-1,0

ROTABL/SHORT,AXIS,1,NEXT
GOTO/1,0,5, 0,1,0

In the above example, either the C-axis or A-axis can move 180 degrees to fulfill the tool axis, but since the user requested that the C-axis move the least, the A-axis was chosen.

6.6 Rotary Axes Clamping - CLAMP

Syntax: **CLAMP/AXIS,n1[...]n4,ON**
OFF

The *CLAMP* command outputs codes to clamp (*ON*) or unclamp (*OFF*) the requested rotary axes. ‘*n*’ specifies which of the rotary axes to clamp/unclamp and can be in the range of 1-4. Usually **PWorks** will automatically clamp and unclamp the rotary axes as needed.

Example:

CLAMP/AXIS,1,2,ON

Clamps Rotary axis #1 and Rotary axis #2.

6.7 Automatic Clamping Of Rotary Axes - CLAMP/AUTO

Syntax: **CLAMP/AUTO,n**

The *CLAMP/AUTO* command controls the automatic clamping of the rotary axes. ‘*n*’ specifies the style of clamping to be done and can be one of the following values.

- 1: Automatic clamping and unclamping of the rotary axes will not be performed.
- 2: Each rotary axis should be unclamped prior to movement and immediately clamped after each move.
- 3: Each rotary axis should be unclamped prior to movement and clamped prior to a move in which this rotary axis does not have any movement. This is the most common style of automatic clamping.
- 4: Each rotary axis should be unclamped prior to any rotary output and clamped prior to pure linear output with no rotary output.
- 5: Each rotary axis should be unclamped prior to any rotary movement and clamped prior to pure linear axes movement (no rotary movement).

The **MPost** utility is used to set the default clamping style.

CHAPTER 7 Machining Modes

7.1 Absolute/Incremental Programming - **MODE/INCR**

Syntax: **MODE/INCR, ON**
OFF

The *MODE/INCR* command controls the output of incremental axis moves or absolute axis positioning.

‘*ON*’ selects incremental output. All axes will be output as incremental distance from the previous absolute axis coordinates.

‘*OFF*’ selects the output of absolute axis coordinate positions.

Example:

MODE/INCR,OFF

All output positions following this command will be output containing their absolute positions.

7.2 Tool Axis Vectors - **MULTAX**

Syntax: **MULTAX/ON**
OFF

The *MULTAX* command is used to pass the programmed tool axis vectors to **PWorks**.

‘*ON*’ specifies that the tool axis vector will be passed to **PWorks** with each cl point.

‘*OFF*’ specifies that the cl points will not contain a tool axis vector. If ‘*OFF*’ is specified, then **PWorks** will assume the last programmed tool axis vector with *MULTAX* turned on as the current vector. The tool axis vector at the beginning of the program is determined by [Menu 2.1.12](#) of the **MPost** utility for the corresponding machine configuration file.

Example:

MULTAX/ON

Each input cl point from the clfile will contain a tool axis vector.

7.3 Primary Spindle Axis - *MODE/TOOL*

Syntax: ***MODE/TOOL,XYPLAN***
ZXPLAN
YZPLAN
i,j,k

This command defines the primary tool axis vector. The primary tool axis is considered to be the programmed tool axis which will move all rotary axes to 0 degrees. You would most likely use the *MODE/TOOL* command when you add an attachment to the machine (such as a 90 degree head) which will change the orientation of the spindle in relationship to the machine table. The primary tool axis can be changed to any vector on machines with no programmable rotary axes to machines with up to four rotary axes.

'*XYPLAN*' specifies that the tool axis is pointing in the positive Z-axis direction when all rotary axes are at 0 degrees. '*ZXPLAN*' selects the positive Y-axis as the primary tool axis. '*YZPLAN*' selects the positive X-axis as the primary tool axis. '*i,j,k*' allows the programmer to define a tool axis vector that will be used as the primary tool axis. Any vector with a real length can be used.

Following are some of the effects that changing the primary tool axis will have.

Rotary Axes	Rotary axes calculations will be based on the primary tool axis.
BREAK	The tool will retract and plunge in the primary tool axis.
CYCLE	Cycle motion will be performed in the primary tool axis.
LOADTL	Tool lengths will be applied along the primary tool axis.
RAPID	Rapid motion can be altered according to the value of the primary tool axis.

Example:

MODE/TOOL,ZXPLAN

Change the primary tool axis to point in the positive Y direction.

MODE/TOOL,-.5,0,.866

Change the primary tool axis to point along a vector of -.5, 0, .866.

7.4 Linear Axes Selection - **MODE/-AXIS**

Syntax: **MODE/XAXIS,n**
 YAXIS
 ZAXIS

The *MODE/-AXIS* command selects the active Primary or Secondary linear axis for output.

‘*XAXIS*’, ‘*YAXIS*’ and ‘*ZAXIS*’ select the output linear axis to activate. ‘*n*’ can be in the range of 1-3. ‘1’ specifies the Primary axis, ‘2’ specifies the Secondary axis, and ‘3’ selects both the Primary and Secondary axes be output.

‘3’ is usually only used when no adjustments for the inactive linear axis need to be made when calculating the position of the active linear axis.

The last active linear axis will remain the active axis when both axes are output.

Example:

MODE/ZAXIS,2

All input Z-axis coordinates after this command will cause the Secondary Z-axis to move. The Primary Z-axis will be locked.

7.5 Rotary Axes Selection - **MODE/AXIS**

Syntax: **MODE/AXIS [,n1 [,n2]]**

The *MODE/AXIS* command activates the specified rotary axes. Although **PWorks** can support up to 4 rotary axes, only 2 can be active at any one time. Enter which of the rotary axis (1-4) you wish to activate, the remaining supported rotary axes will be locked. You can enter either one or two axes.

The active rotary axes will move to fulfill the tool axis vector, while the locked rotary axes must be moved using the *ROTABL* command. This command is usually only used when more than 2 rotary axes are supported.

Example:

MODE/AXIS,1,3

Activates Rotary axes #1 and #3. These rotary axes will move to fulfill the input tool axis vector. Rotary axes #2 and #4 will be locked and must be positioned using either the *ROTABL* or *POSITN* command.

7.6 Multi Heads/Tables Selection - ROTABL/ON

Syntax: **ROTABL/ON** , [AXIS,n,]m1[[[, [THRU] ,mn,]] [...]m9]
OFF

Some multi-head or multi-table machines require that the rotary axes associated with each head/table be addressed separately, even if the position will always be the same. This *ROTABL/ON*,... command specifies which of the physical rotary axes to be turned on and output to the punch file. The command *ROTABL/OFF*,... command specifies which of the physical rotary axes need to be turn off and not output to the punch file.

Up to nine duplicate axes can be turned on or off in one time. ‘*THRU*’ can be used to specify a range of physical axes that need to be enabled or dis-enabled.

The maximum number of duplicated axes for each rotary axis can be set with the *MPost* utility.

7.7 Change Rotary Axes Configuration

Syntax: **ROTABL/TABLE,AAXIS[,XAXIS[,rot1[,XAXIS,rot2]] [...]**
HEAD BAXIS YAXIS YAXIS
CAXIS ZAXIS ZAXIS

PWorks has the ability to change the rotary axes configuration during post processing. This is useful for such machines with modular design that allows the use of different head or table during machining.

“*TABLE*” specifies that the rotary axis being defined is a table, while “*HEAD*” specifies a head. “*AAXIS*” specifies that this axis rotates about the X-axis, “*BAXIS*” specifies the Y-axis, and “*CAXIS*” specifies the Z-axis.

You can optionally specify the initial orientation of the axis when the axis rotates about any other vector than a major axis. This is accomplished by specifying a major axis (*XAXIS*, *YAXIS*, *ZAXIS*) that the axis is oriented about and the angle of rotation of the orientation. You can specify one or two orientation angles. The order of the rotation axes is important and follows the same rules as when defining the physical rotary axes.

All rotary axes must be defined in single command. The “*TABLE/HEAD*” clause is repeated in the command for all rotary axes. The order of the rotation axes is important and must follow the same rules as in *MPost* (table riders, table carriers, head riders, head carriers). For example, use the following command to define a standard AC-style head.

ROTABL/HEAD,CAXIS,AAXIS

It is important that if the pivot length or table origin change when the rotary axes change that a *ROTABL/ORIGIN* command be issued to define the new length/origin. It is also important to define the output register characters if they change from one configuration to another. The following sample Macro shows a method that supports an interchangeable AC-style head and BC-style nutating head.

```

HEAD/Macro,1
$$
$$...HEAD/1
$$...AC-style head
$$
  If (%Arg(1) == 1) Then
    ROTABL/ORIGIN,AXIS,2,0,0,7.
    ROTABL/HEAD,CAXIS,AAXIS
    DEFINE/%Regst(1),"A", %Regst(2),"A", %Regst(3),"A"
    DEFINE/%Regst(4),"C", %Regst(5),"C", %Regst(6),"C"
  $$
  $$...HEAD/2
  $$...BC-nutating head
  $$
  Else If (%Arg(1) == 3) Then
    ROTABL/HEAD,CAXIS,BAXIS,XAXIS,40
    DEFINE/%Regst(1),"B", %Regst(2),"B", %Regst(3),"B"
    DEFINE/%Regst(4),"C", %Regst(5),"C", %Regst(6),"C"
    ROTABL/ORIGIN,AXIS,2,0,0,3
  Else
    HEAD/%Arg(1:%Narg)
  Endif
Termac

```

7.8 Interchange Of Linear And Rotary Axes - *MODE/ROTATE*

Syntax: **MODE/ROTATE, OFF**
 ON [, n]
 AUTO

PWorks has the ability to position a rotary axis table, instead of the planar linear axes, to fulfill a linear move. This capability exists whenever the tool axis is parallel to the rotation axis of a rotary table. The *MODE/ROTATE* command controls the output of linear or polar coordinates. ‘*ON*’ enables polar interpolation. ‘*OFF*’ enables linear interpolation and is the default. ‘*AUTO*’ enables modal polar interpolation. ‘*n*’ specifies the tolerance to use when generating linear moves during polar interpolation.

Polar interpolation is useful when cutting a cylinder or spiral and the linear axis limits are not large enough to cut the full diameter. Naturally, cutting a round object with a rotary axis also

provides for a much nicer finish on the part. When cutting a spiral shape, the linear axes will move in conjunction with the rotary table to fulfill the programmed position.

When enabling polar interpolation, the tool axis must be parallel to a rotary table's axis of rotation and must remain at this orientation for the duration of the polar interpolation mode (until *MODE/ROTATE,OFF* is programmed). For example, when cutting a cylinder on a table mounted with its rotation axis about the Z-axis, then the programmed tool axis vector must be '0, 0, 1' or '0, 0, -1'. An error will be output and polar interpolation will be cancelled if a tool axis change occurs during polar interpolation.

When *MODE/ROTATE,AUTO* is programmed, polar interpolation remains in effect whenever the tool axis is parallel to a table's rotation axis. Whenever the tool axis changes from this orientation, then polar interpolation will be disabled until the tool axis once again become parallel to the axis of rotation. NO error messages will be generated during this mode and polar interpolation will not be cancelled. This mode is generally used when the machine lacks a linear axis, such as the Y-axis.

It is highly recommended that linearization be turned off (*LINTOL/OFF*) during polar interpolation, otherwise a series of small spiral linear moves will be output with the rotary moves, resulting in a straight line cut on the machine. You could actually cut a square on the machine while in polar interpolation with linearization enabled. The moves will be broken up small enough to keep the tool within tolerance of the straight edged sides as specified by 'n'.

Example:

MODE/ROTATE,ON

Enables polar interpolation. The rotary table will move to fulfill any linear moves programmed between *MODE/ROTATE,ON* and *MODE/ROTATE,OFF*. Depending on if cylindrical or spiral motion is programmed, there may also be movement of the linear axes. The tool axis must be parallel to the rotation axis of a rotary table.

7.9 Rotary Axis Linearization - LINTOL

Syntax: **LINTOL/n**

ON

OFF

The *LINTOL* command is used to maintain the tool end point within the tolerance specified when angular changes in the tool axis are programmed. To keep the tool within tolerance the move will be broken up into smaller moves, always keeping the tool end point on the same line as the previous and new positions. Linearization will not be performed when the tool axis vector does not change.

'n' enables linearization and specifies the tolerance to be used during linearization calculations. The output points will never deviate by more than 'n' from the programmed move. 'ON' turns linearization on and specifies a tolerance of .001 (in) or .025 (mm). 'OFF' disables linearization.

This command does not have any effect on RAPID move linearization.

Example:

LINTOL/.005

Enable linearization for rotary axes moves, using a tolerance of .005

LINTOL/ON

Enable linearization for rotary axes moves, using a tolerance of .001 (in) / .025 (mm).

LINTOL/OFF

Disables any linearization of the output axes.

7.10 Rapid Move Rotary Axis Linearization - LINTOL/RAPID

Syntax: **LINTOL/RAPID, n**
ON
OFF

The *LINTOL/RAPID* command is used to maintain the tool end point within the tolerance specified when angular changes in the tool axis are programmed during rapid move. To keep the tool within tolerance the rapid move will be broken up into smaller moves, always keeping the tool end point on the same line as the previous and new positions. Linearization will not be performed when the tool axis vector does not change.

'n' enables linearization and specifies the tolerance to be used during linearization calculations. The output points will never deviate by more than 'n' from the programmed move. 'ON' turns linearization on and specifies a tolerance of .001 (in) or .025 (mm). 'OFF' disables linearization.

Example:

LINTOL/RAPID,.005

Enable linearization for rapid rotary axes moves, using a tolerance of .005

LINTOL/RAPID,ON

Enable linearization for rapid rotary axes moves, using a tolerance of .001 (in) / .025 (mm).

LINTOL/RAPID,OFF

Disables any linearization of the rapid output axes.

7.11 Rotary Axis Positional Tolerance - LINTOL/AXIS

Syntax: **LINTOL/AXIS,n**

Whenever the rotary axes are setup so that the rotation of one of the axis can become parallel to the tool, there becomes a potential problem area where a small change in the tool axis can generate a large movement of the rotary axes. For example, consider a machine with a dual rotary head, an A-axis and a C-axis. A tool axis change from '.0001, 0, .9999' to '0, .0001, .9999' is actually a very small angular change, but on the machine, it will cause the C-axis to move 90 degrees.

The *LINTOL/AXIS* command helps alleviate this problem by controlling the actual angular change in the tool axis which will determine when a change in rotary angles has occurred. 'n' specifies the minimum angular change between tool axis vectors that constitutes a change in the rotary axes. The recommended value for 'n' is one half of the output axis resolution as specified by the *PPTOL* command.

Example:

LINTOL/AXIS,.0005

The rotary axes will be considered the same as the previous position until a tool axis angular change of more than .0005 degrees is encountered

7.12 Simple Cutter Compensation Control - CUTCOM

Syntax: **CUTCOM/ON**

OFF [,MODIFY,n,v]

The *CUTCOM* command outputs the proper coding (set by the *MPost utility*) for the machine to compensate for different sizes of cutter diameters.

'ON' enables cutter compensation and uses the parameters last specified in the expanded syntax of *CUTCOM*, such as *LEFT*, *XYPLAN*, *ENDPT*, etc. See the expanded syntax of *CUTCOM* for a description on how to specify the compensation direction, compensation plane, etc.

'OFF' disables cutter compensation. The optional 'MODIFY' clause is only valid for machines which allow you to specify an entry and/or exit method, such as semi-circle exit, quarter circle exit, etc. 'n' can be in the range of 1-3 and specifies the cutter compensation exit method to use. 'v' specifies the distance/radius for the exit method. See your particular machine's programming

guide for an explanation of different entry/exit methods. 'MODIFY' will be ignored on machines that do not support user programmable entry/exit methods.

Example:

CUTCOM/ON

Enables cutter compensation at the machine control using the parameters specified in the last *CUTCOM/mode* command.

CUTCOM/OFF,MODIFY,1,.03

Disables cutter compensation on the next motion block, using departure method #1 with a distance/radius of .03.

7.13 Cutter Compensation Direction Control - CUTCOM

Syntax: **CUTCOM/LEFT** [,XYPLAN] [,d] [,NORMAL] [,MODIFY,n,v] \$
 RIGHT **YZPLAN** **PERPTO**
 ZXPLAN

 [,MANUAL[,XAXIS,x][,YAXIS,y][,ZAXIS,z]]

The *CUTCOM* command outputs the proper coding (set by the **MPost utility**) for the machine to compensate for different sizes of cutter diameters.

'LEFT' specifies that the machine is to compensate to the left of the cutter path. 'RIGHT' specifies cutter compensation to the right of the cutter path.

'XYPLAN' selects cutter compensation in the XY machining plane. 'ZXPLAN' selects the ZX machining plane. 'YZPLAN' selects the YZ machining plane. If a machining plane is not specified, then the last specified machining plane will be used.

'd' selects a cutter compensation register at the machine control which will contain the compensation amount. This value, when required by the machine control, is usually the same as the tool number.

'NORMAL' specifies that the first cutter compensation vector output after this command will be such that the tool is offset normal to both the current drive surface and the check surface. Use this option when the move approaching the part is not normal to the part.

'PERPTO' specifies that the first cutter compensation vector output after this command will be such that the tool is offset normal to the current check surface only. The 'PERPTO' parameter is only valid with machines which require cutter compensation vectors and the move approaching the part is normal to the part.

The optional '*MODIFY*' clause is only valid for machines which allow you to specify an entry and/or exit method, such as semi-circle entry, quarter circle entry, etc. '*n*' can be in the range of 1-3 and specifies the cutter compensation entry method to use. '*v*' specifies the distance/radius for the entry method. See your particular machine's programming guide for an explanation of different entry/exit method, '*MODIFY*' will be ignored on machines which do not support user programmable entry/exit methods.

The optional '*MANUAL*' clause is only valid for machines which support cutter compensation vectors. The input '*-AXIS*' values will be output as the components of the cutcom vector and will override the cutcom vectors calculated by **PWorks**. The vector defined in the '*MANUAL*' clause will be output on each motion block between *CUTCOM/...,MANUAL,...* and *CUTCOM/OFF*, **PWorks** will not calculate cutcom vectors. '*XAXIS, x*' defines the X-component of the cutcom vector, '*YAXIS, y*' defines the Y-component and '*ZAXIS, z*' defines the Z-component. Only the vector components actually defined in this command will be output.

Example:

CUTCOM/LEFT,30,PERPTO

Enables cutter compensation to the left of the cutter path. Cutter compensation register #30 contains the offset value to use. Cutcom vector support is assumed, with the vector of the first move pointing 180 degrees back from the first motion direction.

CUTCOM/RIGHT,ZXPLAN,MANUAL,XAXIS,.5,ZAXIS,.866

Enables cutter compensation in the ZX machining plane. Cutter compensation will be performed to the right of the tool path. Cutcom vector support is assumed, with the X-component of the vector having a value of .866 on every move following this command until *CUTCOM/OFF* is encountered.

CUTCOM/LEFT,MODIFY,2,.1

Enables cutter compensation to the left of the cutter path using cutcom approach method #2, with a distance/radius of .1.

7.14 3-D Cutter Compensation Control - CUTCOM/ENDPT

Syntax: **CUTCOM/ENDPT** [, *PS*] [, *d*] [, *TOOL, rad, cr*] [, *MAXIPM, f*] \$
DS
[, *ROTREF, n*] [, *MANUAL* [, *XAXIS, i*] [, *YAXIS, j*] \$
[, *ZAXIS, k*]]

This *CUTCOM* command initiates 3-dimension cutter compensation and outputs the proper coding (set by the **MPost utility**) for the machine to compensate different sizes of cutter diameters. See your particular machine's programming guide for an explanation of how to use this 3-D cutter compensation control.

'*ENDPT*' specifies to initiate 3-D cutter compensation and is required for syntax.

'*PS*' specifies to use the part surface normal vectors for cutter compensation. '*DS*' specifies to use the drive surface normal vectors for cutter compensation. '*PS*' is the default.

'*d*' is an optional machine control register that holds the offset value.

'*TOOL*' defines the tool radius '*rad*' and the corner radius '*cr*'. If *TOOL* is not specified, then these parameters are obtained from the last programmed *CUTTER* statement.

'*MAXIPM*' defines the optional maximum feed rate '*f*' that the machine will move during 3-D cutter compensation.

'*ROTREF*' specifies the optional rotary axis movement mode '*n*' during 3-D cutter compensation. Typically, 0 specifies the shortest direction and 1 specifies the absolute position.

'*MANUAL*' allows the programmer to specify a vector that will be output with each point between *CUTCOM/ENDPT* and *CUTCOM/OFF*. The surface normal vectors will not be used if a *MANUAL* vector clause is specified.

Please note that ***the input cfile must contain the surface normal vectors to the part and drive surface in order to effectively use 3-D cutter compensation.*** If these vectors are not present in the cfile, then the tool axis vector will be used. If a simple motion record without the surface normal vectors is encountered during 3-D cutter compensation and surface normal vectors were previously encountered, then the last output normal vector will be used for the simple motion record.

7.15 Slowdown Blocks - SLOWDN/ON

Syntax: **SLOWDN/OFF**

ON[,n] [,NEXT]

AUTO

The *SLOWDN* command controls the output of slowdown blocks. Slowdown blocks are output to maintain a specified tolerance at the machine. Slowdown blocks may contain codes which will control the slowdown features at the machine or may cause **PWorks** to break up moves with a portion of the move being output with a slower feed rate. The format (set by the **MPost utility**) for slowdown blocks depends on the features of the machine control.

'*OFF*' disables the output slowdown blocks. '*ON*' enables the output of slowdown blocks.

When slowdown blocks are generated by **PWorks** by slowing down a portion of the move, then 'n' specifies the tolerance to maintain when generating the slowdown moves.

When slowdown blocks are supported at the machine and slowdown codes will be output, then 'n' specifies the slowdown mode to use at the machine. Check the machine's programming guide for a description of supported slowdown modes.

'NEXT' specifies that the *SLOWDN* command will be treated as a single shot command. Slowdowns will be enabled for the next move only and then immediately cancelled. 'AUTO' is the default qualifier and turns slowdowns on until a *SLOWDN/OFF* command is encountered.

Example:

SLOWDN/ON,.002

Enables slowdown spans until a *SLOWDN/OFF* command is encountered and sets the slowdown tolerance to .002. This syntax assumes **PWorks** will generate slowdown moves.

SLOWDN/ON,NEXT

Causes the next move only to be output in slowdown mode.

SLOWDN/ON,4

Enables slowdown mode #4 at the machine control. This syntax assumes the machine supports slowdown codes.

7.16 Acceleration Blocks - *SLOWDN/RAMP*

Syntax: ***SLOWDN/RAMP, OFF***

ON [, LIMIT, maxvel, axsvel] [, STEP, min, max]

The *SLOWDN/RAMP* command controls the output of acceleration blocks. Acceleration blocks are used to limit the velocity of the tool when accelerating in order to keep the torque on the tool within an acceptable range. They are not used to compensate for limitations of the machine control servos. Acceleration blocks should be enabled if there is a possibility that undue pressure can be applied to the tool during machine acceleration. Acceleration blocks are especially useful when transitioning from multi-axis moves, where the feed rates are slowed down dramatically due to the rotary axes speeds, to straight linear moves, where the feed rate is set back to the programmed feed rate

'OFF' disables acceleration blocks. 'ON' enables acceleration blocks. 'LIMIT' specifies the maximum (*maxvel*) and capped (*axsvel*) vector velocities. 'STEP' specifies the minimum (*min*)

and maximum (*max*) feed rate steps. The **MP**ost utility can be used to set the default values for ‘*macvel*’, ‘*axsvel*’, ‘*min*’ and ‘*max*’.

7.17 Transformation Blocks - TRANS

Syntax: **TRANS/ [TOOL** **]** **[[,] [VECTOR[,i,j,k]]] [[,] AUTO]**
[, NOMORE] **NOW**

This *TRANS* command allows for a simple way to transform a rotated coordinate system to the XY-plane by using the current tool location and tool axis. This is useful for controller routines that require that the XY-plane be active, such as automatic drilling cycles and circular interpolation. The “[Transformation Blocks](#)” section in the ***MPost*** manual contains a detailed description of this feature.

‘*TOOL*’ specifies the current tool location as the origin of the new coordinate system and ‘*VECTOR*’ specifies the current tool axis as the Z-axis of the new coordinate system. Either ‘*TOOL*’, ‘*VECTOR*’ or ‘*TOOL, VECTOR*’ is required for syntax.

'*i, j, k*' specifies the general direction of the X-axis for the new coordinate system and must be preceded by the required word *TOOL* or *VECTOR*.

'*AUTO*' will cause **PWorks** not immediately output a transformation block, but will rather scan the axes locations for a change in tool axis and then output a transformation block to match the new rotation. '*NOW*' is the default setting and will output a transformation block immediately.

TRANS/TOOL,NOMORE will cancel any combination of this version of the *TRANS* command.

7.18 Enable/Disable Transformation of Coordinates - TRANS/TOOL

Syntax: **TRANS/TOOL, ON**
OFF

These commands allow the programmer to enable/disable the transformation of coordinates when a transformation block is enabled. This feature will typically be used from within the XFORM Macro when it is called both prior to and after the motion block is output. When the pre-motion call is used to output the codes used to cancel transformation blocks, then TRANS/TOOL,OFF can be used. TRANS/TOOL,ON can be used in the section of the Macro called after the motion block is output to enable the transformation of coordinates.

The coordinate transformations could also be disabled if the programmer defines their own transformation matrix using the standard TRANS commands.

TRANS/TOOL, ON
OFF

The default condition ON will be set each time a TRANS/TOOL command is processed, other than TRANS/TOOL,OFF.

7.19 Automatic Tool Retraction - RETRACT/TOOL

Syntax: **RETRACT/TOOL** [,ON] [,CLRSRF,i,j,k,d] [,ATANGL,ang] \$
 OFF LENGTH[,ON] [,dis]
 OFF
 [OFFSET,UP [,dis1]] [,FEEDZ,fret[,fpln[,fofs]]]
 LEFT
 RIGHT

The *RETRACT/TOOL* command controls the automatic retraction and repositioning of the tool when (1) rotary motion takes the longest route due to machine limits or (2) the Ultrasonic Cutter changes direction by more degrees than the programmer specifies. The "[Rotary Axes Linearization](#)" section in the *MPost* manual contains a detailed description of this automatic retraction feature. The "[Ultrasonic Blade Cutter Set-up](#)" section describes this feature as it applies to Ultrasonic Cutters.

'*TOOL*' is required for proper syntax. '*ON*' enables the tool retraction feature when the rotary axes take the longest route. '*OFF*' disables this feature.

'*CLRSRF, i, j, k, d*' defines the plane that the tool will retract to when performing the automatic retract move due to the rotary axes taking the longest route. This plane is defined in the machine system, meaning that the tool end point will retract to this plane after it has been adjusted for any rotary tables.

'*LENGTH*' controls the distance the tool is actually retracted. '*ON*' will use the programmed tool length as part of the retract distance. '*OFF*' will disable the tool length from being used. '*dis*' specifies a distance in addition to the tool length to use as the retract distance. In essence, when '*ON*' is specified, then the retract distance will be '*tool length*' + '*dis*'. When '*OFF*' is specified, the retract distance will be simply '*dis*'.

'*ATANGL,ang*' is used with Ultrasonic Cutter machines and specifies the maximum angular change, in degrees, the blade axis can move while cutting. The tool will be automatically retracted, rotated, and repositioned if the direction of the blade changes by more than this amount. You can disable this feature by specifying a value of 90 degrees or more.

'*OFFSET*' specifies whether the tool will be shifted prior to retraction when the rotary axes are taking the longest route.

'*UP*' disables the shifting of the tool, '*LEFT*' shifts the tool to the left of the tool path, and '*RIGHT*' shifts the tool to the right of the tool path. Since *PostWorks* does not have any information on where the tool is in relationship to the part, due care should be practiced when

using this feature so that the tool does not violate the part when shifting the tool. **PostWorks** will check for a closed angled wall situation and temporarily disable the tool offset feature when it determines that the tool will violate the part due to this condition. This is the only checking that is performed for part violation.

'*disI*' specifies the distance to shift the tool. It is recommended that a minimal distance be used, since the goal of this movement is only to minimize dwell marks.

'*FEEDZ*' specifies the retract feed rate (fret), the plunge or repositioning feed rate (fpln) and the feed rate (fofs) for shifting the tool away from the part prior to retracting it and for repositioning the tool after it is plunged. A value of 0 for either of these values specifies the rapid rate.

Example:

```
RETRCT/TOOL,ON,LENGTH,ON,.5,FEEDZ,0,10
```

Enables the automatic retract feature when the rotary axes take the longest route. The retract distance will be the programmed tool length plus .5. The tool will retract at the rapid rate and plunge at 10 FPM.

```
RETRACT/TOOL,ATANGL,5,LENGTH,ON,0
```

Sets the maximum angular direction change for an Ultrasonic Cutter to 5 degrees. The tool will retract a distance equal to the programmed tool length.

7.20 Mill/Turn Turning Mode - **MODE/LATHE**

Syntax: **MODE/LATHE [, XYPLAN]
ZXPLAN**

This command disables the milling features of a Mill/Turn type machine and enables the turning capabilities. Lathe style commands (*TURRET*, *CYCLE/TURN*, etc.) should be used when turning mode is enabled. Use the *MODE/MILL* command to enable milling mode.

'*XYPLAN*' specifies that the input coordinates are programmed in the XY-plane. This is the normal manner for programming lathes. '*ZXPLAN*' specifies that the input coordinates are programmed in the ZX-plane. The output axes will always be in the ZX-plane.

7.21 Mill/Turn Milling Mode - **MODE/MILL,AUTO**

Syntax: **MODE/MILL , AUTO [, XYPLAN]
ZXPLAN**

The *MODE/MILL* command enables the milling features on a Mill/Turn type machine. The milling features of a Mill/Turn machine usually consists of a milling head attachment combined with using the spindle of the lathe as a programmable rotary axis. A third linear axis (Y-axis) may also be supported, along with a second rotary axis. Mill style commands ([LOADTL](#), [CYCLE/DEEP](#), etc.) should be used when milling mode is enabled. Use the *MODE/LATHE* command to enable turning mode.

'*AUTO*' activates milling mode by using the tool axis vector to calculate the rotary positions. The part should be defined as it actually looks, instead of all rotary positions being defined in the XY-plane. If your machine does not have a Y-axis, then you should use the [MODE/ROTATE,ON](#) command to switch the Y-axis positioning to the rotary axis. This command is described previously in this chapter.

'*XYPLAN*' specifies that the input coordinates are programmed in the XY-plane and should be converted to the ZX-plane. '*ZXPLAN*' specifies that the coordinates are programmed in the same plane as the output axes. These parameters allow you to define the milling features of the part in the same orientation as the lathe features.

7.22 Mill/Turn Cylindrical Interpolation - *MODE/MILL,DIAMTR*

Syntax: **MODE/MILL,DIAMTR** [, **AXIS,n,XAXIS**] [, **tol**]
YAXIS
ZAXIS

The *MODE/MILL,DIAMTR* command activated the cylindrical interpolation mode of the machine. The part program following this command should contain a 2-D tool path programmed in the XY-plane. **PostWorks** will internally wrap the tool path around the part using the last programmed X-axis position (input Y-axis position) as the part radius. Cylindrical interpolation eases the programming of grooves on a round part.

'*AXIS,n*' specifies the rotary axis which serves as the rotation axis of the cylinder. This axis must be a rotary table. The default is the first rotary axis.

'*XAXIS*', '*YAXIS*', and '*ZAXIS*' specify the major tool axis vector for use with cylindrical interpolation. This is the axis of the tool when the rotary axis specified by '*AXIS,n*' is at 0 degrees.

Some machines require a minimum movement span value to generate a satisfactory number of points to keep the programmed moves within tolerance of the part. The process of linearization in the machine control can also require some time and this delay may affect the programmed feed rate. '*tol*' specifies the chordal tolerance value to use during cylindrical interpolation and is used by **PostWorks** to calculate the minimum span, utilizing the feedrate value and machine delay. When the feed rate is too high for the programmed tolerance, then it will be slowed down.

7.23 Mill/Turn Polar Interpolation -- **MODE/MILL,FACE**

Syntax: **MODE/MILL,FACE** [,tol]

The *MODE/MILL,FACE* command activates the polar interpolation mode of the machine. The part program following this command should contain a 2-D path programmed in the XY-plane. **PostWorks** will translate the path to the part face and output the appropriate rectangular coordinates. The machine will use the rectangular coordinates to control the rotary axis.

Some machines require a minimum movement span value to generate a satisfactory number of points to keep the programmed moves within tolerance of the part. The process of linearization in the machine control can also require some time and this delay may affect the programmed feed rate. 'tol' specifies the chordal tolerance value to use during polar interpolation and is used by **PostWorks** to calculate the minimum span, utilizing the feedrate value and machine delay. When the feedrate is too high for the programmed tolerance, then it will be slowed down.

CHAPTER 8 Motion Adjustments

8.1 Preset Axis Registers - POSTN

Syntax: **POSTN**/ [**NORMAL**,] [[[,]**X****AXIS**[,**n**] [,**pos**]] \$
 INCR
 [[[,]**Y****AXIS**[,**n**] [,**pos**]] [,]**Z****AXIS**[,**n**] [,**pos**]] \$
 [[[,]**AXIS**[,**n**] [,**pos**]]] [...]

This version of the *POSTN* command outputs a preset axis registers block. A preset axis register block is usually used to enter offsets into registers at the control which will perform a linear compensation for the differences in the part program origin as compared to the origin on the machine.

'*NORMAL*' will output a preset axis registers in absolute mode block. '*INCR*' will output an incremental preset axis registers block. If neither of these parameters are specified, then the axis positions within **PWorks** will also be set to the values specified in this command.

'*-AXIS*' specifies which linear axes will be output in the preset axis registers block. Only the axis included on this command will be included in the preset block. '*AXIS*' selects a rotary axis.

'*n*' is an optional number and selects which of the specific axis to use. On the linear axis specifiers (*X***AXIS**, *Y***AXIS**, *Z***AXIS**), '*n*' can be either 1 (Primary) or 2 (Secondary). On the rotary axes parameter, '*n*' can be in the range of 1-4. If '*n*' is not specified then it will default to 1.

'*pos*' specifies the values to store in the preset axis registers. If this parameter is not specified with its corresponding axis designator, then a default value of 0 will be used.

Example:

```
POSTN/XAXIS,YAXIS,15,AXIS,2,360
```

Presets the X-axis to 0, the Y-axis to 15 and Rotary axis #2 to 360, both in the machine control and within **PWorks**. An absolute preset axis registers block will be output.

```
POSTN/INCR,YAXIS,2,3.5
```

Incrementally presets the secondary Y-axis by 2.5. This command will not change the position of the secondary Y-axis within **PWorks**.

8.2 FROM Preset Axis Registers - POSTN/XYPLAN

Syntax: **POSTN/XYPLAN**

This version of the *POSTN* command outputs a preset axis registers block. Unlike the *POSTN/-AXIS* command, the axes to preset are not contained in this command, but rather the axis coordinates programmed in the next *FROM* statement will be used. All supported axes will be output in the preset axis registers block.

'*XYPLAN*' is required for proper syntax.

Example:

POSTN/XYPLAN

Outputs an absolute preset axis register block when the next *FROM* statement is encountered. The coordinated of the supported axes will be output in the preset block.

8.3 Cancel Preset Axis Registers - POSTN/OFF

Syntax: **POSTN/OFF**

This version of the *POSTN* command cancels, at the machine control, any preset axis registers commands which have been previously programmed. It assumes that the machine control will accept a code to cancel a present axis registers block.

Example:

POSTN/OFF

Outputs a code that will cancel any preset axis registers a block which is in effect.

8.4 Part Origin - ORIGIN

Syntax: **ORIGIN/x,y[,z]**

The *ORIGIN* command defines the origin of the programmed part as compared to the origin of the part on the machine. This command will override the *TRANS/mx* command.

'x' is subtracted from the X dimension, 'y' from the Y dimension and 'z' from the Z dimension.

Example:

ORIGIN/0, 0, 5.25

Defines the part programmed origin (0, 0, 0) as being 0, 0, 5.25 on the machine. 5.25 will be subtracted from the Z-axis.

8.5 CI Point Modifications - TRANS

Syntax: **TRANS/ [SCALE,]n [, LAST]**
x, y, z

The *TRANS* command allows the programmer to alter the input ci points. 'x' is added to the X dimension, 'y' is added to the Y dimension and 'z' is added to the Z dimension. 'n' specifies the value for all 3 axes. 'SCALE' multiplies the axis positions by the values instead of adding them. This command will override the *TRANS/mx* command or by the *TRANS* command with the same format, otherwise it will work in conjunction with each other.

'LAST' applies the *TRANS* values after the points have been adjusted for the rotary axes. Usually, *TRANS* is applied to the input ci points.

Example:

TRANS/0

Cancels any translation amounts which were previously in effect.

TRANS/SCALE, 2, 1, 1

Assumes there is no other TRANS/n or TRANS/x,y,z command specified before this TRANS/SCALE command, the input X-axis coordinates for each point following this command will be doubled until cancelled by another *TRANS/SCALE* or *TRANS/mx* command. Otherwise it will work in conjunction with the current *TRANS/n* or *TRANS/x,y,z* command until cancelled by another TRANS/SCALE or TRANS/mx command.

TRANS/0, .5, 3.2, LAST

Adds .5 to the Y-axis and 3.2 to the Z-axis values of each point after the point has been rotated to compensate for the rotary axis angles.

8.6 CI Point Transformations - TRANS/mx

Syntax: **TRANS/i1, j1, k1, d1, i2, j2, k2, d2, i3, j3, k3, d3**

The *TRANS/mx* command defines a standard 3x4 matrix which is used to transform the input ci points prior to any other adjustment that may be applied to the ci points. This command will

override both the *ORIGIN* and *TRANS* commands. It can be cancelled using the *TRANS/0* command.

Example:

```
TRANS/0, -1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0
```

Defines an XY rotation of 90 degrees to apply to the input cl points.

```
TRANS/1, 0, 0, 2, 0, 1, 0, 2, 0, 0, 1, 2
```

Subtracts 2 from the X-axis, Y-axis, and Z-axis coordinates for each point following this command until cancelled by another *TRANS* command.

8.7 CL Point Translate Up Tool (Spindle) Axis - TRANS/UP

Syntax: **TRANS/UP,value**

The TRANS/UP command allows the programmer to translate all the input cl points up the tool axis (the spindle axis) after the standard translations (ORIGIN, TRANS) and matrix translation (TRANS/MX).

This command is useful if all the following conditions are true

1. An angled head attachment is utilized.
2. One of the rotary axes is a head configuration.
3. The output point to the punch file is the tool tip point.
4. The translated value is the distance between the tool tip and the rotation origin along the axis of the angle head attachment.

Example:

```
TRANS/UP,10
```

The input cl points will be translated 10 units up along the current cl points tool axis.

8.8 Output Axis Modifications - TRANS/-AXIS

Syntax: **TRANS/[SCALE] [[[,]XAXIS[,n] [,val]] \$**
[[[,]YAXIS[,n] [,val]] [[[,]ZAXIS[,n] [,val]] \$
[[[,]AXIS[,n] [,val]]] [...]

The *TRANS/-AXIS* command allows the programmer to alter the output linear and rotary axes. '*-AXIS*', used in conjunction with '*n*', selects which axis to adjust. '*n*' can be 1 (Primary) or 2 (Secondary) with the linear (*XAXIS*, *YAXIS*, *ZAXIS*). '*n*' can be 1 through 4 with the rotary axes (*AXIS*). If '*n*' is not specified then it will default to 1. The values '*val*' will be added to the output axes, unless the '*SCALE*' qualifier is specified, in which case the axes will be multiplied by these values.

Example:

TRANS/XAXIS,10, YAXIS,2,2.4

Adds 10 to the output primary X-axis and 2.4 to the secondary Y-axis, for each position following this command until cancelled by another *TRANS/-AXIS* command.

TRANS/SCALE,AXIS,2,.543

Multiplies Rotary axis #2 by .543 for each position following this command until cancelled by another *TRANS/SCALE,-AXIS* command.

8.9 Limit Planes - *CLRSRF/START*

Syntax: *CLRSRF/STOP*

```

NOMORE
sym      ,dir
z
a,b,c,d
START[,TRFORM,ON ][,IPM ,fd1]      $
                        OFF  MMPM

                        [,RTRCTO,p11[,IPM ,fd2]      $
                        ds1  MMPM
                        OFF

                        [,RAPTO[,p12][,IPM ,fd3]]      $
                        ds2  MMPM

                        [,PLUNGE,p13[,IPM ,fd4]]]
                        ds3  MMPM

```

This form of the *CLRSRF* command specifies a limit plane or planes to be used as the boundary for output axis locations. Up to ten planes may be used for boundaries. Only those locations on the specified side of the plane(s) will be output. Any points on the opposite side of the planes will be dropped. If a move crosses a plane's boundary, then the point on the specified side of the plane and the intersection point of the move and the plane will be output.

'*START*' restarts checking of axis locations, using the planes that were in effect when *CLRSRF/STOP* was invoked. '*STOP*' suspends the checking of cutter locations, but retains the limit planes for future use. '*NOMORE*' suspends the checking of axis locations and cancels any planes currently in effect. The default is '*NOMORE*'.

'*a, b, c, d*' is the canonical form of a plane to use as a boundary. '*sym*' is the name of a plane to use as a boundary. '*z*' is a Z-value for a '*0, 0, 1, z*' plane to use as a boundary.

'*dir*' is the direction modifier specifying which side of the plane boundary the output points are on and is used in conjunction with the '*a, b, c, d*', '*sym*', and '*z*' parameters. It can be one of the following.

NEGX, NEGY, NEGZ, POSX, POSY, POSZ

"*TRFORM,ON*" specifies that the clipping planes will be modified by the "*TRANS/x,y,z*", "*TRANS/matrix*", and "*ORIGIN/x,y,z*" settings. "*TRFORM,OFF*" does not modify the clipping planes and is the default setting.

"*IPM,fd1*" specifies the optional feed rate at which to move the tool across the cutting planes after the tool has been clipped to a plane. A value of zero "0" specifies a rapid move. The general cutting feedrate will be used if not specified.

"*IPM*" and "*MMPM*" can be used interchangeably and do not specify the actual units of the feed rate.

When an area bounded by a clipping plane is reentered the tool will normally move directly from the position where it left the non-clipped area to the location where it reenters the non-clipped area. When the *RTRCTO* parameter clause is specified, the tool will retract to the plane (*pl1*) or at the distance (*ds1*) specified, then move above the reentry point at either the plane "*RAPTO,pl2*" or distance above (*ds2*). The tool will then plunge above this location at the defined plane "*PLUNGE,pl3*" or at a distance (*ds3*) above the reentry location.

The plane specified with the *RAPTO* parameter (*pl2*) is optional. If it is not specified then the *RTRCTO* plane (*pl1*) will be used.

The tool will retract at the feed rate specified by "*fd2*", position above the reentry location using the main feed rate specified in the *CLRSRF* command (*fd1*), plunge above the reentry location using the feed rate specified by "*fd3*", and finally position at the reentry location at the feed rate specified by "*fd4*".

You may specify a value of "0" for the *RTRCTO* feed rates, which signifies that the move should be made at the rapid rate. If a feed rate is not specified for one of these parameters, then the programmed feed rate will be used.

"*RTRCTO,OFF*" disables the retract logic and causes the tool to be repositioned directly to the reentry location.

The direction of the retract motion and the reentry motion will be along the corresponding tool axis vectors at where the motion is clipped.

Example:

```
CLRSRF/0,0,1,10,NEGZ
CLRSRF/-5,POSZ
CLRSRF/START
```

All points over 10 in Z or under -5 in Z will not be output to the control tape. Any moves crossing the limit planes will consist of an intermediate point on the crossed plane.

```
CLRSRF/STOP
```

Limit plane checking is terminated, but the defined planes are retained for future use.

8.10 Creating Intermediate Moves - LINTOL/LINEAR

Syntax: **LINTOL/LINEAR,delta**
TOLER,tol
OFF

This form of the *LINTOL* command breaks up single moves into multiple moves. When *LINTOL* is in effect **PWorks** will break up a move into multiple moves whenever the delta distance is greater than the specified amount or the angle of the move exceeds a specified tolerance.

'*LINEAR*' specifies that the moves should be broken up according to the length of the move. 'delta' is the maximum length of a move to be output. '*TOLER*' specifies that the moves should be broken up when the angle of the move in comparison to one of the 3 major axis (X, Y, Z) exceeds the tolerance specified by 'tol'.

'*OFF*' turns off linearization and can be specified with either '*LINEAR*' or '*TOLER*'.

Circular motion will not be broken up at any time.

Example:

```
LINTOL/LINEAR,.150
```

Any input move which exceeds .150 will be broken up into smaller moves so that no single move is greater than .150.

8.11 Rotary Axis Center - ROTABL/ORIGIN

Syntax: **ROTABL/ORIGIN** [, **AXIS** , **n**] , **x** , **y** , **z**
ROTHED

The *ROTABL/ORIGIN* command defines the absolute center position of the specified rotary axis in relationship to the part program origin (0, 0, 0 location). The linear axes will be rotated around this center point when the rotary axis moves. When defining the center of a rotary head, this command is similar to the *PIVOTZ* command in a custom post-processor.

'*AXIS,n*' selects the rotary axis to define the center point for. 'n' can be in the range of 1-4, with the default being 1.

Example:

```
ROTABLE/ORIGIN,0,10,15
```

Defines the center of Rotary axis #1 as X=0, Y=10, Z=15, as compared to the programmed part origin. It is assumed that this axis is a rotary table which rotates about the X-axis.

8.12 Rotary Axis Circumference - ROTABL/SIZE

Syntax: **ROTABL/SIZE** [, **AXIS** , **n**] , **c**
ROTHED

The *ROTABL/SIZE* command will change the circumference size of the specified rotary axis. This command should only be used when the machine control expects the rotary axis to be output in relationship to the circumference of the rotary axis and not in degrees, for example a Bostomatic machine. The new circumference will be output in a "rotary axis size change" block and will be used internally by **PWorks** to calculate this rotary axis output.

'*SIZE*' is required to differentiate this command from other *ROTABL* commands.

'*AXIS,n*' specifies which of the rotary axes to apply the new circumference to. 'n' can be in the range of 1-4. If this parameter is not specified, then Rotary axis #1 is assumed by default.

'c' specifies the new circumference for the specified rotary axis.

Example:

```
ROTABL/SIZE,AXIS,2,10
```

Changes the circumference of Rotary axis #2 to 10. It is assumed that this rotary axis is not output in degrees.

8.13 Rotary Axis Coordinate Rotations - ROTABL/ADJUST

Syntax: **ROTABL/ADJUST, TABLE, ON**
ROTHED HEAD OFF

The *ROTABL/ADJUST* command controls the rotation of the input coordinates based on the position of the rotary axes. The adjustment of the input coordinates can be separated for rotary tables and rotary heads.

'*TABLE*' specifies that this command controls the table rotations only. '*HEAD*' controls the adjustment for rotary heads. '*ON*' enables the rotation of the input coordinates for this type of rotary axis and is usually used when the part is defined in one coordinate system and the tool axis is used to move the rotary axes. '*OFF*' disables the rotation of the input coordinates for this type or rotary axis. Use '*OFF*' when the part geometry is defined in all planes of rotary axis movement and the *ROTABL/ATANGL* command will be used to position the rotary axes.

The **MPost utility** is used to set the defaults for this command.

8.14 Tolerance Truncating Tool Axis Vector Components - PPTOL/VECTOR

Syntax: **PPTOL/VECTOR, tol**

The *PPTOL/VECTOR* command defines the tolerance "tol" to use for the tool axis vector components (IJK) to determine when a component should be set to zero. Whenever a tool axis component is less than this value, it will be set to zero. This feature is useful for locking out a tool axis component during 4-axis motion.

The **MPost utility** is used to set the defaults for this command.

CHAPTER 9 Circular/Spline Interpolation

9.1 Circular Interpolation Output - ARCSLP

Syntax: **ARCSLP/LINCIR,ON**
OFF

The *ARCSLP* command specifies whether or not to output circular blocks when a circular record is encountered. Circular blocks will only be output when the following conditions are true.

1. The drive surface is either a circle or a cylinder.
2. *ARCSLP/LINCIR,ON* is in effect.
3. A canned cycle is not in effect.
4. The radius is greater than/equal to the minimum radius allowed and is less than/equal to the maximum radius allowed.
5. The radius tolerance is within the allowable range.
6. The linear delta movement is greater than the minimum allowed.
7. Cutter compensation is not active or is in the same plane as the circular record and there is a linear move between *CUTCOM/ON* and the circular move.
8. There is no 3-axis move in the circular record.

Example:

ARCSLP/LINCIR,OFF

Disables the output of circular interpolation records. All input circular motion will be output as a series of linear moves.

9.2 Circular Tolerances - MCHTOL

Syntax: **MCHTOL/cirtol[,cirdlt]**

The *MCHTOL* command defines the tolerance which determines when circular interpolation blocks will be output.

'*cirtol*' defines the tolerance allowed for the radius of the circle or cylinder, calculated using the center line points of the cutter. If a point goes outside this tolerance in comparison with the starting point of the circular block, then the rest of the points in the circular record will be output as linear motion.

'*cirdlt*' defines the minimum linear movement to be output as circular motion. If the delta linear movement of a circular block is less than '*cirdlt*', then the circular record will be output as linear motion.

Example:

```
MCHTOL/,002,.100
```

Each point in an input circular interpolation record will be checked against the radius of the circle as calculated using the starting point of the circular arc. If any of the points differs from this radius by more than .002, then the circular record will be output as a series of linear moves.

The total delta movement of the output circular block will be checked to see if it is at least .100 inch/millimeter. If it is not, then the circular record will be output as a series of linear moves.

9.3 Circular Interpolation Analyzation - SET/LINCIR

Syntax: **SET/***LINEAR*

```
LINCIR [,cirtol] [,ALL ]  
COMBIN
```

This command allows the post to check multiple point-to-point motion blocks for circular movement. The motion does not necessary perpendicular to the tool axis as long as the motion will be output on a major machine plane. If the motion is circular in nature and is not marked as a circular block, then the post will convert it to circular interpolation. Only motion blocks with three or more points will be checked for circular movement. This command is useful when processing input clfiles from CAM systems which do not support circular interpolation, or when driving curves or surfaces which are circular in nature.

'*LINEAR*' disables circular interpolation analyzation. '*LINCIR*' enables analyzation.

'*cirtol*' specifies the tolerance to use when analyzing input points for circular interpolation.

'*ALL*' allows multiple circular records to be output for a single motion record. **PWorks** will check each motion record with over 3 points for separate circular paths within the record. If this keyword is not specified, then all the points within the motion record must be in tolerance of the generated circle for a circular record to be output.

'*COMBIN*' performs the same function as '*ALL*' except this function buffers multiple '*GOTO*' point records into a single motion record prior to circular checks. When '*COMBIN*' is enabled, multiple motion records will be continuously combined into single motion records, thereby distorting the input sequence numbers in the output files. '*COMBIN*' should not be used for **NCL** binary cl motion record.

Example:

```
SET/LINCIR,.002,ALL
```

Circular interpolation analyzation will be performed on all point-to-point motion records. Multiple circular moves which are within a single motion record will be output as separate circular blocks. The tolerance used for determining circular moves is .002.

9.4 Helical Interpolation

The following sections describe post-processor commands which control the output of helical interpolation.

9.4.1 Helical Interpolation Control - COUPLE/AUTO

Syntax: **COUPLE/AUTO, ON**
OFF

The *COUPLE/AUTO* command controls whether helical interpolation records or a series of linear moves will be output for helical moves. 'ON' selects the output of helical interpolation records which consist of circular interpolation codes along with the third linear axis depth and optionally the lead of the depth. 'OFF' specifies that the control does not support helical records, therefore the helical motion will be output as a series of linear moves generated by **PWorks**.

The **MPost** utility is used to set the default output for helical interpolation records.

9.4.2 Helical Interpolation - COUPLE

Syntax: **COUPLE/OFF**
n, ATANGL, a

The *COUPLE* command causes the next circular record to be output as a helical move in the programmed circular plane. The machine will travel the distance specified by 'n' in the linear axis for every 'a' degrees around the circle.

The programmer drives the circle in the circular plane and **PWorks** will control the linear axis. Because of this feature, the programmer should be careful since **PWorks** will contain a different value for the linear axis than the CAM software. This command will stay in effect until a linear motion or *COUPLE/OFF* command is encountered.

'n' is an incremental value from the current linear axis position. A positive value will move the machine in the negative direction and a negative value will move the machine in the positive direction.

'a' specifies the angle to apply the value 'n' to. The machine will move a delta of 'n' for every 'a' degrees around the circle. **PWorks** will generate the helical moves if the machine does not support helical interpolation. Specifying a zero value for the angle causes **PWorks** to use the

entire angular movement of the following circular interpolation record for the incremental distance 'n' specified in the command.

'OFF' cancels helical interpolation. A straight linear motion will also cancel helical interpolation.

Example:

```
COUPLE/1,ATANGL,180
```

Causes the next circular record to be output as a helical move. The linear axis (usually the Z-axis) will move 1.0 inch/millimeter for every 180 degrees around the circle. For example, if the circular arc is 360 degrees, then the linear axis will move a total of 2.0 inches/millimeters.

9.4.3 Helical Interpolation - CYCLE/CIRCUL

Syntax: **CYCLE/CIRCUL,DEPTH,z,RADIUS,d,ON,[TOOL,tdia,] \$**
DIAMTR IN

```
STEP,zi[,IPM ,feed[,rap]][,RAPTO,r] $  
IPR  
MMPM  
MMPR  
  
[,ATANGL,a][,DOWN][,CCLW][,RTRCTO,ret]  
UP CLW
```

The *CYCLE/CIRCUL* command causes the next GOTO point to be output as a helical move.

'z' specifies the depth of the helix, measured from the top of the part. A positive value specifies a helix motion into the part. A negative value specifies a helix motion out of the part.

'd' specifies the radius/diameter of the helix. 'ON' specifies the helix radius/diameter is same as 'd'. 'IN' specifies the helix radius will be equal to '*d - cutter radius*' if RADIUS is specified, or the helix diameter will be equal to "*d - cutter diameter*" if DIAMTR is specified.

'tdia' specifies the cutter diameter which will be used to calculate the helix radius if required. The diameter value specified from the last *CUTTER* statement will be used if not specified.

'zi' specifies the Z-increment per one helical rotation.

'feed' is the cutting feed rate to perform the helical cycle at. It can be specified in Feed per Minute ('IPM' or 'MMPM') or Feed per Spindle Revolutions ('IPR' or 'MMPR').

'*rap*' specifies the feed rate to use when calculating the machining time for rapid moves during the cycle. A value of 0 for '*rap*' will cause the rapid rate to be used.

'*r*' specifies the rapto distance above the top of the part.

'*a*' specified the starting angle of the helix relative to the positive X-axis. The default is 0.

'*DOWN*' specifies that the tool is to remain at the bottom of the helix at the end of the cycle. '*UP*' causes the tool to be retracted to the clearance plane. The default is *DOWN*.

'*CCLW*' and '*CLW*' specifies the helix direction. The default is *CCLW*.

'*RTRCTO*' specifies the retract locations for the helical cycle. '*ret*' specifies the level above to retract to.

The helical motion will be generated as described below:

1. Rapid to the *RAPTO* plane above the helix center.
2. Feed to the starting angle of the helix at the top of the part.
3. Perform the helical motion.
4. Drive around the circle at the bottom of the part.
5. Retract the tool to the '*ret*' level if *UP* is specified.

9.5 Spline Interpolation - ARCSLP/SPLINE

Syntax: **ARCSLP/SPLINE, ON**
OFF

The *ARCSLP/SPLINE* command controls the output of spline interpolation records. Spline interpolation eliminates the small linear moves required to approximate a curve, replacing these moves with a concise curve definition which typically generates less tape and a smoother tool path on the machine.

All cl points within an *ARCSLP/SPLINE* sequence will be evaluated and the best spline that will fit within a tolerance of these points will be determined. **PostWorks** utilizes its own curve fitting algorithms, so no special data is required in the clfile. During spline fitting, post-processor words and circular interpolation records will interrupt the evaluation of a spline. Splines can also be broken up whenever the calculations determine a spline must be split up into two or more splines.

This command will have no effect if the machine controller does not support spline interpolation.

Spline interpolation formats currently supported are for the Siemens 880 control and the Fanuc 16 control. See "[Spline Interpolation Format](#)" section of **MPost** utility for details.

Example:

ARSCLP/SPLINE,ON

Enable the spline evaluation routines. Spline interpolation will be output for all cl points which generate a valid spline.

CHAPTER 10 Canned Cycles

10.1 Cycle Control Commands

The following sections describe post-processor commands which control the output of cycle sequences, including automatic and manual cycles.

10.1.1 Automatic/Canned cycles - CYCLE/AUTO

Syntax: **CYCLE/AUTO,ON**
OFF

The *CYCLE/AUTO* command controls whether automatic (canned) or **PWorks** generated (manual) cycle sequences will be output.

'On' enables the output of automatic cycle sequences when appropriate. See the "Cycle Output control" section for a description of when automatic cycle sequences will be output. 'OFF' disables the output of automatic cycle sequences, **PWorks** will simulate all cycle sequences.

Example:

CYCLE/AUTO,ON

Enables the output of automatic cycle sequences.

10.1.2 Cycle Output Control - CYCLE

Syntax: **CYCLE/ON**
OFF

This version of the *CYCLE* command controls the output of canned (automatic) or **PWorks** generated (manual) cycles. All programmed tool locations between *CYCLE/ON* and *CYCLE/OFF* will be modified to perform the requested cycle. See also the [*CYCLE/mill_mode*](#) and [*CYCLE/lathe_mode*](#) commands

'ON' specifies that the last cycle mode in effect be reinstated. All parameters specified on the last *CYCLE/mode* command will also be reinstated. An error message will be output and this command will be ignored if 'ON' is programmed without a previous *CYCLE/mode* command programmed. 'OFF' terminates all cycles until another *CYCLE* command is programmed.

Mill Considerations

All descriptions for mill cycles assume a tool axis vector of 0,0,1. The cycle motion will be performed along the Z-axis.

The active cycle sequence, except for *CYCLE/MILL* sequences, will remain in effect until explicitly cancelled by another *CYCLE* command. *CYCLE/MILL* sequences will be automatically cancelled after the first point.

The *GOTO* points programmed during a cycle sequence should be defined at the top of the part, since the rapto plane and the final depth are defined as incremental distances from this level.

PWorks will only output automatic cycle sequences when the following conditions are true.

- 1) *CYCLE/AUTO,ON* is in effect.
- 2) A cycle definition code has been defined for the active cycle.
- 3) The cycle motion is along the defined spindle axis.

If *RETRACT/ON* is in effect, then the tool will retract to the Z-level in effect at the time the cycle sequence was initiated, after completing the cycle motion. If *RETRACT/OFF* is in effect, then the tool will retract to the rapto plane, as defined by the '*RAPTO,r*' cycle parameter, after completing the cycle motion. The '*RTRCTO*' cycle parameter can be used to change the retract plane on manual cycles.

The first move in a manual cycle sequence will be output using the *RAPID/TOOL* logic to control the feed rate and positioning mode.

If the *RAPID* command is programmed during a cycle sequence, then the current cycle will be suspended for the rapid motion. The move programmed with *RAPID* will be output as a positioning move and the cycle sequence will be reinstated immediately after this move.

Lathe Considerations

The *GOTO* points programmed during a cycle sequence should be defined at the ending location for the cycle in both the Z-axis and X-axis. *CYCLE* commands can be fully self contained, with the ending position programmed on the *CYCLE* command. In this case, a *GOTO* point is not required.

Cycle sequences are single shot commands and will be automatically cancelled after the first point.

PWorks will only output automatic cycle sequences when the following conditions are true.

- 1) *CYCLE/AUTO,ON* is in effect.
- 2) A cycle definition code has been defined for the active cycle.

10.2 Automatic Mill Cycles - CYCLE/mode

Syntax: **CYCLE/mode**,**FEDTO**,**depth** [, **IPM** ,**feed** [,**rap**]] [,**RAPTO**,**r**] \$
 dia,**ang** **IPR**
 MMPM
 MMPR
 TPI

 [,**DWELL** [,**dw11** [,**dw12**]]] [,**STEP**,**peck1** [,**peck2**]] \$

 [,**OFFSET**,**off1** [,**off2**]] [,**RTRCTO**,**ret** [,**plng**]] \$

 [,**REPEAT**,**rep**] [,**START** [,**seq** [,**inc** [,**frq**]]]] \$

This version of the *CYCLE/mode* command is only valid for mills and outputs an automatic cycle sequence to the Control Tape. The machine control will perform the actual cycle. Please note that the descriptions of the *CYCLE* parameters explain their normal usage and can actually be different depending on how the cycle parameters have been set up in the **MPost** utility.

'mode' specifies the type of canned cycle to output and can be one of the following.

BORE	=	Performs a boring cycle, stopping the spindle at the bottom of the hole and feeding out.
BORE7	=	Performs a boring cycle, stopping the spindle at the bottom of the hole and allowing for a manual feed.
BORE8	=	Performs a boring cycle, stopping the spindle at the bottom of the hole and rapiding out.
BORE9	=	Performs a back boring cycle, with a tool shift at the top of the hole and the cutting being performed from the bottom of the hole.
DEEP	=	Performs a chip breaking drill cycle.
DRILL	=	Performs a drilling cycle with rapid out.
FACE	=	Performs a drilling cycle with a dwell at the bottom of the hole.
MILL	=	Performs a positioning only cycle.
REAM	=	Performs a reaming cycle with feed out.
REVERS	=	Performs a left handed thread tapping cycle.
SHIFT	=	Performs a fine boring cycle, orientating and shifting the tool at the bottom of the hole.
TAP	=	Performs a right handed thread tapping cycle.
THRU	=	Performs a deep hole drilling cycle, with a full retract to the clearance plane after each cut.

'*FEDTO,depth*' specifies the distance below the Z-level of each point to feed to. You can use the '*dia,ang*' parameters instead of the '*depth*' parameter when chamfering a hole with an angle

cutter. 'dia' specifies the diameter of the chamfer and 'ang' is the chamfer angle. The final depth is calculated using the following formula.

$$\text{depth} = .5 * \text{dia} / \text{TAN}(\text{ang}/2)$$

'feed' is the cutting feed rate to perform the cycle at. It can be specified in Feed per Minute ('IPM' or 'MMPM') or Feed per Spindle Revolutions ('IPR' or 'MMPR'). 'TPI' specifies the number of threads per inch/mm and is usually used with a tapping cycle. If 'TPI' is specified and 'feed' is less than 1 (inch) or 24.5 (mm), then 'feed' will be considered to be the thread lead. If 'feed' is greater than 1 (inch) or 25.4 (mm), then 'feed' will be considered to be the number of threads per inch/mm. **PWorks** will calculate the actual cutting feed rate when 'TPI' is specified.'

'rap' specifies the feed rate to use when calculating the machining time for rapid moves during the cycle. A value of 0 for 'rap' will cause the rapid rate to be used.

'RAPTO,r' specifies the distance above the Z-level of each point to rapid to.

'DWELL' specifies that the cycle is to dwell at the bottom of the hole and optionally at the top of the hole on pecking cycles. 'dwl1' specifies the amount of time in seconds to dwell at the bottom of the hole. 'dwl2' specifies the amount of time in seconds to dwell at the top of hole after each tool retraction in a pecking cycle.

'STEP' will cause a multiple pass cycle to be performed. 'peck1' specifies the depth of the first cut. 'peck2' specifies the depth of the last cut. All intermediate cuts will be at a depth that is between 'peck1' and 'peck2'. If 'peck2' is not specified or is the same value as 'peck1', then all passes will make the same size cut into the part.

'OFFSET' specifies an offset amount for back boring or fine boring cycles. 'off1' when specified without the 'off2' parameter, specifies the amount of shift for the tool in a direction opposite of the spindle orientation. When specified with the 'off2' parameter, 'off1' is the amount of shift in the X-axis and 'off2' is the amount of shift in the Y-axis.

'RTRCTO' specifies the retract locations for each cut of a cycle and is usually used with pecking cycles. 'ret' specifies the level above each cut to retract to when performing a pecking cycle. 'plng' specifies the level above the previous cut to plunge to after retracting the tool.

'REPEAT,l' specifies the number of repetitions for each cycles block to perform at equal delta increments.

'START' specifies that an alignment block will be output at each hole during a cycle sequence. 'seq' specifies the starting sequence number for these alignment blocks. If 'n' is not specified, then the normal sequencing scheme as specified in the [SEQNO](#) command will be used. 'inc' specifies the value to increment the alignment block numbers by and will default to 1. 'frq' specifies the frequency of holes which will contain an alignment block and will default to 1.

10.3 Automatic Mill Cycles For Siemens 840D - CYCLE/...,PARAMS,m1,n1,..

Syntax: **CYCLE/ . . . , PARAMS,m1,n1, [. . .]**

The Siemens 840D controller uses *Macro Call CYCLE* output instead of the regular G-code. This version of the *CYCLE* command will cause **PWorks** to utilize the User Defined Blocks to generate an automatic cycle sequence as a *Macro Call* for each point programmed between *CYCLE/mode,...PARAMS,m1,n1,...* and *CYCLE/OFF*. “m1” specifies the parameter defined and “n1” specifies the corresponding assigned parameter value in the *Macro Call CYCLE* output. A maximum of 30 pairs of parameter and value can be defined with this command.

Following steps show how to use this command with the **MPost** Utility to output the required codes.

1. Toggle [Menu item 8.1.3 : “Output Cycles as a Macro Call”](#) in the **MPost** utility to ‘YES’.
2. This will activate the next menu item [8.1.4: “Macro Call Code.”](#). This is defaulted to MA(0). Use another physical register if the physical register MA is used for other purpose.
3. With this activated, the [Cycle Parameter Codes prompts \(8.2\)](#) of the **MPost** utility will expect a Macro parameter position as input instead of a physical register. For example, the drilling cycle has the following syntax:

CYCLE81 (RTP, RFP, SDIS, DP, DPR)

where:

RTP	=	Return plane (Absolute)
RFP	=	Reference plane (Absolute)
SDIS	=	Safety distance (without sign)
DP	=	Final drilling depth (Absolute)
DPR	=	Final drilling depth relative to reference plane (without sign)

Assuming absolute mode is used, the appropriate prompts in the **MPost** utility should have the following values assigned to them to obtain this output for the cycle registers.

- [8.2.1](#) Enter file final depth register: 4
- [8.2.2.](#) Enter To-of-part register: 2
- [8.2.3](#) Enter Rapto plane register: 3
- [8.2.14](#) Enter RTRCTO parameter 1 register: 1

4. Assume the register MA is used for the [Macro Call code](#) output and the register G6 is used for the cycle code output. Modify these two registers to have the format as shown below.

Case A. Assume [Menu item 3.8.4 ‘Separate tape registers with a space in Punch file’](#) is set to YES in the **MPost** Utility.

<u>Register</u>	<u>Begin</u>	<u>Format</u>	<u>Sign</u>	<u>Max</u>		<u>Min</u>		<u>End</u>	<u>Control</u>
				<u>Left</u>	<u>Right</u>	<u>Left</u>	<u>Right</u>		
G6	CYCLE	Decimal			whatever required				Changed
MA	MCALL	Decimal		0	0	0	0		Set

Case B. Assume [Menu item 3.8.4 ‘Separate tape registers with a space in Punch file’](#) is set to NO in the **MPost** Utility.

<u>Register</u>	<u>Begin</u>	<u>Format</u>	<u>Sign</u>	<u>Max</u>		<u>Min</u>		<u>End</u>	<u>Control</u>
				<u>Left</u>	<u>Right</u>	<u>Left</u>	<u>Right</u>		
G6	CYCLE	Decimal			whatever required				Changed
MA	MCALL\	Decimal		0	0	0	0		Set

- User Defined blocks for cycles output as *Macro Calls* are also treated differently than for standard cycle blocks. These User Defined Blocks should contain the order of the Macro parameters as they are assigned to the various cycle registers ([Menu items 8.2](#) of the **MPost** utility). The reordering of the Macro parameters is required in some of the cycle blocks, due to the fact that the cycle parameter may be defined in a different Macro parameter position in varying cycles. For example, the dwell register is the sixth (6th) Macro parameter in the *CYCLE82* block, but the ninth (9th) Macro parameter in the *CYCLE83* block. Therefore, the order of the Macro parameters in the *CYCLE83* block would have to be rearranged prior to output. The following User Defined block for *CYCLE83* blocks (assuming using #13) would accomplish this.

<u>User Defined Block #</u>	<u>Definition</u>
13	1, 2, 3, 4, 5, 9, 7, 8, 6

Sometimes there will be more parameters in the *CYCLE Macro Call* than that can be defined by using the cycle registers. The ‘*PARAMS,m1...*’ is used for this purpose where “*m1*” specifies the value assigned to the first extra parameter, “*m2*” specifies the value assigned to the second extra parameter, “*mn*” specifies the value assigned to the nth parameter.

Finally **PWorks** will attempt to determine if there is a single point or multiple points following a cycle definition block. If there are multiple points, then the [Macro Call code](#), MCALL, will be output along with the cycle definition block.

Example:

MCALL CYCLE81 (10.25, 8.25, .1, 7.5)

The cycle will stay active until a *CYCLE/OFF* is reached, and then the *Macro Call code* will be output on a block by itself to terminate the modal *Macro Call*. When a single point follows the cycle definition block, then this point will be output prior to the cycle definition block and the *Macro Call* code will not be output. The cycle will automatically be cancelled after the one point.

10.4 Manual Mill Cycles

The following sections define the **PWorks** generated (manual) mill cycles supported. The command syntax, available parameters and their functions, and the motion simulation which will be performed for each cycle will be discussed.

10.4.1 Boring Cycle - CYCLE/BORE

Syntax: **CYCLE/BORE,FEDTO,depth** [,IPM ,feed[,rap]] [,RAPTO,r] \$
 dia,ang IPR
 MMPM
 MMPR

 [,DWELL[,dwl]] [,RTRCTO,ret]

This version of the *CYCLE* command will cause **PWorks** to generate a boring sequence for each point programmed between *CYCLE/BORE* and *CYCLE/OFF*.

All parameters on this command will perform the same function as described in the '*Automatic Mill Cycles*' section, except for the '*RTRCTO*' parameter.

'*RTRCTO,ret*' specifies the distance above the Z-level of the point to retract to after the cycle is finished. If '*RTRCTO*' is not specified, then the Z-level implied in the *RETRACT* command will be used.

Cycle Simulation

1. Rapid to XY of point.
2. Rapid to 'r'.
3. Feed in to 'depth'.
4. Optional dwell.
5. Stop spindle.
6. Feed out.
7. Start spindle.

10.4.2 Boring Cycle - CYCLE/BORE7

Syntax: **CYCLE/BORE7**,FEDTO,depth [,IPM ,feed[,rap]][,RAPTO,r] \$
 dia,ang IPR
 MMPM
 MMPR

 [,DWEEL[,dwl]][,RTRCTO,ret]

This version of the *CYCLE* command will cause **PWorks** to generate a boring sequence for each point programmed between *CYCLE/BORE7* and *CYCLE/OFF*.

All parameters on this command will perform the same function as described in the '[Automatic Mill Cycles](#)' section, except for the '*RTRCTO*' parameter.

'*RTRCTO,ret*' specifies the distance above the Z-level of the point to retract to after the cycle is finished. If '*RTRCTO*' is not specified, then the Z-level implied in the *RETRACT* command will be used.

Cycle Simulation

1. Rapid to XY of point.
2. Rapid to 'r'.
3. Feed in to 'depth'.
4. Optional dwell.
5. Stop spindle.
6. STOP (manual feed out).
7. Start spindle.

10.4.3 Boring Cycle - CYCLE/BORE8

Syntax: **CYCLE/BORE8**,FEDTO,depth [,IPM ,feed[,rap]][,RAPTO,r] \$
 dia,ang IPR
 MMPM
 MMPR

 [,DWEEL[,dwl]][,RTRCTO,ret]

This version of the *CYCLE* command will cause **PWorks** to generate a boring sequence for each point programmed between *CYCLE/BORE8* and *CYCLE/OFF*.

All parameters on this command will perform the same function as described in the '[Automatic Mill Cycles](#)' section, except for the '*RTRCTO*' parameter.

'*RTRCTO,ret*' specifies the distance above the Z-level of the point to retract to after the cycle is finished. If '*RTRCTO*' is not specified, then the Z-level implied in the *RETRACT* command will be used.

Cycle Simulation

1. Rapid to XY of point.
2. Rapid to 'r'.
3. Feed in to 'depth'.
4. Optional dwell.
5. Stop spindle.
6. Rapid out.
7. Start spindle.

10.4.4 Back Boring Cycle - CYCLE/BORE9

Syntax: **CYCLE/BORE9**,FEDTO,depth [,IPM ,feed[,rap]][,RAPTO,r] \$
dia,ang IPR
MMPM
MMPR

[,DWEELL[,dwl]][,OFFSET,off1,off2][,RTRCTO,ret]

This version of the *CYCLE* command will cause **PWorks** to generate a back boring sequence for each point programmed between *CYCLE/BORE9* and *CYCLE/OFF*.

All parameters on this command will perform the same function as described in the '[Automatic Mill Cycles](#)' section, except for the '*RTRCTO*' parameter.

'*RTRCTO,ret*' specifies the distance above the Z-level of the point to retract to after the cycle is finished. If '*RTRCTO*' is not specified, then the Z-level implied in the *RETRACT* command will be used.

Cycle Simulation

1. Orientate spindle.
2. Rapid to XY of point.
3. Shift tool.
4. Rapid to 'r'.
5. Shift tool.
6. Start spindle.
7. Feed in to 'depth'.
8. Optional dwell.
9. Orientate spindle.
10. Shift tool.
11. Rapid out.
12. Shift tool.
13. Start spindle.

10.4.5 Chip Breaking Drill Cycle - CYCLE/DEEP

Syntax: **CYCLE/DEEP,FEDTO,depth** [,IPM ,feed[,rap]] [,RAPTO,r] \$
 dia,ang IPR
 MMPM
 MMPR

 [,DWELL[,dw11[,dw12]]] [,STEP,peck1[,peck2]] \$

 [,RTRCTO,ret]

This version of the *CYCLE* command will cause **PWorks** to generate a chip breaking drill sequence for each point programmed between *CYCLE/DEEP* and *CYCLE/OFF*.

All parameters on this command will perform the same function as described in the '[Automatic Mill Cycles](#)' section.

'*RTRCTO,ret*' specifies the distance above the Z-level of the point to retract to after the cycle is finished. If '*RTRCTO*' is not specified, then the Z-level implied in the *RETRACT* command will be used.

Cycle Simulation

1. Rapid to XY of point.
2. Rapid to 'r'.
3. Will a move of 'peck stop past the final depth?
If 'Yes', Go to Step 9.
4. Feed in 'peck'.
5. Optional 'dw11' dwell.
6. Optional retract.
7. Optional 'dw12' dwell.
8. Go to Step 3.
9. Feed in to 'depth'.
10. Optional 'dw11' dwell.
11. Rapid out.

10.4.6 Drill Cycle - CYCLE/DRILL

Syntax: **CYCLE/DRILL, FEDTO, depth** [, IPM , feed[, rap]] [, RAPTO, r] \$
FACE **dia, ang** **IPR**
MMPM
MMPR

[, DWELL[, dwl]] [, RTRCTO, ret[, plng]]

This version of the *CYCLE* command will cause **PWorks** to generate a drilling sequence for each point programmed between *CYCLE/mode* and *CYCLE/OFF*.

Although both of the acceptable cycle mode parameters, 'DRILL' and 'FACE', will be treated the same by **PWorks**, it is recommended to use the following conventions.

DRILL = No dwell specified.
FACE = Use when *DWELL, dwl* is included on the command.

All parameters on this command will perform the same function as described in the '[Automatic Mill Cycles](#)' section, except for the 'RTRCTO' parameter.

'RTRCTO, ret' specifies the distance above the Z-level of the point to retract to after the cycle is finished. If 'RTRCTO' is not specified, then the Z-level implied in the *RETRACT* command will be used.

Cycle Simulation

1. Rapid to XY of point.
2. Rapid to 'r'.
3. Feed in to 'depth'.
4. Optional dwell.
5. Rapid out.

10.4.7 Positioning Cycle - CYCLE/MILL

Syntax: **CYCLE/MILL**,FEDTO,depth [,IPM ,feed[,rap]] [,RAPTO,r] \$
dia,ang IPR
MMPM
MMPR

[,DWELL[,dw11]]

This version of the *CYCLE* command will cause **PWorks** to generate a positioning sequence for the point following the *CYCLE/MILL* command, The cycle will automatically be cancelled after the first point.

All parameters on this command will perform the same function as described in the '[Automatic Mill Cycles](#)' section.

Cycle Simulation

1. Rapid to XY of point.
2. Rapid to 'r'.
3. Feed in to 'depth'.
4. Optional dwell.
5. Cancel cycle.

10.4.8 Reaming Cycle - CYCLE/REAM

Syntax: **CYCLE/REAM,FEDTO,depth** [,IPM ,feed[,rap]] [,RAPTO,r] \$
 dia,ang IPR
 MMPM
 MMPR

 [,DWEEL[,dwl1]] [,RTRCTO,ret]

This version of the *CYCLE* command will cause **PWorks** to generate a reaming sequence for each point programmed between *CYCLE/REAM* and *CYCLE/OFF*.

All parameters on this command will perform the same function as described in the '[Automatic Mill Cycles](#)' sections, except for the '*RTRCTO*' parameter.

'*RTRCTO,ret*' specifies the distance above the Z-level of the point to retract to after the cycle is finished. If '*RTRCTO*' is not specified, then the Z-level implied in the *RETRACT* command will be used.

Cycle Simulation

1. Rapid to XY of point.
2. Rapid to 'r'.
3. Feed in to 'depth'.
4. Optional dwell.
5. Feed out.

10.4.9 Fine Boring Cycle - CYCLE/SHIFT

Syntax: **CYCLE/SHIFT,FEDTO,depth** [,IPM ,feed[,rap]] [,RAPTO,r] \$
 dia,ang IPR
 MMPM
 MMPR

 [,DWELL[,dwl1]] [,OFFSET,off1[,off2]] [,RTRCTO,ret]

This version of the *CYCLE* command will cause **PWorks** to generate a fine boring sequence for each point programmed between *CYCLE/SHIFT* and *CYCLE/OFF*.

All parameters on this command will perform the same function as described in the '[Automatic Mill Cycles](#)' section, except for the '*RTRCTO*' parameter.

'*RTRCTO,ret*' specifies the distance above the Z-level of the point to retract to after the cycle is finished. If '*RTRCTO*' is not specified, then the Z-level implied in the *RETRACT* command will be used.

Cycle Simulation

1. Rapid to XY of point.
2. Rapid to 'r'.
3. Feed in to 'depth'.
4. Optional dwell.
5. Orientate spindle.
6. Shift tool.
7. Feed out.
8. Shift tool.
9. Start spindle.

10.4.10 Hole Tapping Cycle - CYCLE/TAP,REVERS

Syntax: **CYCLE/TAP** ,FEDTO,depth [,IPM ,feed[,rap]] \$
 REVERS dia,ang IPR
 MMPM
 MMPR
 TPI

 [,RAPTO,r] [,DWELL[,dwl1]] [,RTRCTO,ret]

This version of the *CYCLE* command will cause **PWorks** to generate a hole tapping sequence for each point programmed between *CYCLE/mode* and *CYCLE/OFF*.

Although both of the acceptable cycle mode parameters, 'REVERS' and 'TAP'. Will be treated the same by **PWorks**, it is recommended to use the following conventions.

TAP = Cut right handed thread.
REVERS = Cut left handed thread.

PWorks will determine the type of thread to cut depending on the spindle direction in effect at the start of the cycle.

All parameters on this command will perform the same functions as described in the 'Automatic Mill Cycles' section, except for the 'RTRCTO' parameter.

'RTRCTO,ret' specifies the distance above the Z-level of the point to retract to after the cycle is finished. If 'RTRCTO' is not specified, then the Z-level implied in the *RETRCT* command will be used.

Cycle Simulation

1. Rapid to XY of point.
2. Rapid to 'r'.
3. Feed in to 'depth'.
4. Reverse spindle.
5. Optional dwell.
6. Feed out.
7. Reverse spindle.

10.4.11 Deep Hole Drilling Cycle - CYCLE/THRU

Syntax: **CYCLE/THRU**,FEDTO,depth [,IPM ,feed[,rap]] [,RAPTO,r] \$
dia,ang IPR
MMPM
MMPR

[,DWELL[,dw11[,dw12]]] [,STEP,peck1[,peck2]] \$

[,RTRCTO,ret[,plng]]

This version of the *CYCLE* command will cause **PWorks** to generate a deep hole drilling sequence for each point programmed between *CYCLE/THRU* and *CYCLE/OFF*.

All parameters on this command will perform the same function as described in the '[Automatic Mill Cycles](#)' section, except for the '*RTRCTO*' parameter.

'*RTRCTO,ret*' specifies the distance above the Z-level of the point to retract to after each cut. If '*RTRCTO*' is not specified, then the tool will retract to the retract plane implied in the '*RAPTO,r*' parameter.

Cycle Simulation

1. Rapid to XY of point.
2. Rapid to 'r'.
3. Will a move of 'peck' stop past the final depth?
If 'YES', Go to Step 10.
4. Feed in 'peck'.
5. Optional 'dw11' dwell.
6. Retract tool.
7. Optional 'dw12' dwell.
8. Rapid in 'plng' above previous depth.
9. Go to Step 3.
10. Feed in to 'depth'.
11. Optional 'del1' dwell.
12. Rapid out.

10.5 Miscellaneous Mill Cycle Commands

The following sections describe commands which affect mill cycles. Part/fixture avoidance commands and tool retraction commands are discussed in these sections.

10.5.1 Cycle Retract Plane - RETRACT

Syntax: **RETRACT/ON**
OFF

This version of the *RETRACT* command controls the retract plane level for mill cycle sequences. The retract plane defines where the tool will be positioned after a cycle sequence has been completed.

'ON' specifies for the tool to fully retract to the level it was at prior to the cycle being activated. 'OFF' specifies for the tool to retract to the rapto plane as defined by the '*RAPTO,r*' cycle parameter.

Example:

RETRACT/ON

The tool will retract to the clearance plane, defined as the tool position prior to activating the cycle sequence, after performing a cycle sequence.

RETRACT/OFF

The tool will retract to the rapto plane, defined using the '*RAPTO,r*' cycle parameter in the *CYCLE/mode* command, after performing a cycle sequence.

10.5.2 Cycle Avoidance Plane - CLRSRF/AVOID

Syntax: **CLRSRF/AVOID, [i, j, k,] d**

The *CLRSRF/AVOID* command defines a cycle clearance plane that the tool will retract to when the *CYCLE/AVOID* command is encountered. The clearance plane must be defined in reference to the tool end point and not the machine axes position.

'AVOID' differentiates this command from the other *CLRSRF* commands. The *CLRSRF/AVOID* command does not have any effect on any of the other *CLRSRF* commands.

'i, j, k' is the cycle clearance plane vector definition. If this parameter is not specified, then the primary tool axis vector as defined in the *MODE/TOOL* command will be used.

'd' is the distance along the defined vector for the cycle clearance plane. This parameter must be defined.

Example:

```
CLRSRF/AVOID,8.5
```

Defines the clearance plane for the *CYCLE/AVOID* command at a distance of 8.5 along the primary tool axis vector (usually 0,0,1).

```
CLRSRF/AVOID,1,0,0,3.2
```

Defines the clearance plane for the *CYCLE/AVOID* command as being 3.2 in the positive X-axis.

10.5.3 Part/Fixture Avoidance - *CYCLE/AVOID*

Syntax: **CYCLE/AVOID**

The *CYCLE/AVOID* command will cancel the current cycle and retract the tool to the cycle clearance plane defined in the previously programmed *CLRSRF/AVOID* command. After retracting the tool, the cycle mode that was in effect prior to the *CYCLE/AVOID* command will be reinstated.

This command is only valid with mills and can be used with any cycle mode, in both automatic and **PWorks** generated cycles. You can use the *CYCLE/AVOID* command to retract the tool in order to avoid clamps during a cycle sequence.

A *CLRSRF/AVOID* command must have been previously programmed prior to programming a *CYCLE/AVOID* command, or an error message will be output and this command be ignored.

Example:

```
CYCLE/AVOID
```

1. Cancel the active cycle mode.
2. Retract the tool to the *CLRSRF/AVOID* plane.
3. Reactivate the previously cancelled cycle mode.

10.6 Single Pass Lathe Thread Cutting

The following sections describe the post-processor commands which control single pass thread cutting on lathes. Multiple pass thread cutting is also supported, using the *CYCLE/THREAD* command.

10.6.1 Enabling Thread Cutting - THREAD

Syntax: **THREAD/leadk[,leadl][,ZAXIS ,z][,XAXIS ,x][,INCR,k] \$**
ZCOORD XCOORD
[,AUTO]
DECR

The *THREAD/lead* command enables a thread cutting sequence at the machine. The thread cutting sequence will start at the current tool location and either end after the next move or when *THREAD/OFF* is programmed after modal thread cutting has been initiated (*'AUTO'*). This command is only valid with lathes.

'leadk' specifies the lead in the Z-axis for thread cutting. *'leadl'* specifies the lead in the X-axis for tapered threads.

'ZAXIS, z' specifies an incremental distance along the Z-axis which will be used for the end of the thread. *'ZCOORD, z'* specifies the absolute Z-axis position for the end of the thread. *'XAXIS, x'* specifies an incremental distance along the X-axis which will be used for the end of a tapered thread cut. *'XCOORD, x'* specifies the absolute X-axis position for the end of a tapered thread cut. If any of these positioning parameters are specified, then the *THREAD* command will be performed only one time, for the position contained on this command. If none of these parameters are specified, then the next programmed location (*GOTO/z, x*) will be used for the end of the thread cutting sequence.

'INCR' enables increasing variable lead thread cutting. *'DECR'* enables decreasing variable lead thread cutting. *'k'* specifies the amount of increment or decrement for the programmed lead per spindle revolution. If neither *'INCR'* nor *'DEC'* is specified, then an equal lead thread cutting sequence will be initiated.

'AUTO' enables a modal thread cutting sequence, thread cutting will remain in effect until a *THREAD/OFF* command is programmed. If *'AUTO'* is not specified, then thread cutting will be automatically turned off after a single pass.

Example:

```
THREAD/.08
```

Enables constant lead thread cutting, with a lead of .08. Thread cutting will be automatically cancelled after the next move.

```
THREAD/.08,.01,ZAXIS,-1.5,XAXIS,.1
```

Outputs a tapered constant lead thread cutting sequence, with a lead of .08 in the Z-axis and .01 X-axis. The thread will be cut from the current tool location to a distance of 1.5 in the negative Z-axis direction and .1 in the positive X-axis direction.

```
THREAD/.12,INCR,.01,AUTO
```

Enables an increasing lead thread cutting sequence, with an initial lead of .12 and an increment of .01 for each spindle revolution. The thread cutting sequence will remain in effect for all moves until a *THREAD/OFF* command is programmed.

10.6.2 Disabling Thread Cutting - **THREAD/OFF**

Syntax: **THREAD/OFF**

The *THREAD/OFF* command cancels a modal thread cutting sequence, after it was initiated by the *THREAD/lead,AUTO* command. If a thread cutting sequence is not currently in effect, then this command will be ignored.

Example:

```
THREAD/OFF
```

Cancels active modal thread cutting sequence.

10.7 Automatic Lathe Cycles - CYCLE/mode

```
Syntax: CYCLE/mode [,ZAXIS ,z] [,XAXIS ,x] [,FEDTO,depth]      $
          ZCOORD      XCOORD

          [,IPM ,feed[,rap]] [,TPI,leadk[,leadi]]              $
          IPR
          MMPM
          MMPR

          [,RAPTO,r] [,STEP,step1[,step2]]                      $

          [,OFFSET,zoff[,xoff]] [,RTRCTO,ret1[,ret2]]          $

          [,TOOL,t1[,t2]] [,REPEAT,rep] [,ON ]
                                OFF
```

This version of the *CYCLE/mode* command is only valid for lathes and outputs a canned cycle sequence to the Control Tape. The machine control will perform the actual cycle. Please note that the descriptions of the *CYCLE* parameters explain their normal usage and can actually be different depending on how the cycle parameters have been set up in the **MPost** utility.

'mode' specifies the type of canned cycle to output and can be one of the following.

DEEP	=	Performs a grooving cycle in the X-axis.
DRILL	=	Performs a drilling cycle in the Z-axis.
FACE	=	Performs a facing cycle along the X-axis.
ROUGH	=	Performs a turning cycle along the Z-axis.
THREAD	=	Performs a multiple pass thread cutting cycle.
THRU	=	Performs a deep hole drilling cycle in the Z-axis.
TURN	=	Performs a turning cycle along the Z-axis.

'ZAXIS, z' specifies an incremental distance along the Z-axis which will be used for the ending position of the cycle. 'ZCOORD, z' specifies the absolute Z-axis position for the ending position. 'XAXIS, x' specifies an incremental distance along the X-axis which will be used for the ending position of the cycle. 'XCOORD, x' specifies the absolute X-axis position for the ending position. If any of these positioning parameters is specified, then the canned cycle will be performed only one time, for the position contained on this command. If none of these parameters are specified, then the next programmed location (*GOTO/z, x*) will be used for the ending position of the cycle.

'FEDTO,depth' specifies an offset value along the plunging axis for a cycle making a tapered cut.

'feed' is the cutting feed rate to perform the cycle at. It can be specified in Feed per Minute ('IPM' or 'MMPM') or Feed per Spindle Revolution ('IPR' or 'MMPR'). 'rap' specifies the feed rate to

use when calculating the machining time for rapid moves during the cycles,. A value of 0 for '*rap*' will cause the rapid rate to be used.

'*TPI, leadk*' specifies the lead in the Z-axis for a thread cutting cycle. '*leadl*' specifies the lead in the X-axis for tapered threads. '*TPI*' is usually only used with the *CYCLE/THREAD* command.

'*RAPTO, r*' specifies an incremental distance, along the plunging axis, from the current location to rapid to prior to initiating the cycle sequence.

'*STEP*' will cause a multiple pass cycle to be performed. '*step1*' specifies the depth of the first cut. '*step2*' specifies the depth of the last cut. All intermediate cuts will be at a depth that is between '*step1*' and '*step2*'. If '*step2*' is not specified or is the same value as '*step1*', then all passes will make the same size cut into the part.

'*OFFSET*' specifies that a tapered cutting cycle will be performed. '*zoff*' defines an incremental distance from the depth of the cycle, as defined by the ending location, which should be applied to the Z-axis of the cycle at the starting location. '*xoff*' specifies this incremental distance along the X-axis.

'*RTRCTO*' specifies the retract locations for each cut of a cycle and is usually used with multiple pass cycles. '*ret1*' specifies the level above each cut to retract to when performing a multiple pass cycle. '*ret2*' specifies the final location to retract the tool to, as the end of the cycle.

'*TOOL, t1*' specifies the angle, in degrees, of the cutting tool. This parameter is usually only used with the *CYCLE/THREAD* command when chamfering is enables. '*t2*' is provided as an extra parameter for machine controls which require more codes to perform the cycle than are provided here.

'*REPEAT, rep*' specifies the number of times to perform the final (finishing) pass. The default is 1.

'*ON*' specifies that a chamfer should be cut at the end of the thread and is usually only used with the *CYCLE/THREAD* command. The angle of the cutting, as specified by '*TOOL, t1*', will be used to determine the angle of the chamfer. The depth of the chamfer is determined by the depth of cut. '*OFF*' disables chamfering.

10.8 Manual Lathe Cycles

The following sections define the **PWorks** generated (manual) lathe cycles supported. The command syntax, available parameters and their functions, and the motion simulation which will be performed for each cycle will be discussed.

10.8.1 Grooving/Drilling - CYCLE/DEEP,DRILL,THRU

Syntax: **CYCLE/DEEP** [,ZAXIS ,z] [,XAXIS ,x] [,IPM ,feed[,rap]] \$
 DRILL ZCOORD XCOORD IPR
 THRU MMPM
 MMPR

 [,STEP,step1[,step2]] [,RTRCTO,ret1]

This version of the *CYCLE* command will cause **PWorks** to generate either a lathe grooving or lathe drilling sequence. **PWorks** will determine which type of cycle to simulate based on the direction of the final location in comparison to the beginning location.

Although all of the acceptable cycle mode parameters, 'DEEP', 'DRILL', and 'THRU' will be treated the same by **PWorks**, it is recommended to use the following conventions.

DEEP	=	Use for grooving cycle in the X-axis.
DRILL	=	Use for drilling cycle in the Z-axis.
THRU	=	Use for deep hole (peck) drilling cycle in the Z-axis.

All parameters on this command will perform the same function as described in the '[Automatic Lathe Cycles](#)' section.

Cycle Simulation

1. Rapid to 'r'.
2. Will a move of 'peck' stop past the final depth?
If 'Yes', Go to Step 6.
3. Feed in 'peck'.
4. Retract tool at rapid rate.
5. Go to Step 2.
6. Feed in to final depth.
7. Retract tool at rapid rate.

10.8.2 Facing Cycle - CYCLE/FACE

```
Syntax: CYCLE/FACE[,ZAXIS ,z][,XAXIS ,x][,FEDTO,depth] $
          ZCOORD          XCOORD
          [,IPM ,feed[,rap]][,RAPTO,r] $
          IPR
          MMPM
          MMPR
          [,STEP,step1[,step2]][,REPEAT,rep]
```

This version of the *CYCLE* command will cause **PWorks** to generate a lathe facing sequence.

All parameters on this command will perform the same function as described in the '[Automatic Lathe Cycles](#)' section.

Cycle Simulation

1. Rapid to 'r'.
2. Will a move of 'peck' stop past the final depth?
If 'Yes', Go to Step 8.
3. Feed in 'peck'.
4. Feed over to 'x'.
5. Retract tool at rapid rate.
6. Rapid to original position.
7. Go to Step 1.
8. Feed in to 'z' + 'depth'.
9. Feed to 'x' & 'z' (final location).
10. Retract tool at rapid rate.
11. Rapid to original position.
12. Perform Steps 8-11 'rep' times.

10.8.3 Turning Cycle - CYCLE/ROUGH,TURN

Syntax: **CYCLE/ROUGH** [,ZAXIS ,z] [,XAXIS ,x] [,FEDTO,depth] \$
 TURN **ZCOORD** **XCOORD**

 [,IPM ,feed[,rap]] [,RAPTO,r] \$
 IPR
 MMPM
 MMPR

 [,STEP,step1[,step2]] [,REPEAT,rep]

This version of the *CYCLE* command will cause **PWorks** to generate a lathe turning sequence. You can use either the '*ROUGH*' or '*TURN*' parameter at your discretion.

All parameters on this command will perform the same function as described in the '[Automatic Lathe Cycles](#)' section.

Cycle Simulation

1. Rapid to 'r'.
2. Will a move of 'peck' stop past the final depth?
If 'Yes', Go to Step 8.
3. Feed in 'peck'.
4. Feed over to 'z'.
5. Retract tool at rapid rate.
6. Rapid to original position.
7. Got to Step 1.
8. Feed in to 'x' + 'depth'.
9. Feed to 'x' & 'z' (final location).
10. Retract tool at rapid rate.
11. Rapid to original position.
12. Perform Steps 8-11 'rep' times.

10.8.4 Thread Cutting Cycle - CYCLE/THREAD

Syntax: **CYCLE/THREAD**,TPI,leadk[,leadl],OFFSET,offl[,ZAXIS ,z] \$
ZCOORD

[,XAXIS ,x][,FEDTO,depth][,STEP,step1[,step2]]] \$
XCOORD

[,TOOL,t1][,REPEAT,rep][,ON]
OFF

This version of the *CYCLE* command will cause **PWorks** to generate a multiple pass thread cutting sequence.

All parameters on this command will perform the same function as described in the '[Automatic lathe Cycles](#)' section, except for the '*OFFSET*' parameter.

'*OFFSET, offl*' specifies the actual depth of the thread, from the top of the part to the bottom of the thread.

Cycle Simulation

1. Rapid tool along tool angle.
2. Will a move of 'peck' past the top of thread stop past final depth?
If 'Yes', Go to Step 8.
3. Move to top of thread + 'peck' at rapid rate.
4. Is chamfering turned on?
If 'Yes', Cut thread to beginning of chamfer.
Cut chamfer.
If 'No', Cut thread to 'z'.
5. Retract tool at rapid rate.
6. Rapid to original position.
7. Go to Step 1.
8. Move 'x' + 'depth' at rapid rate.
9. Is chamfering turned on?
If 'Yes', Cut thread to beginning of chamfer.
Cut chamfer.
If 'No', Cut thread to 'z'.
10. Retract tool at rapid rate.
11. Rapid to original position.
12. Perform Steps 8-12 'rep' times.

CHAPTER 11 Ultrasonic Cutter

11.1 Knife Blade Tracking Control - MODE/BLADE

Syntax: **MODE/BLADE,OFF**
ON[, BOTH]
FRONT
REAR

One of the capabilities included in the Ultrasonic Cutter support of **PostWorks** is the ability to control the direction of the cutting blade using a rotary spindle axis. When cutting with a knife, it is imperative that the edge of the knife be facing in the direction of the cut, but when using a disk blade or using the machine as a router, then the rotary axis which controls the knife direction is actually used for spindle rotation. In this case the rotary spindle axis should not be output.

The *MODE/BLADE* command controls the output of the rotary spindle axis. 'OFF' disables the output of the spindle axis and allows it to be used to spin the cutting tool (disk or router). 'ON' enables the output of this axis and should be in effect whenever a knife blade is being used.

Most Ultrasonic blades are V-shaped and will cut in either direction, but there are blades that have a flat edge on one side and an angled side on the other. When using this style of blade it is usually desirable to maintain the flat side of the blade against the part. The options: "BOTH", "FRONT", "REAR" are used for this purpose.

'FRONT' specifies the front of the blade should always be pointed in the forward direction. 'REAR' specifies that the back of the blade should always be pointed in the forward direction. 'BOTH' specifies that either blade edge can be pointed in the forward direction. If nothing is specified, it will default to the last specified condition. If this is the first time this command is specified, it will default to the machine configuration setting, i.e. [Menu item 16.2.5](#).

The front of the blade is defined as the blade edge that points in the direction of the default blade direction vector when the blade axis is at 0 degrees.

Refer to the [RETRACT/TOOL](#) command in the Machining Modes chapter for a description of the available retract features when using a knife blade.

Programming a move in *RAPID* mode will disable the blade tracking feature for that move only. It will be reinstated after the *RAPID* move.

Example:

```
MODE/BLADE,OFF
```

Disables the automatic direction tracking feature of the spindle axis. Normal type *SPINDL* commands can be used to control the speed of the spindle rotation.

MODE/BLADE,ON,REAR

Enables the automatic direction tracking feature of the blade axis. The “rear” side of the blade will always point to the motion forward direction.

11.2 Z-axis Variations Motion - HEAD/ZIGZAG

Syntax: **HEAD/ZIGZAG,ON**
OFF
h,d

When using a knife blade, the blade itself is usually quite a bit longer than the depth of cut into the part. Leaving the knife at the same depth throughout a series of long cuts can cause the cutting area of the blade to wear out very quickly.

The *HEAD* command enables a zigzag type cutting motion for the knife. This motion varies the contact area of the blade during a cutting sequence and prolongs the life of the blade.

'h' specifies the length that the knife will travel up and down in the Z-axis variation (zigzag) motion. The first move will be in the down direction at a distance of 'h'. 'd' specifies the linear distance for a full down and up sequence. The knife will plunge 'h' and retract 'h' in the distance specified by 'd'.

'ON' enables the Z-axis variation motion using the previously defined values for 'h' and 'd'. 'OFF' disables the Z-axis variation motion.

Example:

HEAD/ZIGZAG,3,1.0

Enables Z-axis variation motion. The knife will plunge down .3 units and retract .3 units for every 1.0 linear unit traveled.

11.3 Automatic Ultrasonic Blade Alignment - ALIGN/CUT

Syntax: **ALIGN/CUT,OFF**
OMIT
NEXT [, d1 , d2 , ang]
ON

The automatic blade alignment feature for Ultrasonic Cutters is used to align the blade within a predefined span when making long moves with directional changes. The programmer defines both the minimum delta movement for alignment spans and the actual span length that **PWorks** will generate. The alignment span is used to align the blade with the direction of the cut in a short distance, rather than using the entire move to align the blade. On multi-axis moves the move may be broken up into multiple spans aligning the blade, because the blade can change directions multiple times depending on the rotary axis motion.

'*CUT*' is required or proper syntax. '*OMIT*' will turn off the automatic alignment mode for the next cut only. The automatic alignment mode will be reinstated, if it was active prior to the *ALIGN/CUT,OMIT* command, after the next move. '*OFF*' disables the automatic alignment mode until it is reinstated using the *ALIGN/CUT,ON* command.

'*NEXT*' enables the automatic alignment mode for the next cut only. It will be turned off again immediately thereafter. '*ON*' turns on the automatic alignment mode until it is turned off using the *ALIGN/CUT,OFF* command.

'*d1*' specifies the span length of the alignment move. '*d2*' defines the minimum delta movement for alignment spans. '*ang*' defines the minimum angular change of direction for alignment spans. Moves with a change of direction of less than '*ang*' will not have an intermediate alignment span output. The default values for the three parameters and the default automatic blade alignment mode can be set with the **MPost utility**.

Example:

ALIGN/CUT,ON,.1,.2,3

Enables the automatic blade alignment mode. Programmed moves which are at least .2 in length and have a directional change of at least 3 degrees will have an intermediate point at a distance of .1 output. This intermediate point will contain the blade direction for the entire move.

11.4 Ultrasonic Blade Positioning - ALIGN/MOVE

Syntax: **ALIGN/MOVE,OFF**

OMIT

NEXT[,dis,feed1,feed2]

ON

The *ALIGN/MOVE* command controls the blade alignment positioning feature for Ultrasonic Cutters. Blade positioning mode is used to align the blade during *RAPID* or a programmed feed rate move, so that it is in the correct position for the next cutting move.

'*MOVE*' is required for proper syntax. '*OMIT*' will turn off the blade positioning mode for the next move only. The blade positioning mode will be reinstated, if it was active prior to the *ALIGN/MOVE,OMIT* command, for the next *RAPID* move. '*OFF*' disables the blade positioning mode until it is reinstated using the *ALIGN/MOVE,ON* command. '*ON*' turns on the blade positioning mode until it is turned off using the *ALIGN/MOVE,OFF* command. This will only work with *RAPID* move.

'*NEXT*' enables the blade positioning mode for the next cut only. It will be turned off immediately thereafter. '*NEXT*' is usually used for positioning moves which use a programmed feed rate rather than *RAPID*.

'*dis*' specifies the distance above the programmed location at which to align the blade during a positioning move. The blade will be positioned at this location aligned to the angle required for the next cut. '*feed1*' specifies the feed rate to use when positioning the blade above the next cutting location. '*feed1*' will be discarded if the move is specified as a *RAPID* move. '*feed2*' specifies the feed rate to use when plunging the blade from the alignment location to the programmed cutting position. A value of 0 can be specified for either of the '*feed*' parameters to specify a *RAPID* move. The default values for the three parameters and the default automatic blade positioning mode can be set with the ***MPost*** utility.

Example:

ALIGN/MOVE,ON,.1,0.,5.

Enables the blade positioning mode. The blade will be orientated for the next cutting motion on all *RAPID* moves. The blade will move in rapid traverse mode to the alignment position of .1 above the programmed location and will plunge to the programmed location at a feed rate of 5.

11.5 Spline Interpolation - SMOOTH

Syntax: **SMOOTH/ON**
OFF

The *SMOOTH* command controls the output of spline interpolation codes which can be set with the *MPost utility*. 'ON' enables spline interpolation at the machine control. 'OFF' turns off the spline interpolation mode. The locations programmed between *SMOOTH/ON* and *SMOOTH/OFF* will be interpolated at the machine control, creating a smoothly curve path, instead of linear interpolated points output.

11.6 Curve Blending - SMOOTH/ATANGL

Syntax: **SMOOTH/ATANGL,OFF**
ang[,dis]

Curve blending on the control will interpolate a smooth curve through the locations between the *SMOOTH/ATANGL* and *SMOOTH/ATANGL,OFF* commands. Curve blending is more sophisticated than spline interpolation described above, in that a directional move tangent to the beginning of the curve will be output by *PWorks* in order to set the initial direction, and a maximal angle value can be programmed, which is used by the machine control to validate interpolation validity.

During curve blending mode, the control will interpolate a curve through a series of locations which in essence generates a much smoother path than pure linear interpolation will achieve. Large direction changes during curve blending mode will generate greater deviations from the programmed tool path, sometimes creating undesirable results. 'ang' specifies the maximum angular change that can be programmed during curve blending. If a move changes direction by more than this angle while in curve blending mode, then curve blending mode will be disabled for this set of locations.

Curve blending mode requires a short span in the direction of the curve be output prior to the initiating codes for curve blending. 'dis' defines the span distance of this move. *PWorks* will look ahead to the first few moves of the intended curve and output a span at this distance that is tangent to the calculate curve, prior to outputting the curve blending enable codes.

'OFF' marks the end of the curve blending locations.

The curve blending setting, default values for the limit angle and span distance can be set with the *MPost utility*.

Example:

SMOOTH/ATANGL,6,.1

Begins curve blending mode. All locations after this command and the *SMOOTH/ATANGL,OFF* command will generate a smoothly contoured path. The initial move tangent to the curve will be .1. If a direction change of more than 6 degrees is encountered, then the curve blending function will be cancelled and linear interpolation will be reinstated.

11.7 Vacuum Pods

Vacuum Pod fixture is available on most Ultrasonic Cutters and offers an alternative to standard type fixture. The following sections describe the commands used to control the Vacuum Pods.

Some of the *POD* commands address all pods, while others address only a single pod. The single pod commands address the pod using a row and column number 'row, col'. Rows are usually counted along the X-axis and columns along the Y-axis.

The required output codes for the *POD* commands can be set with the ***MPost*** [utility](#).

11.7.1 Vacuum Control - POD/AIR

Syntax: **POD/AIR,ON**
OFF
LOW
HIGH
NOMORE

This command is used to control the vacuum pump and the vacuum pressure to the pods. '*ON*' turns on the vacuum pump and '*OFF*' turns off the vacuum pump.

'*LOW*' turns on a low pressure vacuum to all of the pods. '*HIGH*' turns on a high pressure vacuum. '*NOMORE*' turns the vacuum off.

11.7.2 Disable Operator Vacuum Control - POD/LOCK

Syntax: **POD/LOCK,ON**

The *POD/LOCK,ON* command disables the operator from using the foot switch to control the vacuum to the pods.

11.7.3 Move All Pods Down - POD/DOWN

Syntax: **POD/DOWN**

The *POD/DOWN* command will cause all pods to unclamp, move down to their rest stop, and clamp. The vacuum to the pods will also be turned off.

11.7.4 Addressed POD Vacuum Off - POD/row,col,AIR

Syntax: **POD/row,col,AIR,OFF**

This command causes the vacuum to the specified pod to be turned off. '*row,col*' specifies which pod this command affects.

11.7.5 Clamp Addressed Pod - POD/row,col,CLAMP

Syntax: **POD/row,col,CLAMP,ON**
OFF

This command is used to clamp or unclamp a single pod. '*row,col*' specifies which pod this command affects. '*ON*' clamps the addressed pod. '*OFF*' unclamps the pod.

11.7.6 Addressed Pod Up - POD/row,col,UP

Syntax: **POD/row,col,UP[,CLAMP]**

This command raises the addressed pod to a tool stop position. '*row,col*' specifies which pod this command affects. If '*CLAMP*' is specified, then the pod will first be unclamped, raised to the tool stop position, and then clamped.

APPENDIX A Post-processor Command Summary

```
ALIGN/AXIS, OFF
    AUTO [ ,tol] [[ ,NEXT ] [ ,TRANS] [ ,DOWN]
    RAPID          LIMIT    NOW      UP

ALIGN/CUT, OFF
    OMIT
    NEXT [ ,d1, d2, ang]
    ON

ALIGN/MOVE, OFF
    OMIT
    NEXT [ ,dis, feed1, feed2]
    ON

ARCSLP/LINCIR, ON
    OFF

ARCSLP/SPLINE, ON
    OFF

AUXFUN/n, [ ,NOW ]
    NEXT

BREAK

BREAK/OFF

BREAK/ON    [ ,n] [ ,RAPTO, r] [ ,FEDTO, f]
    AUTO
    TIMES
    DELTA

CHECK/OUT, m [[ [ , ] XAXIS [ ,n] [ ,min, max] ] [[ [ , ] YAXIS [ ,n] [ ,min, max] ]
    $
    [[ [ , ] ZAXIS [ ,n] [ ,min, max] ] [[ [ , ] AXIS [ ,n] [ ,min, max] ] ] [ ... ]

CHECK/ [ [ XAXIS [ ,n] [ ,min, max] ] [[ [ , ] YAXIS [ ,n] [ ,min, max] ]
    $
    [[ [ , ] ZAXIS [ ,n] [ ,min, max] ] [[ [ , ] AXIS [ ,n] [ ,min, max] ] ] [ ... ]

CHECK/LENGTH, ON
    OFF
```

```

CLAMP/AXIS,n1[... ]n4,ON
                        OFF

CLAMP/AUTO,n

CLRSRF/AVOID,[i,j,k,]d

CLRSRF/[NORMAL,][i,j,k,]d

CLRSRF/STOP
    NOMORE
    sym      ,dir
    z
    a,b,c,d
    START[,TRFORM,ON ][,IPM ,fd1][,RTRCTO,p11[,IPM ,fd2]          $
                        OFF      MMPM                        ds1  MMPM
                        OFF
                        [,RAPTO[,p12][,IPM ,fd3]][,PLUNGE,p13[,IPM ,fd4]]
                        ds2      MMPM                        ds3  MMPM

CLRSRF/TOOL,d

COOLNT/ON
    OFF
    FLOOD
    MIST
    AIR

COUPLE/AUTO,ON
        OFF

COUPLE/OFF
    n,ATANGL,a

CUTCOM/ADJUST,ON
        OFF

CUTCOM/ADJUST,n[,PLUS ][,XAXIS[,pos]][,YAXIS[,pos]]          $
                MINUS
                [,ZAXIS[,pos]]

CUTCOM/ENDPT[,PS][,d][,TOOL,rad,cr][,MAXIPM,f][,ROTREF,n]    $
                DS
                [,MANUAL[,XAXIS,i][,YAXIS,j][,ZAXIS,k]]

```

```

CUTCOM/LEFT [,XYPLAN] [,d] [,NORMAL] [,MODIFY,n,v] $
      RIGHT  ZXPLAN      PERTO
            YZPLAN

      [,MANUAL[,XAXIS,x] [,YAXIS,y] [,ZAXIS,z]]

CUTCOM/ON
      OFF[,MODIFY,n,v]

CYCLE/AUTO,ON
      OFF

CYCLE/AVOID

CYCLE/ON
      OFF

CYCLE/CIRCUL,DEPTH,z,RADIUS,d,ON,[TOOL,tdia],STEP,zi $
      DIAMTR      IN

      [,IPM ,feed[,rap]] [,RAPTO,r] [,ATANGL,a] [,DOWN] [,CCLW] $
      IPR                                UP      CLW
      MPPM
      MMPR

      [RTRCTO,ret]

CYCLE/lathe_mode[,ZAXIS ,z] [,XAXIS ,x] [,FEDTO,depth] $
      ZCOORD      XCOORD

      [,IPM ,feed[,rap]] [,TPI,leadk[,leadi]] [,RAPTO,r] $
      IPR
      MPPM
      MMPR

      [,STEP,step1[,step2]] [,OFFSET,zoff[,xoff]] $

      [,RTRCTO,ret1[,ret2]] [,TOOL,t1[,t2]] $

      [,REPEAT,rep] [,ON ]
      OFF

CYLCE/mill_mode,...[,PARAMS,m1,n1[[...][,m30,n30]]]

```

```

CYCLE/mill_mode,FEDTO,depth [,IPM ,feed[,rap]][,RAPTO,r]      $
                        dia,ang  IPR
                        MPPM
                        MMPR
                        TPI

[,DWELL[,dwl1[,dwl2]]][,STEP,peck1[,peck2]]      $

[,OFFSET,off1[,off2]][,RTRCTO,ret[,plng]]      $

[,REPEAT,rep][,START[,seq[,inc[,frq]]]]

DELAY/n[,REV]

DISPLY/ON
      OFF
      AUTO
      ALIGN,n

END

FEDRAT/[IPM      ,][n]
      IPR
      MPPM
      MMPR
      INVERS

FEDRAT/LENGTH,n
      SCALE

FEDRAT/LOCK,ON
      OFF

FEDRAT/MAXIPM,n

FINI

FROM/x,y,z,i,j,k

GOHOME

GOHOME/CHECK[[,XAXIS[,n][,pos]][,YAXIS[,n][,pos]]      $
      AUTO
      FROM
      NEXT

      [,ZAXIS[,n][,pos]][,AXIS[,n][,pos]]][...]

```

```

HEAD/ZIGZAG,ON
            OFF
            h,d

INSERTtext

LEADER/n

LINTOL/n
        ON
        OFF

LINTOL/RAPID,n
            ON
            OFF

LINTOL/ADJUST,ON ,tol[,ATANGL,ang][,LENGTH,t1]
            OFF

LINTOL/AXIS,n

LINTOL/LINEAR,delta
        TOLER,tol
        OFF

LOADTL/SET,parm1,tool1[...] [parm5,tool5]

LOADTL/tn[[,LENGTH],t1][,OFFSET,h[,d]][,LARGE][,HIGH]          $
                                SMALL    LOW

        [,AUTO  ]
        MODIFY
        MANUAL

MACHIN/ PWORKS ,n1[...] [,n10]] [,OPTION,m1,v1[...] ]
        pstname

MAXDPM/n

MCHTOL/cirtol[,cirdlt]

MODE/AXIS[,n1[,n2]]

```

```

MODE/BLADE, OFF
        ON [, BOTH ]
            FRONT
            REAR

MODE/INCR, ON
        OFF

MODE/LATHE [, XYPLAN]
        ZXPLAN

MODE/MILL, AUTO [, XYPLAN]
        ZXPLAN

MODE/MILL, DIAMTR [, AXIS, n, XAXIS] [, tol]
        YAXIS
        ZAXIS

MODE/MILL, FACE [, tol]

MODE/ROTATE, OFF
        ON    [, n]
        AUTO

MODE/TOOL, XYPLAN
        ZXPLAN
        YZPLAN
        i, j, k

MODE/XAXIS, n
        YAXIS
        ZAXIS

MULTAX/ON
        OFF

OPSKIP/ON  [, n1 [...] n9]
        OFF

OPSTOP

ORIGIN/x, y [, z]

PARTNOtext

PLUNGE [/feed]

```



```

POD/AIR, ON
    OFF
    LOW
    HIGH
    NOMORE

POD/LOCK, ON

POD/DOWN

POD/row, col, AIR, OFF

POD/row, col, CLAMP, ON
    OFF

POD/row, col, UP[, CLAMP]

POSITN/[ [XAXIS[,n], pos] [[,] YAXIS[,n], pos] [[,] ZAXIS[,n], pos]      $
    [[,] AXIS[,n], pos]] [...]

POSTN/n

POSTN/[NORMAL,] [[[,] XAXIS[,n] [, pos]] [[,] YAXIS[,n] [, pos]]      $
    INCR
    [[,] ZAXIS[,n] [, pos]] [[,] AXIS[,n] [, pos]] [...]]

POSTN/OFF

POSTN/XYPLAN

PPRINTtext

PPRINT INCLUD/file

PPRINT MACHINE/PWORKS    ,n1[...] [,n10]] [,OPTION,m1,v1[...]
    pstname

PPRINT TLN,tool-description

PPTOL/tol
    [[XAXIS[,n], xtol] [[,] YAXIS[,n], ytol]      $
    [[,] ZAXIS[,n], ztol] [[,] AXIS[,n], atol]] [...]]

```

PPTOL/VECTOR,tol

PREFUN/n, [,NOW]
NEXT

RAPID[/] [MODIFY] [[,]XAXIS] [[,]RTRCTO] [[,]ON]
YAXIS OFF
ZAXIS
TOOL

RAPID/FEDTO,LENGTH[,n]
SCALE

RETRCT[/] [PAST] [[,]XAXIS] [[,]feed]
YAXIS
ZAXIS
TOOL

RETRCT/ON
OFF

RETRCT/TOOL[,ON][,LENGTH[,ON][,dis]][,ATANGL,ang] \$
OFF OFF

[,OFFSET,UP [,dis1]],FEEDZ,fret[,fpln[,fofs]]
LEFT
RIGHT

REWIND

ROTABL/ADJUST, TABLE, ON
ROTHED HEAD OFF

ROTABL/[AXIS,n,] ATANGL,ang[,NEUTRL] [,IPM ,f] [,SAME] \$
ROTHED ORIENT MPPM ROTREF
CLW
CCLW
INCR

[,NOW][,POSITN]
NEXT CUT

ROTABL/NEXT[,AXIS,n],CLW
ROTHED CCLW

ROTABL/ON [,AXIS,n]m1[[[,mk][,THRU],mn]][...]m9]
 ROTHED OFF

ROTABL/ORIGIN[,AXIS,n],x,y,z
 ROTHED

ROTABL/SHORT,COMBIN[,ON]
 ROTHED LARGE NEXT
 PLUS
 AXIS,n

ROTABL/SIZE[,AXIS,n],c
 ROTHED

ROTABL/TABLE,AAXIS[,XAXIS[,rot1[,XAXIS,rot2]][...]
 HEAD BAXIS YAXIS YAXIS
 CAXIS ZAXIS ZAXIS

SELCTL/tn [[,LENGTH],t1][,LARGE][,HIGH][,MODIFY]
 AUTO[,lastn] SMALL LOW NOW
 NEXT,tm

SEQNO/ON
 OFF
 n[,INCR,i[,m]]

SET/LINEAR
 LINCIR[,cirtol][,ALL]
 COMBIN

SLOWDN/OFF
 ON[,n][,NEXT]
 AUTO

SLOWDN/RAMP,OFF
 ON[,LIMIT,maxvel,axsvel][,STEP,min,max]

SMOOTH/ATANGL,OFF
 ang[,dis]

SMOOTH/ON
 OFF

SPINDL/LOCK,ON
 OFF

```

SPINDL/MAXRPM,n

SPINDL/ON
      OFF
      ORIENT

SPINDL/[RPM] [[,]n] [[,]CLW ] [[,]LOW  ] [[,]RADIUS,r]
          SFM          CCLW          MEDIUM          DIAMTR,d
                      BOTH          HIGH
                      AUTO

STOP

THREAD/leadk[,leadi] [,ZAXIS ,z] [,XAXIS ,x] [,INCR,k] [,AUTO]
                      ZCOORD          XCOORD          DECR

THREAD/OFF

TMARK[/] [n] [[,]NOW]

TMARK/AUTO

TOOLNO/ADJUST,ON
          OFF

TOOLNO/ADJUST,n[,d] [,PLUS ] [,XAXIS[,pos]] [,YAXIS[,pos]]          $
          MINUS

          [,ZAXIS[,pos]] [,XYPLAN] [,NOW ]
                      YZPLAN      NEXT
                      ZXPLAN

TOOLNO/OFFSETL,ON
          OFF

TOOLNO/TIMES,ONCE
          ALL

TPRINTtext

TRANS/[SCALE,]n          [,LAST]
          x,y,z

TRANS/i1,j1,k1,d1, i2,j2,k2,d2, i3,j3,k3,d3

```

```

TRANS/[SCALE] [[[,]XAXIS[,n][,val]][[,]YAXIS[,n][,val]]          $
    [[[,]ZAXIS[,n][,val]][[,]AXIS[,n][,val]]][...]

TRANS/[TOOL                ] [[[,] [VECTOR[,i,j,k]]][[,]AUTO]
    [,NOMORE]                NOW

TRANS/TOOL,ON
    OFF

TRANS/UP,n

TURRET/tn[[,RADIUS],r][,OFFSET,h[,d]][,ADJUST,z,x]              $
    [,FRONT][,CLW ][,AUTO  ]
    REAR      CCLW   MODIFY
                MANUAL

UNLOAD/TOOL[,tn]

```

APPENDIX B Special Considerations

B.1 Introduction

This Appendix describes features of **PWorks** which are present to handle certain machine peculiarities. In essence, **PWorks** is a general post-processor made to handle various types of machinery utilizing basically the same style of logic for any type of machine.

However, certain machine configurations require special handling. The [Ultrasonic Cutter](#) is one of these machines (this machine is described in its own chapter). This Appendix describes some of the other special machine configurations.

B.2 AC Style Rotary Head

This section describes special logic available for programming a dual rotary head machine which has the carrier rotary axis rotating about Z. This section will refer to this configuration as an AC (XZ) head, though it can easily be a BC (YZ) head.

First the problem areas of this machine will be discussed.

Users of an AC head will quickly find out that whenever the rider axis (A-axis) approaches zero degrees, but is not quite at zero degrees, the carrier axis (C-axis) will have a tendency to rotate, sometimes quite drastically. This is quite normal and the post-processor is creating the proper output. The following example illustrates this problem.

Example #1:

Input	Output
GOTO/0,0,0, 0,.017,.999	X... Y... Z... A1.0 C0.0
GOTO/0,0,0, .017,0,.999	X... Y... Z... A1.0 C90.0

In the above example, the tool axis vector moved a total of 1.4 degrees, but the C-axis had to rotate a total of 90 degrees to fulfill this tool axis (the A-axis did not have to move at all). The reason for this is that the tool axis moved from a greater Y-axis direction to a greater X-axis direction in a single move. For a head which has a C-axis and another single axis which rotates about either X or Y, to make this move requires that the C-axis rotation be such that the A-axis rotation is in the direction of the X and Y components of the tool axis vector. As you can see in order to fulfill the primary Y-component of the first tool axis vector, the A-axis must rotate about the X-axis and to fulfill the primary X-component of the second tool axis vector, the A-axis must be positioned to rotate about the Y-axis.

A second problem with an AC head, is that there is usually fixed limitation on how far the C-axis can rotate and it is common to exceed the C-axis limits, especially when machining a closed pocket with canted walls. This usually, but not always, causes the rotary axis to take the longest

route to satisfy the tool axis vector, meaning that the C-axis can rotate much more than 90 degrees.

These are the two most common problems with an AC head, so now the topic will move towards the ways to help overcome these by using special features within **PostWorks**.

The first thing that **PWorks** does after loading an *MDF* file is to determine the style of machine being programmed. If it is an AC head, then a special look ahead feature will be turned on. This look ahead feature will sample the clfile whenever the A-axis is at 0 degrees and will search for changes in the C-axis. If a change is found in the C-axis for the following move, then it will be moved to this position in the motion block which still has the A-axis at 0 degrees. Consider the following example.

Example #2:

Input	Output
-----	-----
GOTO/0,0,0, 0,0,1	N1 X... Y... Z... A0.0 C0.0
GOTO/1,0,0, 0,0,1	N2 X... Y... Z... C90.0
GOTO/0,1,0, .017,0,.999	N3 X... Y... Z... A1.0

In the above example **PWorks** finds that the C-axis will move to 90 degrees in block N3, so it positions it in block N2 while the A-axis is still at 0 degrees. This avoids making the long C-axis rotation at the same time that the A-axis is moving, which could very well undercut the part.

By default the C-axis will be positioned to the middle of its limits, The user can override this default setting by placing a *ROTHED/NEXT, (CLW/CCLW)* command just prior to the motion which created block N2. A setting of *CLW* will position the C-axis closest to its positive limit, while a setting *CCLW* will position the C-axis closest to its negative limit. Refer to Example #3 which is the same as Example #2, except it has an added *ROTHED/NEXT* statement. Assume that the C-axis limits are -270 to 270 degrees.

Example #3:

Input	Output
-----	-----
GOTO/0,0,0, 0,0,1	N1 X... Y... Z... A0.0 C0.0
ROTHED/NEXT,CCLW	
GOTO/1,0,0, 0,0,1	N2 X... Y... Z... C-270.0
GOTO/0,1,0, .017,0,.999	N3 X... Y... Z... A1.0

The *ROTHED/NEXT* command is useful when the user knows that the following motion blocks will cause the C-axis to move in generally one direction and the full travel limits are required. Another option which can be used instead of the *ROTHED/NEXT* command is the *ROTHED/ATANGL,ang,ORIENT,NEXT* command, which allows the user to specify an absolute

angular position for the C-axis. Example #4 is similar to Example #3, except that *ROTHED/NEXT* has been replaced with *ROTHED/ATANGL*.

Example #4:

Input	Output
-----	-----
GOTO/0,0,0, 0,0,1	N1 X... Y... Z... A0.0 C0.0
ROTHED/ATANGL,-270,ORIENT,NEXT	
GOTO/1,0,0, 0,0,1	N2 X... Y... Z... C-270.0
GOTO/0,1,0, .017,0,.999	N3 X... Y... Z... A1.0

When using the *ROTHED/ATANGL,ang,ORIENT,NEXT* command it is imperative that the user knows where the C-axis will be positioned on the subsequent block, otherwise this forced rotation will not have any benefit.

All of the above examples rely on the A-axis being at 0 degrees to initiate any look ahead positioning of the C-axis. This condition is mandatory since the C-axis is always governed by the tool axis if it is any direction other than 0,0,1. In cases where the A-axis is not at 0 degrees, **PWorks** has other features which will aid the user in overcoming the peculiarities of this machine.

When the C-axis limits are hit and the post has to unwind the C-axis to fulfill the tool axis vector it will normally take what is called the "longest route" in doing so. A built in feature is the ability to automatically retract the tool when taking the longest route, position the C-axis, and then plunge the tool back to its previous position. With an AC head it is recommended that the longest route logic be set to look at the C-axis. Refer to the *RETRCT/TOOL* command or the **MPost Reference Manual** for further information on Automatic Tool Retraction.

There is also a feature which controls the Maximum Axis Departure. The user has the option of allowing **PWorks** to break up a move whenever a certain axis distance in a single move exceeds a user specified value or an error message can be output when this condition is true. In either case, the C-axis maximum departure should be set to a low value. Having **PWorks** automatically break up the move is not always the most desirable option, as the results in many cases (depending on the C-axis movement) is that the part will still be undercut. The error message output in essence does not make any changes to the program, but rather notifies the user that there could possibly be a trouble area to look at.

This brings us to the final and most important solution for programming an AC head, the motion control defined by the user. This method will usually take at least two times through the post-processor with the user analyzing the output, looking for "longest route" and "maximum departure" error messages, to finalize a program for a machine with an AC head. After analysis, there could be some *ROTHED* commands inserted at key locations within the program to orient the C-axis to its required position.

This will not always solve the problem though, for example when the A-axis is not positioned at 0 degrees. At this time it is up to the user to modify the motion within the program so that the C-axis movement does not cause the tool to violate the part. The most common solution is to back the tool off of the part, position the C-axis, and then reposition the tool to the part. This is similar to Automatic Tool Retraction, except that the user has more control on how the tool is taken off of the part and repositioned. It also does not rely on the rotary axis taking the longest route.

Another option to be considered is moving the part on the machine, usually by using an angle plate, so that the A-axis does not approach 0 degrees. Remember, almost all problems associated with an AC head is the fact that the C-axis wants to float whenever the A-axis is near 0 degrees.

One last method for determining if the tool will violate the part is to use the simulation file output by **PWorks** as input to a visual verification routine, such as **NCL/IPV**. The [simulation file](#) contains all of the movement which the machine will take, including the linearization of the large C-axis moves, which may not be seen if only verifying the APT source file or a reverse post-processed Control Tape file.

To summarize, the following two problem areas can be found when programming for a machine with an AC head.

1. Major fluctuation of the C-axis when the A-axis approaches 0 degrees.
2. Exceeding the C-axis limits during continuous motion.

The features contained in **PWorks** to help alleviate the problems include the following.

1. Automatic positioning of the C-axis when the A-axis is at 0 degrees.
2. The [ROTHED/NEXT](#) and [ROTHED/ATANGL,...,ORIENT](#) commands.
3. [Automatic Tool Retraction](#).
4. Maximum Depart Control.
5. [Simulation file](#).

And lastly, but also very important, what the user can do to solve these problems.

1. Analyze the print file form **PWorks** to identify the problem areas.
2. Alter the motion, bringing the tool away from the part for repositioning of the C-axis if necessary.
3. Use of an angle plate on the machine if feasible.
4. Run visual verification software using the simulation file as input.

B.2.1 Enable/Disable The Automatic Look Ahead Feature - ALIGN/AXIS

Syntax: **ALIGN/AXIS,OFF**

AUTO [,tol] [,NEXT] [,TRANS] [,DOWN]
RAPID **LIMIT** **NOW** **UP**

The *ALIGN/AXIS,mode* command is used to enable/disable and control the look ahead positioning feature for AC-rotary axes style machines. The default condition is enable. Under certain conditions this look ahead position feature must be disabled, such as with a right angled head attachment. If the machine is of any other configuration than an AC-style head (table/head, dual tables, nutating head, etc.), then the look-ahead feature will only be enabled for rapid moves, even if *AUTO* is specified.

‘OFF’ disables the look ahead positioning feature. ‘AUTO’ enables the look ahead positioning feature. ‘RAPID’ specifies only performs the look ahead feature for a change in the C-axis when a rapid move is in effect

‘tol’ specifies a tolerance to use for the I and J components of the tool axis vector to determine when the A-axis should be at zero. The default is .0002 when *AUTO* is in effect and 0 when *OFF* is in effect. The *LINTOL/AXIS* command can also be used to protect against small changes in the tool axis.

‘NEXT’ causes **PostWorks** to look-ahead for the next C-axis movement while the A-axis is at zero. The C-axis will then be positioned to the closest angle that satisfies its next movement. ‘LIMIT’ causes **PostWorks** to continue to scan the clfile until the A-axis moves back to zero or an axis limit is reached. The C-axis will be optimally positioned nearest the limit that will allow it the greatest amount of movement before actually violating a limit.

‘TRANS’ requires that the previous move have the A-axis at zero along with the current move prior to scanning the clfile for the next C-axis movement. This method ensures that the C-axis will be positioned during other axis movement and will not dwell in-place during positioning. ‘NOW’ allows the C-axis to position itself while the A-axis is at zero, even in a single block without any other axis movement. This will happen anyway if linearization is active, since it is necessary in keeping the tool within tolerance of the move, but by enabling it during look-ahead you have the option of retracting the tool during this positioning.

‘DOWN’ causes the C-axis to be positioned in-place when it is positioned in a single block due to the ‘NOW’ setting. ‘UP’ retracts the tool, positions the C-axis, and plunges the tool back to its original position when the C-axis is positioned in a single block during the look-ahead sequence. The positioning method used is the same that is used when retracting the tool due to the longest route being taken. The same distances and feed rates will be used.

B.3 Look Ahead Feature For Machine With a C-axis Table

The look ahead feature for AC-style heads is also applied to machines that have a C-axis table rider with either a table carrier or head as the second rotary axis. When a C-axis table is defined, then the look ahead feature will only apply to rapid moves, since rotating this axis will cause the linear axis to change position in order to maintain the tool relationship with the part. Unlike machines with an AC-style head setup, machines with a C-axis table will not automatically enable the look ahead feature, the *ALIGN/AXIS,AUTO* or *ALIGN/AXIS,RAPID* command must be programmed.

B.4 Tool Axis Adjustment Near Apex Point Of AC-style Rotary Axes Machine - LINTOL/ADJUST

Syntax: **LINTOL/ADJUST,ON** ,tol [,ATANGL,ang] [,DELTA,max] \$
OFF
[,LENGTH,t1]

This allows the user to specify whether the tool axis vector should be adjusted when it is near the apex point of an AC-style rotary axes machine. Of the various “adjustments” that can be made to the tool axis when it is near the apex point, this is the recommended feature to use.

“ON” enables the adjustment of the tool axis vector by moving the ending position of the top of the tool back along the forward vector whenever the tool axis is near the apex point of an AC-style rotary axis. This feature is used to keep the rotary axes from rotating excessively when the apex point is neared. The tool tip will still move to the programmed position and the top of the tool will be within a user specified tolerance of the programmed position. “OFF” disables this adjustment.

“tol” specifies the tolerance value to used to position the top of the tool during tool axis adjustments. The top of the tool will always remain within this tolerance of the programmed position.

“ATANGL,ang” specifies the angular distance that the tool axis vector must be within in reference to the apex point of an AC-style rotary axis before **PostWorks** will consider adjusting the tool axis. This is typically a small value (less than 5 degrees) since the further away the tool axis is from the apex point, the less noticeable the fluctuations in the rotary axes will be and the less benefit this feature will provide.

“DELTA,max” defines the minimum angular delta movement that the rotary axes must move prior to considering adjusting the tool axis vector. This limit is used to keep from adjusting the tool axis vector when the rotary axes do not fluctuate greatly. This is typically a higher value (10 degrees or more).

“*LENGTH,tl*” defines the value to use for the tool length when an actual tool length has not been programmed. The tool length is used to determine where the top of the tool is when performing tool axis adjustments.

The **MPost utility** can be used to set the default values of this command. However, versions of the MDF file prior to 04.19.06 will have the tool axis adjustment feature disabled automatically, while newer versions will enable this feature by default.

INDEX

3-D Cutter Compensation	7-10
Absolute	7-1
AC Style Rotary Head	B-1
Acceleration Block	7-12
ALIGN/AXIS	B-5
ALIGN/CUT	11-3
ALIGN/MOVE	11-4
Alignment Block	2-1, 10-4
ARCSLP	9-1
ARCSLP/SPLINE	9-5
Automatic Cycle	10-1
Automatic Lathe Cycle	10-22
Automatic Mill Cycle	10-3
AUXFUN	2-4
AVOID	10-18
Blade Alignment	11-3
Blade Positioning	11-4
BORE	10-3, 10-7
BORE7	10-3, 10-8
BORE8	10-3, 10-9
BORE9	10-3, 10-10
BREAK	1-2, 2-6, 7-2
C-axis Table	B-6
Chamfer	10-4, 10-23
Change Rotary Axes Configuration	7-4
CHECK	2-3, 5-6, 6-1
Checksumming	2-3
CIRCUL	9-4
Circular Interpolation	9-1
CL Point	7-1
CLAMP	6-5
Clearance Plane	10-18
CLRSRF	4-2, 5-3, 8-5, 10-18
COMBIN	6-4, 9-2
Comments	2-5
Control Tape	B-4
COOLNT	3-6
COUPLE	9-3
Curve Blending	11-5
CUT	5-2

CUTCOM	3-9, 3-10, 7-8, 7-9
CUTTER	7-11, 9-4
Cutter Compensation	3-2, 3-5, 3-7, 7-8, 7-9, 9-1
CYCLE	7-2, 9-1, 9-4, 10-1
Cycles For Siemens 840D	10-5
Cylindrical Interpolation	7-16
Date	2-5
DEEP	10-3, 10-11, 10-22, 10-24
DELAY	4-7
DISPLY	2-5
DRILL	10-3, 10-12, 10-22, 10-24
END	1-5
End-of -Program	1-5
End-of-sequence	2-7
ENDPT	7-10
External File	1-3
FACE	10-3, 10-12, 10-22, 10-25
FEDRAT	4-1, 4-4, 4-5
Feed Rate	4-1, 10-4
Feed Rate Control Point	4-1, 5-2
FINI	1-5
Fixture Offset	3-9, 3-10
FROM	5-1
G-code	2-4
GOHOME	5-6
Grooving	10-24
HEAD/ZIGZAG	11-2
Helical Interpolation	9-3, 9-4
Home Position	5-6
INCLUD	1-3
Incremental	7-1
INSERT	2-3
Interference Zone	6-1
Intermediate Moves	8-7
LATHE	7-15
Lathe	10-2
LEADER	1-4
LENGTH	2-3, 4-1
Limit Planes	8-5
LINCIR	9-1, 9-2
LINEAR	9-2
Linear Axes	7-3
Linearization	7-6, 7-17
LINTOL	7-6, 7-8, 8-7, B-6
LOADTL	2-7, 3-1, 3-2, 4-2, 5-4, 7-2
LOCK	4-5, 4-7

Look Ahead	B-5
MACHIN	1-1
Machine Descriptor File	1-1
Machine Dwells	4-7
Machining Plane	3-8, 7-9
Manual Lathe Cycle	10-23
Manual Mill Cycle	10-7
Matrix	8-3
MAXDPM	4-4
MAXIPM	4-4
MAXRPM	4-6
MCHTOL	9-1
M-code	2-4
MILL	7-15, 10-3, 10-13
Mill	3-2, 10-1
Mill/Turn	7-15
MODE	3-8, 7-1, 7-2, 7-3, 7-15
MODE/BLADE	11-1
MULTAX	7-1
Multi Heads/Tables	7-4
NCL/IPV	B-4
OFSETL	3-9
Operator Message	2-5
OPSKIP	2-2
OPSTOP	3-6
ORIENT	4-5
ORIGIN	8-2, 8-8
OUT	6-1
Override	1-2, 4-5, 4-7
PARTNO	1-1
PIVOTZ	8-8
PLUNGE	2-7, 5-5
Polar Interpolation	7-5, 7-17
Positioning	5-1
POSITN	5-1, 5-2
POSTN	1-4, 8-1, 8-2
PPRINT	2-5
PPTOL	6-2
PREFUN	2-4
Preset Axis Registers	8-1, 8-2
Print Descriptor File	3-6
Print File	B-4
Program Number	1-2
Punch Header File	3-6
RAMP	7-12
RAPID	4-2, 10-2, 11-1

REAM	10-3, 10-14
RETRCT	2-7, 4-2, 5-3, 5-4, 7-14, 10-2, 10-18, B-3
REVERS	10-3, 10-16
REWIND	1-5
Rewind Stop	1-4
ROTABL	5-2, 6-3, 6-4, 7-4, 8-8
Rotary Axes	4-4, 5-2, 6-3, 6-5, 7-2, 7-3
Rotary Axis	7-6, 8-8, 8-9
ROTATE	7-5
ROTHED	5-2, 8-8, B-2, B-4
ROUGH	10-22, 10-26
SCALE	4-1
SELCTL	3-1
SEQNO	2-1, 10-4
Sequence Number	2-1
SET	9-2
SHIFT	10-3, 10-15
SHORT	6-4
Shortest Route	6-4
Simulation file	B-4
SLOWDN	7-11, 7-12
Slowdown Block	7-11
SMOOTH	11-5
SMOOTH/ATANGL	11-5
SPINDL	4-5, 4-6
Spindle	3-1, 4-1, 4-5, 6-1, 7-2, 10-2
Spline Interpolation	9-5, 11-5
STEP	9-4
STOP	2-7, 3-6
Surface Normal Vector	7-11
TAP	10-3, 10-16
TAPBRK	2-8
Tape Break	2-6
Tapping	10-16
THREAD	10-20, 10-21, 10-22, 10-27
Thread Cutting	10-20, 10-21
THRU	10-3, 10-17, 10-22, 10-24
Time	2-5
TLNAME	3-6
TMARK	1-4, 2-1, 2-7
TOLER	8-7
Tolerance	6-2, 7-8, 9-1
Tool Axis Adjustment	B-6
Tool Axis Vector	4-2, 5-2, 6-3, 7-1, 7-6, 7-7
tool change	3-2
Tool Description	3-6

Tool Length Offset	3-2, 3-7
TOOLNO	3-7
TPRINT	2-5
TRANS	7-13, 8-3, 8-4
Transformation Block	7-13
Travel Limits	6-1, 6-3
TURN	10-22, 10-26
TURRET	3-2, 3-4
Ultrasonic Cutter	7-14, 11-1
UNLOAD	3-4
Vacuum Pods	11-6
Z-axis Variations Motion	11-2