



# ***POST PROCESSOR WRITER'S REFERENCE***

Copyright © 1984-2010 Geometric Technologies, Inc. All Rights Reserved.



**Geometric**

Geometric Technologies, Inc. is a subsidiary of Geometric Ltd.

January 2010

---

# Table of Contents

<b>CHAPTER 1 SYSTEM COMMANDS</b>	<b>1-1</b>
ABS.....	1-2
SIN.....	1-3
COS.....	1-4
TAN.....	1-5
ACOS.....	1-6
ASIN.....	1-7
ATAN2.....	1-8
ATAN.....	1-9
SQRT.....	1-10
LOOKUP.....	1-11
RESETEOL.....	1-12
STRCAT.....	1-13
CALL.....	1-14
REPLACE.....	1-15
SETON.....	1-16
SETOFF.....	1-17
SYS_CANNED.....	1-18
GOTO.....	1-19
RETURN.....	1-20
OFFSET_INC.....	1-21
OFFSET_XYZ.....	1-22
ADD_MACRO_START.....	1-23
ADD_MACRO_END.....	1-24
SPACES.....	1-25
GETTOOLS.....	1-26
GET_OPER_COMMENTS.....	1-27
OPEN_NEXT.....	1-28
ROUNDOFF.....	1-30
:QUALIFIED_TOOLING=.....	1-31
INCON.....	1-32
INCOFF.....	1-33
APPEND.....	1-34
TRANSFORM.....	1-35
RUN_PROCESS.....	1-37
QUERY_SYSTEM().....	1-38
KILLSYSFILE().....	1-39
 <b>CHAPTER 2 SYSTEM TEXT LINE COMMANDS</b>	 <b>2-1</b>
L.....	2-2
l.....	2-3
N.....	2-4
T.....	2-5
t.....	2-6
@.....	2-7
-.....	2-8
+.....	2-9
".....	2-10

## Table of Contents

#.....	2-11
.....	2-12
% .....	2-13
<b>CHAPTER 3 SYSTEM CCL COMMANDS</b>	<b>3-1</b>
GET_DATA() .....	3-2
GET_POINT() .....	3-3
SET_COLOR() .....	3-4
SET_TEXT_COLOR() .....	3-5
SET_LAYER().....	3-6
ADD_PUNCH_PATTERN() .....	3-7
ADD_PUNCH_PATH() .....	3-9
ADD_REPOSITION() .....	3-10
ADD_CAD().....	3-11
MAKE_FILLET() .....	3-13
ADD_PUNCH_TOOL() .....	3-14
SELECT_TOOL().....	3-16
START_COMPLEX() .....	3-17
END_COMPLEX().....	3-18
START_GROUP().....	3-19
END_GROUP() .....	3-20
<b>CHAPTER 4 ATTRIBUTE COMMANDS</b>	<b>4-1</b>
:ATTRNAME= .....	4-2
:ATTREND .....	4-3
:ATTRTYPE= .....	4-4
:ATTRVTYPE= .....	4-6
:ATTRID= .....	4-7
:ATTREMARK .....	4-8
:ATTRLISTDEF .....	4-9
:ATTRLIST= .....	4-10
:ATTRSELSTR= .....	4-11
:ATTRSEL= .....	4-12
:ATTRUSED= .....	4-13
:ATTRDEFAULT= .....	4-14
:ATTRINLEN= .....	4-15
:ATTRSHORT= .....	4-16
:ATTRLONG= .....	4-17
:ATTRHIGH= .....	4-18
:ATTRLOW= .....	4-19
:ATTRTEXT= .....	4-20
:ATTRTITLE= .....	4-21
:ATTRSPACES= .....	4-22
:CODETYPE= .....	4-23
:CODE= .....	4-25
:WORD_ADDRESS_BEF= .....	4-26
:WORD_ADDRESS_AFT= .....	4-27
:LEFT_PLACES= .....	4-28
:RIGHT_PLACES= .....	4-29
:CANNOT_BE_DECIMAL .....	4-30
:CANNOT_BE_LEADING .....	4-31
:CANNOT_BE_TRAILING .....	4-32

:CANNOT_BE_SIGNED .....	4-33
:MUST_BE_DECIMAL .....	4-34
:MUST_BE_LEADING.....	4-35
:MUST_BE_TRAILING.....	4-36
:MUST_BE_SIGNED.....	4-37
:MUST_BE_LEADING_SPACES .....	4-38
:MUST_BE_TRAILING_SPACES .....	4-39
:MODAL .....	4-40
:UNITFLAG= .....	4-41
:METRIC_UNITS.....	4-42
:ATTRFUNC= .....	4-43
:ATTRCFUNC=.....	4-44
:SELECT=.....	4-45
:VAR= .....	4-46
:VARB= .....	4-47
:ATTRVCNT= .....	4-48
:COLUMN= .....	4-49
:LEFT_JUST=.....	4-50
:RIGHT_JUST= .....	4-51
:ATTRLNG=.....	4-52

**CHAPTER 5 MISCELLANEOUS COMMANDS****5-1**

:ATTRID .....	5-2
:IDHIGH .....	5-3
:ATTRMACHINE .....	5-4
:SECTION.....	5-5
:OPERID .....	5-6
:OPERSUB.....	5-7
:OPERLIST .....	5-8
:OPEREND .....	5-9
:DEFINE .....	5-10
:LIBRARY .....	5-11
:INCLUDE .....	5-12
FLAGGED(variable, bit value).....	5-13

**CHAPTER 6 SYSTEM TOOL COMMANDS****6-1**

:STATION_NUM .....	6-2
:AUTOINDEX .....	6-3
:KEYSIZE .....	6-4
:KEYED .....	6-5
:LARGEDIAM.....	6-6
:XWDEAD.....	6-7
:YHDEAD.....	6-8
:XLORANGE.....	6-9
:YLORANGE.....	6-10
:XHRANGE.....	6-11
:YHIRANGE.....	6-12

<b>CHAPTER 7 SYSTEM HEADER COMMANDS</b>	<b>7-1</b>
:BCL_FORMAT .....	7-2
:SYSTEM.....	7-3
:LEADING.....	7-4
:TRAILING.....	7-5
:DECIMAL .....	7-6
:QUAD.....	7-7
:SPACE .....	7-8
:ARCS.....	7-9
:METRIC_SHIFT .....	7-10
:G_LEFT_PLACES .....	7-11
:G_RIGHT_PLACES.....	7-12
:G_INT_LEFT_PLACES.....	7-13
:INT_LEADING .....	7-14
:INT_TRAILING .....	7-15
:PQCOMP .....	7-16
:QUALIFIED_TOOLING.....	7-17
:SINGLE_MACROS .....	7-18
:MIRROR_MACROS .....	7-19
:MACROS_REDEFINE .....	7-20
:MULT_MACROS .....	7-21
:LAYOUT_MACROS .....	7-22
:MACROS_CALL .....	7-23
:MACROS_MULT .....	7-24
:MACROS_LAYOUT .....	7-25
:MACROS_MAIN .....	7-26
:MACROS_OUT .....	7-27
:MACROS_TAPE.....	7-28
:MACROS_XYZ .....	7-29
:MACRO_ROTATE .....	7-30
:MACRO_ROTATE_X .....	7-31
:MACRO_ROTATE_Y .....	7-32
:MACRO_ROTATE_Z.....	7-33
:EDM4AXIS .....	7-34
:TAPER.....	7-35
:ARC_TO_ARC.....	7-36
:SHARP_CORNER .....	7-37
:EQUAL_CORNER.....	7-38
:INDEPENDENT_CORNER.....	7-39
:CONIC_CORNER.....	7-40
:CHAMFER_CORNER .....	7-41
:TAPER_DURING .....	7-42
:LOOK_AHEAD .....	7-43
:TAPER_FILLET.....	7-44
:LIVE_Y_AXIS .....	7-45
:OD_MILL .....	7-46
:OD_DRILL .....	7-47
:OD_ARC.....	7-48
:FACE_MILL.....	7-49
:FACE_DRILL.....	7-50
:FACE_ARC .....	7-51

:LATHE .....	7-52
:5AXIS_MILLING .....	7-53
:4AXIS_X_MILLING .....	7-54
:4AXIS_Y_MILLING .....	7-55
:HELICAL .....	7-56
:MAXIMUM_LINE .....	7-57
:USE_SPECIAL_TOOL_TYPE .....	7-58
:SLOW_INDEXER .....	7-59
<b>CHAPTER 8 SYSTEM VARIABLES</b>	<b>8-1</b>
System Variables .....	8-2
<b>CHAPTER 9 SYSTEM CONSTANTS</b>	<b>9-1</b>
System Symbolic Constants .....	9-2
Mill Operation Symbolic Constants .....	9-5
Drill Operation Symbolic Constants .....	9-6
Lathe Operation Symbolic Constants .....	9-7
Additional System Constants .....	9-8
<b>CHAPTER 10 PROGRAMMING EXAMPLES</b>	<b>10-1</b>
Example 1 .....	10-2
Example 2 .....	10-6
Example 3 .....	10-12
<b>CHAPTER 11 ADD'L. SYSTEM HEADER COMMANDS</b>	<b>11-1</b>
WORLD_POSITIONING .....	11-2
RIGHT_ANGLE_SHEAR_ATTACHED .....	11-3
LASER_PLASMA_CUT_DATA .....	11-4
VECTOR_COMP .....	11-5
NO_SET_FILE .....	11-6
TRAPDOOR .....	11-7
MOVE_CLAMP .....	11-8
SORTER_ARM .....	11-9
MILL_OD_CYLINDRICAL .....	11-10
MILL_FACE_POLAR .....	11-11
MACROS_ROTATE .....	11-12
DUAL_SPINDLE .....	11-13
LENGTH_DIAM_OFFSET_FROM_TOOL .....	11-14
<b>CHAPTER 12 ADDITIONAL SYSTEM VARIABLES</b>	<b>12-1</b>
System Variables .....	12-2
<b>CHAPTER 13 ADDITIONAL SYSTEM COMMANDS</b>	<b>13-1</b>
SYS_CANNED .....	13-2
GETID .....	13-4
OPENTXT .....	13-5
CLOSETXT .....	13-6
SETTXT .....	13-7
UPPERTXT .....	13-8
LOWERTXT .....	13-9
ORIGINALTXT .....	13-10
OPENRWTXT .....	13-11
GETTXT .....	13-12

## Table of Contents

GETMCS .....	13-13
STRGLEN.....	13-14
LEFTSTRG.....	13-15
RIGHTSTRG .....	13-16
MIDSTRG.....	13-17
STRGUPPER.....	13-18
STRGLOWER .....	13-19
GET_SELECT STRING.....	13-20
FASTLINE.....	13-22
<b>CHAPTER 14 ADD'L. CALC SECTIONS &amp; OPERIDS</b>	<b>14-1</b>
CALC_ARC_MOVE_ZX.....	14-2
CALC_ARC_MOVE_YZ.....	14-3
CALC_ARC_MOVE_ANYPLANE.....	14-4
CALC_POST_INITIALIZE .....	14-5
CALC_TOOL_INITIALIZE.....	14-6
CALC_RAPID_MOVE_SHEAR .....	14-7
CALC_INIT_TOOL_CHANGE_SHEAR.....	14-8
CALC_SUB_TOOL_CHANGE_SHEAR.....	14-9
CALC_EVERY_MOVE_SHEAR.....	14-10
CALC_FULL_SHEAR.....	14-11
CALC_HALF_SHEAR_X.....	14-12
CALC_HALF_SHEAR_Y.....	14-13
CALC_FULL_SHEAR_DIAGONAL.....	14-14
CALC_HALF_SHEAR_DIAGONAL.....	14-15
CALC_REPOSITION_SHEAR.....	14-16
CALC_RAPID_TO_TRAP DOOR .....	14-17
CALC_PROFILE_DRILL_LASER .....	14-18
CALC_GET_TAPER_EDM.....	14-19
:OPERID .....	14-20
CALC_SLOWDOWN_SPEED .....	14-21
CALC_SHIFT_TOOL_LATHE .....	14-22
CALC_CUTTER_COMP_LATHE .....	14-23
CALC_ALLOW_RAPID_DURING_DRILL .....	14-24
CALC_SET_PRE_POSITION_ROTARY_TYPE .....	14-25
CALC_ADD_REAR_SYNC_CODE .....	14-26
CALC_ADD_FROMT_SYNC_CODE .....	14-27
<b>APPENDIX A USING AN ACCESS DATABASE DURING POSTING</b>	<b>A-1</b>
<b>APPENDIX B CALC SECTIONS</b>	<b>B-1</b>



---

# **Chapter 1 System Commands**

---

---

# ABS

## ***Purpose***

Returns the absolute value of an argument or number.

## ***Syntax***

**ABS (arg)**

## ***Comments***

<i>Parameter</i>	<i>Description</i>
arg	can be any expression or number

## ***Example***

Check if two points are the same:

```
:C: IF ABS(X_START-X_END)<.00005 THEN CALL(SAME) ENDIF
```

Assign the absolute value of a number to another variable:

```
:C: DISTANCE=(ABS( INC_X_END) )
```

---

# SIN

## ***Purpose***

Returns the sine of an angle in radians.

## ***Syntax***

**SIN(ang)**

## ***Comments***

<i>Parameter</i>	<i>Description</i>
ang	an angle in radians

To convert degrees to radians, multiply by (180/PI).

## ***Example***

```
:C: Y_POS=(ABS_J_CENTER+(ARC_RADIUS*SIN(ARC_END_ANGLE*(PI/180))))
```

---

# COS

## ***Purpose***

Returns the cosine of an angle in radians.

## ***Syntax***

**COS (ang)**

## ***Comments***

<i>Parameter</i>	<i>Description</i>
ang	an angle in radians

To convert degrees to radians, multiply by (PI/180).

## ***Example***

```
:C: X_POS=(ABS_I_CENTER+(ARC_RADIUS*COS(ARC_END_ANGLE*(PI/180))))
```

---

# TAN

## ***Purpose***

Returns the tangent of an angle in radians.

## ***Syntax***

**TAN ( ang )**

## ***Comments***

<i>Parameter</i>	<i>Description</i>
ang	an angle in radians

To convert degrees to radians, multiply by (PI/180).

## ***Example***

```
:C: HEIGHT=(RADIUS/TAN( (TOOL_ANGLE*(PI/180) ) /2 ) )
```

---

# ACOS

## ***Purpose***

Returns the arc cosine of a value from 1 to -1.

## ***Syntax***

**ACOS ( *r* )**

## ***Comments***

<i>Parameter</i>	<i>Description</i>
<i>r</i>	must be a value between 1 and -1

ACOS returns a value in radians between 0 and 3.141593.

## ***Example***

:C: Y= ( ACOS ( X ) )

---

# ASIN

## ***Purpose***

Returns the arc sine of a value from 1 to -1.

## ***Syntax***

**ASIN(*r*)**

## ***Comments***

<i>Parameter</i>	<i>Description</i>
<i>r</i>	must be a value between 1 and -1

ASIN returns a value in radians between 0 and 3.141593.

## ***Example***

:C: Y= (ASIN(X) )

---

# ATAN2

## ***Purpose***

Returns the arc tangent of dx and dy (i.e., the slope of a line).

## ***Syntax***

**ATAN2 (dx, dy)**

## ***Comments***

ATAN2 returns a value between 3.141593 and -3.141593.

## ***Example***

:C: V= (ATAN2 (DX, DY) )



---

# ATAN

## ***Purpose***

Returns the arc tangent of x.

## ***Syntax***

**ATAN (x)**

## ***Comments***

ATAN returns a value between (3.141593/2) and (-3.141593/2).

## ***Example***

:C: V= ( ATAN (X) )

---

# SQRT

## ***Purpose***

Return the square root of x.

## ***Syntax***

**SQRT (x)**

## ***Comments***

None.

## ***Example***

:C: V= ( SQRT (DX\*DX+DY\*DY) )

---

# LOOKUP

## ***Purpose***

To find a value within an array.

## ***Syntax***

**LOOKUP (ARRAY, VALUE, INDEX)**

## ***Comments***

<i>Parameter</i>	<i>Description</i>
ARRAY	the name of the array to search
VALUE	the value to search for
INDEX	the position in the array where the value was found (if INDEX=-1 the system failed to find the value in the array)

## ***Example***

The example below seeds an array then searches the array:

```
:C: ARRAY(1)=10 ARRAY(2)=11 ARRAY(3)=12
:C: VALUE=11
:C: LOOKUP (ARRAY, VALUE, INDEX)
:C: IF INDEX<>-1 THEN CALL(FOUND_IN_ARRAY) ENDIF
```

(The system will return a 2 in the variable INDEX)

```
:C: ARRAY(1)=10 ARRAY(2)=11 ARRAY(3)=12
:C: VALUE=20
:C: LOOKUP (ARRAY, VALUE, INDEX)
:C: IF INDEX=-1 THEN CALL(NOT_FOUND_IN_ARRAY) ENDIF
```

(The system will return a -1 in the variable INDEX)

---

# RESETEOL

## ***Purpose***

To remove (delete) the <EOL> (end of line) character from the last line of code output, so more code can be added.

## ***Syntax***

**RESETEOL**

## ***Comments***

## ***Example***

```
:C: RESETEOL CALL(ADD_END_OF_TAPE)
```

```
:SECTION=ADD_END_OF_TAPE
```

```
:T:<M02><EOL>
```

---

# STRCAT

## ***Purpose***

To append one string to another (i.e., concatenate strings).

## ***Syntax***

**STRCAT(string1, string2)**

## ***Comments***

<i>Parameter</i>	<i>Description</i>
string1	a character variable that will get the string attached to it
string2	a character variable that is the string to attach

## ***Example***

```
:C: STRING1={ProCAD} STRING2={/CAM}  
:C: STRCAT(STRING1, STRING2)  
:C: CALL(OUTPUT_STRING1)  
  
:SECTION=OUTPUT_STRING1  
:T:<N><STRING1><EOL>
```

Result: N10 ProCAD/CAM

---

# CALL

## ***Purpose***

To call another section of the post.

## ***Syntax***

### **A**

**CALL(section)**

### **B**

**CALL(section(arga, argb, ...))**

## ***Comments***

<i>Parameter</i>	<i>Description</i>
section	must be an existing section in the post. The CALL function can only call a SECTION.
arga	an argument that is passed to the section (Syntax B)
argb	an argument that is passed to the section (Syntax B)

When passing arguments, the section must be defined to be capable of accepting arguments. See `SECTION=` below.

## ***Example***

### **A**

```
:C: CALL(LINE_MOVE_MILL)
:SECTION=LINE_MOVE_MILL
:T:<N><G:01><X><Y><Z><F><attributes><EOL>
```

### **B**

```
:C: CALL(CALC_ESTIMATE_TIME(DISTANCE,OPR_X_FEED,TIME))
:SECTION=CALC_ESTIMATE_TIME(DIS,FEED,TIME)
:C: IF FEED=0 THEN RETURN ENDIF
:C: TIME=(TIME+(DIS/FEED))
```

---

# REPLACE

## ***Purpose***

To replace all occurrences of string A with string B throughout the entire tape.

## ***Syntax***

**REPLACE(stringa,stringb)**

## ***Comments***

<i>Parameter</i>	<i>Description</i>
stringa	the name of a character variable
stringb	the name of a character variable

## ***Example***

This example will replace the comment "Total hits=xxx", which was output at the beginning of the tape, with the correct number of hits known at the end of the tape. A temporary string must be used to transfer the number of hits stored in an integer to a string.

```
:SECTION=CALC_END_OF_TAPE
:C: STRA={Total hits=xxx}
:C: STRB={Total hits=}
:C: STRC=TOTAL_HITS
:C: STRCAT(STRB,STRC)
:C: REPLACE(STRB,STRC)
```

---

# SETON

## ***Purpose***

To set a parameter to output code.

## ***Syntax***

**SETON (<name>)**

## ***Comments***

<i>Parameter</i>	<i>Description</i>
name	the name of the parameter to be set on

Parameters must be defined as MODAL parameters in order for this command to have any effect.

## ***Example***

```
:SECTION=CALC_LINE_MOVE_MILL  
:C: SETON (<X>)  
:C: CALL (LINE_MOVE_MILL)  
:SECTION=LINE_MOVE_MILL  
:T: <N><G:01><X><Y><EOL>
```



---

# SETOFF

## ***Purpose***

To set a parameter to not output code.

## ***Syntax***

**SETOFF ( <name> )**

## ***Comments***

<i>Parameter</i>	<i>Description</i>
name	the name of the parameter to be set off

Parameters must be defined as MODAL parameters in order for this command to have any effect.

## ***Example***

:C: SETOFF (<F>)

The system will execute all sections responsible for outputting the parameter, however the result will not be sent to the tape.

---

# SYS\_CANNED

## Purpose

To break an entity not supported by the post into a series of entities that are supported by the post. Typically, this command is used the explode line, grid, arc and bolt hole patterns into single points.

## Syntax

**SYS\_CANNED(type,section)**

## Comments

Parameter	Description
type	the type of breakup and is a constant <ol style="list-style-type: none"> <li>1 Single points</li> <li>2 Lines, arcs and bolt holes (use only on grids and big hole patterns)</li> <li>3 Breaks a thread cycle into diameters (use only on threading cycles)</li> </ol>
section	section that will handle the exploded entity The word <code>SYSTEM</code> instructs the system to call the appropriate sections.

## Example

```
:SECTION=CALC_ARC_PATTERN_PUNCH
:C: SYS_CANNED(1,CALC_SINGLE_HIT_PUNCH)
:SECTION=CALC_GRID_PATTERN_PUNCH
:C: SYS_CANNED(2,SYSTEM)
:SECTION=CALC_MACHINE_THREAD_LATHE
:C: SYS_CANNED(3,CALC_MULTIPLE_THREAD_LATHE)
```

---

Warning: A `SYS_CANNED` command cannot be executed while inside of another `SYS_CANNED` cycle. In the example below grids are broken into line patterns, but since the post does not support line patterns, another `SYS_CANNED` command is executed. This is an error.

---

```
:SECTION=CALC_GRID_PATTERN_PUNCH
:C: SYS_CANNED(2,SYSTEM)
:SECTION=CALC_LINE_PATTERN_PUNCH
:C: SYS_CANNED(1,CALC_SINGLE_HIT_PUNCH)
```

The post should have been written:

```
:SECTION=CALC_GRID_PATTERN_PUNCH
:C: SYS_CANNED(1,CALC_SINGLE_HIT_PUNCH)
```

---

# GOTO

## ***Purpose***

Branches to a specific line number in the current section.

## ***Syntax***

**GOTO**number

## ***Comments***

<i>Parameter</i>	<i>Description</i>
number	any number from 0 to 9999

## ***Type***

Calculation section only

## ***Example***

To make a looping portion of code that loops until a condition is satisfied:

```
:C: LOOP=1
:C1: CALL(OUTPUT_TOOL)
:C: LOOP=(LOOP+1)
:C: IF LOOP>TOTAL_NUMBER_OF_TOOLS THEN RETURN ENDIF
:C: GOTO1
```

RETURN

---

# RETURN

## ***Purpose***

To end the current section and return processing control to the system.

## ***Syntax***

**RETURN**

## ***Comments***

None

## ***Example***

```
:C: IF TOOL=LAST_TOOL THEN RETURN ENDIF  
:C: CALL (CALC_TOOL_CHANGE_TIME)  
:C: CALL (SUB_TOOL_CHANGE)
```

---

# OFFSET\_INC

## ***Purpose***

To incrementally add an offset to all horizontal and vertical axis information.

## ***Syntax***

**OFFSET\_INC(hoff,voff)**

## ***Comments***

<i>Parameter</i>	<i>Description</i>
hoff	decimal variable containing amount of offset to add to all horizontal axis information
voff	decimal variable containing amount of offset to add to all vertical axis information

## ***Example***

Add 10" offset to X and 5" offset to Y, then cancel the offset:

```
:C: X_OFFSET=10 Y_OFFSET=5
:C: OFFSET_INC(X_OFFSET,Y_OFFSET)
:C: CALL(LINE_MOVE)
:C: X_OFFSET=-10 Y_OFFSET=-5
:C: OFFSET_INC(X_OFFSET,Y_OFFSET)
```

---

# OFFSET\_XYZ

## ***Purpose***

To incrementally add an offset to all X,Y,Z axis information. This command is used only in Mill.

## ***Syntax***

**OFFSET\_XYZ (x,y,z)**

## ***Comments***

<i>Parameter</i>	<i>Description</i>
x	decimal variable containing amount of offset to add to all X axis information
y	decimal variable containing amount of offset to add to all Y axis information
z	decimal variable containing amount of offset to add to all Z axis information

## ***Example***

Add 10" offset to X and 5" offset to Y and 3" offset to Z, then cancel the offsets:

```
:C: X_OFFSET=10 Y_OFFSET=5 Z_OFFSET=3  
:C: OFFSET_XYZ (X_OFFSET,Y_OFFSET,Z_OFFSET)  
:C: CALL (LINE_MOVE)  
:C: OFFSET_XYZ (-X_OFFSET,-Y_OFFSET,-Z_OFFSET)
```

---

# ADD\_MACRO\_START

## ***Purpose***

To define the start of a macro. This command redirects all code to a secondary file, which is used to store all subprograms prior to the system inserting the subprogram file before or after the main tape.

## ***Syntax***

**ADD\_MACRO\_START**

## ***Comments***

None.

## ***Example***

:C: ADD\_MACRO\_START

---

# ADD\_MACRO\_END

## ***Purpose***

To define the end of a macro. This command cancels an ADD\_MACRO\_START command and redirects all code back to the main program tape.

## ***Syntax***

**ADD\_MACRO\_END**

## ***Comments***

## ***Example***

:C: ADD\_MACRO\_END



---

# SPACES

## ***Purpose***

To allow spaces to be output or not to be output to the tape. This command is used to override the global post definition of : SPACES=FALSE in the header in <CONTROLLER.SRC).

## ***Syntax***

**SPACES (YES)**

**SPACES (NO)**

## ***Comments***

This command can be executed prior to calling a template section; however, it is recommended you use ATTRSPACE instead.

## ***Example***

**:C: SPACES (YES)**

**:C: CALL (SETUP\_SHEET)**

---

# GETTOOLS

***Purpose***

To retrieve from the system all tools used within the part being posted.

***Syntax***

**GETTOOLS (type, section)**

***Comments***

<i>Parameter</i>	<i>Description</i>
type	the type of sorting method
	1 Sorted by turret location
	2 Sorted by order used in part
section	the section that will be called each time a tool is loaded.
	The word SYSTEM will instruct the system to output a tool setup sheet.

***Example***

:C: GETTOOLS (2,CALC\_PRELOAD\_TOOL)

:C: GETTOOLS (1,SYSTEM)

---

# GET\_OPER\_COMMENTS

## ***Purpose***

To get the comments from the current operation. This command outputs the comments entered in the Comments dialog box. Used in Mill and Lathe only.

## ***Syntax***

**GET\_OPER\_COMMENTS(section)**

## ***Comments***

<i>Parameter</i>	<i>Description</i>
section	the section that will be called each time a comment is loaded. The word SYSTEM instructs the system to output the comments directly to the tape.

## ***Example***

```
:SECTION=CALC_SUB_TOOL_CHANGE_MILL
:C: GET_OPER_COMMENTS(CALC_OUTPUT_OPER_COMMENT)

:SECTION=CALC_OUTPUT_OPER_COMMENT
:C: CALL(OUTPUT_OPER_COMMENT)

:SECTION=OUTPUT_OPER_COMMENT
:T:<N><OPR_COMMENT><EOL>
```

---

# OPEN\_NEXT

## Purpose

To break a ".TXT" file and start a new tape output file. Typically, this command is used only for some older controllers that have a limit on the number of lines per file.

## Syntax

**OPEN\_NEXT(char\_str(arg), str\_len(arg), int\_number(arg))**

## Comments

<i>Parameter</i>	<i>Description</i>
char_str(agr)	file name as in - A0000001.TXT. A0000001 is the File name. It will assign the ".TXT" to it.
Str_len(agr)	the " = Length of file name "A0000001 is (8) Characters long
int_number(agr)	used for incrementing the last digits of the program name

## Example

```
:SECTION=CALC_BREAK_PROGRAM
```

This example will break a program at a certain line count. It will string cat until it has built a line.

The example file name = A1000001.TXT

program\_letter = A

PROGRAM\_PREFIX = program\_name = 1000

SUB\_COUNT = 1

```
:C: IF (LINE_COUNT+2)>line_number THEN GOTO2 ENDIF
:C: IF line_number<>(LINE_COUNT+2) THEN RETURN ENDIF
:C2: SUB_COUNT=(SUB_COUNT+1)
```

```
PROGRAM_PREFIX = Character Var. - String length = 5
SUB_ID          = Character Var.
ZEROS           = Character Var.
program_letter = Character setup Var. - String length = 1
SUB_COUNT       = Integer Var. - Max. length = 3
:C: PROGRAM_PREFIX=program_letter
:C: SUB_ID=program_number
:C: STRCAT(PROGRAM_PREFIX,SUB_ID)
:C: IF SUB_COUNT>99 THEN
:C: SUB_ID=SUB_COUNT
:C: STRCAT(PROGRAM_PREFIX,SUB_ID)
:C: GOTO1 ENDIF
:C: IF SUB_COUNT>9 THEN
:C: ZEROS={0}
:C: SUB_ID=SUB_COUNT
:C: STRCAT(ZEROS,SUB_ID)
:C: STRCAT(PROGRAM_PREFIX,ZEROS)
```

```

:C: GOTO1
:C: ENDIF
:C: ZEROS={00}
:C: SUB_ID=SUB_COUNT
:C: STRCAT(ZEROS, SUB_ID)
:C: STRCAT(PROGRAM_PREFIX, ZEROS)
:C1: S_SUB_COUNT_START=SUB_COUNT

```

This will call end of tape.

```

:C: CALL(SUB_OUTPUT_END)
:C: IF SUB_COUNT=1 THEN
:C: CALL(END_OF_TAPE_PUNCH)
:C: ELSE

```

This will call end of sub.

```

:C: CALL(SUB_END)
:C: ENDIF
:C: TOTAL_BYTE_COUNT=(TOTAL_BYTE_COUNT+BYTE_COUNT)

```

This will open new file.

```

:C: OPEN_NEXT(PROGRAM_PREFIX, 8, SUB_COUNT)

```

This will call start of sub.

```

:C: IF CANT_OUTPUT_PROGRAM=YES THEN RETURN ENDIF
:C: CALL(SUB_OUTPUT_START)

```

---

# ROUND OFF

## *Purpose*

To round off a number. This command is no longer used since all posts round-off automatically for inch or metric.

## *Syntax*

**ROUND OFF (number, bucket, result, places, type)**

## *Comments*

<i>Parameter</i>	<i>Description</i>
number	a decimal number or argument containing the value to be rounded off
bucket	a decimal variable that will receive the difference between the original number and the rounded off number (bucket is also added to the original number before any rounding occurs)
result	a decimal variable that will receive the rounded off result
places	the number of places to the right of the decimal
type	the type of mode: 1-English , 2-Metric

## *Example*

```
:C: BUCKET=0
:C: ROUND OFF (ABS_X_END, BUCKET, X_POS, G_RIGHT_PLACES, METRIC_OUT)
```

---

## :QUALIFIED\_TOOLING=

### **Purpose**

To retrieve tool offset information from a fixed structured external file.

### **Syntax**

**:QUALIFIED\_TOOLING=\PROCAD\TOOL\TOOLFILE.F6M**

### **Comments**

The above syntax: shows where the external file is located.

This command should be put in the post header info.

Below is an example of what the external file might look like.

1111 or 2222 must be entered in the TOOL COMMENT in the tool pulldown for this to work.

<i>ID</i>	<i>ZGL</i>	<i>XGL</i>	<i>Tool</i>	<i>Comment</i>
1111,	5,	10,	7,	80 Degree Diamond.
2222,	6.375,	12.5,	11,	.5 Diameter Drill.

The external file has 5 fields.

1. Field 1 is used by PROUNIV.EXE. It will match the "ID" field with the TOOL HOLDER\_NAME in lathe and TOOL\_COMMENT in mill.
2. Field 2 is a decimal field for the Z gauge.
3. Field 3 is a decimal field for the X gauge.
4. Field 4 is a integer field for the tool number.
5. Field 5 is a character field.

A comma is used as a field delimiter.

### **Example**

```
:T: Z FEED=<#:TOOL_ZGL> X FEED=<#:TOOL_XGL>
:T: RPM=<"%4T":TOOL_QTN> COMMENT=<TOOL_QT_COMMENT><EOL>
```

---

# INCON

## ***Purpose***

To output incremental movements.

## ***Syntax***

**:C: INCON**

## ***Comments***

At the time INCON is executed the variables XAXIS, YAXIS and ZAXIS are set to incremental distances.

This command is not used much, because it does not handle incremental roundoff.

## ***Example***

```
:SECTION=CALC_LINE_MOVE_MILL
:C: INCON
:C: X_POS=XAXIS
:C: Y_POS=YAXIS
:C: CALL(LINE_MOVE_MILL)

:SECTION=LINE_MOVE_MILL
:T:<N><G:01> X<#:X_POS> Y<#:Y_POS><F><EOL>
```



---

# INCOFF

## ***Purpose***

To cancel incremental movements.

## ***Syntax***

**:C: INCOFF**

## ***Comments***

At the time INCOFF is executed the variables XAXIS, YAXIS and ZAXIS are set to absolute distances. This command is the system default.

This command is not used much, because it does not handle incremental roundoff.

## ***Example***

```
:SECTION=CALC_LINE_MOVE_MILL
:C: INCOFF
:C: X_POS=XAXIS
:C: Y_POS=YAXIS
:C: CALL(LINE_MOVE_MILL)

:SECTION=LINE_MOVE_MILL
:T:<N><G:01> X<#:X_POS> Y<#:Y_POS><F><EOL>
```

# APPEND

## ***Purpose***

To append another file into current part that is being posted.

## ***Syntax***

**:C: APPEND(STRG)**

## ***Comments***

<i>Parameter</i>	<i>Description</i>
strg	The strg must be defined in the attribute section as a string variable. The strg should hold the path and filename you want to append.

The APPEND command will put whatever information is in the file you append into the file you are posting. This command could be used for a special cycle that is hardcoded and happens all the time.

## ***Example***

```
:SECTION=CALC_APPEND_FILE  
:C: STRG={C:\PROCAM\DRILL.TXT}  
:C: APPEND(STRG)
```

---

# TRANSFORM

## ***Purpose***

To allow the post to output world coordinates when it is outputting in machine coordinates.

There are specific machines that when implementing rotary axis preposition moves that the first move needs to be in world coordinates then switch back to machine coordinates for all other moves until a new rotary position is called.

Supported in CAMWorks 2007 SP2 and higher. Not supported in any ProCAM product.

## ***Syntax***

**:C: TRANSFORM**

## ***Comments***

Associated commands and variables:

These variables need to be set to the current machine values as shown.

```
:C: TRANS_START_X=ABS_X_END
:C: TRANS_START_Y=ABS_Y_END
:C: TRANS_START_Z=ABS_Z_END
```

These variables need to be assigned a vector number depending on the direction of the rotary motion. Below shows us that the 4th axis rotates about X and the 5th axis about Y. The vector numbers can have a range from -1 to +1. If you are using this in conjunction with 5axis multiaxis operations and are using a \*.KIN file, then you can use the second example below which will use the \*.KIN file to get the vector numbers.

```
:C: TRANS_ROTAXISDIR_4X=1.0
:C: TRANS_ROTAXISDIR_4Y=0.0
:C: TRANS_ROTAXISDIR_4Z=0.0
:C: TRANS_ROTAXISDIR_5X=0.0
:C: TRANS_ROTAXISDIR_5Y=1.0
:C: TRANS_ROTAXISDIR_5Z=0.0
```

2nd example:

```
:C: IF KIN_HAVE_KINEMATICS=TRUE THEN
:C: TRANS_ROTAXISDIR_4X=KIN_ROTAXISDIR_4X
:C: TRANS_ROTAXISDIR_4Y=KIN_ROTAXISDIR_4Y
:C: TRANS_ROTAXISDIR_4Z=KIN_ROTAXISDIR_4Z
:C: TRANS_ROTAXISDIR_5X=KIN_ROTAXISDIR_5X
:C: TRANS_ROTAXISDIR_5Y=KIN_ROTAXISDIR_5Y
:C: TRANS_ROTAXISDIR_5Z=KIN_ROTAXISDIR_5Z
:C: ENDIF
```

These variables are setting the current rotated angle.

```
:C: TRANS_ROTANGLE_A=ROT_TILT_A
:C: TRANS_ROTANGLE_B=ROT_TILT_B
```

## TRANSFORM

This command will perform the transform calculations.

```
:C: TRANSFORM
```

The lines below show how you can set post variables equal to the transformed numbers once the transform is done.

```
:C: X_POS=TRANS_END_X
:C: Y_POS=TRANS_END_Y
:C: Z_POS=TRANS_END_Z
```

### **Example**

This is an example of the order and how you could implement this command.

```
:C: TRANS_START_X=ABS_X_END
:C: TRANS_START_Y=ABS_Y_END
:C: TRANS_START_Z=ABS_Z_END
:C: TRANS_ROTAXISDIR_4X=1.0
:C: TRANS_ROTAXISDIR_4Y=0.0
:C: TRANS_ROTAXISDIR_4Z=0.0
:C: TRANS_ROTAXISDIR_5X=0.0
:C: TRANS_ROTAXISDIR_5Y=1.0
:C: TRANS_ROTAXISDIR_5Z=0.0
:C: TRANS_ROTANGLE_A=ROT_TILT_A
:C: TRANS_ROTANGLE_B=ROT_TILT_B
:C: TRANSFORM
:C: X_POS=TRANS_END_X
:C: Y_POS=TRANS_END_Y
:C: Z_POS=TRANS_END_Z
:C: CALL(TRANSLATED_OUTPUT)
```

This is another example of the order and how you could implement this command.

```
:C: TRANS_START_X=ABS_X_END
:C: TRANS_START_Y=ABS_Y_END
:C: TRANS_START_Z=ABS_Z_END
:C: IF KIN_HAVE_KINEMATICS=TRUE THEN
:C: TRANS_ROTAXISDIR_4X=KIN_ROTAXISDIR_4X
:C: TRANS_ROTAXISDIR_4Y=KIN_ROTAXISDIR_4Y
:C: TRANS_ROTAXISDIR_4Z=KIN_ROTAXISDIR_4Z
:C: TRANS_ROTAXISDIR_5X=KIN_ROTAXISDIR_5X
:C: TRANS_ROTAXISDIR_5Y=KIN_ROTAXISDIR_5Y
:C: TRANS_ROTAXISDIR_5Z=KIN_ROTAXISDIR_5Z
:C: ENDIF
:C: TRANS_ROTANGLE_A=ROT_TILT_A
:C: TRANS_ROTANGLE_B=ROT_TILT_B
:C: TRANSFORM
:C: X_POS=TRANS_END_X
:C: Y_POS=TRANS_END_Y
:C: Z_POS=TRANS_END_Z
:C: CALL(TRANSLATED_OUTPUT)
```

---

# RUN\_PROCESS

## ***Purpose***

To call any \*.exe file and execute it. It will also allow you to either continue posting while executing or stop posting until process is done.

## ***Syntax***

**:C: RUN\_PROCESS**

## ***Comments***

Supported in CAMWorks 2007 SP2 and higher. Not supported in any ProCAM product.

## ***Example***

In the example below:

**PROCESS\_COMMAND** determines the file and path to execute.

**WAIT\_FOR\_PROCESS=TRUE or FALSE** sets whether posting continues as the program is executed or waits until executed program is done.

This example will execute NOTEPAD.EXE and wait until you close NOTEPAD. This process can be executed in any CALC section.

```
:C: PROCESS_COMMAND={C:\WINDOWS\notepad.exe}  
:C: WAIT_FOR_PROCESS=TRUE  
:C: RUN_PROCESS
```

---

## QUERY\_SYSTEM()

### ***Purpose***

This command will query any object ID you pass to it. It can get information from an object that is in a Setup or operation dialog box. Available in Mill, Turn and Mill/Turn.

### ***Syntax***

**QUERY\_SYSTEM()**

### ***Comments***

Supported in CAMWorks 2009 and higher. Not supported in any ProCAM product.

Associated commands and variables: The example code below shows the other variables used with this command and that QUERY\_SYSTEM() will first determine what has been selected for X Axis machining direction on the Axis tab in the Setup dialog box. If the value is “1” then “Angle” was selected and we can now do another QUERY\_SYSTEM() to find the angle value that will be passed to QUERY\_DEC\_VALUE.

### ***Example***

Currently QUERY\_INT\_X\_SETUP\_DIR and QUERY\_DEC\_X\_SETUP\_ANGLE are the only object ID’s set. In the future you can send in an enhancement request to query other objects and we will add them when R&D resources are available.

```
:C: QUERY_ITEM_ID=QUERY_INT_X_SETUP_DIR
:C: QUERY_SYSTEM()
:C: IF QUERY_RESULT=1 THEN
:C:     IF QUERY_INT_VAL=1 THEN
:C:         QUERY_ITEM_ID=QUERY_DEC_X_SETUP_ANGLE
:C:         QUERY_SYSTEM()
:C:         IF QUERY_RESULT=1 THEN
:C:             MY_OPER_SETUP_ANGLE=QUERY_DEC_VAL
:C:             CALL (TEST_QUERY)
:C:         ENDIF
:C:     ENDIF
:C: ENDIF
```

---

# KILLSYSFILE()

## *Purpose*

This command will delete any file that has a path in the variable called SYSFILENAME.

## *Syntax*

```
:C: SYSFILENAME="drive:\folder\foldername"  
:C: KILLSYSFILE()
```

## *Example*

```
:C: SYSFILENAME="C:\TEST\TESTFILE.TXT"  
:C: KILLSYSFILE()
```

KILLSYSFILE()



---

## **Chapter 2 System Text Line Commands**

---

L

---

**L**

### ***Purpose***

Tells system to output leading zeros.

### ***Syntax***

**L**

### ***Comments***

The example is output if the number is 1 inch

N10 G01 X001

### ***Example***

:SECTION=LINE\_MOVE\_MILL

:T:<N><G:01> X<"#34L":ABS\_X\_END><EOL>

---

**I*****Purpose***

A lowercase "l" tells system to output leading spaces.

***Syntax***

**1**

***Comments***

The example is output if the number is 1 inch

N10 G01 X\_\_1

***Example***

:SECTION=LINE\_MOVE\_MILL

:T:<N><G:01> X<"#341":ABS\_X\_END><EOL>

N

---

# N

## ***Purpose***

An uppercase "N" tells system not to convert output if decimal.

## ***Syntax***

**N**

## ***Comments***

The example is output if the number is 180 degrees

N10 G01 A180

## ***Example***

:SECTION=LINE\_MOVE\_MILL

:T:<N><G:01> A<"#33N":ARC\_START\_ANGLE><EOL>

---

# T

## ***Purpose***

An uppercase "T" tells the system to output trailing zeros.

## ***Syntax***

**T**

## ***Comments***

The example is output if the number is 1 inch

N10 G01 X10000

## ***Example***

:SECTION=LINE\_MOVE\_MILL

:T:<N><G:01> X<"#34T":ABS\_X\_END><EOL>

t

---

**t**

### ***Purpose***

A lowercase "t" tells the system to output trailing spaces.

### ***Syntax***

t

### ***Comments***

The example is output if the number is 1 inch

N10 G01 X1\_\_\_\_\_

### ***Example***

:SECTION=LINE\_MOVE\_MILL

:T:<N><G:01> X<"#34t":ABS\_X\_END><EOL>

---

## @

### **Purpose**

An @ tells system to output a space in place of a plus or minus sign if a sign is not output.

### **Syntax**

@

### **Comments**

The below example is output if the number is 1 inch

N10 G01 X\_1

### **Example**

:SECTION=LINE\_MOVE\_MILL

:T:<N><G:01> X<"#-34@":ARC\_START\_ANGLE><EOL>

-

---

-

### ***Purpose***

Tells system to output a minus sign if number is negative.

### ***Syntax***

-

### ***Comments***

The below example is output if the number is negative 1 inch.

N10 G01 X-1

### ***Example***

:SECTION=LINE\_MOVE\_MILL

:T:<N><G:01> X<"#-34":ARC\_START\_ANGLE><EOL>



---

+

***Purpose***

Forces system to output a plus or minus sign.

***Syntax***

+

***Comments***

The below example is output if the number is negative 1 inch.

N10 G01 X+1

***Example***

:SECTION=LINE\_MOVE\_MILL

:T:<N><G:01> X<"#+34":ARC\_START\_ANGLE><EOL>

"

---

"

### ***Purpose***

Double quotation marks are used to group output format.

### ***Syntax***

" "

### ***Comments***

None.

### ***Example***

#### **#1: Not using HEADER default decimal output**

```
:SECTION=LINE_MOVE_MILL  
:T:<N><G:01> X<"#-34":ARC_START_ANGLE><EOL>
```

#### **#2: Using HEADER defaults**

```
:SECTION=LINE_MOVE_MILL  
:T:<N><G:01> X<#:ARC_START_ANGLE><EOL>
```

---

**#*****Purpose***

Tells system the output is decimal.

***Syntax***

**#**

***Comments***

None.

***Example*****#1: Not using HEADER default decimal output**

```
:SECTION=LINE_MOVE_MILL
:T:<N><G:01> X<"#-34":ARC_START_ANGLE><EOL>
```

**#2: Using HEADER defaults**

```
:SECTION=LINE_MOVE_MILL
:T:<N><G:01> X<#:ARC_START_ANGLE><EOL>
```

---

## ***Purpose***

A decimal point (.) forces the system to output a decimal to tape.

## ***Syntax***

.

## ***Comments***

None.

## ***Example***

### **#1: Not using a decimal point**

```
:SECTION=LINE_MOVE_MILL  
:T:<N><G:01> X<"#-34":ARC_START_ANGLE><EOL>
```

```
N100 G01 X?
```

### **#2: Using a decimal point**

```
:SECTION=LINE_MOVE_MILL  
:T:<N><G:01> X<"#-3.4":ARC_START_ANGLE><EOL>
```

```
N10 G01 X?.
```

---

%

***Purpose***

Tells system the output is integer.

***Syntax***

%

***Comments***

None.

***Example*****#1: Not Using HEADER default integer output**

```
:SECTION=START_OF_TAPE_MILL
:T:O<"%4LT":program_number><EOL>
Returns: O0001
```

**#2: Using HEADER defaults**

```
:SECTION=START_OF_TAPE_MILL
:T:O<%:program_number><EOL>
```

%

---

## **Chapter 3 System CCL Commands**

---

GET\_DATA()

---

## GET\_DATA()

### ***Purpose***

To build popup for attribute list.

### ***Syntax***

**:C: GET\_DATA(listname)**

### ***Comments***

This command will invoke an attribute list for the user to answer in CAD.

### ***Example***

```
:ATTRNAME=ellipse
:ATTRTYPE=LIST
:ATTRSEL=N
:ATTRTITLE=Ellipse
:ATTRLIST=major
:ATTRLISTDEF=10
:ATTRLIST=minor
:ATTRLISTDEF=5
:ATTRLIST=nsegments
:ATTRLISTDEF=120
:ATTRUSED=1
:ATTRDEFAULT=1
:ATTREND

:SECTION=CALC_MAIN
:C: GET_DATA(ellipse)
```



---

# GET\_POINT()

## ***Purpose***

To call pick point and return result.

## ***Syntax***

**:C: GET\_POINT()**

## ***Comments***

This command invokes user to snap a point, which will return the result in the system variables ABS\_X\_END and ABS\_Y\_END.

## ***Example***

```
:SECTION=CALC_MAIN  
:C: GET_POINT()  
:C: OFF_IN_X=ABS_X_END  
:C: OFF_IN_Y=ABS_Y_END
```

SET\_COLOR()

---

## SET\_COLOR()

### ***Purpose***

To set current system color in CAD.

### ***Syntax***

**:C: SET\_COLOR(color)**

### ***Comments***

This command sets the system color from 0-15:

BLACK	=	0
BLUE	=	1
GREEN	=	2
CYAN	=	3
RED	=	4
MAGENTA	=	5
BROWN	=	6
WHITE	=	7
GREY	=	8
LIGHT_BLUE	=	9
LIGHT_GREEN	=	10
LIGHT_CYAN	=	11
LIGHT_RED	=	12
LIGHT_MAGENTA	=	13
LIGHT_YELLOW	=	14
BRIGHT_WHITE	=	15

### ***Example***

This example sets the system color to BLUE.

```
:SECTION=CALC_MAIN  
:C: SET_COLOR(1)
```

---

# SET\_TEXT\_COLOR()

## ***Purpose***

To set current system text color in CAD.

## ***Syntax***

**:C: SET\_TEXT\_COLOR(color)**

## ***Comments***

This command sets the system text color from 0-15:

BLACK	=	0
BLUE	=	1
GREEN	=	2
CYAN	=	3
RED	=	4
MAGENTA	=	5
BROWN	=	6
WHITE	=	7
GREY	=	8
LIGHT_BLUE	=	9
LIGHT_GREEN	=	10
LIGHT_CYAN	=	11
LIGHT_RED	=	12
LIGHT_MAGENTA	=	13
LIGHT_YELLOW	=	14
BRIGHT_WHITE	=	15

## ***Example***

This example sets the system text color to BLUE.

:SECTION=CALC\_MAIN

**:C: SET\_TEXT\_COLOR(1)**

SET\_LAYER()

---

## SET\_LAYER()

### ***Purpose***

To set current system layer in CAD.

### ***Syntax***

**:C: SET\_LAYER(layer)**

### ***Comments***

This command sets the system layer from 0-256.

### ***Example***

This example sets the system layer to "1".

:SECTION=CALC\_MAIN

**:C: SET\_LAYER(1)**

---

# ADD\_PUNCH\_PATTERN()

## ***Purpose***

To add a punch pattern.

## ***Syntax***

**:C: ADD\_PUNCH\_PATTERN(pattern)**

## ***Comments***

You must set all the system variables first before you add the pattern in CAM.

## ***Examples***

### **#1: Example of a grid pattern**

```
:SECTION=CALC_MAIN
:C: SELECT_TOOL(1)
:C: ABS_X_START=5.
:C: ABS_Y_START=5.
:C: ABS_X_END=10.
:C: ABS_Y_END=10.
:C: ANGLE=0
:C: NUM_HITS_X=5
:C: NUM_HITS_Y=5
:C: DIST_BET_HOLES_X=1.
:C: DIST_BET_HOLES_Y=1.
:C: HORIZ_OR_VERT=HORIZONTAL
:C: ADD_PUNCH_PATTERN(MGRID)
```

### **#2: Example of a clockwise arc pattern**

```
:SECTION=CALC_MAIN
:C: SELECT_TOOL(1)
:C: ABS_X_START=5.
:C: ABS_Y_START=5.
:C: ABS_X_END=10.
:C: ABS_Y_END=10.
:C: ABS_I_CENTER=10.
:C: ABS_J_CENTER=5.
:C: NUM_HITS=5
:C: MOVE_TYPE=CW_ARC
:C: ADD_PUNCH_PATTERN(MARC)
```

ADD\_PUNCH\_PATTERN()

### #3: Example of a counterclockwise bolt circle pattern

```
:SECTION=CALC_MAIN
:C: SELECT_TOOL(1)
:C: ABS_X_START=20.
:C: ABS_Y_START=15.
:C: ABS_X_END=20.
:C: ABS_Y_END=15.
:C: ABS_I_CENTER=15.
:C: ABS_J_CENTER=15.
:C: NUM_HITS=8
:C: ARC_START_ANGLE=22.5
:C: INC_ANGLE=45
:C: MOVE_TYPE=CCW_ARC
:C: ADD_PUNCH_PATTERN(MCIRCLE)
```

### #4: Example of a line at angle pattern

```
:SECTION=CALC_MAIN
:C: SELECT_TOOL(1)
:C: ABS_X_START=5.
:C: ABS_Y_START=5.
:C: ABS_X_END=10.
:C: ABS_Y_END=10.
:C: NUM_HITS=5
:C: MOVE_TYPE=LINE
:C: ADD_PUNCH_PATTERN(MLINE)
```

---

# ADD\_PUNCH\_PATH()

## ***Purpose***

To add a punch path - nibble line or arc.

## ***Syntax***

**:C: ADD\_PUNCH\_PATH(path)**

## ***Comments***

You must set all the system variables first before you add the path in CAM.

## ***Example***

### **#1: Example of a line path**

```
:SECTION=CALC_MAIN  
:C: SELECT_TOOL(1)  
:C: ABS_X_START=5.  
:C: ABS_Y_START=5.  
:C: ABS_X_END=10.  
:C: ABS_Y_END=10.  
:C: PITCH=.125  
:C: MOVE_TYPE=LINE  
:C: ADD_PUNCH_PATH(MLINE)
```

### **#2: Example of an clockwise arc path**

```
:SECTION=CALC_MAIN  
:C: SELECT_TOOL(1)  
:C: ABS_X_START=5.  
:C: ABS_Y_START=5.  
:C: ABS_X_END=10.  
:C: ABS_Y_END=10.  
:C: ABS_I_CENTER=10.  
:C: ABS_J_CENTER=5.  
:C: PITCH=.125  
:C: MOVE_TYPE=CW_ARC  
:C: ADD_PUNCH_PATH(MARC)
```

---

## ADD\_REPOSITION()

### ***Purpose***

To add a reposition in CAM.

### ***Syntax***

**:C: ADD\_REPOSITION()**

### ***Comments***

You must set all the system variables first before you add the reposition in CAM.

### ***Example***

The example below will rapid to "X" and "Y" to ten inches and then reposition the sheet in "X" 20 inches incrementally.

```
:SECTION=CALC_MAIN  
:C: SELECT_TOOL(1)  
:C: ABS_X_END=10.  
:C: ABS_Y_END=10.  
:C: INC_X_END=20  
:C: ADD_REPOSITION()
```



---

# ADD\_CAD()

## ***Purpose***

To add a CAD entity.

## ***Syntax***

**:C: ADD\_CAD(position)**

## ***Comments***

You must set all the system variables first before you add the entity in CAD. You add CAD by PREVIOUS position, CURRENT position or NEXT position.

## ***Examples***

### **#1: Example of a current line entity**

```
:SECTION=CALC_MAIN
:C: ABS_X_START=5.
:C: ABS_Y_START=5.
:C: ABS_X_END=10.
:C: ABS_Y_END=10.
:C: MOVE_TYPE=MLINE
:C: ADD_CAD(CURRENT)
```

### **#2: Example of an current clockwise arc entity**

```
:SECTION=CALC_MAIN
:C: ABS_X_START=5.
:C: ABS_Y_START=5.
:C: ABS_X_END=10.
:C: ABS_Y_END=10.
:C: ABS_I_CENTER=10.
:C: ABS_J_CENTER=5.
:C: MOVE_TYPE=MCW_ARC
:C: ADD_CAD(CURRENT)
```

### **#3: Example of a previous line entity**

```
:SECTION=CALC_MAIN
:C: P_ABS_X_START=5.
:C: P_ABS_Y_START=5.
:C: P_ABS_X_END=10.
:C: P_ABS_Y_END=10.
:C: P_MOVE_TYPE=MLINE
:C: ADD_CAD(PREVIOUS)
```

### **#4: Example of a previous clockwise arc entity**

ADD\_CAD()

```
:SECTION=CALC_MAIN  
:C: P_ABS_X_START=5.  
:C: P_ABS_Y_START=5.  
:C: P_ABS_X_END=10.  
:C: P_ABS_Y_END=10.  
:C: P_ABS_I_CENTER=10.  
:C: P_ABS_J_CENTER=5.  
:C: P_MOVE_TYPE=MCW_ARC
```

---

# MAKE\_FILLET()

## ***Purpose***

To make a fillet between two CAD entities.

## ***Syntax***

**:C: MAKE\_FILLET(radius)**

## ***Comments***

You must set all the system variables first before you make the fillet in CAD.

## ***Example***

```
:SECTION=CALC_MAIN
:C: ARC_RAD(.1)
:C: P_ABS_X_START=5.
:C: P_ABS_Y_START=5.
:C: P_ABS_X_END=10.
:C: P_ABS_Y_END=5.
:C: P_MOVE_TYPE=MLINE
:C: N_ABS_X_START=10.
:C: N_ABS_Y_START=5.
:C: N_ABS_X_END=10.
:C: N_ABS_Y_END=10.
:C: N_MOVE_TYPE=MLINE
:C: MAKE_FILLET(ARC_RAD)
:C: ADD_CAD(PREVIOUS)
:C: ADD_CAD(CURRENT)
:C: P_ABS_X_START=N_ABS_X_START
:C: P_ABS_Y_START=N_ABS_Y_START
:C: P_ABS_X_END=N_ABS_X_END
:C: P_ABS_Y_END=N_ABS_Y_END
:C: P_MOVE_TYPE=N_MOVE_TYPE
:C: ADD_CAD(NEXT)
```

---

# ADD\_PUNCH\_TOOL()

## ***Purpose***

To add a punch tool in CAM.

## ***Syntax***

**:C: ADD\_PUNCH\_TOOL(tool)**

## ***Comments***

You must set all the system tool variables first before you add the tool in CAM.

TOOL\_TYPE:

ROUND	=	1
RECTANGLE	=	2
TRIANGLE	=	3
CROSS	=	4
OBROUND	=	5
SQUARE	=	6
RECRAD	=	7
DOUBLED	=	8
SINGLED	=	9
SPETOOL	=	10

## ***Examples***

### **#1**

```
:SECTION=CALC_MAIN
:C: TOOL_TYPE=1
:C: TOOL_DIAMETER=.5
:C: TOOL_LOAD_ANGLE=0
:C: TOOL_COMMENT={.5 ROUND PUNCH}
:C: TOOL_DISCRIPTION={ROUND PUNCH}
:C: ADD_PUNCH_TOOL(TOOL)
```

### **#2**

```
:SECTION=CALC_MAIN
:C: TOOL_TYPE=2
:C: TOOL_LENGTH=.5
:C: TOOL_WIDTH=.25
:C: TOOL_LOAD_ANGLE=0
:C: TOOL_COMMENT={.5 x .25 RECTANGLE PUNCH}
:C: TOOL_DISCRIPTION={RECTANGLE PUNCH}
:C: ADD_PUNCH_TOOL(TOOL)
```

### **#3**

```
:SECTION=CALC_MAIN
```

## ADD\_PUNCH\_TOOL()

```
:C: TOOL_TYPE=7
:C: TOOL_LENGTH=.5
:C: TOOL_WIDTH=.25
:C: TOOL_CORNER_RADIUS=.01
:C: TOOL_LOAD_ANGLE=0
:C: TOOL_COMMENT={.5 x .25 RECTANGLE RADIUS PUNCH}
:C: TOOL_DISCRIPTION={RECTANGLE RADIUS PUNCH}
:C: ADD_PUNCH_TOOL(TOOL)
```

SELECT\_TOOL()

---

## SELECT\_TOOL()

### ***Purpose***

To select a punch tool in CAM.

### ***Syntax***

**:C: SELECT\_TOOL(tool)**

### ***Comments***

If the tool already exists in the part tool list then you can use this command. If the tool does not exist in the current part tool list, you must use the `ADD_PUNCH_TOOL ( )` command.

### ***Example***

```
:SECTION=CALC_MAIN  
:C: TOOL=1  
:C: SELECT_TOOL(TOOL)
```

---

# START\_COMPLEX()

## ***Purpose***

To start a CAM boundary.

## ***Syntax***

**:C: START\_COMPLEX()**

## ***Comments***

This command will initiate a CAM boundary. Make sure you use this command before you add your first entity in CAD.

## ***Example***

```
:SECTION=CALC_MAIN  
:C: START_COMPLEX()  
:C: ABS_X_START=5.  
:C: ABS_Y_START=5.  
:C: ABS_X_END=10.  
:C: ABS_Y_END=10.  
:C: MOVE_TYPE=MLINE  
:C: ADD_CAD(CURRENT)
```

END\_COMPLEX()

---

## END\_COMPLEX()

### ***Purpose***

To end a CAM boundary.

### ***Syntax***

**:C: END\_COMPLEX ( )**

### ***Comments***

This command will end a CAM boundary. Make sure this command is after your last entity in CAD.

### ***Example***

```
:SECTION=CALC_MAIN  
:C: START_COMPLEX ( )  
:C: ABS_X_START=5 .  
:C: ABS_Y_START=5 .  
:C: ABS_X_END=10 .  
:C: ABS_Y_END=10 .  
:C: MOVE_TYPE=MLINE  
:C: ADD_CAD (CURRENT)  
:C: END_COMPLEX ( )
```



---

# START\_GROUP()

## ***Purpose***

To start a group of entities.

## ***Syntax***

**:C: START\_GROUP ( )**

## ***Comments***

This command will initiate a group of entities. Make sure you use this command before you add your first entity in CAD.

## ***Example***

```
:SECTION=CALC_MAIN  
:C: START_GROUP ( )  
:C: ABS_X_START=5.  
:C: ABS_Y_START=5.  
:C: ABS_X_END=10.  
:C: ABS_Y_END=10.  
:C: MOVE_TYPE=MLINE  
:C: ADD_CAD (CURRENT)
```

END\_GROUP()

---

## END\_GROUP()

### ***Purpose***

To end a group of entities.

### ***Syntax***

**:C: END\_GROUP ( )**

### ***Comments***

This command will end a group. Make sure this command is after your last entity in CAD.

### ***Example***

```
:SECTION=CALC_MAIN  
:C: START_GROUP ( )  
:C: ABS_X_START=5 .  
:C: ABS_Y_START=5 .  
:C: ABS_X_END=10 .  
:C: ABS_Y_END=10 .  
:C: MOVE_TYPE=MLINE  
:C: ADD_CAD (CURRENT)  
:C: END_GROUP ( )
```

---

## Chapter 4 Attribute Commands

---

:ATTRNAME=

---

## :ATTRNAME=

### ***Purpose***

Start attribute definition.

### ***Syntax***

**:ATTRNAME=name**

### ***Comments***

This command starts an attribute definition. name identifies the attribute.

### ***Example***

```
:ATTRNAME=MACHINE NAME  
:ATTRTYPE=DESCRIPTOR  
:ATTRVTYPE=CHARACTER  
:ATTRID=501  
:ATTREND
```

---

## :ATTREND

### ***Purpose***

Ends an attribute definition.

### ***Syntax***

**:ATTREND**

### ***Comments***

This command ends an attribute definition.

### ***Example***

```
:ATTRNAME=MACHINE NAME  
:ATTRTYPE=DESCRIPTOR  
:ATTRVTYPE=CHARACTER  
:ATTRID=501  
:ATTREND
```

:ATTRTYPE=

---

## :ATTRTYPE=

### **Purpose**

Defines attribute type.

### **Syntax**

```
:ATTRTYPE=POST  
:ATTRTYPE=DESCRIPTOR  
:ATTRTYPE=VALUE  
:ATTRTYPE=DISPLAY  
:ATTRTYPE=SELECT  
:ATTRTYPE=LIST
```

### **Comments**

<i>Parameter</i>	<i>Description</i>
POST	Defined only in the post and can only be used while posting.
DESCRIPTOR	Machine descriptors. These attributes have to be defined in MASTER.ATR and defined in the post.
VALUE	Used for Setup and Attachable attributes that require entering a value. These attributes have to be defined in MASTER.ATR and defined in the post.
DISPLAY	Used in HARDCODE attributes that are displayed only (they do not require a choice or value). These attributes have to be defined in MASTER.ATR and defined in the post.
SELECT	Used for attributes that require selection from a list of choices. These attributes have to be defined in MASTER.ATR if they are to be used for Setup or Attachable type attributes. If used for posting only, then they need to be defined only in the post.
LIST	Used for Setup attributes. This has to be the last attribute defined. These attributes have to be defined in MASTER.ATR and defined in the post.

### **Example**

```
:ATTRNAME=x sheet width  
:ATTRTYPE=VALUE  
:ATTRVTYPE=DECIMAL  
:ATTREMARK=X Sheet Width  
:ATTRSEL=N  
:ATTRINLEN=10  
:ATTRSHORT=X Sheet Width  
:ATTRLONG=Enter X Sheet Width  
:ATTRHIGH=9999  
:ATTRLOW=9999  
:ATTRDEFAULT=0  
:ATTRUSED=1
```

:ATTRTYPE=

:ATTREND

:ATTRNAME=optional stop  
**:ATTRTYPE=DISPLAY**  
:ATTRREMARK=optional stop M01  
:ATTRSEL=Y  
:ATTRTEXT=  
:ATTRFUNC=optional\_stop  
:CODETYPE=HARDCODE  
:CODE=|M01  
:ATTRUSED=1  
:ATTREND

:ATTRNAME=work chute  
**:ATTRTYPE=SELECT**  
:ATTRVTYPE=INTEGER  
:ATTRSEL=Y  
:ATTRSHORT=Work Chute  
:ATTRTITLE=Work Chute (open/close)  
:ATTRSELSTR=M80  
:ATTRDEFAULT=1  
:ATTRUSED=1  
:ATTREND

:ATTRNAME=setup  
**:ATTRTYPE=LIST**  
:ATTRSEL=N  
:ATTRTITLE=Setup  
:ATTRLIST=program number  
:ATTRLISTDEF=1  
:ATTRLIST=x sheet width  
:ATTRLISTDEF=0  
:ATTRLIST=y sheet width  
:ATTRLISTDEF=0  
:ATTRUSED=1  
:ATTRDEFAULT=1  
:ATTREND

:ATTRVTYPE=

---

## :ATTRVTYPE=

### ***Purpose***

Variable type.

### ***Syntax***

**:ATTRVTYPE=INTEGER**  
**:ATTRVTYPE=DECIMAL**  
**:ATTRVTYPE=CHARACTER**

### ***Comments***

If no ATTRVTYPE is put in the attribute definition then it assumes it is a character variable type. If, however, you define the attribute with a :VAR=? the ?= another attribute that has been defined before this one and it had a ATTRVTYPE defined, then this attribute will be the same variable type as in the :VAR= command.

### ***Example***

```
:ATTRNAME=x sheet width
:ATTRTYPE=VALUE
:ATTRVTYPE=DECIMAL
:ATTREMARK=X Sheet Width
:ATTRSEL=N
:ATTRINLEN=10
:ATTRSHORT=X Sheet Width
:ATTRLONG=Enter X Sheet Width
:ATTRHIGH=9999
:ATTRLOW=9999
:ATTRDEFAULT=0
:ATTRUSED=1
:ATTREND
```



---

## :ATTRID=

### ***Purpose***

Attribute ID number.

### ***Syntax***

**:ATTRID=501**

### ***Comments***

This command determines the ID number for any attribute that is defined in the MASTER.ATR file.

### ***Example***

```
:ATTRNAME=MACHINE NAME  
:ATTRTYPE=DESCRIPTOR  
:ATTRVTYPE=CHARACTER  
:ATTRID=501  
:ATTREND
```

:ATTREMARK

---

## :ATTREMARK

### ***Purpose***

Information for programmer only.

### ***Syntax***

**:ATTREMARK=text**

### ***Comments***

This command allows the programmer to document the attribute for future reference.

### ***Example***

```
:ATTRNAME=x sheet width
:ATTRTYPE=VALUE
:ATTRVTYPE=DECIMAL
:ATTREMARK=X Sheet Width
:ATTRSEL=N
:ATTRINLEN=10
:ATTRSHORT=X Sheet Width
:ATTRLONG=Enter X Sheet Width
:ATTRHIGH=9999
:ATTRLOW=9999
:ATTRDEFAULT=0
:ATTRUSED=1
:ATTREND
```

---

# :ATTRLISTDEF

## ***Purpose***

List string default.

## ***Syntax***

**:ATTRLISTDEF=n**

## ***Comments***

This command allows you to set the list default value for a Setup select type attribute.

## ***Example***

```
:ATTRNAME=setup
:ATTRTYPE=LIST
:ATTRSEL=N
:ATTRTITLE=Setup
:ATTRLIST=program number
:ATTRLISTDEF=1
:ATTRLIST=x sheet width
:ATTRLISTDEF=0
:ATTRLIST=y sheet width
:ATTRLISTDEF=0
:ATTRUSED=1
:ATTRDEFAULT=1
:ATTREND
```

:ATTRLIST=

---

## :ATTRLIST=

### ***Purpose***

List string for list type attribute.

### ***Syntax***

**:ATTRLIST=attribute name**

### ***Comments***

This command allows you to insert a previously defined attribute into a list.

### ***Example***

```
:ATTRNAME=setup
:ATTRTYPE=LIST
:ATTRSEL=N
:ATTRTITLE=Setup
:ATTRLIST=program number
:ATTRLISTDEF=1
:ATTRLIST=x sheet width
:ATTRLISTDEF=0
:ATTRLIST=y sheet width
:ATTRLISTDEF=0
:ATTRUSED=1
:ATTRDEFAULT=1
:ATTREND
```

---

## :ATTRSELSTR=

### ***Purpose***

Select string for select type.

### ***Syntax***

**:ATTRSELSRT=string**

### ***Comments***

This command is used in select type attributes to specify the choices.

### ***Example***

```
:ATTRNAME=init machine comp
:ATTRTYPE=SELECT
:ATTREMARK=Comp lft, rgt, cancel
:ATTRSEL=N
:ATTRTITLE=Laser Compensation
:ATTRLIST=program number
:ATTRSELSTR=Left
:ATTRSELSTR=Right
:ATTRSELSTR=Cancel
:ATTRDEFAULT=1
:ATTRUSED=1
:ATTREND
```

:ATTRSEL=

---

## :ATTRSEL=

### ***Purpose***

Selectable switch.

### ***Syntax***

**:ATTRSEL=Y or N**

### ***Comments***

This command determines whether an attribute displays in the Select Attribute dialog box or in the Setup Information dialog box. If set to Y (yes), the attribute displays in the Select Attribute dialog box and cannot be listed in the Setup Information dialog box. If set to N (no), the attribute displays in the Setup Information dialog box.

### ***Example***

```
:ATTRNAME=init machine comp
:ATTRTYPE=SELECT
:ATTREMARK=Comp lft, rgt, cancel
:ATTRSEL=N
:ATTRTITLE=Laser Compensation
:ATTRLIST=program number
:ATTRSELSTR=Left
:ATTRSELSTR=Right
:ATTRSELSTR=Cancel
:ATTRDEFAULT=1
:ATTRUSED=1
:ATTREND
```

---

## :ATTRUSED=

### ***Purpose***

Flag for attribute used or not.

### ***Syntax***

**:ATTRUSED=1 or 0**

### ***Comments***

This command tells the compiler if an attribute is used or not. If set to 1, the attribute is included in the compile. If set to 0, the attribute is not compiled.

### ***Example***

```
:ATTRNAME=init machine comp
:ATTRTYPE=SELECT
:ATTREMARK=Compt lft, rgt, cancel
:ATTRSEL=N
:ATTRTITLE=Laser Compensation
:ATTRLIST=program number
:ATTRSELSTR=Left
:ATTRSELSTR=Right
:ATTRSELSTR=Cancel
:ATTRDEFAULT=1
:ATTRUSED=1
:ATTREND
```

:ATTRDEFAULT=

---

## :ATTRDEFAULT=

### ***Purpose***

Define default for attribute.

### ***Syntax***

**:ATTRDEFAULT=value**

### ***Comments***

This command sets the default value for input. This command overrides ATTRLISTDEF.

### ***Example***

```
:ATTRNAME=x sheet width
:ATTRTYPE=VALUE
:ATTRVTYPE=DECIMAL
:ATTREMARK=X Sheet Width
:ATTRSEL=N
:ATTRINLEN=10
:ATTRSHORT=X Sheet Width
:ATTRLONG=Enter X Sheet Width
:ATTRHIGH=9999
:ATTRLOW=9999
:ATTRDEFAULT=0
:ATTRUSED=1
:ATTREND
```



---

## :ATTRINLEN=

### ***Purpose***

Define input length for attribute.

### ***Syntax***

**:ATTRINLEN=value**

### ***Comments***

This command sets the input length.

For decimal input (3.4), set this value to 10, which includes the decimal point, the + sign and the - sign.

For integers, the length is normally 4.

### ***Example***

```
:ATTRNAME=x sheet width
:ATTRTYPE=VALUE
:ATTRVTYPE=DECIMAL
:ATTREMARK=X Sheet Width
:ATTRSEL=N
:ATTRINLEN=10
:ATTRSHORT=X Sheet Width
:ATTRLONG=Enter X Sheet Width
:ATTRHIGH=9999
:ATTRLOW=9999
:ATTRDEFAULT=0
:ATTRUSED=1
:ATTREND
```

:ATTRSHORT=

---

## :ATTRSHORT=

### ***Purpose***

Defines the text that displays in the dialog boxes for value type attributes.

### ***Syntax***

**:ATTRSHORT=***text*

### ***Comments***

This command defines the text that displays in the dialog box for value type attributes. When the post is compiled, this text is output to the .LNG file and can be translated if necessary.

### ***Example***

```
:ATTRNAME=x sheet width
:ATTRTYPE=VALUE
:ATTRVTYPE=DECIMAL
:ATTREMARK=X Sheet Width
:ATTRSEL=N
:ATTRINLEN=10
:ATTRSHORT=X Sheet Width
:ATTRLONG=ENTER X Sheet Width
:ATTRHIGH=9999
:ATTRLOW=9999
:ATTRDEFAULT=0
:ATTRUSED=1
:ATTREND
```

---

## :ATTRLONG=

### ***Purpose***

Defines the text that displays on the prompt line for value type attributes.

### ***Syntax***

**:ATTRLONG=***text*

### ***Comments***

This command defines the text that displays on the prompt line for value type attributes. When the post is compiled, this text is output to the .LNG file and can be translated if necessary.

### ***Example***

```
:ATTRNAME=x sheet width
:ATTRTYPE=VALUE
:ATTRVTYPE=DECIMAL
:ATTREMARK=X Sheet Width
:ATTRSEL=N
:ATTRINLEN=10
:ATTRSHORT=X Sheet Width
:ATTRLONG=ENTER X Sheet Width
:ATTRHIGH=9999
:ATTRLOW=9999
:ATTRDEFAULT=0
:ATTRUSED=1
:ATTREND
```

:ATTRHIGH=

---

## :ATTRHIGH=

### ***Purpose***

Answer high.

### ***Syntax***

**:ATTRHIGH=value**

### ***Comments***

This command defines the maximum value that can be entered for the attribute. This command is used with value type attributes.

### ***Example***

```
:ATTRNAME=x sheet width
:ATTRTYPE=VALUE
:ATTRVTYPE=DECIMAL
:ATTREMARK=X Sheet Width
:ATTRSEL=N
:ATTRINLEN=10
:ATTRSHORT=X Sheet Width
:ATTRLONG=ENTER X Sheet Width
:ATTRHIGH=9999
:ATTRLOW=9999
:ATTRDEFAULT=0
:ATTRUSED=1
:ATTREND
```

---

## :ATTRLOW=

### **Purpose**

Answer low.

### **Syntax**

**:ATTRLOW=value**

### **Comments**

This command defines the lowest value that can be entered for the attribute. This command is used with value type attributes. If the value type is an integer, you cannot define a negative value. If the value type is decimal, you can define a negative value.

### **Example**

```
:ATTRNAME=x sheet width
:ATTRTYPE=VALUE
:ATTRVTYPE=DECIMAL
:ATTREMARK=X Sheet Width
:ATTRSEL=N
:ATTRINLEN=10
:ATTRSHORT=X Sheet Width
:ATTRLONG=ENTER X Sheet Width
:ATTRHIGH=9999
:ATTRLOW=9999
:ATTRDEFAULT=0
:ATTRUSED=1
:ATTREND
```

:ATTRTEXT=

---

## :ATTRTEXT=

### ***Purpose***

Text for display type attributes.

### ***Syntax***

**:ATTRTEXT=text**

### ***Comments***

This command defines the text for display type attributes. This command is not used.

### ***Example***

---

## :ATTRTITLE=

### ***Purpose***

Text for select type attributes in the Select Attribute dialog box.

### ***Syntax***

**:ATTRTITLE=text**

### ***Comments***

This command defines the text that identifies the attribute in the Select Attribute dialog box.

### ***Example***

```
:ATTRNAME=init machine comp
:ATTRTYPE=SELECT
:ATTREMARK=Comp Lft, Rgt, Cancel
:ATTRSEL=N
:ATTRTITLE=Laser Compensation
:ATTRSELSTR=Left
:ATTRSELSTR=Right
:ATTRSELSTR=Cancel
:ATTRDEFAULT=1
:ATTRUSED=1
:ATTREND
```

:ATTRSPACES=

---

## :ATTRSPACES=

### ***Purpose***

Code space flag.

### ***Syntax***

**:ATTRSPACES=YES or NO**

### ***Comments***

This command allows spaces in the code output even when the global :SPACE=FALSE is set in the ? file.

### ***Example***

```
:ATTRNAME=TOOL COMMENT
:ATTRTYPE=POST
:ATTREMARK=
:CODETYPE=FORMAT
:WORD_ADDRESS_BEH= | (
:LEFT_PLACES=0
:RIGHT_PLACES=0
:UNITFLAG=NON_CONVERT
:ATTRSPACES=YES
:MODAL=YES
:ATTRUSED=1
:ATTREND
```



---

# :CODETYPE=

## Purpose

Define codeblock type.

## Syntax

```
:CODETYPE=FORMAT
:CODETYPE=SELECT
:CODETYPE=SELECT_FORMAT
:CODETYPE=HARDCODE
```

## Comments

<i>Parameter</i>	<i>Description</i>
FORMAT	Defines how you want your output to be
SELECT	defines a selectable output.
SELECT_FORM	Defines two variables. The first is a select and the second
AT	is a format type.
HARDCODE	Defines hardcoded output.

## Example

```
:ATTRNAME=F
:ATTRTYPE=POST
:ATTRVTYPE=DECIMAL
:ATTREMARK=Feedrate IPM/MPM
:CODETYPE=FORMAT
:ATTRCFUNC=CALC_REG_F_(MACH,REG_E,F)
:WORD_ADDRESS_BE=|F
:LEFT_PLACES=3
:RIGHT_PLACES=4
:CANNOT_BE_SIGNED
:MODAL=YES
:ATTREND

:ATTRNAME=DEBUG
:ATTRTYPE=POST
:ATTRVTYPE=INTEGER
:ATTREMARK=Debug
:CODETYPE=SELECT
:SELECT=1
:CODE=| | | |Line|Move
:SELECT=2
:CPDE=| | | |Arc|Move
:ATTRUSED=1
:ATTREND
```

:CODETYPE=

```
:ATTRNAME=F
:ATTRTYPE=POST
:ATTRVTYPE=DECIMAL
:ATTREMARK=Feedrate IPM/MPM
:CODETYPE=SELECT_FORMAT
:ATTRCFUNC=CALC_REG_F_(MACH,REG_E,F)
:VAR=OPR_FEED_TYPE
:SELECT=1
:WORD_ADDRESS_BE= | F |
:VARB=F
:LEFT_PLACES=2
:RIGHT_PLACES=4
:CANNOT_BE_SIGNED
:MODAL=YES
:SELECT=2
:WORD_ADDRESS_BE= | F |
:VARB=F
:LEFT_PLACES=3
:RIGHT_PLACES=2
:CANNOT_BE_SIGNED
:MODAL=YES
:ATTREND

:ATTRNAME=BLOCK DELETE
:ATTRTYPE=POST
:ATTREMARK=
:CODETYPE=HARDCODE
:CODE=/
:ATTREND
```

---

## :CODE=

### ***Purpose***

Define code for code block.

### ***Syntax***

**:CODE=hardcode**

### ***Comments***

This command allows you to define a constant, not a variable.

### ***Example***

```
:ATTRNAME=BLOCK DELETE
:ATTRTYPE=POST
:ATTREMARK=
:CODETYPE=HARDCODE
:CODE= /
:ATTREND
```

:WORD\_ADDRESS\_BEF=

---

## :WORD\_ADDRESS\_BEF=

### ***Purpose***

Word address before variable.

### ***Syntax***

**:WORD\_ADDRESS\_BEF=output**

### ***Comments***

This command allows you to define what to output in a FORMAT type attribute before the number has been output. Use the pipe (|) to put a space in the output.

### ***Example***

```
:ATTRNAME=PART NAME
:ATTRTYPE=POST
:ATTREMARK=
:CODETYPE=FORMAT
::WORD_ADDRESS_BEF= | ( | PART | NAME=
:CAR=PART NAME
:WORD_ADDRESS_AFT= | )
:LEFT_PLACES=0
:RIGHT_PLACES=0
:UNITFLAG=NON_CONVERT
:ATTRUSED=1
:ATTREND
```

---

## :WORD\_ADDRESS\_AFT=

### ***Purpose***

Word address after variable.

### ***Syntax***

**:WORD\_ADDRESS\_BEF=output**

### ***Comments***

This command allows you to define what to output in a FORMAT type attribute after the number has been output. Use the pipe (|) to put a space in the output.

### ***Example***

```
:ATTRNAME=PART NAME
:ATTRTYPE=POST
:ATTREMARK=
:CODETYPE=FORMAT
:WORD_ADDRESS_BEF= ( | PART | NAME=
:CAR=PART NAME
::WORD_ADDRESS_AFT= | )
:LEFT_PLACES=0
:RIGHT_PLACES=0
:UNITFLAG=NON_CONVERT
:ATTRUSED=1
:ATTREND
```

:LEFT\_PLACES=

---

## :LEFT\_PLACES=

### ***Purpose***

Number of places to the left of the implied decimal.

### ***Syntax***

**:LEFT\_PLACES=value**

### ***Comments***

This command allows you to override the global G\_LEFT\_PLACES command for this attribute only.

### ***Example***

```
:ATTRNAME=F
:ATTRTYPE=POST
:ATTRVTYPE=DECIMAL
:ATTREMARK=Feedrate IPM/MPM
:CODETYPE=FORMAT
:ATTRCFUNC=CALC_REG_F_(MACH,REG_E,F)
:WORD_ADDRESS_BE=|F
:LEFT_PLACES=3
:RIGHT_PLACES=4
:CANNOT_BE_SIGNED
:MODAL=YES
:ATTREND
```

---

## :RIGHT\_PLACES=

### ***Purpose***

Number of places to the right of the implied decimal.

### ***Syntax***

**:RIGHT\_PLACES=value**

### ***Comments***

This command allows you to override the global G\_RIGHT\_PLACES command for this attribute only.

### ***Example***

```
:ATTRNAME=F
:ATTRTYPE=POST
:ATTRVTYPE=DECIMAL
:ATTREMARK=Feedrate IPM/MPM
:CODETYPE=FORMAT
:ATTRCFUNC=CALC_REG_F (MACH, REG_E, F)
:WORD_ADDRESS_BEF= | F
:LEFT_PLACES=3
:RIGHT_PLACES=4
:CANNOT_BE_SIGNED
:MODAL=YES
:ATTREND
```

:CANNOT\_BE\_DECIMAL

---

## :CANNOT\_BE\_DECIMAL

### ***Purpose***

Forces integer output.

### ***Syntax***

**:CANNOT\_BE\_DECIMAL**

### ***Comments***

This command allows you to override the global DECIMAL=TRUE command for this attribute only.

### ***Example***

```
:ATTRNAME=N
:ATTRTYPE=POST
:ATTRVTYPE=INTEGER
:ATTREMARK=Sequence Number
:CODETYPE=FORMAT
:ATTRCFUNC=CALC_REG_N (SEQ, MAX_SEQUENCE)
:WORD_ADDRESS_BE= N
:VAR=SEQ
:LEFT_PLACES=4
:RIGHT_PLACES=0
:CANNOT_BE_DECIMAL
:UNITFLAG=NON_CONVERT
:ATTREND
```



---

## :CANNOT\_BE\_LEADING

### ***Purpose***

Global leading format flag.

### ***Syntax***

**:CANNOT\_BE\_LEADING**

### ***Comments***

This command allows you to override the global :LEADING=TRUE command for this attribute only. No leading zeros will be output.

### ***Example***

```
:ATTRNAME=TIME HOURS
:ATTRTYPE=POST
:ATTRVTYPE=INTEGER
:ATTREMARK=Time in Hours
:CODETYPE=FORMAT
:ATTRCFUNC=CALC_REG_N (SEQ, MAX_SEQUENCE)
:WORD_ADDRESS_BEF= | ESTIMATED | MACHINE | TIME=
:WORD_ADDRESS_AFT= | HRS. | |
:LEFT_PLACES=3
:RIGHT_PLACES=04
:CANNOT_BE_LEADING
:UNITFLAG=NON_CONVERT
:ATTREND
```

:CANNOT\_BE\_TRAILING

---

## :CANNOT\_BE\_TRAILING

### ***Purpose***

Global trailing format flag.

### ***Syntax***

**:CANNOT\_BE\_TRAILING**

### ***Comments***

This command allows you to override the global :TRAILING=TRUE command for this attribute only. No trailing zeros will be output.

### ***Example***

```
:ATTRNAME=TIME HOURS
:ATTRTYPE=POST
:ATTRVTYPE=INTEGER
:ATTREMARK=Time in Hours
:CODETYPE=FORMAT
:ATTRCFUNC=CALC_REG_N (SEQ, MAX_SEQUENCE)
:WORD_ADDRESS_BEF= | ESTIMATED | MACHINE | TIME=
:WORD_ADDRESS_AFT= | HRS. | |
:LEFT_PLACES=3
:RIGHT_PLACES=04
:CANNOT_BE_TRAILING
:UNITFLAG=NON_CONVERT
:ATTREND
```

---

## :CANNOT\_BE\_SIGNED

### ***Purpose***

Signed format definition.

### ***Syntax***

**:CANNOT\_BE\_SIGNED**

### ***Comments***

This command prevents the attribute from outputting a + or - sign to the output field. Only a positive number will be output.

### ***Example***

```
:ATTRNAME=TIME HOURS
:ATTRTYPE=POST
:ATTRVTYPE=INTEGER
:ATTREMARK=Time in Hours
:CODETYPE=FORMAT
:ATTRCFUNC=CALC_REG_N (SEQ, MAX_SEQUENCE)
:WORD_ADDRESS_BEF= | ESTIMATED | MACHINE | TIME=
:WORD_ADDRESS_AFT= | HRS. | |
:LEFT_PLACES=3
:RIGHT_PLACES=04
:CANNOT_BE_SIGNED
:UNITFLAG=NON_CONVERT
:ATTREND
```

:MUST\_BE\_DECIMAL

---

## :MUST\_BE\_DECIMAL

### ***Purpose***

Global decimal format flag.

### ***Syntax***

**:MUST\_BE\_DECIMAL**

### ***Comments***

This command sets the output to always have a decimal even if the output equals zero.

### ***Example***

```
:ATTRNAME=F
:ATTRTYPE=POST
:ATTRVTYPE=DECIMAL
:ATTREMARK=Feedrate IPM/MPM
:CODETYPE=FORMAT
:ATTRCFUNC=CALC_REG_F_(MACH,REG_E,F)
:WORD_ADDRESS_BE=|F
:LEFT_PLACES=3
:RIGHT_PLACES=4
:MUST_BE_DECIMAL
:MODAL=YES
:ATTREND
```

---

## :MUST\_BE\_LEADING

### ***Purpose***

Global leading format flag.

### ***Syntax***

**:MUST\_BE\_LEADING**

### ***Comments***

This command overrides the global :LEADING=FALSE command and always outputs leading zeros.

### ***Example***

```
:ATTRNAME=F
:ATTRTYPE=POST
:ATTRVTYPE=DECIMAL
:ATTREMARK=Feedrate IPM/MPM
:CODETYPE=FORMAT
:ATTRCFUNC=CALC_REG_F_(MACH,REG_E,F)
:WORD_ADDRESS_BEF=|F
:LEFT_PLACES=3
:RIGHT_PLACES=4
:MUST_BE_LEADING
:MODAL=YES
:ATTREND
```

:MUST\_BE\_TRAILING

---

## :MUST\_BE\_TRAILING

### ***Purpose***

Global trailing format flag.

### ***Syntax***

**:MUST\_BE\_TRAILING**

### ***Comments***

This command overrides the global `:TRAILING=FALSE` command and always outputs trailing zeros.

### ***Example***

```
:ATTRNAME=F
:ATTRTYPE=POST
:ATTRVTYPE=DECIMAL
:ATTREMARK=Feedrate IPM/MPM
:CODETYPE=FORMAT
:ATTRCFUNC=CALC_REG_F_(MACH,REG_E,F)
:WORD_ADDRESS_BE=|F
:LEFT_PLACES=3
:RIGHT_PLACES=4
:MUST_BE_TRAILING
:MODAL=YES
:ATTREND
```

---

## :MUST\_BE\_SIGNED

### ***Purpose***

Signed format definition.

### ***Syntax***

**:MUST\_BE\_SIGNED**

### ***Comments***

This command forces the attribute to output a + or - sign to the output file.

### ***Example***

```
:ATTRNAME=F
:ATTRTYPE=POST
:ATTRVTYPE=DECIMAL
:ATTREMARK=Feedrate IPM/MPM
:CODETYPE=FORMAT
:ATTRCFUNC=CALC_REG_F_(MACH,REG_E,F)
:WORD_ADDRESS_BE=|F
:LEFT_PLACES=3
:RIGHT_PLACES=4
:MUST_BE_SIGNED
:MODAL=YES
:ATTREND
```

:MUST\_BE\_LEADING\_SPACES

---

## :MUST\_BE\_LEADING\_SPACES

### ***Purpose***

Global leading spaces format flag.

### ***Syntax***

**:MUST\_BE\_LEADING\_SPACES**

### ***Comments***

This command forces leading spaces in the output file.

If G\_LEFT\_SPACES=3 and the number you output is -1., then you get a minus sign, one space, then 1. (- 1.). If the number output is 1., you will get two spaces, then 1. ( 1.).

This command can be used for controllers where each space means something.



---

## :MUST\_BE\_TRAILING\_SPACES

### ***Purpose***

Global trailing spaces format flag.

### ***Syntax***

**:MUST\_BE\_TRAILING\_SPACES**

### ***Comments***

This command forces trailing spaces in the output file.

If G\_RIGHT\_SPACES=4 and the number you output is -1., then you get a minus sign, 1., then four spaces (-1. ). If the number output is 1.1, you will get the number and three spaces, then 1. (1.1 ).

:MODAL

---

## :MODAL

### ***Purpose***

Modality for code block.

### ***Syntax***

**:MODAL=YES or NO**

### ***Comments***

If MODAL=YES, then the attribute does not get output to the output file again until the attribute's value changes.

MODAL=NO forces the attribute to be output to the output file. If this command is not used, the default is NO.

### ***Example***

```
:ATTRNAME=F
:ATTRTYPE=POST
:ATTRVTYPE=DECIMAL
:ATTREMARK=Feedrate IPM/MPM
:CODETYPE=FORMAT
:ATTRCFUNC=CALC_REG_F_(MACH,REG_E,F)
:WORD_ADDRESS_BE=|F
:LEFT_PLACES=3
:RIGHT_PLACES=4
:MUST_BE_SIGNED
:MODAL=YES
:ATTREND
```

---

## :UNITFLAG=

### ***Purpose***

Global English/Metric definition.

### ***Syntax***

**:UNITFLAG=CONVERT or NON\_CONVERT**

### ***Comments***

The default for this command is CONVERT. If the part is saved as metric, then the output will be converted to metric. If set to NON\_CONVERT, then the output will always be in inches.

### ***Example***

```
:ATTRNAME=TIME HOURS
:ATTRTYPE=POST
:ATTRVTYPE=INTEGER
:ATTREMARK=Time in Hours
:CODETYPE=FORMAT
:ATTRCFUNC=CALC_REG_N (SEQ, MAX_SEQUENCE)
:WORD_ADDRESS_BEF= | ESTIMATED | MACHINE | TIME=
:WORD_ADDRESS_AFT= | HRS. | |
:LEFT_PLACES=3
:RIGHT_PLACES=04
:CANNOT_BE_SIGNED
::UNITFLAG=NON_CONVERT
:ATTREND
```

:METRIC\_UNITS

---

## :METRIC\_UNITS

### ***Purpose***

Metric unit definition.

### ***Syntax***

```
:METRIC_UNITS=MM  
:METRIC_UNITS=CM  
:METRIC_UNITS=M
```

### ***Comments***

This command converts the output to always be whatever METRIC-UNITS is set to: MM (millimeters), CM (centimeters) or M (meters). The default is MM (millimeters). Some controllers may require a different metric unit.

### ***Example***

```
:ATTRNAME=REDUCED FEED  
:ATTRTYPE=POST  
:ATTRVTYPE=DECIMAL  
:ATTREMARK=Reduced feed rate  
:CODETYPE=FORMAT  
:WORD_ADDRESS_BEf=|F  
:VAR=REDUCED FEED  
:LEFT_PLACES=4  
:RIGHT_PLACES=0  
:MODAL=YES  
:CANNOT_BE_SIGNED  
:CANNOT_BE_DECIMAL  
:MUST_BE_TRAILING  
:METRIC_UNITS=M  
:ATTREND
```

---

## :ATTRFUNC=

### ***Purpose***

Define the function to process the attribute.

### ***Syntax***

**:ATTRFUNC=function**

### ***Comments***

This command allows you to call a section you specify. When this attribute is attached to an entity, it will then goto the specified function (:SECTION=) to output the code.

### ***Example***

In this example, the output is M01.

```
:ATTRNAME=optional stop
:ATTRTYPE=DISPLAY
:ATTREMARK=Optional Stop M01
:ATTRSEL=Y
:ATTRTEXT=
:ATTRFUNC=optional_stop
:CODETYPE=HARDCODE
:CODE= | M01
:ATTRUSED=1
:ATTREND

:SECTION=optional_stop
:T:<optional_stop>
```

:ATTRCFUNC=

---

## :ATTRCFUNC=

### ***Purpose***

Define function to process the attribute.

### ***Syntax***

**:ATTRCFUNC=function**

### ***Comments***

This command allows you to call a section you specify. When this attribute is used, it will first goto the specified function (:SECTION=), then return back to the attribute and output the contents.

### ***Example***

```
:ATTRNAME=R
:ATTRTYPE=POST
:ATTRVTYPE=DECIMAL
:ATTREMARK=R Radius
:CODETYPE=FORMAT
:ATTRCFUNC=CALC_RADIUS (R, MACH, REG_R)
:WORD_ADDRESS_BEV= | R
:MODAL=YES
:ATTREND

:SECTION=CALC_RADIUS (RVAL, MACH, REGISTER)
*
*   Arc Radius
*
:C: IF ATTROVERRIDE=YES THEN
:C: RVAL=ATTRDVALUE ELSE RVAL=ARC_RADIUS ENDIF
:C: SETON ()
:C: MACH(REGISTER)=RVAL
:C: IF ARC_INC_ANGLE>HALF_CIRCLE OR
:C: ARC_INC_ANGLE=HALF_CIRCLE
:C: THEN RVAL= (-MACH(REGISTER)) ENDIF
```

---

## :SELECT=

### ***Purpose***

First value for codeblock.

### ***Syntax***

**:SELECT=text**

### ***Comments***

This command allows you have hardcoded selectable output.

### ***Example***

```
:ATTRNAME=G CODE
:ATTRTYPE=POST
:ATTREMARK=G code parameters
:CODETYPE=SELECT
:VAR=MOVE TYPE
:SELECT=LINE
:CODE= | G01
:MODAL=YES
:SELECT=CW ARC
:CODE= | G02
:MODAL=YES
:SELECT=CCW ARC
:CODE= | G03
:MODAL=YES
:SELECT=RAPID
:CODE= | G00
:MODAL=YES
:ATTREND
```

:VAR=

---

## :VAR=

### ***Purpose***

First variable for code block.

### ***Syntax***

**:VAR=variable**

### ***Comments***

This command defines the first variable in a :CODETYPE=SELECT\_FORMAT type attribute.

### ***Example***

```
:ATTRNAME=X
:ATTRTYPE=POST
:ATTRVTYPE=DECIMAL
:ATTREMARK=X End
:CODETYPE=SELECT_FORMAT
:ATTRCFUNC=CALC_ENDPOINT(X,Y_POS,GC,GG,G_GROUP,MACH,PREV,REG_X,
                        RAD_OR_DIAM,SX)

:VAR=CURRENT_MODE
:SELECT=1
:WORD_ADDRESS_BEf= | X
:VERB=X
:MODAL=YES
:SELECT=2
:WORD_ADDRESS_BEf= | U
:VERB=X
:CODE= | G03
:MODAL=YES
:ATTREND
```



---

## :VARB=

### ***Purpose***

Second variable for code block.

### ***Syntax***

**:VARB=variable**

### ***Comments***

This command defines the second variable in a :CODETYPE=SELECT\_FORMAT type attribute.

### ***Example***

```
:ATTRNAME=X
:ATTRTYPE=POST
:ATTRVTYPE=DECIMAL
:ATTREMARK=X End
:CODETYPE=SELECT_FORMAT
:ATTRCFUNC=CALC_ENDPOINT(X,Y_POS,GC,GG,G_GROUP,MACH,PREV,REG_X,
                        RAD_OR_DIAM,SX)

:VAR=CURRENT_MODE
:SELECT=1
:WORD_ADDRESS_BEf= | X
:VERB=X
:MODAL=YES
:SELECT=2
:WORD_ADDRESS_BEf= | U
:VERB=X
:CODE= | G03
:MODAL=YES
:ATTREND
```

:ATTRVCNT=

---

## :ATTRVCNT=

### ***Purpose***

Defines attribute as an array.

### ***Syntax***

**:ATTRVCNT=value**

### ***Comments***

This command defines the attribute as an array and the value indicates the size of the array.

### ***Example***

```
:ATTRNAME=GC  
:ATTRTYPE=POST  
:ATTRVTYPE=INTEGER  
:ATTREMARK=G Codes  
:ATTREND
```

---

## :COLUMN=

### ***Purpose***

Position of code output.

### ***Syntax***

**:COLUMN=value**

### ***Comments***

This command determines the position of the code output. :COLUMN=9 positions the output in the 10<sup>th</sup> place from the left. This command is used for some older controllers that require fielded output where every space means something.

### ***Example***

```
:ATTRNAME=S
:ATTRTYPE=POST
:ATTRVTYPE=INTEGER
:ATTREMARK=Spindle RPM
:CODETYPE=FORMAT
:ATTRCFUNC=CALC_INT_REGISTER(S,MACH,OPR_SPEED,REG_S)
:WORD_ADDRESS_BEf=SP=
:WORD_ADDRESS_AfT=|RPM
:LEFT_PLACES=4
:RIGHT_PLACES=0
:COLUMN=9
:RIGHT_JUST=4
:ATTREND
```

```
N001      S1000
```

:LEFT\_JUST=

---

## :LEFT\_JUST=

### ***Purpose***

Left justification of the code output.

### ***Syntax***

**:LEFT\_JUST=value**

### ***Comments***

This command determines the position of the code output from where you were last, starting from the left.

### ***Example***

```
:ATTRNAME=S
:ATTRTYPE=POST
:ATTRVTYPE=INTEGER
:ATTREMARK=Spindle RPM
:CODETYPE=FORMAT
:ATTRCFUNC=CALC_INT_REGISTER(S,MACH,OPR_SPEED,REG_S)
:WORD_ADDRESS_BEf=SP=
:WORD_ADDRESS_AfT=| RPM
:LEFT_PLACES=4
:RIGHT_PLACES=0
:LEFT_JUST=7
:ATTREND
```

N001S1000 M03

---

## :RIGHT\_JUST=

### ***Purpose***

Right justification of the code output.

### ***Syntax***

**:RIGHT\_JUST=value**

### ***Comments***

This command determines the position of the code output from where you were last, starting from the right.

### ***Example***

```
:ATTRNAME=S
:ATTRTYPE=POST
:ATTRVTYPE=INTEGER
:ATTREMARK=Spindle RPM
:CODETYPE=FORMAT
:ATTRCFUNC=CALC_INT_REGISTER(S,MACH,OPR_SPEED,REG_S)
:WORD_ADDRESS_BEf=SP=
:WORD_ADDRESS_AfT=| RPM
:LEFT_PLACES=4
:RIGHT_PLACES=0
:RIGHT_JUST=7
:ATTREND
```

N001 S1000M03

:ATTRLNG=

---

## :ATTRLNG=

### ***Purpose***

Add attribute to language file.

### ***Syntax***

**:ATTRLNG=YES or NO**

### ***Comments***

This command defines whether the attribute will be added to the language file (.LNG) when the post is compiled. The compiler automatically knows which attributes to add to the language file and this command is rarely used.

### ***Example***

```
:ATTRNAME=R
:ATTRTYPE=POST
:ATTRVTYPE=DECIMAL
:ATTREMARK=R Radius
:CODETYPE=FORMAT
:ATTRCFUNC=CALC_RADIUS (R, MACH, REG_R)
:WORD_ADDRESS_BEf= | R
:ATTRLNG=YES
:MODAL=YES
:ATTREND
```

---

## **Chapter 5 Miscellaneous Commands**

---

:ATTRID

---

## :ATTRID

### ***Purpose***

Attribute ID number.

### ***Syntax***

**:ATTRID=501**

### ***Comments***

This command determines the ID number for any attribute that is defined in the MASTER.ATR file.

### ***Example***

```
:ATTRNAME=MACHINE NAME  
:ATTRTYPE=DESCRIPTOR  
:ATTRVTYPE=CHARACTER  
:ATTRID=501  
:ATTREND
```



---

## :IDHIGH

### ***Purpose***

Highest attribute ID in MASTER.ATR file.

### ***Syntax***

**:IDHIGH=17501**

### ***Comments***

This command determines the highest ID number used in the MASTER.ATR file. This command should be put at the beginning of the MASTER.ATR file.

### ***Example***

None.

:ATTRMACHINE

---

## :ATTRMACHINE

### ***Purpose***

Set of machine-dependent attributes.

### ***Syntax***

**:ATTRMACHINE=PUNCH**

### ***Comments***

After this command is put in the post, any attachable attribute that is defined after it will be assumed to be the system you entered in this command.

### ***Example***

None.

---

## :SECTION

### ***Purpose***

Start function definitions.

### ***Syntax***

**:SECTION=CALC\_REMOVE\_OFFSET**

### ***Comments***

This command determines the name of a section and what type of section it is. If the :SECTION=CALC\_? section equals starts with a CALC, then it is a calculation section; otherwise, it is a template section.

### ***Example***

```
:SECTION=CALC_REMOVE_OFFSET
:C: IF DEFINING_MACRO=YES AND JUST_STARTED_MACRO=1
:C: THEN SAV_MODE=CURRENT_MODE JUST_STARTED_MACRO=2
:C: ENDIF
:C: IF OFFSET_RESIDENT=NO THEN RETURN ENDIF
:C: X_OFFSET=(LAST_X-SYS_X_OFFSET)
:C: Y_OFFSET=(LAST_Y-SYS_Y_OFFSET)
:C: CALL(OFFSET_PART_PUNCH)
:C: OFFSET_RESIDENT=NO

:SECTION=OFFSET_PART_PUNCH
:T:<SEQ><ABS_PRESET><X_OFFSET><Y_OFFSET><EOL>
```

:OPERID

---

## :OPERID

### *Purpose*

Mill and Lathe operation ID name.

### *Syntax*

**:OPERID=MILL\_PROFILING**

### *Comments*

This command determines the name of an operation ID. Any OPERLIST question asked after this command will be added to the operation questions in CAD.

Operation List:

DRILLING	MILL_DRILLING	LATHE_DRILLING	EDM_PROFILE
SPOT_DRILLING	MILL_PROFILING	LATHE_PROFILING	EDM_SKIM
PECKING	MILL_LACE	LATHE_ROUGHING	EDM_CORE
TAPPING	MILL_POCKET	LATHE_GROOVING	EDM_MACROS
BORING	MILL_MISC	LATHE_THREADING	
HIGH_SPEED_PECKING	MILL_SPECIAL	LATHE_MISC	
VARIABLE_PECKING	MILL_MACRO	LATHE_SPECIAL	
REVERSE_TAPPING	MILL_UV_CUT	LATHE_CUTOFF	
REAMING	MILL_SLICE_CUT		
REAMING_DWELL	MILL_ROUGH_CUT		
BORE_DWELL	MILL_CURVE_CUT		
BACK_BORING	MILL_TOPO_CUT		
FINE_BORING	MILL_FREEFORM_CUT		
	MILL_PENCIL_CUT		

### *Example*

```
:OPERID=MILL_PROFILING  
:OPERLIST=machine compensation  
:OPERLIST=abs inc  
:OPERLIST=work coord  
:OPERLIST=coolant  
:OPEREND
```

---

## :OPERSUB

### ***Purpose***

Mill drilling operation ID name.

### ***Syntax***

**:OPERSUB=DRILLING**

### ***Comments***

This command determines the name of a drilling operation ID. Any OPERLIST questions asked after this command will be added to the operation questions in CAD. This command is used in a drilling operation only. It should be placed after the

:OPERID=MILL\_DRILLING.

### ***Example***

```
:OPERID=MILL_DRILLING
:OPERSUB=DRILLING
:OPERLIST=abs inc
:OPERLIST=work coord
:OPERLIST=coolant
:OPEREND
```

:OPERLIST

---

## :OPERLIST

### ***Purpose***

Mill and Lathe operation question.

### ***Syntax***

**:OPERLIST=abs inc**

### ***Comments***

This command determines the name of an attribute question that is to be asked in CAD.

### ***Example***

```
:OPERID=MILL_PROFILING  
:OPERLIST=machine compensation  
:OPERLIST=abs inc  
:OPERLIST=work coord  
:OPERLIST=coolant  
:OPEREND
```

---

## :OPEREND

### ***Purpose***

Mill and Lathe operation definition end.

### ***Syntax***

**:OPEREND**

### ***Comments***

This command determines the end of an operation questions.

### ***Example***

```
:OPERID=MILL_PROFILING  
:OPERLIST=machine compensation  
:OPERLIST=abs inc  
:OPERLIST=work coord  
:OPERLIST=coolant  
:OPEREND
```

:DEFINE

---

## :DEFINE

### ***Purpose***

Hardcoded definitions.

### ***Syntax***

```
:DEFINE RAPID_Z_UP=8
```

### ***Comments***

This command sets a define variable equal to a value. In the example below  
RAPID\_Z\_UP=8 sets the value to eight. This command needs to be set at the beginning of  
the attribute list in the library.

### ***Example***

```
:DEFINE RAPID_Z_UP=8
```



---

## :LIBRARY

### ***Purpose***

Library definition.

### ***Syntax***

**:LIBRARY=\MILL\LIBRARY\GENERAL.LIB**

### ***Comments***

This command defines the name and location of the post library. The example below shows that the GENERAL.LIB file is located in the \MILL\LIBRARY directory.

### ***Example***

**:LIBRARY=\MILL\LIBRARY\GENERAL.LIB**

:INCLUDE

---

## **:INCLUDE**

### ***Purpose***

Include definition.

### ***Syntax***

**: INCLUDE=\MILL\TOOLS\GENERAL.T32**

### ***Comments***

This command defines the name and location of another file to be used when compiling.

### ***Example***

: INCLUDE=\MILL\TOOLS\GENERAL.T32

---

## FLAGGED(variable, bit value)

### ***Purpose***

CAMWorks 2007 and later: This command is used in the Turn system only for the first approach and last retract moves of an operation. It stores multiple bits of information about those moves and depending on the flag it is looking for, it will pass a True or False value.

### ***Syntax***

**FLAGGED(CAM\_MOVE\_FLAG, CAM\_APPROACH)**

### ***Comments***

CAM\_MOVE\_FLAG is a multiple bit variable and CAM\_APPROACH is the value it will look for. The bit value of CAM\_APPROACH is 2. What the command FLAGGED is going to return is a True or False value. In the CAM\_MOVE\_FLAG variable is CAM\_APPROACH's bit value equal to True or False.

### ***Example***

```
:C: IF FLAGGED(CAM_MOVE_FLAG,CAM_APPROACH)=TRUE THEN
:C:   IF FLAGGED(CAM_MOVE_FLAG, CAM_MOVE_X)=TRUE AND
:C:   FLAGGED(CAM_MOVE_FLAG, CAM_MOVE_Z)=TRUE THEN
:C:     CALL(RAPID_MOVE_LATHE)
:C:     RETURN
:C:   ELSE
:C:     IF FLAGGED(CAM_MOVE_FLAG,CAM_MOVE_X)=TRUE THEN
:C:       CALL(RAPID_MOVE_LATHE_X)
:C:       RETURN
:C:     ELSE
:C:       IF FLAGGED(CAM_MOVE_FLAG,CAM_MOVE_Z)=TRUE THEN
:C:         CALL(RAPID_MOVE_LATHE_Z)
:C:         RETURN
:C:       ENDIF
:C:     ENDIF
:C:   ENDIF
:C: ENDIF
```

FLAGGED(variable, bit value)

---

## **Chapter 6 System Tool Commands**

---

:STATION\_NUM

---

## :STATION\_NUM

### ***Purpose***

Tool station number,

### ***Syntax***

**: STATION\_NUM=01**

### ***Comments***

This command defines the tool number.

### ***Example***

```
STATION_NUM=01  
AUTOINDEX=NO  
KEYSIZE=4  
KEYED=YES  
LARGEDIAM=3.000000  
XWDEAD=8.000000  
YHDEAD=4.000000  
XLORANGE=-1.000000  
YLORANGE=-1.000000  
XHIRANGE=50.000000  
YHIRANGE=30.000000
```

---

# :AUTOINDEX

## ***Purpose***

Tool station number.

## ***Syntax***

**:AUTOINDEX=YES or NO**

## ***Comments***

This command defines whether this is auto indexable or not. This should be set to NO if not a punch. This should be set to YES if punch station is auto indexable.

## ***Example***

```
STATION_NUM=01
AUTOINDEX=NO
KEYSIZE=4
KEYED=YES
LARGEDIAM=3.000000
XWDEAD=8.000000
YHDEAD=4.000000
XLORANGE=-1.000000
YLORANGE=-1.000000
XHRANGE=50.000000
YHIRANGE=30.000000
```

:KEYSIZE

---

## :KEYSIZE

### ***Purpose***

Tool station key size.

### ***Syntax***

**:KEYSIZE=4**

### ***Comments***

This command defines what size the key is:

- 1 = .5
- 2 = 1.25
- 3 = 2.0
- 4 = 3.5
- 5 = 4.5
- 6 = greater than 4.5

If not a punch tool, then set KEYSIZE=5

### ***Example***

```
STATION_NUM=01
AUTOINDEX=NO
KEYSIZE=4
KEYED=YES
LARGEDIAM=3.000000
XWDEAD=8.000000
YHDEAD=4.000000
XLORANGE=-1.000000
YLORANGE=-1.000000
XHIRANGE=50.000000
YHIRANGE=30.000000
```



---

## :KEYED

### ***Purpose***

Tool station keyed.

### ***Syntax***

**:KEYED=YES or NO**

### ***Comments***

This command defines if the tool is keyed. Normally, you would set this to YES.

### ***Example***

```
STATION_NUM=01
AUTOINDEX=NO
KEYSIZE=4
KEYED=YES
LARGEDIAM=3.000000
XWDEAD=8.000000
YHDEAD=4.000000
XLORANGE=-1.000000
YLORANGE=-1.000000
XHIRANGE=50.000000
YHIRANGE=30.000000
```

:LARGEDIAM

---

## :LARGEDIAM

### ***Purpose***

Tool station largest diameter used.

### ***Syntax***

**:LARGEDIAM=3.000000**

### ***Comments***

This command defines how big a diameter tool you can use in this station.

### ***Example***

```
STATION_NUM=01
AUTOINDEX=NO
KEYSIZE=4
KEYED=YES
LARGEDIAM=3.000000
XWDEAD=8.000000
YHDEAD=4.000000
XLORANGE=-1.000000
YLORANGE=-1.000000
XHIRANGE=50.000000
YHIRANGE=30.000000
```

---

## :XWDEAD

### ***Purpose***

Define X dead zone.

### ***Syntax***

**:XWDEAD=8.000000**

### ***Comments***

This command defines how big the "X" width dead zone of a punch clamp is. If the system is not a punch, laser or plasma then it does not matter what you set this to.

### ***Example***

```
STATION_NUM=01
AUTOINDEX=NO
KEYSIZE=4
KEYED=YES
LARGEDIAM=3.000000
XWDEAD=8.000000
YHDEAD=4.000000
XLORANGE=-1.000000
YLORANGE=-1.000000
XHRANGE=50.000000
YHIRANGE=30.000000
```

:YHDEAD

---

## :YHDEAD

### ***Purpose***

Define Y dead zone.

### ***Syntax***

**:YHDEAD=4.000000**

### ***Comments***

This command defines how big the "Y" height dead zone of a punch clamp is. If the system is not a punch, laser or plasma, you can set this to anything.

### ***Example***

```
STATION_NUM=01
AUTOINDEX=NO
KEYSIZE=4
KEYED=YES
LARGEDIAM=3.000000
XWDEAD=8.000000
YHDEAD=4.000000
XLORANGE=-1.000000
YLORANGE=-1.000000
XHRANGE=50.000000
YHIRANGE=30.000000
```

---

# :XLORANGE

## ***Purpose***

Define X travel low range.

## ***Syntax***

**:XLORANGE=1.000000**

## ***Comments***

This command defines how far to the left side of the table you can travel or how far in the X minus direction you can travel. If the system is not a punch, plasma or laser, then you can set this to anything.

## ***Example***

```
STATION_NUM=01
AUTOINDEX=NO
KEYSIZE=4
KEYED=YES
LARGEDIAM=3.000000
XWDEAD=8.000000
YHDEAD=4.000000
XLORANGE=-1.000000
YLORANGE=-1.000000
XHRANGE=50.000000
YHIRANGE=30.000000
```

:YLORANGE

---

## :YLORANGE

### ***Purpose***

Define Y travel low range.

### ***Syntax***

**:YLORANGE=1.000000**

### ***Comments***

This command defines how far in the Y minus direction you can travel. If the system is not a punch, plasma or laser, then you can set this to anything.

### ***Example***

```
STATION_NUM=01
AUTOINDEX=NO
KEYSIZE=4
KEYED=YES
LARGEDIAM=3.000000
XWDEAD=8.000000
YHDEAD=4.000000
XLORANGE=-1.000000
YLORANGE=-1.000000
XHIRANGE=50.000000
YHIRANGE=30.000000
```

---

# :XHIRANGE

## ***Purpose***

Define X travel high range.

## ***Syntax***

**:XHIRANGE=50.000000**

## ***Comments***

This command defines how far in the X plus direction you can travel. If the system is not a punch, plasma or laser, then you can set this to anything.

## ***Example***

```
STATION_NUM=01
AUTOINDEX=NO
KEYSIZE=4
KEYED=YES
LARGEDIAM=3.000000
XWDEAD=8.000000
YHDEAD=4.000000
XLORANGE=-1.000000
YLORANGE=-1.000000
XHIRANGE=50.000000
YHIRANGE=30.000000
```

:YHIRANGE

---

## :YHIRANGE

### ***Purpose***

Define Y travel high range.

### ***Syntax***

**:YHIRANGE=50.000000**

### ***Comments***

This command defines how far in the Y plus direction you can travel. If the system is not a punch, plasma or laser, then you can set this to anything.

### ***Example***

```
STATION_NUM=01
AUTOINDEX=NO
KEYSIZE=4
KEYED=YES
LARGEDIAM=3.000000
XWDEAD=8.000000
YHDEAD=4.000000
XLORANGE=-1.000000
YLORANGE=-1.000000
XHRANGE=50.000000
YHIRANGE=30.000000
```



---

## **Chapter 7 System Header Commands**

---

:BCL\_FORMAT

---

## :BCL\_FORMAT

### ***Purpose***

Output BCL format.

### ***Syntax***

**:BCL\_FORMAT=TRUE or FALSE**

### ***Comments***

This command determines whether the output is BCL or not. BCL stands for Binary Cutter Location. The file is a fixed binary output called \*.BCL.

---

# :SYSTEM

## ***Purpose***

System type.

## ***Syntax***

**: SYSTEM=PUNCH**

## ***Comments***

This command determines what system you are in:

PLASMA

PUNCH

PLASMA / PUNCH

LASER

EDM

LATHE

MILL

LATHE / MILL

LATHE4AX

:LEADING

---

## :LEADING

### ***Purpose***

Global leading flag.

### ***Syntax***

**:LEADING=TRUE or FALSE**

### ***Comments***

This command determines if the code output has leading zeros or not.

---

## :TRAILING

### ***Purpose***

Global trailing flag.

### ***Syntax***

**:TRAILING=TRUE OR FALSE**

### ***Comments***

This command determines if the code output has trailing zeros or not.

:DECIMAL

---

## :DECIMAL

### ***Purpose***

Global decimal flag.

### ***Syntax***

**:DECIMAL=TRUE or FALSE**

### ***Comments***

This command if code output has decimals or not.

---

## :QUAD

### ***Purpose***

Global quadrant arc flag.

### ***Syntax***

**:QUAD=TRUE, FALSE, NO181 or COORD**

### ***Comments***

This command determines if code output has quadrant arcs or not. I

If QUAD=TRUE, a 360 degree arc will take four blocks of code to generate that arc.

If QUAD=FALSE, a 360 degree arc will take only one block of code to generate that arc.

If QUAD=NO181, the code will generate that arc in two moves.

If QUAD=COORD, the code will generate tiny line moves set by the `chord_length` attribute instead of arcs.

:SPACE

---

## :SPACE

### ***Purpose***

Global spaces flag.

### ***Syntax***

**:SPACE=TRUE or FALSE**

### ***Comments***

This command determines if code output has spaces or not.



---

## :ARCS

### ***Purpose***

Global arc flag.

### ***Syntax***

**:ARCS=RADIAL or CENTER**

### ***Comments***

This command determines if arcs will output a radial command or I's and J's.

If set to RADIAL, then arc command output will output R's.

If set to CENTER, then arc command output will output I's and J's.

:METRIC\_SHIFT

---

## :METRIC\_SHIFT

### ***Purpose***

Global metric shift flag.

### ***Syntax***

**:METRIC\_SHIFT=1**

### ***Comments***

This defines the amount of shift in the output when switching to metric.

---

## :G\_LEFT\_PLACES

### ***Purpose***

Global decimal places to the left of the decimal.

### ***Syntax***

**:G\_LEFT\_PLACES=3**

### ***Comments***

This defines the decimal places to the left of the decimal.

:G\_RIGHT\_PLACES

---

## :G\_RIGHT\_PLACES

### ***Purpose***

Global decimal places to the right of the decimal.

### ***Syntax***

**:G\_RIGHT\_PLACES=3**

### ***Comments***

This defines the decimal places to the right of the decimal.

---

## :G\_INT\_LEFT\_PLACES

### ***Purpose***

Global integer places to the left of the decimal.

### ***Syntax***

**:G\_INT\_LEFT\_PLACES=2**

### ***Comments***

This defines the integer places to the left of the decimal.

:INT\_LEADING

---

## :INT\_LEADING

### ***Purpose***

Global integer leading flag.

### ***Syntax***

**:INT\_LEADING=TRUE or FALSE**

### ***Comments***

This command defines if integer has leading zeros or not.

---

## :INT\_TRAILING

### ***Purpose***

Global integer trailing flag.

### ***Syntax***

**:INT\_TRAILING=TRUE or FALSE**

### ***Comments***

This command defines if integer has trailing zeros or not.

:PQCOMP

---

## :PQCOMP

### ***Purpose***

P and Q compensation flag.

### ***Syntax***

**:PQCOMP=TRUE or FALSE**

### ***Comments***

This command is reserved for future use.



---

## :QUALIFIED\_TOOLING

### ***Purpose***

External tool file.

### ***Syntax***

**:QUALIFIED\_TOOLING=\PROCAD\TPPL\TOOLFILE.F6M**

### ***Comments***

This command can be used if you want a specific tool file with tool offset lengths. The file has 5 fields in it.

Field 1 is a record field

Fields 2 and 3 are decimal fields

Field 4 is an integer field

Field 5 is a character field.

Commas are used as delimiters.

### ***Example***

Below is an example of what the file might have in it.

```
*****
**      Z Feed    X Feed    RPM      Comment
**-----
*1,    5,         10,         1200,   1" End Mill   *
*2,    6.375,     12.5,       1100,   2" Ball Nose  *
*****
```

:SINGLE\_MACROS

---

## :SINGLE\_MACROS

### ***Purpose***

Single macro flag.

### ***Syntax***

**:SINGLE\_MACROS=YES or NO**

### ***Comments***

This command defines if post can output single macro calls.

---

## :MIRROR\_MACROS

### ***Purpose***

Mirror macro flag.

### ***Syntax***

**:MIRROR\_MACROS=YES or NO**

### ***Comments***

This command is reserved for future use.

:MACROS\_REDEFINE

---

## :MACROS\_REDEFINE

### ***Purpose***

Redefine macro flag.

### ***Syntax***

**:MACROS\_REDEFINE=YES or NO**

### ***Comments***

This command determines if a macro has to be redefined each time it is called.

---

## :MULT\_MACROS

### ***Purpose***

Multiple macro flag.

### ***Syntax***

**:MULT\_MACROS=YES or NO**

### ***Comments***

This command defines if post can output multiple macro calls.

:LAYOUT\_MACROS

---

## :LAYOUT\_MACROS

### ***Purpose***

Layout of macro's flag.

### ***Syntax***

**:LAYOUT\_MACROS=YES or NO**

### ***Comments***

This command defines if post can output multiple macro calls.

---

## :MACROS\_CALL

### ***Purpose***

Calling single macro flag.

### ***Syntax***

**:MACROS\_CALL=BEFORE or AFTER**

### ***Comments***

This command defines if post will call a single macro before or after the subroutine. Used only if MACROS\_MAIN=DURING.

:MACROS\_MULT

---

## :MACROS\_MULT

### ***Purpose***

Calling multiple macro flag.

### ***Syntax***

**:MACROS\_MULT=BEFORE or AFTER**

### ***Comments***

This command defines if post will call a single macro before or after the subroutine. Used only if MACROS\_MAIN=DURING.



---

## :MACROS\_LAYOUT

### ***Purpose***

Macro layout flag.

### ***Syntax***

**:MACROS\_LAYOUT=BEFORE or AFTER**

### ***Comments***

This command defines if post will call a section called  
CALC\_MULTIPLE\_MACRO\_DEFINE\_PUNCH.

:MACROS\_MAIN

---

## :MACROS\_MAIN

### ***Purpose***

Subroutine call flag.

### ***Syntax***

**:MACROS\_MAIN=BEFORE, DURING or AFTER**

### ***Comments***

This command defines if the subroutine will be called before, after or during the main program.

---

## :MACROS\_OUT

### ***Purpose***

Macro output flag.

### ***Syntax***

**:MACROS\_OUT=CALLED or NESTED**

### ***Comments***

If set to CALLED, then the output will be in the order it was created in CAD.

If set to NESTED, then output will be in the reverse order.

:MACROS\_TAPE

---

## :MACROS\_TAPE

### ***Purpose***

Macro output flag.

### ***Syntax***

**:MACROS\_TAPE=SAME or SEPARATE**

### ***Comments***

If set to SAME, then the output will be in the same file \*.TXT.

If set to SEPARATE, then output of subroutine will be in a file called \*.SUB.

---

## :MACROS\_XYZ

### ***Purpose***

Allow Z axis in macros.

### ***Syntax***

**:MACROS\_XYZ=TRUE or FALSE**

### ***Comments***

This command allows the Z axis to step and repeat in a macro. This can only be done in Mill.

:MACRO\_ROTATE

---

## :MACRO\_ROTATE

### ***Purpose***

Allow macro rotation.

### ***Syntax***

**:MACRO\_ROTATE=TRUE or FALSE**

### ***Comments***

If you set to MACRO\_ROTATE\_X=FALSE, MACRO\_ROTATE\_Y=FALSE and MACRO\_ROTATE\_Z=FALSE, then XY numbers will rotate instead of an angle. This can only be done in Mill.

---

## :MACRO\_ROTATE\_X

### ***Purpose***

Allow X axis rotation.

### ***Syntax***

**:MACRO\_ROTATE\_X=TRUE or FALSE**

### ***Comments***

Allows macro to rotate about the X axis. This can only be done in Mill.

:MACRO\_ROTATE\_Y

---

## :MACRO\_ROTATE\_Y

### ***Purpose***

Allow Y axis rotation.

### ***Syntax***

**:MACRO\_ROTATE\_Y=TRUE or FALSE**

### ***Comments***

Allows macro to rotate about the Y axis. This can only be done in Mill.



---

## :MACRO\_ROTATE\_Z

### ***Purpose***

Allow Z axis rotation.

### ***Syntax***

**:MACRO\_ROTATE\_Z=TRUE or FALSE**

### ***Comments***

Allows macro to rotate about the Z axis. This can only be done in Mill.

:EDM4AXIS

---

## :EDM4AXIS

### ***Purpose***

EDM 4 axis definition.

### ***Syntax***

**:EDM4AXIS=TRUE or FALSE**

### ***Comments***

Defines whether this post can do X,Y,U,V 4-axis output. This can only be done in EDM.

---

## :TAPER

### ***Purpose***

EDM 2 axis taper definition.

### ***Syntax***

**:TAPER=TRUE or FALSE**

### ***Comments***

Defines whether this post can do 2 axis with taper output. This can only be done in EDM.

:ARC\_TO\_ARC

---

## :ARC\_TO\_ARC

### ***Purpose***

EDM 4 axis arc definition.

### ***Syntax***

**:ARC\_TO\_ARC=TRUE or FALSE**

### ***Comments***

Defines whether this post can do 4 axis GO2 or GO3 output if top and bottom surface arcs have equal radii. This can only be done in EDM.

---

## :SHARP\_CORNER

### ***Purpose***

EDM 2 axis corner definition.

### ***Syntax***

**:SHARP\_CORNER=TRUE or FALSE**

### ***Comments***

Defines whether this post can do 2 axis sharp corners. This can only be done in EDM.

:EQUAL\_CORNER

---

## **:EQUAL\_CORNER**

### ***Purpose***

EDM 2 axis corner definition.

### ***Syntax***

**:EQUAL\_CORNER=TRUE or FALSE**

### ***Comments***

Defines whether this post can do 2 axis equal corners. This can only be done in EDM.

---

## **:INDEPENDENT\_CORNER**

### ***Purpose***

EDM 2 axis corner definition.

### ***Syntax***

**:INDEPENDENT\_CORNER=TRUE or FALSE**

### ***Comments***

Defines whether this post can do 2 axis independent corners. This can only be done in EDM.

:CONIC\_CORNER

---

## :CONIC\_CORNER

### ***Purpose***

EDM 2 axis corner definition.

### ***Syntax***

**:CONIC\_CORNER=TRUE or FALSE**

### ***Comments***

Defines whether this post can do 2 axis conic corners. This can only be done in EDM.



---

## :CHAMFER\_CORNER

### ***Purpose***

EDM 2 axis corner definition.

### ***Syntax***

**:CHAMFER\_CORNER=TRUE or FALSE**

### ***Comments***

Defines whether this post can do 2 axis chamfer corners. This can only be done in EDM.

:TAPER\_DURING

---

## :TAPER\_DURING

### ***Purpose***

EDM 2 axis corner definition.

### ***Syntax***

**:TAPER\_DURING=TRUE or FALSE**

### ***Comments***

Defines whether this post can do 2 axis taper during a move. This can only be done in EDM.

---

## **:LOOK\_AHEAD**

### ***Purpose***

EDM look ahead definition.

### ***Syntax***

**:LOOK\_AHEAD=1**

### ***Comments***

Defines how many entities to look ahead for compensation. This can only be done in EDM.

:TAPER\_FILLET

---

## :TAPER\_FILLET

### ***Purpose***

EDM taper fillet definition.

### ***Syntax***

**:TAPER\_FILLET=TRUE or FALSE**

### ***Comments***

Defines if controller can do taper filleting. This can only be done in EDM.

---

## :LIVE\_Y\_AXIS

### ***Purpose***

Lathe/Mill Y axis definition.

### ***Syntax***

**:LIVE\_Y\_AXIS=TRUE or FALSE**

### ***Comments***

Defines if controller can do Y axis moves in Live C post. This can only be done in Lathe/Mill.

:OD\_MILL

---

## :OD\_MILL

### ***Purpose***

Lathe/Mill OD mill definition.

### ***Syntax***

**:OD\_MILL=FREE, FIXED or BOTH**

### ***Comments***

Defines if the rotation axis on OD milling can be free, fixed or both free and fixed. This can only be done in Lathe/Mill.

---

## :OD\_DRILL

### ***Purpose***

Lathe/Mill OD drill definition.

### ***Syntax***

**:OD\_DRILL=FREE, FIXED or BOTH**

### ***Comments***

Defines if the rotation axis on OD drilling can be free, fixed or both free and fixed. This can only be done in Lathe/Mill.

:OD\_ARC

---

## :OD\_ARC

### ***Purpose***

Lathe/Mill OD arc definition.

### ***Syntax***

**:OD\_ARC=FREE, FIXED or BOTH**

### ***Comments***

Defines if the rotation axis on OD arcs can be free, fixed or both free and fixed. This can only be done in Lathe/Mill.



---

## :FACE\_MILL

### ***Purpose***

Lathe/Mill Face mill definition.

### ***Syntax***

**:FACE\_MILL=FREE, FIXED or BOTH**

### ***Comments***

Defines if the rotation axis on face milling can be free, fixed or both free and fixed. This can only be done in Lathe/Mill.

:FACE\_DRILL

---

## :FACE\_DRILL

### ***Purpose***

Lathe/Mill Face drill definition.

### ***Syntax***

**:FACE\_DRILL=FREE, FIXED or BOTH**

### ***Comments***

Defines if the rotation axis on face drilling can be free, fixed or both free and fixed. This can only be done in Lathe/Mill.

---

## :FACE\_ARC

### ***Purpose***

Lathe/Mill Face arc definition.

### ***Syntax***

**:FACE\_ARC=FREE, FIXED or BOTH**

### ***Comments***

Defines if the rotation axis on face arcs can be free, fixed or both free and fixed. This can only be done in Lathe/Mill.

:LATHE

---

## :LATHE

### ***Purpose***

Mill 4<sup>th</sup> axis lathe definition.

### ***Syntax***

**:LATHE=TRUE or FALSE**

### ***Comments***

This command should be set to FALSE if doing a mill 4<sup>th</sup> axis cutting post.

---

## **:5AXIS\_MILLING**

### ***Purpose***

Allows access to 5 axis milling.

### ***Syntax***

**:5AXIS\_MILLING=TRUE or FALSE**

### ***Comments***

This command should be set to FALSE if not using 5 axis work. This can only be done in 3D CAD.

:4AXIS\_X\_MILLING

---

## :4AXIS\_X\_MILLING

### ***Purpose***

Allows access to 4 axis milling.

### ***Syntax***

**:4AXIS\_X\_MILLING=TRUE or FALSE**

### ***Comments***

This command should be set to FALSE if not using 4 axis work.

This command should be set to TRUE if rotating about the X axis.

This can only be done in 3D CAD.

---

## :4AXIS\_Y\_MILLING

### ***Purpose***

Allows access to 4 axis milling.

### ***Syntax***

**:4AXIS\_Y\_MILLING=TRUE or FALSE**

### ***Comments***

This command should be set to FALSE if not using 4 axis work.

This command should be set to TRUE if rotating about the Y axis.

This can only be done in 3D CAD.

:HELICAL

---

## :HELICAL

### ***Purpose***

This command allows helical arc moves at start of profile.

### ***Syntax***

**:HELICAL=TRUE or FALSE**

### ***Comments***

This command is reserved for future use.



---

## :MAXIMUM\_LINE

### ***Purpose***

This command lets you set the maximum line length of \*.TXT files.

### ***Syntax***

**:MAXIMUM\_LINE=100**

### ***Comments***

The system default for this command is 100. If 100 is not a problem, then you can leave this command out of your source.

:USE\_SPECIAL\_TOOL\_TYPE

---

## :USE\_SPECIAL\_TOOL\_TYPE

### ***Purpose***

This command allows special tooling.

### ***Syntax***

**:USE\_SPECIAL\_TOOL\_TYPE=TRUE or FALSE**

### ***Comments***

If not using special tool type, set this command to FALSE.

If this command is set to TRUE, a special tool dialog box displays.

---

## :SLOW\_INDEXER

### ***Purpose***

Allows optimized autoindex output.

### ***Syntax***

**:SLOW\_INDEXER=TRUE or FALSE**

### ***Comments***

If this command should be set to TRUE, the system will generate optimized autoindex angles. It will punch all entities with the same autoindex angle, then go to the next autoindex angle, then the next until finished.

:SLOW\_INDEXER

---

## **Chapter 8 System Variables**

---

## System Variables

<i>Variable</i>	<i>Type</i>	<i>Usage</i>
<b>NO</b>	CHARACTER	Its value equals 0.
<b>YES</b>	CHARACTER	Its value equals 1.
<b>ABS_X_END</b> <b>ABS_Y_END</b> <b>ABS_Z_END</b> <b>ABS_C_END</b> <b>ABS_U_END</b> <b>ABS_V_END</b> <b>ABS_W_END</b>	METRIC DECIMAL	Absolute X,Y,Z,C end point of move. EDM: Absolute U,V secondary surface end point of move. Lathe: Absolute U,W front turret end point of move.
<b>N_ABS_X_END</b> <b>N_ABS_Y_END</b> <b>N_ABS_Z_END</b> <b>N_ABS_C_END</b> <b>N_ABS_U_END</b> <b>N_ABS_V_END</b> <b>N_ABS_W_END</b>		Absolute X,Y,Z,C end point of next move. EDM: Absolute U,V secondary surface end point of next move. Lathe: Absolute U,W front turret end point of next move.
<b>P_ABS_X_END</b> <b>P_ABS_Y_END</b> <b>P_ABS_Z_END</b> <b>P_ABS_C_END</b> <b>P_ABS_U_END</b> <b>P_ABS_V_END</b> <b>P_ABS_W_END</b>		Absolute X,Y,Z,C end point of previous move. EDM: Absolute U,V secondary surface end point of previous move. Lathe: Absolute U,W front turret end point of previous move.
<b>O_ABS_X_END</b> <b>O_ABS_Y_END</b>		Punch: Original absolute X,Y end point of a move.
<b>ABS_X_START</b> <b>ABS_Y_START</b> <b>ABS_Z_START</b> <b>ABS_C_START</b> <b>ABS_U_START</b> <b>ABS_V_START</b> <b>ABS_W_START</b>	METRIC DECIMAL	Absolute X,Y,Z,C start point of move. EDM: Absolute U,V secondary surface start point of move. Lathe: Absolute U,W front turret start point of move
<b>N_ABS_X_START</b> <b>N_ABS_Y_START</b> <b>N_ABS_Z_START</b> <b>N_ABS_C_START</b> <b>N_ABS_U_START</b> <b>N_ABS_V_START</b> <b>N_ABS_W_START</b>	METRIC DECIMAL	Absolute X,Y,Z,C start point of next move. EDM: Absolute U,V secondary surface start point of next move. Lathe: Absolute U,W front turret start point of next move.

<i>Variable</i>	<i>Type</i>	<i>Usage</i>
<b>P_ABS_X_START</b>		Absolute X,Y,Z,C start point of previous move.
<b>P_ABS_Y_START</b>		
<b>P_ABS_Z_START</b>		EDM: Absolute U,V secondary surface start point of previous move.
<b>P_ABS_C_START</b>		
<b>P_ABS_U_START</b>		Lathe: Absolute U,W front turret start point of previous move.
<b>P_ABS_V_START</b>		
<b>P_ABS_W_START</b>		
<b>O_ABS_X_START</b>		Punch: Original absolute X,Y start point of a move.
<b>O_ABS_Y_START</b>		
<b>INC_X_END</b>	METRIC DECIMAL	X,Y,Z,C Signed incremental distance of move.
<b>INC_Y_END</b>		
<b>INC_Z_END</b>		EDM: U,V secondary surface signed incremental distance of move.
<b>INC_C_END</b>		
<b>INC_U_END</b>		Lathe: U,W front turret signed incremental distance of move.
<b>INC_V_END</b>		
<b>INC_W_END</b>		
<b>N_INC_X_END</b>		X,Y,Z,C Signed incremental distance of next move.
<b>N_INC_Y_END</b>		
<b>N_INC_Z_END</b>		EDM: U,V secondary surface signed incremental distance of next move.
<b>N_INC_C_END</b>		
<b>N_INC_U_END</b>		Lathe: U,W front turret signed incremental distance of next move.
<b>N_INC_V_END</b>		
<b>N_INC_W_END</b>		
<b>P_INC_X_END</b>	METRIC DECIMAL	X,Y,Z,C Signed incremental distance of previous move.
<b>P_INC_Y_END</b>		
<b>P_INC_Z_END</b>		EDM: U,V secondary surface signed incremental distance of previous move.
<b>P_INC_C_END</b>		
<b>P_INC_U_END</b>		Lathe: U,W front turret signed incremental distance of previous move.
<b>P_INC_V_END</b>		
<b>P_INC_W_END</b>		
<b>ABS_I_CENTER</b>	METRIC DECIMAL	I,J absolute center point of arc.
<b>ABS_J_CENTER</b>		EDM: K,L absolute center point of secondary surface arc.
<b>ABS_K_CENTER</b>		Lathe: K,L absolute center point of front turret arc.
<b>ABS_L_CENTER</b>		
<b>N_ABS_I_CENTER</b>		I,J absolute center point of next arc.
<b>N_ABS_J_CENTER</b>		EDM: K,L absolute center point of next secondary surface arc.
<b>N_ABS_K_CENTER</b>		Lathe: K,L absolute center point of next front turret arc.
<b>N_ABS_L_CENTER</b>		

<i>Variable</i>	<i>Type</i>	<i>Usage</i>
<b>P_ABS_I_CENTER</b> <b>P_ABS_J_CENTER</b> <b>P_ABS_K_CENTER</b> <b>P_ABS_L_CENTER</b>		I,J absolute center point of previous arc. EDM: K,L absolute center point of previous secondary surface arc. Lathe: K,L absolute center point of previous front turret arc.
<b>INC_I_CENTER</b> <b>INC_J_CENTER</b> <b>INC_K_CENTER</b> <b>INC_L_CENTER</b>	METRIC DECIMAL	I,J signed incremental distance from start of arc to center. K,L signed incremental distance from start of secondary surface arc to center.
<b>N_INC_I_CENTER</b> <b>N_INC_J_CENTER</b> <b>N_INC_K_CENTER</b> <b>N_INC_L_CENTER</b>	METRIC DECIMAL	I,J signed incremental distance from next start of arc to center. K,L signed incremental distance from next start of secondary surface arc to center.
<b>P_INC_I_CENTER</b> <b>P_INC_J_CENTER</b> <b>P_INC_K_CENTER</b> <b>P_INC_L_CENTER</b>	METRIC DECIMAL	I,J signed incremental distance from previous start of arc to center. K,L signed incremental distance from previous start of secondary surface arc to center.
<b>REV_INC_I_CENTER</b> <b>REV_INC_J_CENTER</b> <b>REV_INC_K_CENTER</b> <b>REV_INC_L_CENTER</b>	METRIC DECIMAL	I,J signed incremental distance from center to start of arc. K,L signed incremental distance from center to start of secondary surface arc.
<b>N_REV_INC_I_CENTER</b> <b>N_REV_INC_J_CENTER</b> <b>N_REV_INC_K_CENTER</b> <b>N_REV_INC_L_CENTER</b>	METRIC DECIMAL	I,J signed incremental distance from center to start of next arc. K,L signed incremental distance from center to start of next secondary surface arc.
<b>P_REV_INC_I_CENTER</b> <b>P_REV_INC_J_CENTER</b> <b>P_REV_INC_K_CENTER</b> <b>P_REV_INC_L_CENTER</b>	METRIC DECIMAL	I,J signed incremental distance from center to start of previous arc. K,L signed incremental distance from center to start of previous secondary surface arc.
<b>DISPLACED_X</b> <b>DISPLACED_Y</b> <b>DISPLACED_Z</b>	METRIC DECIMAL	5 axis modified X,Y,Z end point of move.
<b>N_DISPLACED_X</b> <b>N_DISPLACED_Y</b> <b>N_DISPLACED_Z</b>		5 axis modified X,Y,Z end point of next move.



<i>Variable</i>	<i>Type</i>	<i>Usage</i>
<b>P_DISPLACED_X</b> <b>P_DISPLACED_Y</b> <b>P_DISPLACED_Z</b>		5 axis modified X,Y,Z end point of previous move.
<b>ROT_TILT_A</b>	DECIMAL	5 axis "A" absolute rotation move.
<b>N_ROT_TILT_A</b>		5 axis "A" absolute rotation of next move.
<b>P_ROT_TILT_A</b>	DECIMAL	5 axis "A" absolute rotation of previous move.
<b>INC_ROT_TILT_A</b>		5 axis "A" signed incremental rotation move.
<b>N_INC_ROT_TILT_A</b>		5 axis "A" signed incremental rotation of next move.
<b>N_INC_ROT_TILT_A</b>		5 axis "A" signed incremental rotation of previous move.
<b>ROT_TILT_B</b>	DECIMAL	5 axis "B" absolute tilt move.
<b>N_ROT_TILT_B</b>		5 axis "B" absolute tilt of next move.
<b>P_ROT_TILT_B</b>		5 axis "B" absolute tilt of previous move.
<b>INC_ROT_TILT_B</b>		5 axis "B" signed incremental tilt move.
<b>N_INC_ROT_TILT_B</b>		5 axis "B" signed incremental tilt of next move.
<b>N_INC_ROT_TILT_B</b>		5 axis "B" signed incremental tilt of previous move.
<b>VECTOR_I</b>	DECIMAL	5 axis "I" axis vector move.
<b>N_VECTOR_I</b>		5 axis next "I" axis vector move.
<b>P_VECTOR_I</b>		5 axis previous "I" axis vector move.
<b>VECTOR_J</b>	DECIMAL	5 axis "J" axis vector move.
<b>N_VECTOR_J</b>		5 axis next "J" axis vector move.
<b>P_VECTOR_J</b>		5 axis previous "J" axis vector move.
<b>VECTOR_K</b>	DECIMAL	5 axis "K" axis vector move.
<b>N_VECTOR_K</b>	DECIMAL	5 axis next "K" axis vector move.
<b>P_VECTOR_K</b>		5 axis previous "K" axis vector move.

<i>Variable</i>	<i>Type</i>	<i>Usage</i>
<b>ARC_RADIUS</b>	METRIC DECIMAL	Arc radius.
<b>N_ARC_RADIUS</b>		Next arc radius.
<b>P_ARC_RADIUS</b>		Previous arc radius.
<b>O_ARC_RADIUS</b>		Punch: Original radius.
<b>S_ARC_RADIUS</b>		EDM: Secondary surface arc radius. Lathe: Front turret arc radius.
<b>S_N_ARC_RADIUS</b>		EDM: Next secondary surface arc radius. Lathe: Next front turret arc radius.
<b>S_P_ARC_RADIUS</b>		EDM: Previous secondary surface arc radius. Lathe: Previous front turret arc radius.
<b>DISTANCE</b>	METRIC DECIMAL	Length of a move.
<b>N_DISTANCE</b>		Length of next move.
<b>P_DISTANCE</b>		Length of previous move.
<b>S_DISTANCE</b>		EDM: Length of secondary surface move. Lathe: Length of front turret move.
<b>S_N_DISTANCE</b>		EDM: Length of next secondary surface move. Lathe: Length of next front turret move.
<b>S_P_DISTANCE</b>		EDM: Length of previous secondary surface move. Lathe: Length of previous front turret move.
<b>O_DISTANCE</b>		Punch: Original move length.
<b>ARC_START_ANGLE</b>	DECIMAL	Absolute arc start angle.
<b>N_ARC_START_ANGLE</b>		Next absolute arc start angle.
<b>P_ARC_START_ANGLE</b>		Previous absolute arc start angle.
<b>S_ARC_START_ANGLE</b>		EDM: Absolute arc start angle of secondary surface. Lathe: Absolute arc start angle of front turret.

<i>Variable</i>	<i>Type</i>	<i>Usage</i>
<b>S_N_ARC_START_ANGLE</b>		EDM: Next absolute arc start angle of secondary surface. Lathe: Next absolute arc start angle of front turret.
<b>S_P_ARC_START_ANGLE</b>		EDM: Previous absolute arc start angle of secondary surface. Lathe: Previous absolute arc start angle of front turret.
<b>O_ARC_START_ANGLE</b>		Punch: Original absolute arc start angle.
<b>ARC_END_ANGLE</b>	DECIMAL	Absolute arc end angle.
<b>N_ARC_END_ANGLE</b>		Next absolute arc end angle.
<b>P_ARC_END_ANGLE</b>		Previous absolute arc end angle.
<b>S_ARC_END_ANGLE</b>		EDM: Absolute arc end angle of secondary surface. Lathe: Absolute arc end angle of front turret.
<b>S_N_ARC_END_ANGLE</b>	DECIMAL	EDM: Next absolute arc end angle of secondary surface. Lathe: Next absolute arc end angle of front turret.
<b>S_P_ARC_END_ANGLE</b>		EDM: Previous absolute arc end angle of secondary surface. Lathe: Previous absolute arc end angle of front turret.
<b>O_ARC_END_ANGLE</b>		Punch: Original absolute arc end angle.
<b>ARC_INC_ANGLE</b>	DECIMAL	Absolute incremental arc angle.
<b>N_ARC_INC_ANGLE</b>		Next absolute incremental arc angle.
<b>P_ARC_INC_ANGLE</b>		Previous absolute incremental arc angle.
<b>S_ARC_INC_ANGLE</b>		EDM: Absolute incremental arc angle of secondary surface. Lathe: Absolute incremental arc angle of front turret.

<i>Variable</i>	<i>Type</i>	<i>Usage</i>
<b>S_N_ARC_INC_ANGLE</b>		EDM: Next absolute incremental arc angle of secondary surface. Lathe: Next absolute incremental arc angle of front turret.
<b>S_P_ARC_INC_ANGLE</b>		EDM: Previous absolute incremental arc angle of secondary surface. Lathe: Previous absolute incremental arc angle of front turret.
<b>O_ARC_INC_ANGLE</b>		Punch: Original absolute incremental arc angle.
<b>ANGLE</b>	DECIMAL	Angle of a Line, Rapid, Grid, Line Pattern or Window move.
<b>N_ANGLE</b>	DECIMAL	Angle of next Line, Rapid, Grid, Line Pattern or Window move.
<b>P_ANGLE</b>		Angle of previous Line, Rapid, Grid, Line Pattern or Window move.
<b>O_ANGLE</b>		Punch: Angle of original Line, Rapid, Grid, Line Pattern or Window move.
<b>TOOL_INDEX_ANGLE</b>	DECIMAL	Punch: Autoindex tool angle.
<b>N_TOOL_INDEX_ANGLE</b>		Punch: Next autoindex tool angle.
<b>P_TOOL_INDEX_ANGLE</b>		Punch: Previous autoindex tool angle.
<b>INC_TOOL_INDEX_ANGLE</b>		Punch: Signed incremental autoindex tool angle.
<b>N_INC_TOOL_INDEX_ANGLE</b>		Punch: Next signed incremental autoindex tool angle.
<b>P_INC_TOOL_INDEX_ANGLE</b>		Punch: Previous signed incremental autoindex tool angle.
<b>PITCH</b>	METRIC DECIMAL	Punch: Nibbling pitch of move.
<b>NPITCH</b>		Punch: Nibbling pitch of next move.
<b>PPITCH</b>		Punch: Nibbling pitch of previous move.
<b>OPITCH</b>		Punch: Original nibbling pitch of move.

<i>Variable</i>	<i>Type</i>	<i>Usage</i>
<b>N_OPITCH</b>		Punch: Original nibbling pitch of next move.
<b>P_OPITCH</b>	METRIC DECIMAL	Punch: Original nibbling pitch of previous move.
<b>MICRO_START</b>	METRIC DECIMAL	Punch: Signed incremental microjoint start distance.
<b>N_MICRO_START</b>		Punch: Next signed incremental microjoint start distance.
<b>P_MICRO_START</b>		Punch: Previous signed incremental microjoint start distance.
<b>MICRO_END</b>	METRIC DECIMAL	Punch: Signed incremental microjoint end distance.
<b>N_MICRO_END</b>		Punch: Next signed incremental microjoint end distance.
<b>P_MICRO_END</b>		Punch: Previous signed incremental microjoint end distance.
<b>P_MICROJOINT</b>	METRIC DECIMAL	Punch: Previous signed incremental microjoint distance.
<b>N_MICROJOINT</b>		Punch: Next signed incremental microjoint distance.
<b>MACRO_TIME</b>	DECIMAL	Internal time calculation for macros.
<b>MACRO_A</b> <b>MACRO_B</b> <b>MACRO_C</b> <b>MACRO_D</b> <b>MACRO_E</b> <b>MACRO_F</b> <b>MACRO_G</b> <b>MACRO_H</b> <b>MACRO_I</b> <b>MACRO_J</b>	DECIMAL	Internal macro variables used to pass information from the defined subroutine to the main calling program. MACRO_A and MACRO_B can also pass information from the main calling program to defined subroutine. MACRO_A will pass the 4th axis absolute preposition angle and MACRO_B will pass the 5th axis absolute preposition angle. These values will only be the preposition angles of the first time they are called. For example, if you are doing a tombstone and the first time the subroutine is called the 4th axis is at 90 degrees and the 5th axis is at 0 degrees, then MACRO_A will have a value of 90 and MACRO_B will have a value of zero.

## System Variables

<i>Variable</i>	<i>Type</i>	<i>Usage</i>
<b>TOOL_ZGL</b>	METRIC DECIMAL	Mill and Lathe: 2nd field of a fixed external file.
<b>TOOL_XGL</b>	METRIC DECIMAL	Mill and Lathe: 3rd field of a fixed external file.`
<b>TOOL_QTN</b>	METRIC DECIMAL	Mill and Lathe 4th field of a fixed external file.
<b>TOOL_QT_COMMENT</b>	CHARACTER	Mill and Lathe: 5th field of a fixed external file.
<b>PI</b>	DECIMAL	Mathematical constant = 3.14159265.
<b>BYTE_COUNT</b>	INTEGER	Byte count of the text output file *.txt.
<b>LINE_COUNT</b>	INTEGER	Line count of the text output file *.txt.
<b>METRIC_FLAG</b>	INTEGER	Metric in flag: 0=inch 1=metric.
<b>METRIC_OUT</b>	INTEGER	Metric out flag: 0=inch 1=metric.
<b>SEQ_INCREMENT</b>	INTEGER	Sets the sequence increment.
<b>MOVE_TYPE</b>	INTEGER	Movement type - Line, Arc, Rapid, etc.
<b>N_MOVE_TYPE</b>		Next movement type - Line, Arc, Rapid, etc.
<b>P_MOVE_TYPE</b>		Previous movement type - Line, Arc, Rapid, etc.
<b>S_MOVE_TYPE</b>		EDM and 4 axis Lathe: Secondary or U,V plane.
<b>S_N_MOVE_TYPE</b>	INTEGER	EDM and 4 axis Lathe: Next secondary or U,V plane.
<b>S_P_MOVE_TYPE</b>		EDM and 4 axis Lathe: Previous secondary or U,V plane.
<b>SEQ</b>	INTEGER	Sequence number.
<b>OPR_TYPE</b>	INTEGER	Mill or Lathe operation type - MILL_DRILLING, PROFILING, etc.

<i>Variable</i>	<i>Type</i>	<i>Usage</i>
<b>ATTROVERRIDE</b>	INTEGER	: <X:x_preset> - anything after the colon makes ATTROVERRIDE=YES.
<b>ATTRVALUE</b>	INTEGER	<S:1000> - since "S" is an INTEGER, the value is ATTRVALUE.
<b>ATTRDVALUE</b>	METRIC DECIMAL	<X:x_preset> since "X" is a DECIMAL, the value is ATTRDVALUE.
<b>ATTRCVALUE</b>	METRIC DECIMAL	<P:{MILLING}> since "P" is a CHARACTER, the value is ATTRCVALUE.
<b>DIST_BET_HOLES_X</b> <b>DIST_BET_HOLES_Y</b>	METRIC DECIMAL	Signed incremental X and Y distance between holes in a grid.
<b>N_DIST_BET_HOLES_X</b> <b>N_DIST_BET_HOLES_Y</b>		Signed incremental X and Y distance between holes in next grid.
<b>P_DIST_BET_HOLES_X</b> <b>P_DIST_BET_HOLES_Y</b>		Signed incremental X and Y distance between holes in previous grid.
<b>DIST_BET_PARTS_X</b> <b>DIST_BET_PARTS_Y</b>	METRIC DECIMAL	Signed incremental X and Y distance between parts in a macro.
<b>N_DIST_BET_PARTS_X</b> <b>N_DIST_BET_PARTS_Y</b>	METRIC DECIMAL	Signed incremental X and Y distance between parts in next macro.
<b>P_DIST_BET_PARTS_X</b> <b>P_DIST_BET_PARTS_Y</b>		Signed incremental X and Y distance between parts in previous macro.
<b>HEIGHT</b>	METRIC DECIMAL	Tool compensated window Y height.
<b>N_HEIGHT</b>		Next tool compensated in window Y height.
<b>P_HEIGHT</b>		Previous tool compensated in window Y height.
<b>WIDTH</b>	METRIC DECIMAL	Tool compensated window X width.
<b>N_WIDTH</b>		Next tool compensated in window X width.
<b>P_WIDTH</b>		Previous tool compensated in window X width.
<b>WINDOW_ORIGIN_X</b> <b>WINDOW_ORIGIN_Y</b>	METRIC DECIMAL	Absolute window origin.
<b>N_WINDOW_ORIGIN_X</b> <b>N_WINDOW_ORIGIN_Y</b>		Next absolute window origin.

<i>Variable</i>	<i>Type</i>	<i>Usage</i>
<b>P_WINDOW_ORIGIN_X</b> <b>P_WINDOW_ORIGIN_Y</b>		Previous absolute window origin.
<b>WINDOW_HEIGHT</b>	METRIC DECIMAL	CAD entity window Y height.
<b>N_WINDOW_HEIGHT</b>		Next CAD entity window Y height.
<b>P_WINDOW_HEIGHT</b>		Previous CAD entity window Y height.
<b>WINDOW_WIDTH</b>	METRIC DECIMAL	CAD entity window X width.
<b>N_WINDOW_WIDTH</b>		Next CAD entity window X width.
<b>P_WINDOW_WIDTH</b>		Previous CAD entity window X width.
<b>TOOL_LENGTH</b> <b>TOOL_WIDTH</b> <b>TOOL_DIAMETER</b>	METRIC DECIMAL	Tool dimensions.
<b>N_TOOL_LENGTH</b> <b>N_TOOL_WIDTH</b> <b>N_TOOL_DIAMETER</b>		Next entity tool dimensions.
<b>NC_TOOL_LENGTH</b> <b>NC_TOOL_WIDTH</b> <b>NC_TOOL_DIAMETER</b>		Next tool change tool dimensions.
<b>SCALLOP</b>	METRIC DECIMAL	Scallop height when nibbling.
<b>N_SCALLOP</b>		Next scallop height when nibbling.
<b>P_SCALLOP</b>		Previous scallop height when nibbling.
<b>TOOL_CORNER_RADIUS</b>	METRIC DECIMAL	Corner radius value of a corner radius punch tool.
<b>ISCOLLOP</b>	METRIC DECIMAL	Inside scollop height of an arc or circle when nibbling.
<b>OSCOLLOP</b>	METRIC DECIMAL	Outside scollop height of an arc or circle when nibbling.
<b>TOOL_LOAD_ANGLE</b>	DECIMAL	Load angle of the tool in a punch turret.
<b>TOOL</b>	INTEGER	Tool number.
<b>N_TOOL</b>	INTEGER	Next entity tool number.
<b>NC_TOOL</b>	INTEGER	Next tool change tool number.



<i>Variable</i>	<i>Type</i>	<i>Usage</i>
<b>TOOL_TYPE</b>	INTEGER	Tool type: 201 (Endmill) 202 (Ballnose) 203 (Hognose) 221 (Tapered Endmill) 222 (Tapered Ballnose) 223 (Tapered Hognose) 204 (Drill) 205 (Bore) 206 (Ream) 207 (Tap) 208 (Center Drill) 209 (Corner Round) 210 (Counter Sink) 211 (SP Thread Mill) 212 (MP Thread Mill) 213 (Dovetail) 214 (Keyway) 215 (Unknown) 216 (Lollipop) 217 (Facemill) 218 (Unknown) 301 (Round) 302 (Square) 303 (Triangle) 304 (Diamond) 305 (Diamond) 306 (Diamond) 315 (Diamond) 316 (Thread) 308 (Groove) 309 (Drill) 310 (Center Drill) 311 (Tap) 317 (Trigon) 318 (Hexagon)
<b>N_TOOL_TYPE</b>		Next entity tool type.
<b>NC_TOOL_TYPE</b>		Next tool change tool type.
<b>NUM_HITS_X</b> <b>NUM_HITS_Y</b>	INTEGER	Number of hits in X & Y on a grid or window.
<b>N_NUM_HITS_X</b> <b>N_NUM_HITS_Y</b>		Number of hits in X & Y on next grid or window.
<b>P_NUM_HITS_X</b> <b>P_NUM_HITS_Y</b>		Number of hits in X & Y on previous grid or window.
<b>NUM_HITS</b>	INTEGER	Total number of hits in a grid or window.

<i>Variable</i>	<i>Type</i>	<i>Usage</i>
<b>N_NUM_HITS</b>		Total number of hits on next grid or window.
<b>P_NUM_HITS</b>		Total number of hits on previous grid or window.
<b>NUM_PARTS_X</b> <b>NUM_PARTS_Y</b>	INTEGER	Number of parts in X & Y in a macro.
<b>N_NUM_PARTS_X</b> <b>N_NUM_PARTS_Y</b>		Number of parts in X & Y in next macro.
<b>P_NUM_PARTS_X</b> <b>P_NUM_PARTS_Y</b>		Number of parts in X & Y in previous macro.
<b>HORIZ_OR_VERT</b>	INTEGER	Direction of punching in a grid, window or multiple macro call.
<b>N_HORIZ_OR_VERT</b>	INTEGER	Direction of punching in next grid, window or multiple macro call.
<b>P_HORIZ_OR_VERT</b>		Direction of punching in previous grid, window or multiple macro call.
<b>O_DIRECTION</b>	INTEGER	Offset direction of tool: O_DIRECTION= OFFSET_LEFT or OFFSET_RIGHT CENTER_LEFT or CENTER_RIGHT
<b>SYSTEM_COMP</b>	INTEGER	Offset direction of tool: SYSTEM_COMP= OFFSET_LEFT or OFFSET_RIGHT CENTER_LEFT or CENTER_RIGHT
<b>FIXED</b>	INTEGER	Punch: Tool index angle. FIXED=YES or NO.
<b>FRAME</b>	INTEGER	Punch: Grid or window. FRAME=YES or NO
<b>SKIP_1ST_HIT</b>	INTEGER	Punch: Skip first hit. Mill: Skip first hole. - SKIP_1ST_HIT=YES or NO
<b>CURRENT_SYSTEM</b>	INTEGER	Plasma/punch post: What system are you currently in. CURRENT_SYSTEM=PLASMA or PUNCH

<i>Variable</i>	<i>Type</i>	<i>Usage</i>
<b>PREV_SYSTEM</b>	INTEGER	Plasma/punch post: What system you were in. PREV_SYSTEM=PLASMA or PUNCH
<b>NEXT_SYSTEM</b>	INTEGER	Plasma/punch post: What system you will be in. NEXT_SYSTEM=PLASMA or PUNCH
<b>PART_TYPE</b>	INTEGER	Plasma/punch post: What type of part is it. PART_TYPE=PLASMA or PUNCH
<b>CURRENT_MACRO_NUMBER</b>	INTEGER	Current macro name.
<b>EOL</b>	CHARACTER	End of line parameter. The default is EOL={ } Example EOL={ * } .
<b>PART_NAME</b>	CHARACTER	Name of the CAD part *.PRT.
<b>POST_NAME</b>	CHARACTER	Name of the post *.CTL
<b>CURRENT_MACRO_NAME</b>	CHARACTER	Current macro name.
<b>MACRO_NAME</b>	CHARACTER	Macro name.
<b>TOOL_DESCRIPTION</b>	CHARACTER	Tool Description.
<b>N_TOOL_DESCRIPTION</b>		Next entity tool Description.
<b>NC_TOOL_DESCRIPTION</b>		Next tool change tool Description.
<b>TOOL_COMMENT</b>	CHARACTER	Tool comment.
<b>N_TOOL_COMMENT</b>		Next entity tool comment.
<b>NC_TOOL_COMMENT</b>		Next tool change tool comment.
<b>TOOL_MATERIAL</b>	CHARACTER	Tool material from feed and speed table
<b>N_TOOL_MATERIAL</b>	CHARACTER	Next entity tool material from feed and speed table
<b>NC_TOOL_MATERIAL</b>	CHARACTER	Next tool change tool material from feed and speed table.
<b>TOOL_SERIAL_NUM</b>	CHARACTER	Tool serial number from feed and speed table.
<b>N_TOOL_SERIAL_NUM</b>		Next entity tool serial number from feed and speed table.

<i>Variable</i>	<i>Type</i>	<i>Usage</i>
<b>NC_TOOL_SERIAL_NUM</b>		Next tool change tool serial number from feed and speed table.
<b>DIE_SERIAL_NUM</b>	CHARACTER	Die serial number from feed and speed table.
<b>N_DIE_SERIAL_NUM</b>		Next entity die serial number from feed and speed table.
<b>NC_DIE_SERIAL_NUM</b>		Next tool change die serial number from feed and speed table.
<b>TOOL_DIE_CLEAR</b>	METRIC DECIMAL	Tool die clearance amount from feed and speed table.
<b>N_TOOL_DIE_CLEAR</b>		Next entity tool die clearance amount from feed and speed table.
<b>NC_TOOL_DIE_CLEAR</b>		Next tool change tool die clearance amount from feed and speed table.
<b>TOOL_SUB_TYPE</b>	INTEGER	Tool sub type: TOOL_SUB_TYPE= FLAT_BOTTOM HEELED FORM SHEAR_PROOF LOUVER MARKING
<b>N_TOOL_SUB_TYPE</b>	INTEGER	Next entity tool sub type: N_TOOL_SUB_TYPE= FLAT_BOTTOM HEELED FORM SHEAR_PROOF LOUVER MARKING
<b>NC_TOOL_SUB_TYPE</b>		Next tool change tool sub type: NC_TOOL_SUB_TYPE= FLAT_BOTTOM HEELED FORM SHEAR_PROOF LOUVER MARKING
<b>TOOL_SPEC_TYPE</b>	INTEGER	Tool special type: TOOL_SPEC_TYPE= AIR_BLOW PRESS_RAISE PRESSURE_IGNORE

<i>Variable</i>	<i>Type</i>	<i>Usage</i>
<b>N_TOOL_SPEC_TYPE</b>		Next entity tool special type: N_TOOL_SPEC_TYPE= AIR_BLOW PRESS_RAISE PRESSURE_IGNORE
<b>NC_TOOL_SPEC_TYPE</b>		Next tool change tool special type: NC_TOOL_SPEC_TYPE= AIR_BLOW PRESS_RAISE PRESSURE_IGNORE
<b>TOOL_NUM_TEETH</b>	INTEGER	Tool: Number of flutes from feed and speed table.
<b>N_TOOL_NUM_TEETH</b>		Next tool: Number of flutes from feed and speed table.
<b>NC_TOOL_NUM_TEETH</b>		Previous tool: Number of flutes from feed and speed table.
<b>FILLET_RADIUS</b>	METRIC DECIMAL	EDM: Fillet radius for XY surface.
<b>N_FILLET_RADIUS</b>		EDM: Next fillet radius for XY surface.
<b>P_FILLET_RADIUS</b>		EDM: Previous fillet radius for XY surface.
<b>S_FILLET_RADIUS</b>		EDM: Fillet radius for UV surface.
<b>S_N_FILLET_RADIUS</b>		EDM: Next fillet radius for UV surface.
<b>S_P_FILLET_RADIUS</b>		EDM: Previous fillet radius for UV surface.
<b>TAPER_ANGLE_START</b>	DECIMAL	EDM: Absolute taper angle start and end.
<b>N_TAPER_ANGLE_START</b>		EDM: Next absolute taper angle start and end.
<b>P_TAPER_ANGLE_START</b>		EDM: Previous absolute taper angle start and end.
<b>TAPER_ANGLE_END</b>	DECIMAL	EDM: Absolute taper angle end.
<b>N_TAPER_ANGLE_END</b>		EDM: Next absolute taper angle end.
<b>P_TAPER_ANGLE_END</b>		EDM: Previous absolute taper angle end.
<b>PROGRAM_SURF</b>	DECIMAL	EDM: X,Y surface or plane of a part.
<b>OTHER_SURF</b>	DECIMAL	EDM: U,V surface or plane of a part.
<b>XY_GUIDE_OFFSET</b>	DECIMAL	EDM: X,Y surface or plane guide offset of a part.

<i>Variable</i>	<i>Type</i>	<i>Usage</i>
<b>UV_GUIDE_OFFSET</b>	DECIMAL	EDM: U,V surface or plane guide offset of a part.
<b>CORNER_TYPE</b>	INTEGER	EDM: Taper corner type: CORNER_TYPE= SHARP_CORNER EQUAL_CORNER INDEPENDANT_CORNER CONIC_CORNER CHAMFER_CORNER
<b>N_CORNER_TYPE</b>		EDM: Next taper corner type: N_CORNER_TYPE= SHARP_CORNER EQUAL_CORNER INDEPENDANT_CORNER CONIC_CORNER CHAMFER_CORNER
<b>P_CORNER_TYPE</b>		EDM: Previous taper corner type: P_CORNER_TYPE= SHARP_CORNER EQUAL_CORNER INDEPENDANT_CORNER CONIC_CORNER CHAMFER_CORNER
<b>LEADIN</b>	INTEGER	Profile leadin flag: LEADIN=YES OR NO
<b>N_LEADIN</b>		Profile next leadin flag: LEADIN=YES OR NO.
<b>P_LEADIN</b>		Profile previous leadin flag: LEADIN=YES OR NO.
<b>LEADOUT</b>	INTEGER	Profile leadout flag: LEADOUT=YES OR NO.
<b>N_LEADOUT</b>		Profile next leadout flag: LEADOUT=YES OR NO.
<b>P_LEADOUT</b>		Profile previous leadout flag: LEADOUT=YES OR NO.
<b>WIRE_INCLINATION</b>	INTEGER	EDM: Wire taper direction: WIRE_INCLINATION= LEFT RIGHT
<b>EDM_MODE</b>	INTEGER	EDM mode. EDM_MODE= EDM4AXIS TAPER

<i>Variable</i>	<i>Type</i>	<i>Usage</i>
<b>G_RIGHT_PLACES</b>	INTEGER	Global number of places to the right of decimal point.
<b>G_LEFT_PLACES</b>	INTEGER	Global number of places to the left of decimal point.
<b>POWER_REG</b>	INTEGER	EDM: Power register.
<b>OFFSET_REG</b>	INTEGER	EDM: Offset tool register.
<b>PATH_TYPE</b>	INTEGER	EDM: Path type. PATH_TYPE= 1 (Profiling) 2 ( Tab cutting.
<b>TURRET</b>	INTEGER	4-axis Lathe: Turret. TURRET=FRONT OR REAR.
<b>N_TURRET</b>		4-axis Lathe: Turret next operation. N_TURRET= FRONT REAR
<b>P_TURRET</b>		4-axis Lathe: Turret previous operation. P_TURRET= FRONT REAR
<b>FRONT_TOOL</b>	INTEGER	4-axis Lathe: Tool selection. STATION_NUM=F01
<b>REAR_TOOL</b>	INTEGER	4-axis Lathe: Tool selection. STATION_NUM=R01
<b>STOCK_DIAMETER</b>	METRIC DECIMAL	4-axis Lathe: Stock diameter.
<b>OD</b>	INTEGER	Lathe: Outside diameter flag. OD=1 or 0.
<b>ID</b>	INTEGER	Lathe: Inside diameter flag: ID=1 or 0.
<b>OPR_FEED_FPR</b>	METRIC DECIMAL	Lathe: Feed per revolution.
<b>OPR_FEED_FPM</b>	METRIC DECIMAL	Lathe: Feed per minute.
<b>OPR_SPEED_SFPM</b>	METRIC DECIMAL	Lathe: Constant surface feed per minute.
<b>OPR_SPEED_RPM</b>	INTEGER	Lathe: Spindle speed.

<i>Variable</i>	<i>Type</i>	<i>Usage</i>
<b>OPR_FEED_TYPE</b>	INTEGER	Lathe: Feed type. OPR_FEED_TYPE= FPR FPM.
<b>OPR_SPEED_TYPE</b>	INTEGER	Lathe: Speed type. OPR_SPEED_TYPE= SFPM RPM
<b>OPR_X_POSITION</b>	METRIC DECIMAL	Lathe: X end operation position.
<b>OPR_Z_POSITION</b>	METRIC DECIMAL	Lathe: Z end operation position.
<b>OPR_X_PART_CLEARANCE</b>	METRIC DECIMAL	Lathe: Unsigned incremental roughing X finish allowance.
<b>OPR_Z_PART_CLEARANCE</b>	METRIC DECIMAL	Lathe: Unsigned incremental roughing Z finish allowance.
<b>OPR_CYCLE_CLEARANCE</b>	METRIC DECIMAL	Lathe: Unsigned incremental cycle clearance.
<b>OPR_THREAD_LENGTH</b>	METRIC DECIMAL	Lathe: Unsigned incremental thread length.
<b>OPR_THREAD_MINOR_DIAM</b>	METRIC DECIMAL	Lathe: Thread minor diameter.
<b>OPR_THREAD_FIRST_CUT</b>	METRIC DECIMAL	Lathe: Thread first pass amount.
<b>OPR_THREAD_LEADIN</b>	METRIC DECIMAL	Lathe: Thread unsigned incremental leadin amount.
<b>OPR_THREAD_CHAMFER</b>	METRIC DECIMAL	Lathe: Thread unsigned incremental chamfer amount.
<b>OPR_THREAD_MIN_CUT</b>	METRIC DECIMAL	Lathe: Thread unsigned minimum cut amount.
<b>OPR_THREAD_PITCH</b>	METRIC DECIMAL	Lathe: Thread pitch, feed or lead amount.
<b>OPR_THREAD_ANGLE</b>	DECIMAL	Lathe: Thread angle.
<b>OPR_INFEEED_ANGLE</b>	DECIMAL	Lathe: Thread infeed angle.
<b>OPR_THREAD_NUM_SPRING</b>	INTEGER	Lathe: Thread number of spring passes.



<i>Variable</i>	<i>Type</i>	<i>Usage</i>
<b>OPR_INFEED_TYPE</b>	INTEGER	Lathe thread infeed type. OPR_INFEED_TYPE= STRAIGHT_INFEED ANGLE_INFEED
<b>OPR_FIXED_ANGLE</b>	DECIMAL	Lathe/Mill: OD or FACE fixed angle.
<b>OPR_CUT_AMOUNT</b>	METRIC DECIMAL	Lathe: Roughing cycle cut amount.
<b>OPR_POST_CYCLE_TYPE</b>	INTEGER	Lathe: Cycle type. OPR_POST_CYCLE_TYPE= SYSTEM MACHINE
<b>OPR_CYCLE_TYPE</b>	INTEGER	Lathe: Cycle type. OPR_CYCLE_TYPE= TURNING FACING  Lathe Canned Drilling cycle OPR_CYCLE_TYPE= SINGLE_DEPTH CONSTANT PERCENTAGE
<b>OPR_RETRACT_TYPE</b>	INTEGER	Lathe: Retract type. OPR_RETRACT_TYPE= SINGLE_RETRACT MULTIPLE_RETRACT
<b>TOOL_TIP_CENTER</b>	INTEGER	Lathe: Tool compensation. TOOL_TIP_CENTER= 1 (center) 2 (tip)
<b>OPR_DEPTH_TYPE</b>	INTEGER	Lathe: Drilling depth type. OPR_DEPTH_TYPE= 0 (tip) 1 (flat)
<b>OPR_END_RETRACT_TYPE</b>	INTEGER	Lathe: Retract type. OPR_END_RETRACT_TYPE= 2 (x,z preset) 1 (x,z position) 0 (none) 4 (both)
<b>OPR_CMODE</b>	INTEGER	Lathe/Mill.
<b>OPR_CFIXED</b>	INTEGER	Lathe/Mill: OPR_CFIXED=CFIXED or CFREE.
<b>MOVE_COUNT</b>	INTEGER	Lathe: Cycle movement count. Used for canned Roughing cycles.
<b>LATHE_TOOL_NAME</b>	CHARACTER	Lathe: Tool name.

<i>Variable</i>	<i>Type</i>	<i>Usage</i>
<b>LATHE HOLDER_NAME</b>	CHARACTER	Lathe: Holder name.
<b>OPR_Z_FACE</b>	METRIC DECIMAL	Mill: Absolute Z face of part.
<b>OPR_Z_RAPID_PLANE</b>	METRIC DECIMAL	Mill: Absolute Z rapid plane.
<b>OPR_Z_CLEARANCE</b>	METRIC DECIMAL	Mill: Absolute Z clearance plane.
<b>OPR_Z_DEPTH</b>	METRIC DECIMAL	Mill: Absolute Z depth.
<b>OPR_CLEARANCE</b>	METRIC DECIMAL	Mill: Incremental Z clearance distance.
<b>OPR_LEADIN</b>	METRIC DECIMAL	Mill: Incremental leadin distance.
<b>OPR_LEADOUT</b>	METRIC DECIMAL	Mill: Incremental leadout distance.
<b>OPR_Z_FIRST_PECK</b>	METRIC DECIMAL	Mill/Lathe: Incremental drill first peck amount.
<b>OPR_Z_SUB_PECK</b>	METRIC DECIMAL	Mill/Lathe: Incremental drill subsequent peck amount.
<b>OPR_Z_MIN_PECK</b>	METRIC DECIMAL	Mill/Lathe: Incremental drill minimum peck amount.
<b>OPT_Z_FINISH</b>	METRIC DECIMAL	Mill: Lacing and pocketing Z finish allowance.
<b>OPR_Z_DIST_ALONG</b>	METRIC DECIMAL	Mill: Z distance along a wall.
<b>OPR_Z_FIRST_CUT</b>	METRIC DECIMAL	Mill: Profiling, lacing or pocketing Z first depth.
<b>OPR_Z_SUB_CUT</b>	METRIC DECIMAL	Mill: Profiling, lacing or pocketing Z subsequent depth.
<b>OPT_Z_RETRACT_AMOUNT</b>	METRIC DECIMAL	Mill: Profiling, lacing or pocketing Z retract amount.
<b>OPR_LACE_CUT</b>	METRIC DECIMAL	Mill: Lacing cut amount.
<b>OPR_CORNER_CLEAR</b>	METRIC DECIMAL	Mill: Profiling corner clearance amount.

<i>Variable</i>	<i>Type</i>	<i>Usage</i>
<b>OPR_CORNER_EXT</b>	METRIC DECIMAL	Mill: Profiling corner extension amount.
<b>OPR_Z_FEED</b>	METRIC DECIMAL	Mill: Z feed.
<b>OPR_X_FEED</b>	METRIC DECIMAL	Mill: XY feed.
<b>OPR_Z_PER_PECK</b>	DECIMAL	Mill: variable drill pecking percentage.
<b>OPR_LACE_ANGLE</b>	DECIMAL	Mill: lacing angle.
<b>ROTATE_ANGLE_X</b>	DECIMAL	Mill: 4 <sup>th</sup> axis rotation about the X axis in degrees.
<b>ROTATE_ANGLE_Y</b>	DECIMAL	Mill: 4 <sup>th</sup> axis rotation about the Y axis in degrees.
<b>ROTATE_ANGLE_Z</b>	DECIMAL	Mill: 4 <sup>th</sup> axis rotation about the Z axis in degrees.
<b>OPR_SPEED</b>	INTEGER	Mill: Spindle speed.
<b>OPR_SPEED_DIR</b>	INTEGER	Mill/Lathe: Spindle direction. OPR_SPEED_DIR=1, CW OPR_SPEED_DIR=2, CCW
<b>OPR_DRILL_CYCLE_TYPE</b>	INTEGER	Mill: Drilling cycle type. OPR_DRILL_CYCLE_TYPE= DRILLING SPOT_DRILLING PECKING HIGH_SPEED_PECKING VARIABLE_PECKING TAPPING REVERSE_TAPPING FINE_BORING REAMING REAMING_DWELL BORING BORING_DWELL BACK_BORING
<b>OPR_RETRACT_TYPE</b>	INTEGER	Mill: Retract type. OPR_RETRACT_TYPE= CLEARANCE_PLANE RAPID_PLANE
<b>OPR_CORNER_TYPE</b>	INTEGER	Mill: Profiling corner type. OPR_CORNER_TYPE= ROUND_CORNERS SHARP_CORNERS SQUARE_CORNERS TRIANGLE_CORNERS FANUC_CORNERS

<i>Variable</i>	<i>Type</i>	<i>Usage</i>
<b>OPR_Z_CYCLE_TYPE</b>	INTEGER	Mill: Z cycle type. OPR_Z_CYCLE_TYPE= 1,z_depth 2,z_wall 3,z_extrude
<b>OPR_Z_CUT_METHOD_TYPE</b>	INTEGER	Mill: Z cut type. OPR_Z_CUT_METHOD_TYPE= 1,z_depth 2,distance along wall
<b>OPR_CLEARANCE_TYPE</b>	INTEGER	Mill: Z clearance type. OPR_Z_CLEARANCE_TYPE= 1,single clearance 2,multiple clearance
<b>MACRO_ROTATE_AXIS</b>	INTEGER	Mill: 4 <sup>th</sup> axis rotation. MACRO_ROTATE_AXIS= X_AXIS (about X axis) Y_AXIS (about Y axis) Z_AXIS (about Z axis)
<b>OPR_COMMENT</b>	CHARACTER	Mill/Lathe: Operation comments.
<b>NEW_SFPM</b>	INTEGER	Lathe: Flag for figuring out surface feed per minute in versions 70d and above. NEW_SFPM=0 (if part was saved before version 8.0). NEW_SFPM=1 (if part was saved in version 8.0 and above).
<b>CCL_STATUS</b>	INTEGER	Macro CCL file status.
<b>OPR_AXIS_TYPE</b>	INTEGER	Mill: Operation type: OPR_AXIS_TYPE= 3333 if 3D 4444 if 4AXIS_X_MILLING 5555 if 4AXIS_Y_MILLING 6666 if 5AXIS_MILLING
<b>ROTATE_TILE</b>	INTEGER	4 and 5 axis rotate and tilt combinations. ROTATE_TILT= 6 (rotate table and tilt head about x-axis) 18 (rotate table and tilt head about y-axis) 10 (rotate table and tilt table about x-axis for 4th axis rotation use this for rotate x-axis "A" axis is being held on 4th axis) 34 (rotate table and tilt table about y-axis for 4th axis rotation use this for rotate y-axis "A" axis is being held on 4th axis) 11 (rotate arm and tilt head)

<i>Variable</i>	<i>Type</i>	<i>Usage</i>
<b>ARM_LEN</b>	METRIC DECIMAL	5 axis arm extension length.
<b>HEAD_LEN</b>	METRIC DECIMAL	5 axis head length.
<b>MILL_FACE_INC</b>	METRIC DECIMAL	The incremental distance between holes in a line pattern when drilling.
<b>Z_AXIS</b> <b>X_AXIS</b> <b>Y_AXIS</b> <b>I_AXIS</b> <b>J_AXIS</b> <b>K_AXIS</b> <b>ABS_PRESET_X</b> <b>ABS_PRESET_Y</b> <b>ABS_PRESET_Z</b> <b>ABS_PRESET_C</b> <b>X_START_POSITION</b> <b>Z_START_POSITION</b> <b>ATTRLVALUE</b> <b>OPR_LOOKAHEAD</b>	METRIC DECIMAL	Reserved for future use.
<b>Q_TAPER_ANGLE_START</b> <b>P_Q_TAPER_ANGLE_START</b> <b>N_Q_TAPER_ANGLE_START</b> <b>Q_TAPER_ANGLE_END</b> <b>P_Q_TAPER_ANGLE_END</b> <b>N_Q_TAPER_ANGLE_END</b> <b>R_TAPER_ANGLE_START</b> <b>P_R_TAPER_ANGLE_START</b> <b>N_R_TAPER_ANGLE_START</b> <b>R_TAPER_ANGLE_END</b> <b>P_R_TAPER_ANGLE_END</b> <b>N_R_TAPER_ANGLE_END</b>	DECIMAL	Reserved for future use.
<b>DIRECTION</b> <b>ESC</b> <b>RECNUM</b> <b>NEXT_OPR_TOOL</b> <b>MONTH</b> <b>DAY</b> <b>YEAR</b> <b>CURRENT_MACRO_QUADRANT</b> <b>CURRENT_MACRO_DEFINED</b> <b>MACRO_COUNT</b> <b>MACRO_ERROR</b> <b>MACRO_DEFINED</b> <b>SYS_COUNT</b>	INTEGER	Reserved for future use.

<i>Variable</i>	<i>Type</i>	<i>Usage</i>
<b>TIME</b> <b>DISK_DRIVE_1</b> <b>DISK_DRIVE_2</b> <b>TAB_CHAR</b>	CHARACTER	Reserved for future use.
<b>INC_ANGLE</b>  <b>N_INC_ANGLE</b>  <b>P_INC_ANGLE</b>	DECIMAL	Punch: Incremental angle of an arc pattern.  Punch: Incremental angle of next arc pattern.  Punch: Incremental angle of previous arc pattern.
<b>DIST_BET_PARTS_X</b> <b>DIST_BET_PARTS_Y</b>  <b>N_DIST_BET_PARTS_X</b> <b>N_DIST_BET_PARTS_Y</b>  <b>P_DIST_BET_PARTS_X</b> <b>P_DIST_BET_PARTS_Y</b>	DECIMAL	The signed incremental distance between parts in a multiple macro call.  The signed incremental distance between parts in the next multiple macro call.  The signed incremental distance between parts in the previous multiple macro call.

---

## **Chapter 9 System Constants**

---

---

# System Symbolic Constants

<i>Name</i>	<i>Value</i>
CENTER	0
RADIAL	1
ARCS	0
NO	0
YES	1
OFF	0
ON	1
F	0
T	1
FALSE	0
TRUE	1
ZERO	0
LEFT	1
RIGHT	2
HORIZONTAL	1
VERTICAL	0
MPOINT	POINT
MLINE	LINE
MCW_ARC	ARC   ARC_DIR
MCCW_ARC	ARC
MARC	ARC
MCIRCLE	CIRCLE
MTEXT	0x10
MGRID	0x10
LINE	1
CW_ARC	2
CCW_ARC	3
RAPID	4
SINGLE_HIT	5
ABSOLUTE	1
INCREMENTAL	2
HALF_CIRCLE	180
FULL_CIRCLE	360
NONE	0
ENGLISH	1
METRIC	2
MILL	MILL
LATHE	LATHE
PUNCH	PUNCH
PLASMA	PLASMA
LASER	LASER
EDM	EDM
WASINO	WASINO
JIG	JIG



<i>Name</i>	<i>Value</i>
BOTH	0
SHARP_CORNER	1009
EQUAL_CORNER	1010
INDEPENDENT_CORNER	2010
CONIC_CORNER	1011
CHAMFER_CORNER	2009
EDM4AXIS	1
TAPER	2
PROFILE	1
TABCUT	2
OFFSET_LEFT	0
OFFSET_RIGHT	1
CENTER_LEFT	2
CENTER_RIGHT	3
Z_DEPTH	1
Z_WALL	2
Z_EXTRUDE	3
DIST_ALONG	2
SINGLE	1
MULTIPLE	2
CW	1
CCW	2
FPR	1
FPM	2
SFPM	1
RPM	2
SYSTEM	1
MACHINE	2
TIP	0
FLAT	1
CONSTANT	1
PERCENTAGE	2
SINGLE_DEPTH	0
SINGLE_RETRACT	1
MULTIPLE_RETRACT	0
TURNING	0
FACING	1
CONSTANT_DEPTH	0
CONSTANT_CUT	1
STRAIGHT_INFEEED	0
ANGLE_INFEEED	1
ROUND_CORNERS	0
SHARP_CORNERS	1
SQUARE_CORNERS	2
TRIANGLE_CORNERS	3
FANUC_CORNERS	4
DRILLING	1
SPOT_DRILLING	2

## System Symbolic Constants

<i>Name</i>	<i>Value</i>
PECKING	3
TAPPING	4
BORING	5
HIGH_SPEED_PECKING	6
VARIABLE_PECKING	7
REVERSE_TAPPING	8
REAMING_DWELL	10
REAMING	9
BORE_DWELL	11
BACK_BORING	12
FINE_BORING	13
CLEARANCE_PLANE	1
RAPID_PLANE	2
FRONT	1
REAR	2
HOME	2
XZPOS	1
CFIXED	2
CFREE	1
MILL_OD	256
MILL_FACE	512
X_AXIS	7
Y_AXIS	8
Z_AXIS	6
ROUND	1
RECTANGLE	2
TRIANGLE	3
CROSS	4
OBROUND	5
SQUARE	6
RECRAD	7
DOUBLED	8
SINGLED	9
SPETOOL	10
PREVIOUS	1
NEXT	2
CURRENT	0
FLAT_BOTTOM	1
HEELED	2
SHEAR_PROOF	3
LOUVER	4
FORM	5
MARKING	6

---

## Mill Operation Symbolic Constants

<i>Name</i>	<i>Value</i>
MILL_DRILLING	1001
MILL_PROFILING	1002
MILL_LACE	1003
MILL_POCKET	1004
MILL_MISC	1005
MILL_SPECIAL	1006
MILL_MACRO	1007
MILL_UV_CUT	1007
MILL_SLICE_CUT	1008
MILL_ROUGH_CUT	1009
MILL_CURVE_CUT	1010

---

## Drill Operation Symbolic Constants

<i>Name</i>	<i>Value</i>
DRILLING	2001
SPOT_DRILLING	2002
PECKING	2003
TAPPING	2004
BORING	2005
HIGH_SPEED_PECKING	2006
VARIABLE_PECKING	2007
REVERSE_TAPPING	2008
REAMING_DWELL	2010
REAMING	2009
BORE_DWELL	2011
BACK_BORING	2012
FINE_BORING	2013

---

## Lathe Operation Symbolic Constants

<i>Name</i>	<i>Value</i>
LATHE_DRILLING	1011
LATHE_PROFILING	1012
LATHE_ROUGHING	1013
LATHE_GROOVING	1014
LATHE_THREADING	1015
LATHE_MISC	1016
LATHE_SPECIAL	1017

---

# Additional System Constants

<i>Name</i>	<i>Value</i>
OPEN SHAPE	0
CLOSED SHAPE	1
CIRCLE SHAPE	2
DROP	1
TILT	1
MAIN_SPINDLE	1
SUB_SPINDLE	2
CAMWORKS	1
PROCAM_2D	2
PROCAM_3D	3

---

## Chapter 10 Programming Examples

---

The examples in this chapter apply to changing mill posts.

## Example 1

### Adding a Setup Question and Using its Value to Change the Output

1. Make sure you copy the original source files to a folder where you can copy them back into this folder (master.atr, class.src and class.lib).
2. Add an attribute to (master.atr) called “head position”. This attribute will be a select type as described below. You will place it after ID number 17501.

```
*-----
:ATTRNAME=shots to be fired
:ATTRTYPE=VALUE
:ATTRVTYPE=DECIMAL
:ATTRID=17501
:ATTREND
*-----
:ATTRNAME=head position ←----- Attrname
:ATTRTYPE=SELECT
:ATTRID=17502 ←----- New ID number
:ATTREND
```

3. Change the IDHIGH=17502 at the start of the (master.atr) file.  
Note: if you use an ID number lower than 17501, then you would not have to do this step.
4. Open (class.src) and add the new attrname you created in master.atr in the setup list as shown below. Since this parameter will be asked in the Setup Information, the value is valid for the whole program.

```
*-----
* Define Setup Questions
*-----
:ATTRNAME=setup
:ATTRTYPE=LIST
:ATTRSEL=N
:ATTRTITLE=Setup
:ATTRLIST=program number
:ATTRLISTDEF=1
:ATTRLIST=material
:ATTRLISTDEF=STEEL
:ATTRLIST=thickness
:ATTRLISTDEF=1
*:ATTRLIST=chord length
*:ATTRLISTDEF=.01
:ATTRLIST=material_type
:ATTRLISTDEF=
:ATTRLIST=head position ←----- Add this line
:ATTRLISTDEF=1 ←----- Add this line
```



```
:ATTRUSED=1
:ATTRDEFAULT=1
:ATTREND
```

5. Go to the template section :SECTION=INIT\_TOOL\_CHANGE\_MILL in (class.src) and add the line below. Since this will only happen at the start of the program, you only need to change the init tool change section.

```
*
:SECTION=INIT_TOOL_CHANGE_MILL
:T:<N><M:HEAD_P><EOL> ←----- Add this line
:T:<N><TOOL_COMMENT><EOL>
:T:<N><T><M:06><EOL>
:T:<N><S!><M!:SPINDLE_DIR><EOL>
:T:<N><G!:work_coord><EOL>
:T:<N><M!:COOLANT_TYPE><EOL>
```

6. Open (class.lib) file and add the new attribute “head position” that you created in (master.atr). Even though you defined it in (master.atr), you still need to define it in the library file. See below for position.

```
*----- ←----- Add these lines
:ATTRNAME=head position
:ATTRTYPE=SELECT
:ATTREMARK=Spindle Head Position
:ATTRSEL=N
:ATTRTITLE=Spindle Head Position
:ATTRSELSTR=High
:ATTRSELSTR=Medium
:ATTRSELSTR=Low
:ATTRDEFAULT=1
:ATTRUSED=1
:ATTREND ←----- To here
*-----
:ATTRNAME=material type
:ATTRTYPE=SELECT
:ATTREMARK=Material type
:ATTRSEL=N
:ATTRTITLE=Material Type
:ATTRSELSTR=Rolled Steel
:ATTRSELSTR=Aluminum
:ATTRSELSTR=Stainless Steel
:ATTRDEFAULT=1
:ATTRUSED=1
:ATTREND
```

7. From the top of (class.lib), search for NAME=SEQ CONFIG.

## Example 1

8. Add the post type attribute listed below.

```
*----- ←----- Add these lines
:ATTRNAME=HEAD P
:ATTRTYPE=POST
:ATTRVTYPE=INTEGER
:ATTREMARK=Head Position
:ATTREND ←----- To here
*-----

:ATTRNAME=SEQ CONFIG
:ATTRTYPE=POST
:ATTRVTYPE=INTEGER
:ATTREMARK=Config. seq. numbers
:ATTREND
```

9. You need to add some logic to :SECTION=CALC\_INIT\_TOOL\_CHANGE\_MILL.

Since this section is not in the (class.lib), you need to copy it from (general.lib) file and place it at the end of the (class.lib) file as shown below. Now add the following lines of logic.

```
*-----
:ATTRNAME=CURRENT MACRO NAME
:ATTRTYPE=POST
:ATTREMARK=
:CODETYPE=FORMAT
:WORD_ADDRESS_BEf= | (
:VAR=CURRENT MACRO NAME
:WORD_ADDRESS_AfT=)
:LEFT_PLACES=0
:RIGHT_PLACES=0
:UNITFLAG=NON_CONVERT
:ATTRSPACES=YES
*:MODAL=YES
:ATTRUSED=1
:ATTREND
*-----

:SECTION=CALC_INIT_TOOL_CHANGE_MILL
:C: IF SECTIONEXIST(DEBUG) THEN
:C: DEBUG=4 CALL (DEBUG)
:C: ENDIF
*
*:C: IF OPER_COUNT>1 THEN CALL (CALC_SUB_TOOL_CHANGE_MILL) RETURN
ENDIF
:C: IF MACH(REG_T2) <> 0 THEN CALL (CALC_SUB_TOOL_CHANGE_MILL) RETURN
ENDIF
*
* If you are defining a macro then you stop here!
*
:C: P_MOVE_TYPE=TOOL_CHANGE
:C: IF DEFINING_MACRO=(YES) THEN CALL (CALC_CHECK_OPER_COMMENTS)
RETURN ENDIF
```

```

:C: IF MACH(REG_T) <> 0 AND MACH(REG_T) = TOOL THEN RETURN ENDIF
:C: CALC_CHANGE_TOOL = 1
:C: TOOL_ARRAY (ARRAY_COUNT) = NC_TOOL
:C: TOOL_DIAM_ARRAY (ARRAY_COUNT) = NC_TOOL_DIAMETER
:C: IF NC_TOOL = (-1) THEN
TOOL_DIAM_ARRAY (ARRAY_COUNT) = TOOL_DIAM_ARRAY (0) ENDIF
:C: NEXT_TOOL = TOOL_ARRAY (ARRAY_COUNT)
:C: POT_NUMBER = 10
:C: IF TOOL_DIAMETER > LARGE_POT THEN POT_NUMBER = 90 ENDIF
:C: NEXT_POT_NUMBER = 10
:C: IF TOOL_DIAM_ARRAY (ARRAY_COUNT) > LARGE_POT THEN
NEXT_POT_NUMBER = 90 ENDIF
:C: ARRAY_COUNT = (ARRAY_COUNT + 1)
:C: IF SECTIONEXIST (OUTPUT_ESTIMATED_TIME) THEN
:C: CALL (CALC_TOOL_CHANGE_TIME)
:C: ENDIF
:C: IF TOOL_COMMENT = { } THEN
:C: SETOFF (<TOOL_COMMENT>) ELSE
:C: SETON (<TOOL_COMMENT>)
:C: ENDIF
:C: IF head_position=1 THEN HEAD_P=101 ENDIF <---- Add these lines
:C: IF head_position=2 THEN HEAD_P=102 ENDIF
:C: IF head_position=3 THEN HEAD_P=103 ENDIF <----- To here
:C: IF SECTIONEXIST (INIT_PRELOAD_TOOL_CHANGE_MILL) THEN
:C: CALL (CALC_INIT_PRELOAD_TOOL_CHANGE)
:C: CALL (CALC_CHECK_OPER_COMMENTS)
:C: MACH (REG_T) = TOOL
:C: FIRST_TOOL = TOOL
:C: LAST_TOOL = TOOL
:C: RETURN
:C: ENDIF
:C: IF SECTIONEXIST (INIT_TOOL_CHANGE_MILL) THEN
:C: CALL (INIT_TOOL_CHANGE_MILL)
:C: ENDIF
:C: CALL (CALC_CHECK_OPER_COMMENTS)
:C: MACH (REG_T) = TOOL
:C: FIRST_TOOL = TOOL
:C: LAST_TOOL = TOOL

```

10. Now assuming you put master.atr in the same folder as your source, you can save all the files you edited and exit your editor.

To compile, you will need to type this at your **DOS** prompt –

**WINMAKE CLASS.SRC MASTER.ATR ?:\PROCAD\CTL.**

If you have installed the UPG, then you can compile the source in the UPG by selecting the File menu and picking the Compile Post command.

## Example 2

### Adding an Operation Question and Using its Value to Change the Output

1. If you want to save the previous example, then copy the source files (master.atr, class.src and class.lib) to a save folder and recopy the original files back into the class folder.
2. Add an attribute to (master.atr) called “changing pallets”. This attribute will be a select type as described below. You will place it after ID number 17502

```
*-----
:ATTRNAME=shots to be fired
:ATTRTYPE=VALUE
:ATTRVTYPE=DECIMAL
:ATTRID=17501
:ATTREND
*-----
:ATTRNAME=changing pallets ←----- Attrname
:ATTRTYPE=SELECT
:ATTRID=17502 ←----- New ID number
:ATTREND
```

3. Change the IDHIGH=17502 at the start of the (master.atr) file.  
Note: If you use an ID number that is lower than 17501, then you would not have to do this step.
4. Open (class.src) and add the new attrname you created in master.atr to all the operation lists as shown below. Since this question will be asked in the operation list, then the value is valid for that operation only. Make sure you add the line to all the :OPERID= (Operations). If you add it to just the Drilling one in the example below, then the drilling operation is the only operation that will get the question asked.

```
*-----
* Operation List Questions
*-----
:OPERID=MILL_DRILLING
:OPERSUB=DRILLING
:OPERLIST=abs inc
:OPERLIST=work coord
:OPERLIST=coolant
:OPERLIST=changing pallets ←----- Add this line to all
":OPERID=" lists
:OPEREND
```

- Go to the template section :SECTION=INIT\_TOOL\_CHANGE\_MILL in (class.src) and add the line below.

Since this will only happen at every tool change , you need to change the init tool change section and the subtool change sections.

```
*
:SECTION=INIT_TOOL_CHANGE_MILL
:T:<N><TOOL_COMMENT><EOL>
:T:<N><T><M:06><EOL>
:T:<N><S!><M!:SPINDLE_DIR><EOL>
:T:<N><G!:work_coord><EOL>
:T:<N><P_CHANGE><EOL> ←-----Add this line
:T:<N><M!:COOLANT_TYPE><EOL>
*

:SECTION=SUB_TOOL_CHANGE_MILL
:T:<N><G:00><G:91><G:28> Z0<EOL>
:T:<N><TOOL_COMMENT><EOL>
:T:<N><T><M:06><EOL>
:T:<N><S!><M!:SPINDLE_DIR><EOL>
:T:<N><G!:work_coord><EOL>
:T:<N><P_CHANGE><EOL> ←-----Add this line
:T:<N><M!:COOLANT_TYPE><EOL>
```

- Open (class.lib) file and add the new attribute “changing pallets” you created in (master.atr). Even though you defined it in (master.atr), you still need to define it in the library file. See below for position.

```
*----- ←----- Add these lines
:ATTRNAME=changing pallets
:ATTRTYPE=SELECT
:ATTREMARK=Changing Pallets
:ATTRSEL=N
:ATTRTITLE=Changing Pallets
:ATTRSELSTR=No
:ATTRSELSTR=Left
:ATTRSELSTR=Right
:ATTRDEFAULT=1
:ATTRUSED=1
:ATTREND ←----- To here
*-----

:ATTRNAME=material type
:ATTRTYPE=SELECT
:ATTREMARK=Material type
:ATTRSEL=N
:ATTRTITLE=Material Type
:ATTRSELSTR=Rolled Steel
:ATTRSELSTR=Aluminum
:ATTRSELSTR=Stainless Steel
:ATTRDEFAULT=1
:ATTRUSED=1
:ATTREND
```

## Example 2

7. Add the attribute listed below at the end of (class.lib) file.

Notice that the attrname P CHANGE is a select type attribute. What ever the value of P CHANGE equals is the select value. If the value of P CHANGE is not 2 or 3, then there is no output. In the lower case attribute “changing pallets” the :ATTRSELSTR= are No, Left or Right. The value of “changing pallets” equals one (1) if you pick the first one in the list, etc. So if you select “No” in the operation question, then the value of “changing pallets” equals one (1) and you will not get any output from P\_CHANGE.

```
*-----
:ATTRNAME=CURRENT MACRO NAME
:ATTRTYPE=POST
:ATTREMARK=
:CODETYPE=FORMAT
:WORD_ADDRESS_BEf= | (
:VAR=CURRENT MACRO NAME
:WORD_ADDRESS_AfT=)
:LEFT_PLACES=0
:RIGHT_PLACES=0
:UNITFLAG=NON_CONVERT
:ATTRSPACES=YES
*:MODAL=YES
:ATTRUSED=1
:ATTREND
*----- ←----- Add these lines
:ATTRNAME=P CHANGE
:ATTRTYPE=POST
:ATTRVTYPE=INTEGER
:ATTREMARK=Pallet Change
:CODETYPE=SELECT
:SELECT=2
:CODE=M51
:SELECT=3
:CODE=M52
:ATTREND ←----- To here
```

8. Now you need to add some logic to handle the output. You need to have this happen at every tool change, so you have to change the calc sections  
:SECTION=CALC\_INIT\_TOOL\_CHANGE\_MILL and  
:SECTION=CALC\_SUB\_TOOL\_CHANGE\_MILL. Since both of these sections are not in (class.lib), you have to copy them from (general.lib) to the end of (class.lib). See below for position.

```
*-----
:SECTION=CALC_INIT_TOOL_CHANGE_MILL
:C: IF SECTIONEXIST(DEBUG) THEN
:C: DEBUG=4 CALL (DEBUG)
:C: ENDEF
*
*:C: IF OPER_COUNT>1 THEN CALL (CALC_SUB_TOOL_CHANGE_MILL) RETURN
ENDEF
```

```

:C: IF MACH(REG_T2)<>0 THEN CALL(CALC_SUB_TOOL_CHANGE_MILL) RETURN
ENDIF
*
*   If you are defining a macro then you stop here!
*
:C: P_MOVE_TYPE=TOOL_CHANGE
:C: IF DEFINING_MACRO=(YES) THEN CALL(CALC_CHECK_OPER_COMMENTS)
RETURN ENDIF
:C: IF MACH(REG_T)<>0 AND MACH(REG_T)=TOOL THEN RETURN ENDIF
:C: CALC_CHANGE_TOOL=1
:C: TOOL_ARRAY(ARRAY_COUNT)=NC_TOOL
:C: TOOL_DIAM_ARRAY(ARRAY_COUNT)=NC_TOOL_DIAMETER
:C: IF NC_TOOL=(-1) THEN
TOOL_DIAM_ARRAY(ARRAY_COUNT)=TOOL_DIAM_ARRAY(0) ENDIF
:C: NEXT_TOOL=TOOL_ARRAY(ARRAY_COUNT)
:C: POT_NUMBER=10
:C: IF TOOL_DIAMETER>LARGE_POT THEN POT_NUMBER=90 ENDIF
:C: NEXT_POT_NUMBER=10
:C: IF TOOL_DIAM_ARRAY(ARRAY_COUNT)>LARGE_POT THEN
NEXT_POT_NUMBER=90 ENDIF
:C: ARRAY_COUNT=(ARRAY_COUNT+1)
:C: IF SECTIONEXIST(OUTPUT_ESTIMATED_TIME) THEN
:C: CALL(CALC_TOOL_CHANGE_TIME)
:C: ENDIF
:C: IF TOOL_COMMENT={} THEN
:C: SETOFF(<TOOL_COMMENT>) ELSE
:C: SETON(<TOOL_COMMENT>)
:C: ENDIF
:C: P_CHANGE=changing pallets ←----- ----- Add this line
:C: IF SECTIONEXIST(INIT_PRELOAD_TOOL_CHANGE_MILL) THEN
:C: CALL(CALC_INIT_PRELOAD_TOOL_CHANGE)
:C: CALL(CALC_CHECK_OPER_COMMENTS)
:C: MACH(REG_T)=TOOL
:C: FIRST_TOOL=TOOL
:C: LAST_TOOL=TOOL
:C: RETURN
:C: ENDIF
:C: IF SECTIONEXIST(INIT_TOOL_CHANGE_MILL) THEN
:C: CALL(INIT_TOOL_CHANGE_MILL)
:C: ENDIF
:C: CALL(CALC_CHECK_OPER_COMMENTS)
:C: MACH(REG_T)=TOOL
:C: FIRST_TOOL=TOOL
:C: LAST_TOOL=TOOL
*

:SECTION=CALC_SUB_TOOL_CHANGE_MILL
:C: IF SECTIONEXIST(DEBUG) THEN
:C: DEBUG=5 CALL(DEBUG)
:C: ENDIF
*
*   Startup Seton Codes
*

```

## Example 2

```
:C: CALL(CALC_BEF_SETON_CODES)
:C: G=GC(G_LEN_COMP) SETON(<G>)
:C: M=MC(M_COOL_OFF) SETON(<M>)
*
*   If you are defining a macro then you stop here!
*
:C: P_MOVE_TYPE=TOOL_CHANGE
:C: IF DEFINING_MACRO=(YES) THEN CALL(CALC_CHECK_OPER_COMMENTS)
RETURN ENDIF
:C: IF MACH(REG_T)<>0 AND MACH(REG_T)=TOOL THEN RETURN ENDIF
:C: CALC_CHANGE_TOOL=(CALC_CHANGE_TOOL+1)
:C: IF OFFSET_RESIDENT=YES THEN CALL(CALC_REMOVE_OFFSET) ENDIF
:C: IF TOOL=NC_TOOL THEN NC_TOOL=(-1) ENDIF
:C: TOOL_ARRAY(ARRAY_COUNT)=NC_TOOL
:C: TOOL_DIAM_ARRAY(ARRAY_COUNT)=NC_TOOL_DIAMETER
:C: IF NC_TOOL=(-1) THEN
TOOL_DIAM_ARRAY(ARRAY_COUNT)=TOOL_DIAM_ARRAY(0) ENDIF
:C: NEXT_TOOL=TOOL_ARRAY(ARRAY_COUNT)
:C: POT_NUMBER=10
:C: IF TOOL_DIAMETER>LARGE_POT THEN POT_NUMBER=90 ENDIF
:C: NEXT_POT_NUMBER=10
:C: IF TOOL_DIAM_ARRAY(ARRAY_COUNT)>LARGE_POT THEN
NEXT_POT_NUMBER=90 ENDIF
:C: ARRAY_COUNT=(ARRAY_COUNT+1)
:C: CALL(CALC_TOOL_CHANGE_TIME)
:C: IF TOOL_COMMENT={} THEN
:C: SETOFF(<TOOL_COMMENT>) ELSE
:C: SETON(<TOOL_COMMENT>)
:C: ENDIF
:C: P_CHANGE=changing pallets ←----- Add this line
:C: IF SECTIONEXIST(SUB_PRELOAD_TOOL_CHANGE_MILL) THEN
:C: CALL(CALC_SUB_PRELOAD_TOOL_CHANGE)
:C: CALL(CALC_CHECK_OPER_COMMENTS)
:C: CALL(CALC_AFT_SETON_CODES)
:C: MACH(REG_T)=TOOL
:C: LAST_TOOL=TOOL
:C: MACH(REG_Z)=MILL_Z_HOME
:C: RETURN
:C: ENDIF
:C: IF SECTIONEXIST(SUB_TOOL_CHANGE_MILL) THEN
:C: CALL(SUB_TOOL_CHANGE_MILL)
:C: ENDIF
:C: CALL(CALC_CHECK_OPER_COMMENTS)
:C: CALL(CALC_AFT_SETON_CODES)
:C: MACH(REG_T)=TOOL
:C: LAST_TOOL=TOOL
:C: MACH(REG_Z)=MILL_Z_HOME
```



9. Assuming you put master.atr in the same folder as your source, you can save all the files you edited and exit your editor.

To compile you will need to type this at your **DOS** prompt –

**WINMAKE CLASS.SRC MASTER.ATR [Drive]:\PROCAD\CTL.**

If you have installed the UPG, then you can compile the source in the UPG by selecting the File menu and picking "Compile Post".

## Example 3

### Adding an Operation Question and Using its Value to Change the Output Depending on if it is Modal or Not

1. If you want to save the previous example, copy the source files (master.atr, class.src and class.lib) to a save folder and recopy the original files back into the class folder.
2. You are going to add an attribute to (class.lib) called “spindle range”. Since this attribute is already defined in (master.atr), you do not have to edit (master.atr). Add “spindle range” to (class.lib) as shown below.

```
*----- ←----- Add these lines
:ATTRNAME=spindle range
:ATTRTYPE=VALUE
:ATTRVTYPE=INTEGER
:ATTREMARK=Spindle Range
:ATTRSEL=N
:ATTRINLEN=3
:ATTRSHORT=Spindle Range
:ATTRLONG=ENTER Spindle Range
:ATTRHIGH=41
:ATTRLOW=40
:ATTRDEFAULT=40
:ATTRUSED=1
:ATTREND ←----- To here
*-----

:ATTRNAME=material type
:ATTRTYPE=SELECT
:ATTREMARK=Material type
:ATTRSEL=N
:ATTRTITLE=Material Type
:ATTRSELSTR=Rolled Steel
:ATTRSELSTR=Aluminum
:ATTRSELSTR=Stainless Steel
:ATTRDEFAULT=1
:ATTRUSED=1
:ATTREND
```

3. Open (class.src) and add the new attrname you created to all the operation lists as shown below.

Since this question will be asked in the operation list, then the value is valid for that operation only. Make sure you add the line to all the :OPERID= (Operations). If you add it to just the Drilling one in the example below, then the drilling operation is the only operation that will get the question asked.

```
*-----
* Operation List Questions
*-----
:OPERID=MILL_DRILLING
:OPERSUB=DRILLING
:OPERLIST=abs inc
:OPERLIST=work coord
:OPERLIST=coolant
:OPERLIST=spindle range ←-- Add this line to all ":OPERID=" lists
:OPEREND
```

4. Find :SECTION=INIT\_TOOL\_CHANGE\_MILL and add the following code to both tool change sections as shown below.

```
*
:SECTION=INIT_TOOL_CHANGE_MILL
:T:<N><TOOL_COMMENT><EOL>
:T:<N><T><M:06><EOL>
:T:<N><M:spindle_range><EOL> ←----- Add this line
:T:<N><S!><M!:SPINDLE_DIR><EOL>
:T:<N><G!:work_coord><EOL>
:T:<N><M!:COOLANT_TYPE><EOL>
*

:SECTION=SUB_TOOL_CHANGE_MILL
:T:<N><G:00><G:91><G:28> Z0<EOL>
:T:<N><TOOL_COMMENT><EOL>
:T:<N><T><M:06><EOL>
:T:<N><M:spindle_range><EOL> ←----- Add this line
:T:<N><S!><M!:SPINDLE_DIR><EOL>
:T:<N><G!:work_coord><EOL>
:T:<N><M!:COOLANT_TYPE><EOL>
```

### Example 3

5. In the same file go to :SECTION=CALC\_INIT\_MCODES and add the bold underlined lines as shown below. Adding these lines of code to the (class.src) makes these codes modal.

```
:SECTION=CALC_INIT_MCODES
*-----*
*           M Code           M Group           M Modal           *
*-----*
:C: MC (M_STOP)           = 0   MG (M_STOP)           = 0   MM (M_STOP)           = NO
:C: MC (M_OPT_STOP)       = 1   MG (M_OPT_STOP)       = 0   MM (M_OPT_STOP)       = NO
:C: MC (M_PROG_END)       = 2   MG (M_PROG_END)       = 0   MM (M_PROG_END)       = NO
:C: MC (M_SPIN_CW)        = 3   MG (M_SPIN_CW)        = 1   MM (M_SPIN_CW)        = YES
:C: MC (M_SPIN_CCW)       = 4   MG (M_SPIN_CCW)       = 1   MM (M_SPIN_CCW)       = YES
:C: MC (M_SPIN_STOP)      = 5   MG (M_SPIN_STOP)      = 1   MM (M_SPIN_STOP)      = YES
:C: MC (M_TOOL_CHANGE)    = 6   MG (M_TOOL_CHANGE)    = 0   MM (M_TOOL_CHANGE)    = NO
:C: MC (M_COOL_MIST)      = 7   MG (M_COOL_MIST)      = 2   MM (M_COOL_MIST)      = YES
:C: MC (M_COOL_FLOOD)     = 8   MG (M_COOL_FLOOD)     = 2   MM (M_COOL_FLOOD)     = YES
:C: MC (M_COOL_OFF)       = 9   MG (M_COOL_OFF)       = 2   MM (M_COOL_OFF)       = YES
:C: MC (M_LOCK_OFF)       = 10  MG (M_LOCK_OFF)       = 3   MM (M_LOCK_OFF)       = YES
:C: MC (M_LOCK_ON)        = 11  MG (M_LOCK_ON)        = 3   MM (M_LOCK_ON)        = YES
:C: MC (M_ORIENT)         = 19  MG (M_ORIENT)         = 0   MM (M_ORIENT)         = NO
:C: MC (M SPIN LOW)      = 40  MG (M SPIN LOW)      = 4  MM (M SPIN LOW)      = YES
:C: MC (M SPIN HI)       = 41  MG (M SPIN HI)       = 4  MM (M SPIN HI)       = YES
:C: MC (M_END_PROG)       = 30  MG (M_END_PROG)       = 0   MM (M_END_PROG)       = NO
:C: MC (M_SUB_CALL)       = 98  MG (M_SUB_CALL)       = 0   MM (M_SUB_CALL)       = NO
:C: MC (M_SUB_END)        = 99  MG (M_SUB_END)        = 0   MM (M_SUB_END)        = NO
```

6. Assuming you put master.atr in the same folder as your source, you can save all the files you edited and exit your editor.
7. To compile you will need to type this at your **DOS** prompt –

**WINMAKE CLASS.SRC MASTER.ATR ?:\PROCAD\CTL.**

If you have installed the UPG, then you can compile the source in the UPG by selecting the File menu and picking "Compile Post".

---

## **Chapter 11 Add'l. System Header Commands**

---

---

# WORLD\_POSITIONING

## ***Purpose***

Allows world coordinate posted output when indexing in 4 and 5 axis assembly parts using CAMWorks 2005 or newer versions.

## ***Syntax***

**:WORLD\_POSITIONING=TRUE or FALSE**

## ***Comments***

This command should be set to FALSE if you do not need world coordinate posted output in 4 and 5 axis indexing.

This command should be set to TRUE if you need world coordinate posted output in 4 and 5 axis indexing, which means that the posted output will not translate the numbers when indexing to another plane.

---

## RIGHT\_ANGLE\_SHEAR\_ATTACHED

### ***Purpose***

Allows the system to exit into Shear for a combination PUNCH/SHEAR machine.

### ***Syntax***

**:RIGHT\_ANGLE\_SHEAR\_ATTACHED=TRUE or FALSE**

### ***Comments***

This command should be set to FALSE if you don't need shear.

This command should be set to TRUE if you need shear and even though the post  
“:SYSTEM=PUNCH” you can use this header command to exit into shear.

---

## LASER\_PLASMA\_CUT\_DATA

### ***Purpose***

Allows the laser and plasma systems to use the external fabrication database for special cutting parameters while posting.

### ***Syntax***

**:LASER\_PLASMA\_CUT\_DATA=TRUE or FALSE**

### ***Comments***

This command should be set to FALSE if you don't need special cutting parameters output to the post.

This command should be set to TRUE if you need special cutting parameters output to the post. The files that are used when accessing the data are "FABDBENGLISH.MDB" in Inch or "FABDBMETRIC.MDB" in metric. The post needs to be setup to use this function. See using access database in the online help.



---

## VECTOR\_COMP

### ***Purpose***

Allows a CAMWorks mill post to output X,Y,Z,I, and J in vector coordinates.

### ***Syntax***

**:VECTOR\_COMP=TRUE or FALSE**

### ***Comments***

This command should be set to FALSE if you do not need vector coordinates output.

This command should be set to TRUE if you need vector coordinates output. This will only apply in the CAMWorks advanced cutting operations. The post system variable "V\_COMP" will be set to "0" if operations not using this option and will be set to "1" if it is. Post system variables "XC", "YC", "ZC", "IC", "JC" and "KC" hold endpoints for the posted output. XC=ABS\_X\_END, YC=ABS\_Y\_END, ZC=ABS\_Z\_END, IC=I\_VECTOR, JC=J\_VECTOR and KC=K\_VECTOR.

---

## NO\_SET\_FILE

### ***Purpose***

Allows the posted \*.set file to be created or not.

### ***Syntax***

**:NO\_SET\_FILE=TRUE or FALSE**

### ***Comments***

This command should be set to FALSE if you want the \*.set file to be created every time you post. If you do not want the \*.set file to be created then this needs to be set to TRUE.

---

# TRAPDOOR

## ***Purpose***

Allows post to create an auto Open Chute to be attached to a closed boundary.

## ***Command***

**:TRAPDOOR=FALSE, DROP or TILT**

## ***Comments***

This command should be set to FALSE if you do not have a trap door. If you have a trap door you can set this to DROP or TILT depending on the style of the machines trap door. Only one style of trap door can be set per post.

---

## MOVE\_CLAMP

### ***Purpose***

Allows punch system to move a clamp manually and then passes information to the post for output.

### ***Command***

**:MOVE\_CLAMP=TRUE or FALSE**

### ***Comments***

This command should be set to FALSE if you do not have a machine that supports commands to move the clamps.

Set this command to TRUE if you have a machine that supports output code to move the clamps.

When you trigger a clamp move it is then passed to the punch post calc section CALC\_MOVE\_CLAMP. It stores the clamp positioned amount in absolute numbers only in post system variables CLAMP1\_POSITION, CLAMP2\_POSITION, CLAMP3\_POSITION, CLAMP4\_POSITION. Up to 4 clamps are supported for this command.

---

# SORTER\_ARM

## ***Purpose***

Allows Laser, Plasma or Punch system to have sorter arm options available in ProCAM 2D.

## ***Command***

**:SORTER\_ARM=TRUE or FALSE**

## ***Comments***

This command should be set to FALSE if you do not have a machine that supports sorter arm commands.

Set this command to TRUE if you have a machine that supports output code to support sorter arm commands.

When you trigger a sorter arm move it does an auto attachment of “sorter arm pickup” attribute, then it attaches a “sorter arm release” attribute and then attaches a “sorter arm hit releases part” attribute. In the definition of each of these attributes is or should be defined an ATTRFUNC command to call a calc section for necessary output for this operation.

Available post variables are ARM\_PICKUP\_X, ARM\_PICKUP\_Y, ARM\_DESTINATION\_X, ARM\_DESTINATION\_Y, ARM\_OFFSET, ARM\_ACTIVE\_CUPS.

---

## MILL\_OD\_CYLINDRICAL

### ***Purpose***

Allows Lathe/Mill (Live C) machines to use special G-code for doing milling on the OD in CAMWorks only.

### ***Command***

**:MILL\_OD\_CYLINDRICAL=TRUE or FALSE**

### ***Comments***

This command should be set to FALSE if your machine does not support special G-code for doing milling on the OD.

Set this command to TRUE if you have a machine that supports the special G-code for doing milling on the OD.

When your machine supports this the post will output cylindrical G-code output. The reason for using this is the feedrate is in either IPM or MPPM and not degree minutes, which requires a different degree minute feedrate on almost every line of code. Another reason is it can have machine compensation added, plus it is shorter code to do the same operation then in degree minutes.

---

# MILL\_FACE\_POLAR

## ***Purpose***

Allows Lathe/Mill (Live C) machines to use special G-code for doing milling on the FACE in CAMWorks only.

## ***Command***

**:MILL\_FACE\_POLAR=TRUE or FALSE**

## ***Comments***

This command should be set to FALSE if your machine does not support special G-code for doing milling on the FACE.

Set this command to TRUE if you have a machine that supports the special G-code for doing milling on the FACE.

When your machine supports this the post will output polar G-code output. The reason for using this is the feedrate is in either IPM or MPPM and not degree minutes, which requires a different degree minute feedrate on almost every line of code. Another reason is it can have machine compensation added, plus it is shorter code to do the same operation then in degree minutes.

---

# MACROS\_ROTATE

## ***Purpose***

Allows Laser or Plasma controllers to utilize rotating a macro and then call it with a subroutine call.

## ***Command***

**:MACROS\_ROTATE=TRUE or FALSE**

## ***Comments***

This command should be set to FALSE if your machine does not support rotating a macro and then call it.

Most machines do not allow you to create a macro at lets say zero degrees and call it then rotate it at 90 degrees and call the same subroutine. Normally what has to happen is once you rotate the macro the system has to recreate another macro define and then call that macro. The post variable that holds the rotated angle is ROTATE\_ANGLE\_Z.



---

# DUAL\_SPINDLE

***Purpose***

Allows Turn and Mill-Turn systems to machine on different spindles.

***Syntax***

**:DUAL\_SPINDLE=TRUE or FALSE**

***Comments***

For more information, see DSPINDLE in Chapter 12.

LENGTH\_DIAM\_OFFSET\_FROM\_TOOL

---

## LENGTH\_DIAM\_OFFSET\_FROM\_TOOL

### ***Purpose***

Tells CAMWorks whether the post supports using the Coolant, Length comp and diameter comp from the tool definition or the post.

### ***Syntax***

**:LENGTH\_DIAM\_OFFSET\_FROM\_TOOL=TRUE or FALSE**

### ***Comments***

Supported in CAMWorks 2008 or later. Not supported in any ProCAM product.

---

## Chapter 12 Additional System Variables

---

This chapter contains system variables that were not documented previously and that are new in CAMWorks releases.

---

## System Variables

<i><b>Variable</b></i>	<i><b>Type</b></i>	<i><b>Usage</b></i>
<b>DRILL_RAPID_X</b>	METRIC DECIMAL	<p>Drilled cycle rapid plane in X, Y and Z.</p> <p>CAMWorks 2005 or later: this variable will output the world coordinate correctly if the post has <code>WORLD_POSITIONING</code> set to <code>TRUE</code>.</p> <p>ProCAM II 2004 or later: <code>WORLD_POSITIONING</code> does not need to be set in the post.</p>
<b>DRILL_RAPID_Y</b>	METRIC DECIMAL	<p>Drilled cycle rapid plane in X, Y and Z.</p> <p>CAMWorks 2005 or later: this variable will output the world coordinate correctly if the post has <code>WORLD_POSITIONING</code> set to <code>TRUE</code>.</p> <p>ProCAM II 2004 or later: <code>WORLD_POSITIONING</code> does not need to be set in the post.</p>
<b>DRILL_RAPID_Z</b>	METRIC DECIMAL	<p>Drilled cycle rapid plane in X, Y and Z.</p> <p>CAMWorks 2005 or later: this variable will output the world coordinate correctly if the post has <code>WORLD_POSITIONING</code> set to <code>TRUE</code>.</p> <p>ProCAM II 2004 or later: <code>WORLD_POSITIONING</code> does not need to be set in the post.</p>
<b>DRILL_CLEAR_X</b>	METRIC DECIMAL	<p>Drilled cycle clearance plane in X, Y and Z.</p> <p>CAMWorks 2005 or later: this variable will output the world coordinate correctly if the post has <code>WORLD_POSITIONING</code> set to <code>TRUE</code>.</p> <p>ProCAM II 2004 or later: <code>WORLD_POSITIONING</code> does not need to be set in the post.</p>
<b>DRILL_CLEAR_Y</b>	METRIC DECIMAL	<p>Drilled cycle clearance plane in X, Y and Z.</p> <p>CAMWorks 2005 or later: this variable will output the world coordinate correctly if the post has <code>WORLD_POSITIONING</code> set to <code>TRUE</code>.</p> <p>ProCAM II 2004 or later: <code>WORLD_POSITIONING</code> does not need to be set in the post.</p>
<b>DRILL_CLEAR_Z</b>	METRIC DECIMAL	<p>Drilled cycle clearance plane in X, Y and Z.</p> <p>CAMWorks 2005 or later: this variable will output the world coordinate correctly if the post has <code>WORLD_POSITIONING</code> set to <code>TRUE</code>.</p> <p>ProCAM II 2004 or later: <code>WORLD_POSITIONING</code> does not need to be set in the post.</p>

---

<b>Variable</b>	<b>Type</b>	<b>Usage</b>
<b>DRILL_FACE_X</b>	METRIC DECIMAL	<p>Drilled cycle face plane in X, Y and Z.</p> <p>CAMWorks 2005 or later: this variable will output the world coordinate correctly if the post has <code>WORLD_POSITIONING</code> set to <code>TRUE</code>.</p> <p>ProCAM II 2004 or later: <code>WORLD_POSITIONING</code> does not need to be set in the post.</p>
<b>DRILL_FACE_Y</b>	METRIC DECIMAL	<p>Drilled cycle face plane in X, Y and Z.</p> <p>CAMWorks 2005 or later: this variable will output the world coordinate correctly if the post has <code>WORLD_POSITIONING</code> set to <code>TRUE</code>.</p> <p>ProCAM II 2004 or later: <code>WORLD_POSITIONING</code> does not need to be set in the post.</p>
<b>DRILL_FACE_Z</b>	METRIC DECIMAL	<p>Drilled cycle face plane in X, Y and Z.</p> <p>CAMWorks 2005 or later: this variable will output the world coordinate correctly if the post has <code>WORLD_POSITIONING</code> set to <code>TRUE</code>.</p> <p>ProCAM II 2004 or later: <code>WORLD_POSITIONING</code> does not need to be set in the post.</p>
<b>DRILL_DEPTH_X</b>	METRIC DECIMAL	<p>Drilled cycle drill depth in X, Y and Z.</p> <p>CAMWorks 2005 or later: this variable will output the world coordinate correctly if the post has <code>WORLD_POSITIONING</code> set to <code>TRUE</code>.</p> <p>ProCAM II 2004 or later: <code>WORLD_POSITIONING</code> does not need to be set in the post.</p>
<b>DRILL_DEPTH_Y</b>	METRIC DECIMAL	<p>Drilled cycle drill depth in X, Y and Z.</p> <p>CAMWorks 2005 or later: this variable will output the world coordinate correctly if the post has <code>WORLD_POSITIONING</code> set to <code>TRUE</code>.</p> <p>ProCAM II 2004 or later: <code>WORLD_POSITIONING</code> does not need to be set in the post.</p>
<b>DRILL_DEPTH_Z</b>	METRIC DECIMAL	<p>Drilled cycle drill depth in X, Y and Z.</p> <p>CAMWorks 2005 or later: this variable will output the world coordinate correctly if the post has <code>WORLD_POSITIONING</code> set to <code>TRUE</code>.</p> <p>ProCAM II 2004 or later: <code>WORLD_POSITIONING</code> does not need to be set in the post.</p>

## System Variables

<b>Variable</b>	<b>Type</b>	<b>Usage</b>
<b>DRILL_SAFE_X</b>	METRIC DECIMAL	<p>Drilled cycle safe retract plane in X, Y and Z.</p> <p>CAMWorks 2005 or later: this variable will output the world coordinate correctly if the post has <code>WORLD_POSITIONING</code> set to <code>TRUE</code>.</p> <p>ProCAM II 2004 or later: <code>WORLD_POSITIONING</code> does not need to be set in the post.</p>
<b>DRILL_SAFE_Y</b>	METRIC DECIMAL	<p>Drilled cycle safe retract plane in X, Y and Z.</p> <p>CAMWorks 2005 or later: this variable will output the world coordinate correctly if the post has <code>WORLD_POSITIONING</code> set to <code>TRUE</code>.</p> <p>ProCAM II 2004 or later: <code>WORLD_POSITIONING</code> does not need to be set in the post.</p>
<b>DRILL_SAFE_Z</b>	METRIC DECIMAL	<p>Drilled cycle safe retract plane in X, Y and Z.</p> <p>CAMWorks 2005 or later: this variable will output the world coordinate correctly if the post has <code>WORLD_POSITIONING</code> set to <code>TRUE</code>.</p> <p>ProCAM II 2004 or later: <code>WORLD_POSITIONING</code> does not need to be set in the post.</p>
<b>DRILL_PECK_DEPTH_X</b>	METRIC DECIMAL	<p>Pecking cycle peck depth per peck in X, Y and Z. Works in conjunction with <code>SYS_CANNED(????)</code>. See <code>SYS_CANNED</code>.</p> <p>CAMWorks 2005 or later: this variable will output the world coordinate correctly if the post has <code>WORLD_POSITIONING</code> set to <code>TRUE</code>.</p> <p>ProCAM II 2004 or later: <code>WORLD_POSITIONING</code> does not need to be set in the post.</p>
<b>DRILL_PECK_DEPTH_Y</b>	METRIC DECIMAL	<p>Pecking cycle peck depth per peck in X, Y and Z. Works in conjunction with <code>SYS_CANNED(????)</code>. See <code>SYS_CANNED</code>.</p> <p>CAMWorks 2005 or later: this variable will output the world coordinate correctly if the post has <code>WORLD_POSITIONING</code> set to <code>TRUE</code>.</p> <p>ProCAM II 2004 or later: <code>WORLD_POSITIONING</code> does not need to be set in the post.</p>

<b>Variable</b>	<b>Type</b>	<b>Usage</b>
<b>DRILL_PEEK_DEPTH_Z</b>	METRIC DECIMAL	Pecking cycle peck depth per peck in X, Y and Z. Works in conjunction with SYS_CANNED(????). See SYS_CANNED. CAMWorks 2005 or later: this variable will output the world coordinate correctly if the post has WORLD_POSITIONING set to TRUE. ProCAM II 2004 or later: WORLD_POSITIONING does not need to be set in the post.
<b>DRILL_PEEK_RAPID_TO_X</b>	METRIC DECIMAL	Pecking cycle rapid back into hole clearance plane in X, Y and Z. Works in conjunction with SYS_CANNED(????). See SYS_CANNED. CAMWorks 2005 or later: this variable will output the world coordinate correctly if the post has WORLD_POSITIONING set to TRUE. ProCAM II 2004 or later: WORLD_POSITIONING does not need to be set in the post.
<b>DRILL_PEEK_RAPID_TO_Y</b>	METRIC DECIMAL	Pecking cycle rapid back into hole clearance plane in X, Y and Z. Works in conjunction with SYS_CANNED(????). See SYS_CANNED. CAMWorks 2005 or later: this variable will output the world coordinate correctly if the post has WORLD_POSITIONING set to TRUE. ProCAM II 2004 or later: WORLD_POSITIONING does not need to be set in the post.
<b>DRILL_PEEK_RAPID_TO_Z</b>	METRIC DECIMAL	Pecking cycle rapid back into hole clearance plane in X, Y and Z. Works in conjunction with SYS_CANNED(????). See SYS_CANNED. CAMWorks 2005 or later: this variable will output the world coordinate correctly if the post has WORLD_POSITIONING set to TRUE. ProCAM II 2004 or later: WORLD_POSITIONING does not need to be set in the post.

<b>Variable</b>	<b>Type</b>	<b>Usage</b>
<b>INIT_TOOL_LENGTH</b>	METRIC DECIMAL	<p>Stores the internal tool length that is defined in the tool definition and is used in the system calc section called</p> <p>:SECTION=CALC_TOOL_INITIALIZE. This is used when the machine is 4 or 5 axis and the head rotates and or tilts. In this case the posted output may need to have the numbers modified for tool length.</p> <p>Note: Can be used only with CAMWorks 2005 or ProCAM II 2004 or later versions.</p>
<b>ARC_DEVIATION</b>	METRIC DECIMAL	<p>Stores a deviation amount for doing a SYS_CANNED(5,???) arc breakup in 4 and 5 axis milling with arcs that are not on the top plane. This is also used in a CAMWorks Live C post for OD and FACE milling for arc breakup, but not using SYS_CANNED.</p> <p>If you are creating a new post and your machine does not support arcs on different planes, then you need to add the question "max_arc_dev" to either the Setup Info or operation questions to set the deviation amount. This value will then set the post variable ARC_DEVIATION to this value.</p> <p>:C: ARC_DEVIATION=max_arc_dev :C: SYS_CANNED(5,CALC_BREAK_ARC)</p>
<b>SHAPE_INSIDE</b>	INTEGER	<p>Stores a 0 or 1 and is used if the laser or plasma tool path is closed. If it stores a 0, then the tool path is to the outside of the geometry if it stores a 1, then the toolpath is to the inside of the geometry.</p>
<b>SHAPE_TYPE</b>	INTEGER	<p>This variable stores a 0, 1 or 2 and is used for checking the type of laser or plasma tool path. If it stores a 0, then the tool path is open. If it stores a 1, then the tool path is closed. If it stores a 2, then it is a full circle toolpath. The system constants that are associated are</p> <p>OPEN_SHAPE=0 CLOSED_SHAPE=1 CIRCLE_SHAPE=2</p>
<b>SHAPE_DIAMETER</b>	METRIC DECIMAL	<p>Stores the diameter of a laser or plasma toolpath if SHAPE_TYPE is set to CIRCLE_SHAPE.</p>



<b>Variable</b>	<b>Type</b>	<b>Usage</b>
<b>BOUNDARY_AREA</b>	METRIC DECIMAL	Stores the boundary area of a closed laser of plasma toolpath. Used in conjunction with the external database. If it is not a closed toolpath, then the value will be set to a -1.
<b>V_COMP</b>	INTEGER	Stores a 0 or 1 depending on whether the post has a header command :VECTOR_COMP set to TRUE and you are in CAMWorks 3 Axis cutting operations. It will store a 1 if you are in an Advanced cutting operation, if not then the value is 0.
<b>XC</b>	METRIC DECIMAL	Stores the X end point of a line movement in a CAMWorks 3 Axis cutting operation if the post has the :VECTOR_COMP header set to TRUE.
<b>YC</b>	METRIC DECIMAL	Stores the Y end point of a line movement in a CAMWorks 3 Axis cutting operation if the post has the :VECTOR_COMP header set to TRUE.
<b>ZC</b>	METRIC DECIMAL	Stores the Z end point of a line movement in a CAMWorks 3 Axis cutting operation if the post has the :VECTOR_COMP header set to TRUE.
<b>IC</b>	METRIC DECIMAL	Stores the vector of “I” in a line movement in a CAMWorks 3 Axis cutting operation if the post has the :VECTOR_COMP header set to TRUE.
<b>JC</b>	METRIC DECIMAL	Stores the vector of “J” in a line movement in a CAMWorks 3 Axis cutting operation if the post has the :VECTOR_COMP header set to TRUE.
<b>KC</b>	METRIC DECIMAL	This variable stores the vector of “K” in a line movement in a CAMWorks 3 Axis cutting operation if the post has the :VECTOR_COMP header set to TRUE.
<b>OPR_CLEARANCE</b>	INTEGER	Stores the EDM operation clearance amount.
<b>OPR_GLUE_STOP</b>	INTEGER	Stores the EDM Glue Stop option setting. If you are using a glue stop, it is set to 1 (YES) . If not, then it is set to 0 (NO) .
<b>OPR_GLUE_DISTANCE</b>	INTEGER	Stores the EDM Glue Stop distance amount. If you are using a glue stop, it is set the distance entered in the operation dialog box.

## System Variables

<b>Variable</b>	<b>Type</b>	<b>Usage</b>
<b>CLAMP1_POSITION</b>	METRIC DECIMAL	Stores the punch manual clamp move, which will be an absolute distance. This will be available only if the post has the post header command : MOVE_CLAMP=TRUE.
<b>CLAMP2_POSITION</b>	METRIC DECIMAL	Stores the punch manual clamp move, which will be an absolute distance. This will be available only if the post has the post header command : MOVE_CLAMP=TRUE.
<b>CLAMP3_POSITION</b>	METRIC DECIMAL	Stores the punch manual clamp move, which will be an absolute distance. This will be available only if the post has the post header command : MOVE_CLAMP=TRUE.
<b>CLAMP4_POSITION</b>	METRIC DECIMAL	Stores the punch manual clamp move, which will be an absolute distance. This will be available only if the post has the post header command : MOVE_CLAMP=TRUE.
<b>KOMBID</b>	CHARACTER	Stores CAMWorks assembly mode tool ID information per tool change.
<b>NEXT_KOMBID</b>	CHARACTER	Stores CAMWorks assembly mode tool ID information per tool change for the next tool to be output.
<b>NEXT_MOVE_KOMBID</b>	CHARACTER	Stores CAMWorks assembly mode tool ID information for the next movement.
<b>OPR_Z_ROTARY_RETRACT_PLANE</b>	METRIC DECIMAL	Stores the CAMWorks assembly mode operation global retract plane in Z axis when you are doing a 4 or 5 axis rotary position move to another plane. The value is also assigned to the last Z value (ABS_Z_END) just before you rotate to another plane.
<b>ARM_PICKUP_X</b>	METRIC DECIMAL	Stores the ProCAM 2D sorter arm operations pickup position in the X axis.
<b>ARM_PICKUP_Y</b>	METRIC DECIMAL	Stores the ProCAM 2D sorter arm operations pickup position in the Y axis.

<b>Variable</b>	<b>Type</b>	<b>Usage</b>
<b>ARM_DESTINATION_X</b>	METRIC DECIMAL	Stores the ProCAM 2D sorter arm operations destination position in the X axis
<b>ARM_DESTINATION_Y</b>	METRIC DECIMAL	Stores the ProCAM 2D sorter arm operations destination position in the Y axis.
<b>ARM_OFFSET</b>	METRIC DECIMAL	Stores the ProCAM 2D sorter arm operations offset distance.
<b>ARM_ACTIVE_CUPS</b>	CHARACTER	Stores the ProCAM 2D sorter arm operations active cups.
<b>OPR_POLAR</b>	INTEGER	<p>OPR_POLAR=TRUE or FALSE</p> <p>Set to FALSE if your machine does not support special G-code for doing milling on the OD and or FACE.</p> <p>Set this command to TRUE if you have a machine that supports the special G-code for milling on the OD and or FACE.</p> <p>When you are in a lathe/mill milling operation, you can select Fixed or Free for the Rotary Axis Mode on the NC tab. If you select Free, you can select Polar/Cylindrical interpolation. When you select the Polar/Cylindrical interpolation, this variable will store a value of 1. Otherwise, it is set to zero.</p>
<b>OPR_B_AXIS</b>	METRIC DECIMAL	Stores the B axis in an absolute angle from defined setup parameters. This variable can be used only in CAMWorks 2005 or later versions for lathe/mill (live c) combination machines
<b>INC_B_END</b>	METRIC DECIMAL	<p>Stores the B axis in an incremental angle from last position. This variable will be available in a future CAMWorks version that has implemented 5 axis cutting in the lathe/mill (live c) system.</p> <p>Options:</p> <p>N_INC_B_END N_ = next move</p> <p>P_INC_B_END P_ = previous move</p>

## System Variables

<b>Variable</b>	<b>Type</b>	<b>Usage</b>
<b>ABS_B_END</b>	METRIC DECIMAL	Stores the B axis in an absolute end angle from zero position. This variable will be available in a future CAMWorks version that has implemented 5 axis cutting in the lathe/mill (live c) system. Options: N_ABS_B_END N_ = next move P_ABS_B_END P_ = previous move
<b>ABS_B_START</b>	METRIC DECIMAL	Stores the B axis in an absolute start angle from zero position. This variable will be available in a future CAMWorks version that has implemented 5 axis cutting in the lathe/mill (live c) system. Options: N_ABS_B_START N_ = next move P_ABS_B_START P_ = previous move
<b>MCS_X_OFFSET</b>	METRIC DECIMAL	Stores the Machine Coordinate System as an absolute position in X using ProCAM II or CAMWorks assembly mode.
<b>MCS_Y_OFFSET</b>	METRIC DECIMAL	Stores the Machine coordinate System as an absolute position in Y using ProCAM II or CAMWorks assembly mode.
<b>MCS_Z_OFFSET</b>	METRIC DECIMAL	Stores the Machine coordinate System as an absolute position in Z using ProCAM II or CAMWorks assembly mode.
<b>LASER_DBPATH</b>	CHARACTER	Laser or Plasma: Stores the system database path. Is used with the OPENDB command. It stores either the Metric database path or the English database path depending on the current system selected when posting.
<b>ENGLISH_LASERDB</b>	CHARACTER	Laser or Plasma: Stores the system database path. Is used with the OPENDB command. It stores the English database path.
<b>METRIC_LASERDB</b>	CHARACTER	Laser or Plasma: Stores the system database path. Is used with the OPENDB command. It stores the Metric database path.

<b>Variable</b>	<b>Type</b>	<b>Usage</b>
<b>LD_METRIC</b>	CHARACTER	Laser or Plasma: When using the standard database, this variable stores what units the database is using. Either English or Metric. The file for English is FABDBENGLISH.MDB and for Metric FABDBMETRIC.MDB. These files are in the \procad\defaults folder. The post header command called <code>LASER_PLASMA_CUT_DATA</code> must be set to <code>TRUE</code> .
<b>LD_MATERIAL</b>	CHARACTER	Laser or Plasma: When using the standard database, this variable stores what Material type is being used. The file for English is FABDBENGLISH.MDB and for Metric FABDBMETRIC.MDB. These files are in the \procad\defaults folder. The post header command called <code>LASER_PLASMA_CUT_DATA</code> must be set to <code>TRUE</code> .
<b>LD_THICKNESS</b>	DECIMAL	Laser or Plasma: When using the standard database, this variable stores what the Material thickness is. The file for English is FABDBENGLISH.MDB and for Metric FABDBMETRIC.MDB. These files are in the \procad\defaults folder. The post header command called <code>LASER_PLASMA_CUT_DATA</code> must be set to <code>TRUE</code> .

## System Variables

<b>Variable</b>	<b>Type</b>	<b>Usage</b>
<b>LD_DATA_GROUP</b>	CHARACTER	<p>Laser or Plasma: When using the standard database, this variable stores the Data Group. These are the different data groups:</p> <p>General  Pierce Conditions  Area size 1 Cut Conditions  Area size 2 Cut Conditions  Area size 3 Cut Conditions  Area size 4 Cut Conditions  Area size 5 Cut Conditions  Line Length 1 Cut Conditions  Line Length 2 Cut Conditions  Line Length 3 Cut Conditions  Line Length 4 Cut Conditions  Line Length 5 Cut Conditions  Arc Radius 1 Cut Conditions  Arc Radius 2 Cut Conditions  Arc Radius 3 Cut Conditions  Arc Radius 4 Cut Conditions  Arc Radius 5 Cut Conditions  External Boundary Parameters  Internal Boundary Parameters  Hole size 1 Parameters  Hole size 2 Parameters  Hole size 3 Parameters  Hole size 4 Parameters  Hole size 5 Parameters  Hole size 1 cut Conditions  Hole size 2 cut Conditions  Hole size 3 cut Conditions  Hole size 4 cut Conditions  Hole size 5 cut Conditions  Microjoints</p> <p>The file for English is FABDBENGLISH.MDB and for Metric FABDBMETRIC.MDB. These files are in the \procad\defaults folder. The post header command called <code>LASER_PLASMA_CUT_DATA</code> must be set to TRUE.</p>
<b>LD_PART_CLEARANCE</b>	DECIMAL	<p>Laser or Plasma: When using the standard database, this variable stores what the Part clearance is. The file for English is FABDBENGLISH.MDB and for Metric FABDBMETRIC.MDB. These files are in the \procad\defaults folder. The post header command called <code>LASER_PLASMA_CUT_DATA</code> must be set to TRUE.</p>

<b>Variable</b>	<b>Type</b>	<b>Usage</b>
<b>LD_SENSOR_RADIUS</b>	DECIMAL	Laser or Plasma: When using the standard database, this variable stores what the Sensor radius is. The file for English is FABDBENGLISH.MDB and for Metric FABDBMETRIC.MDB. These files are in the \procad\defaults folder. The post header command called <code>LASER_PLASMA_CUT_DATA</code> must be set to <code>TRUE</code> .
<b>LD_GO_AROUND_DISTANCE</b>	DECIMAL	Laser or Plasma: When using the standard database, this variable stores what the Go around distance is. The file for English is FABDBENGLISH.MDB and for Metric FABDBMETRIC.MDB. These files are in the \procad\defaults folder. The post header command called <code>LASER_PLASMA_CUT_DATA</code> must be set to <code>TRUE</code> .
<b>LD_AUTOBREAK_IN_LENGTH</b>	DECIMAL	Laser or Plasma: When using the standard database, this variable stores what the Auto break distance into the cut is. The file for English is FABDBENGLISH.MDB and for Metric FABDBMETRIC.MDB. These files are in the \procad\defaults folder. The post header command called <code>LASER_PLASMA_CUT_DATA</code> must be set to <code>TRUE</code> .
<b>LD_AUTOBREAK_OUT_LENGTH</b>	DECIMAL	Laser or Plasma: When using the standard database, this variable stores what the Auto break distance out of the cut is. The file for English is FABDBENGLISH.MDB and for Metric FABDBMETRIC.MDB. These files are in the \procad\defaults folder. The post header command called <code>LASER_PLASMA_CUT_DATA</code> must be set to <code>TRUE</code> .
<b>LD_SYSTEM_COMP</b>	INTEGER	<p>Laser or Plasma: When using the standard database, this variable stores what the System comp is:</p> <ul style="list-style-type: none"> <li>(3) for n/a</li> <li>(2) for Off</li> <li>(1) for On</li> </ul> <p>The file for English is FABDBENGLISH.MDB and for Metric FABDBMETRIC.MDB. These files are in the \procad\defaults folder. The post header command called <code>LASER_PLASMA_CUT_DATA</code> must be set to <code>TRUE</code>.</p>

## System Variables

<b>Variable</b>	<b>Type</b>	<b>Usage</b>
<b>LD_CC_MODE</b>	INTEGER	<p>Laser or Plasma: When using the standard database, this variable stores what the Cutting condition mode is:</p> <ul style="list-style-type: none"><li>(1) Area</li><li>(2) Line Length/Arc Radius</li></ul> <p>The file for English is FABDBENGLISH.MDB and for Metric FABDBMETRIC.MDB. These files are in the \procad\defaults folder. The post header command called <code>LASER_PLASMA_CUT_DATA</code> must be set to <code>TRUE</code>.</p>
<b>LD_DIRECTION</b>	INTEGER	<p>Laser or Plasma: When using the standard database, this variable stores what the Cutting direction is:</p> <ul style="list-style-type: none"><li>(1) Clockwise</li><li>(2) Counter Clockwise</li><li>(3) Original</li><li>(4) n/a</li></ul> <p>The file for English is FABDBENGLISH.MDB and for Metric FABDBMETRIC.MDB. These files are in the \procad\defaults folder. The post header command called <code>LASER_PLASMA_CUT_DATA</code> must be set to <code>TRUE</code>.</p>
<b>LD_START_POSITION</b>	INTEGER	<p>Laser or Plasma: When using the standard database, this variable stores what the where cutting is to start from:</p> <ul style="list-style-type: none"><li>(1) Auto</li><li>(2) Long edge midpoint</li><li>(3) n/a</li></ul> <p>The file for English is FABDBENGLISH.MDB and for Metric FABDBMETRIC.MDB. These files are in the \procad\defaults folder. The post header command called <code>LASER_PLASMA_CUT_DATA</code> must be set to <code>TRUE</code>.</p>
<b>LD_LIFTHEAD</b>	INTEGER	<p>Laser or Plasma: When using the standard database, this variable stores whether you need the Lift Head option or not:</p> <ul style="list-style-type: none"><li>(1) On</li><li>(2) Off</li><li>(3) n/a</li></ul> <p>The file for English is FABDBENGLISH.MDB and for Metric FABDBMETRIC.MDB. These files are in the \procad\defaults folder. The post header command called <code>LASER_PLASMA_CUT_DATA</code> must be set to <code>TRUE</code>.</p>



<b>Variable</b>	<b>Type</b>	<b>Usage</b>
<b>LD_START_AT_HOLE_CENTER</b>	INTEGER	<p>Laser or Plasma: When using the standard database, this variable stores if you want to start at hole center or not:</p> <ul style="list-style-type: none"> <li>(1) On</li> <li>(2) Off</li> <li>(3) n/a</li> </ul> <p>The file for English is FABDBENGLISH.MDB and for Metric FABDBMETRIC.MDB. These files are in the \procad\defaults folder. The post header command called <code>LASER_PLASMA_CUT_DATA</code> must be set to <code>TRUE</code>.</p>
<b>LD_LEADIN_MODE</b>	INTEGER	<p>Laser or Plasma: When using the standard database, this variable stores what type of leadin is used.</p> <ul style="list-style-type: none"> <li>(1) Arc</li> <li>(2) n/a</li> <li>(3) None</li> <li>(4) Parallel</li> <li>(5) Perpendicular</li> </ul> <p>The file for English is FABDBENGLISH.MDB and for Metric FABDBMETRIC.MDB. These files are in the \procad\defaults folder. The post header command called <code>LASER_PLASMA_CUT_DATA</code> must be set to <code>TRUE</code>.</p>
<b>LD_LEADIN_LENGTH</b>	DECIMAL	<p>Laser or Plasma: When using the standard database, this variable stores what the leadin distance is. The file for English is FABDBENGLISH.MDB and for Metric FABDBMETRIC.MDB. These files are in the \procad\defaults folder. The post header command called <code>LASER_PLASMA_CUT_DATA</code> must be set to <code>TRUE</code>.</p>
<b>LD_LEADIN_ANGLE</b>	DECIMAL	<p>Laser or Plasma: When using the standard database, this variable stores what the leadin angle is. The file for English is FABDBENGLISH.MDB and for Metric FABDBMETRIC.MDB. These files are in the \procad\defaults folder. The post header command called <code>LASER_PLASMA_CUT_DATA</code> must be set to <code>TRUE</code>.</p>

## System Variables

<b>Variable</b>	<b>Type</b>	<b>Usage</b>
<b>LD_LEADIN_ARC_RADIUS</b>	INTEGER	Laser or Plasma: When using the standard database, this variable stores what the leadin arc radius is. The file for English is FABDBENGLISH.MDB and for Metric FABDBMETRIC.MDB. These files are in the \procad\defaults folder. The post header command called <code>LASER_PLASMA_CUT_DATA</code> must be set to <code>TRUE</code> .
<b>LD_LEADIN_ARC_ANGLE</b>	DECIMAL	Laser or Plasma: When using the standard database, this variable stores what the leadin arc angle is. The file for English is FABDBENGLISH.MDB and for Metric FABDBMETRIC.MDB. These files are in the \procad\defaults folder. The post header command called <code>LASER_PLASMA_CUT_DATA</code> must be set to <code>TRUE</code> .
<b>LD_OVERLAP</b>	INTEGER	Laser or Plasma: When using the standard database, this variable stores what the leadin overlap is. The file for English is FABDBENGLISH.MDB and for Metric FABDBMETRIC.MDB. These files are in the \procad\defaults folder. The post header command called <code>LASER_PLASMA_CUT_DATA</code> must be set to <code>TRUE</code> .
<b>LD_LEADOUT_MODE</b>	INTEGER	<p>Laser or Plasma: When using the standard database, this variable stores what type of leadout is used.</p> <ul style="list-style-type: none"><li>(1) Clockwise</li><li>(2) Counter Clockwise</li><li>(3) Original</li><li>(4) None</li></ul> <p>The file for English is FABDBENGLISH.MDB and for Metric FABDBMETRIC.MDB. These files are in the \procad\defaults folder. The post header command called <code>LASER_PLASMA_CUT_DATA</code> must be set to <code>TRUE</code>.</p>

<b>Variable</b>	<b>Type</b>	<b>Usage</b>
<b>LD_LEADOUT_LENGTH</b>	DECIMAL	Laser or Plasma: When using the standard database, this variable stores what the leadout length is. The file for English is FABDBENGLISH.MDB and for Metric FABDBMETRIC.MDB. These files are in the \procad\defaults folder. The post header command called <code>LASER_PLASMA_CUT_DATA</code> must be set to <code>TRUE</code> .
<b>LD_LEADOUT_ANGLE</b>	DECIMAL	Laser or Plasma: When using the standard database, this variable stores what the leadout angle is. The file for English is FABDBENGLISH.MDB and for Metric FABDBMETRIC.MDB. These files are in the \procad\defaults folder. The post header command called <code>LASER_PLASMA_CUT_DATA</code> must be set to <code>TRUE</code> .
<b>LD_LEADOUT_ARC_RADIUS</b>	DECIMAL	Laser or Plasma: When using the standard database, this variable stores what the leadout arc radius is. The file for English is FABDBENGLISH.MDB and for Metric FABDBMETRIC.MDB. These files are in the \procad\defaults folder. The post header command called <code>LASER_PLASMA_CUT_DATA</code> must be set to <code>TRUE</code> .
<b>LD_LEADOUT_ARC_ANGLE</b>	DECIMAL	Laser or Plasma: When using the standard database, this variable stores what the leadout arc angle is. The file for English is FABDBENGLISH.MDB and for Metric FABDBMETRIC.MDB. These files are in the \procad\defaults folder. The post header command called <code>LASER_PLASMA_CUT_DATA</code> must be set to <code>TRUE</code> .
<b>LD_LEADOUT_OVERLAP</b>	DECIMAL	Laser or Plasma: When using the standard database, this variable stores what the leadout overlap distance is. The file for English is FABDBENGLISH.MDB and for Metric FABDBMETRIC.MDB. These files are in the \procad\defaults folder. The post header command called <code>LASER_PLASMA_CUT_DATA</code> must be set to <code>TRUE</code> .

## System Variables

<b>Variable</b>	<b>Type</b>	<b>Usage</b>
<b>LD_SIZE</b>	DECIMAL	Laser or Plasma: When using the standard database, this variable stores the size of the Area, Line Length, Arc Radius and Holes. The file for English is FABDBENGLISH.MDB and for Metric FABDBMETRIC.MDB. These files are in the \procad\defaults folder. The post header command called <code>LASER_PLASMA_CUT_DATA</code> must be set to <code>TRUE</code> .
<b>LD_COLOR</b>	INTEGER	<p>Laser or Plasma: When using the standard database, this variable stores the color of the different path sizes and shapes.</p> <ul style="list-style-type: none"><li>(1) BLACK</li><li>(2) BLUE</li><li>(3) GREEN</li><li>(4) CYAN</li><li>(5) RED</li><li>(6) MAGENTA</li><li>(7) BROWN</li><li>(8) GRAY</li><li>(9) WHITE</li><li>(10) LTBLUE</li><li>(11) LTGREEN</li><li>(12) LTCYAN</li><li>(13) LTRED</li><li>(14) LTMAGEN</li><li>(15) YELLOW</li><li>(16) HIWHITE</li></ul> <p>The file for English is FABDBENGLISH.MDB and for Metric FABDBMETRIC.MDB. These files are in the \procad\defaults folder. The post header command called <code>LASER_PLASMA_CUT_DATA</code> must be set to <code>TRUE</code>.</p>
<b>LD_SUBROUTINE_NUMBER</b>	INTEGER	Laser or Plasma: When using the standard database, this variable stores the subroutine number used. The file for English is FABDBENGLISH.MDB and for Metric FABDBMETRIC.MDB. These files are in the \procad\defaults folder. The post header command called <code>LASER_PLASMA_CUT_DATA</code> must be set to <code>TRUE</code> .
<b>LD_FEEDRATE</b>	DECIMAL	Laser or Plasma: When using the standard database, this variable stores the federate used. The file for English is FABDBENGLISH.MDB and for Metric FABDBMETRIC.MDB. These files are in the \procad\defaults folder. The post header command called <code>LASER_PLASMA_CUT_DATA</code> must be set to <code>TRUE</code> .

<b>Variable</b>	<b>Type</b>	<b>Usage</b>
<b>LD_FEEDRATE_PERCENT</b>	INTEGER	Laser or Plasma: When using the standard database, this variable stores the federate percentage used. The file for English is FABDBENGLISH.MDB and for Metric FABDBMETRIC.MDB. These files are in the \procad\defaults folder. The post header command called <code>LASER_PLASMA_CUT_DATA</code> must be set to <code>TRUE</code> .
<b>LD_POWER_LEVEL</b>	INTEGER	Laser or Plasma: When using the standard database, this variable stores the size of the Area, Line Length, Arc Radius and Holes. The file used for English is FABDBENGLISH.MDB and for Metric FABDBMETRIC.MDB. These files are in the \procad\defaults folder. The post header command called <code>LASER_PLASMA_CUT_DATA</code> must be set to <code>TRUE</code> .
<b>LD_ASSIST_GAS</b>	CHARACTER	<p>Laser or Plasma: When using the standard database, this variable stores the assist gas type.</p> <ul style="list-style-type: none"> <li>(1) Oxygen</li> <li>(2) Nitrogen</li> <li>(3) Carbon Dioxide</li> <li>(4) n/a</li> </ul> <p>The file for English is FABDBENGLISH.MDB and for Metric FABDBMETRIC.MDB. These files are in the \procad\defaults folder. The post header command called <code>LASER_PLASMA_CUT_DATA</code> must be set to <code>TRUE</code>.</p>
<b>LD_GAS_PRESSURE</b>	INTEGER	Laser or Plasma: When using the standard database, this variable stores the gas pressure used. The file for English is FABDBENGLISH.MDB and for Metric FABDBMETRIC.MDB. These files are in the \procad\defaults folder. The post header command called <code>LASER_PLASMA_CUT_DATA</code> must be set to <code>TRUE</code> .

## System Variables

<b>Variable</b>	<b>Type</b>	<b>Usage</b>
<b>LD_2ND_ASSIST_GAS</b>	CHARACTER	<p>Laser or Plasma: When using the standard database, this variable stores the second assist gas type:</p> <ul style="list-style-type: none"><li>(1) Oxygen</li><li>(2) Nitrogen</li><li>(3) Carbon Dioxide</li><li>(4) n/a</li></ul> <p>The file for English is FABDBENGLISH.MDB and for Metric FABDBMETRIC.MDB. These files are in the \procad\defaults folder. The post header command called <code>LASER_PLASMA_CUT_DATA</code> must be set to <code>TRUE</code>.</p>
<b>LD_2ND_GAS_PRESSURE</b>	INTEGER	<p>Laser or Plasma: When using the standard database, this variable stores the second gas pressure used. The file for English is FABDBENGLISH.MDB and for Metric FABDBMETRIC.MDB. These files are in the \procad\defaults folder. The post header command called <code>LASER_PLASMA_CUT_DATA</code> must be set to <code>TRUE</code>.</p>
<b>LD_LASER_MODE</b>	CHARACTER	<p>Laser or Plasma: When using the standard database, this variable stores the laser mode type:</p> <ul style="list-style-type: none"><li>(3) Continuous Wave</li><li>(4) Dynamic Power</li><li>(1) Gated Pulse</li><li>(5) n/a</li><li>(2) Super Power</li></ul> <p>The file for English is FABDBENGLISH.MDB and for Metric FABDBMETRIC.MDB. These files are in the \procad\defaults folder. The post header command called <code>LASER_PLASMA_CUT_DATA</code> must be set to <code>TRUE</code>.</p>
<b>LD_FREQUENCY</b>	INTEGER	<p>Laser or Plasma: When using the standard database, this variable stores the laser frequency. The file for English is FABDBENGLISH.MDB and for Metric FABDBMETRIC.MDB. These files are in the \procad\defaults folder. The post header command called <code>LASER_PLASMA_CUT_DATA</code> must be set to <code>TRUE</code>.</p>

<b>Variable</b>	<b>Type</b>	<b>Usage</b>
<b>LD_DUTY_CYCLE</b>	INTEGER	Laser or Plasma: When using the standard database, this variable stores the laser duty cycle. The file for English is FABDBENGLISH.MDB and for Metric FABDBMETRIC.MDB. These files are in the \procad\defaults folder. The post header command called <code>LASER_PLASMA_CUT_DATA</code> must be set to <code>TRUE</code> .
<b>LD_COOLANT_MODE</b>	CHARACTER	<p>Laser or Plasma: When using the standard database, this variable stores the coolant type used.</p> <p>(1) On (2) Off (3) n/a</p> <p>The file for English is FABDBENGLISH.MDB and for Metric FABDBMETRIC.MDB. These files are in the \procad\defaults folder. The post header command called <code>LASER_PLASMA_CUT_DATA</code> must be set to <code>TRUE</code>.</p>
<b>LD_OFFSET_VALUE</b>	INTEGER	Laser or Plasma: When using the standard database, this variable stores the offset value used. The file for English is FABDBENGLISH.MDB and for Metric FABDBMETRIC.MDB. These files are in the \procad\defaults folder. The post header command called <code>LASER_PLASMA_CUT_DATA</code> must be set to <code>TRUE</code> .
<b>LD_DURATION</b>	INTEGER	Laser or Plasma: When using the standard database, this variable stores the duration amount. The file for English is FABDBENGLISH.MDB and for Metric FABDBMETRIC.MDB. These are in the \procad\defaults folder. The post header command called <code>LASER_PLASMA_CUT_DATA</code> must be set to <code>TRUE</code> .

## System Variables

<b>Variable</b>	<b>Type</b>	<b>Usage</b>
<b>LD_START_AT_MICROJOINT</b>	INTEGER	<p>Laser or Plasma: When using the standard database, this variable stores whether you are using microjoints:</p> <ul style="list-style-type: none"><li>(1) On</li><li>(2) Off</li><li>(3) n/a</li></ul> <p>The file for English is FABDBENGLISH.MDB and for Metric FABDBMETRIC.MDB. These are in the \procad\defaults folder. The post header command called <code>LASER_PLASMA_CUT_DATA</code> must be set to <code>TRUE</code>.</p>
<b>LD_COMMENT</b>	CHARACTER	<p>Laser or Plasma: When using the standard database, this variable stores the comment. The file for English is FABDBENGLISH.MDB and for Metric FABDBMETRIC.MDB. These are in the \procad\defaults folder. The post header command called <code>LASER_PLASMA_CUT_DATA</code> must be set to <code>TRUE</code>.</p>
<b>LD_END_AT_HOLE_CENTER</b>	INTEGER	<p>Laser or Plasma: When using the standard database, this variable stores whether you are ending at hole center or not.</p> <ul style="list-style-type: none"><li>(1) On</li><li>(2) Off</li><li>(3) n/a</li></ul> <p>The file for English is FABDBENGLISH.MDB and for Metric FABDBMETRIC.MDB. These are in the \procad\defaults folder. The post header command called <code>LASER_PLASMA_CUT_DATA</code> must be set to <code>TRUE</code>.</p>
<b>LD_OPTION_START_POSITION</b>	INTEGER	<p>Laser or Plasma: When using the standard database, this variable stores the optional start position. The file for English is FABDBENGLISH.MDB and for Metric FABDBMETRIC.MDB. These are in the \procad\defaults folder. The post header command called <code>LASER_PLASMA_CUT_DATA</code> must be set to <code>TRUE</code>.</p>



<b>Variable</b>	<b>Type</b>	<b>Usage</b>
<b>LD_CHOICE_START_POSITION</b>	INTEGER	Laser or Plasma: When using the standard database, this variable stores the user's choice start position. The file for English is FABDBENGLISH.MDB and for Metric FABDBMETRIC.MDB. These are in the \procad\defaults folder. The post header command called <code>LASER_PLASMA_CUT_DATA</code> must be set to <code>TRUE</code> .
<b>LD_USER_INT01 through LD_USER_INT10</b>	INTEGER	Laser or Plasma: When using the standard database, this variable stores the expansion integer fields. The file for English is FABDBENGLISH.MDB and for Metric FABDBMETRIC.MDB. These are in the \procad\defaults folder. The post header command called <code>LASER_PLASMA_CUT_DATA</code> must be set to <code>TRUE</code> .
<b>LD_USER_DBL01 through LD_USER_DBL10</b>	DECIMAL	When using the Plasma or Laser standard database this variable stores the expansion decimal fields. The file for English is FABDBENGLISH.MDB and for Metric FABDBMETRIC.MDB. These files are in the \procad\defaults folder. The post header command called <code>LASER_PLASMA_CUT_DATA</code> must be set to <code>TRUE</code> .
<b>RUNNING_SYSTEM</b>	INTEGER	The current running system. <code>RUNNING_SYSTEM=CAMWORKS</code> <code>RUNNING_SYSTEM=PROCAM_2D</code> <code>RUNNING_SYSTEM=PROCAM_3D</code>  <code>CAMWORKS=1</code> <code>PROCAM_2D=2</code> <code>PROCAM_3D=3</code>
<b>SKIM_CUT</b>	INTEGER	ProCAM 2D EDM: Stores the EDM skim cut. <code>SKIM_CUT=CURRENT</code> <code>SKIM_CUT=PREVIOUS</code> <code>SKIM_CUT=NEXT</code>  <code>CURRENT=0</code> <code>PREVIOUS=1</code> <code>NEXT=2</code>
<b>TAB_CUT</b>	INTEGER	ProCAM 2D EDM: Stores the EDM tan cut. <code>TAB_CUT=CURRENT</code> <code>TAB_CUT=PREVIOUS</code> <code>TAB_CUT=NEXT</code>

## System Variables

<b>Variable</b>	<b>Type</b>	<b>Usage</b>
<b>PART_TOTAL_SKIM_CUTS</b>	INTEGER	ProCAM 2D: Stores the total number of skim cuts in a part file.
<b>PART_TOTAL_TAB_CUTS</b>	INTEGER	ProCAM 2D: Stores the total number of tab cuts in a part file.
<b>OPER_TOTAL_SKIM_CUTS</b>	INTEGER	ProCAM 2D: Stores the total number of skim cuts in an operation.
<b>OPER_TOTAL_TAB_CUTS</b>	INTEGER	ProCAM 2D: Stores the total number of tab cuts in an operation.
<b>OPER_CURRENT_SKIM_CUT</b>	INTEGER	ProCAM 2D EDM: Stores the EDM operation type. SKIM_CUT=CURRENT SKIM_CUT=PREVIOUS SKIM_CUT=NEXT CURRENT=0 PREVIOUS=1 NEXT=2
<b>OPER_CURRENT_TAB_CUT</b>	INTEGER	ProCAM 2D EDM: Stores the EDM operation type. TAB_CUT=CURRENT TAB_CUT=PREVIOUS TAB_CUT=NEXT CURRENT=0 PREVIOUS=1 NEXT=2
<b>DSPINDLE</b>	INTEGER	CAMWorks and ProCAM 2D: Stores whether you are using a main or sub spindle in the lathe system. DSPINDLE=MAIN_SPINDLE DSPINDLE=SUB_SPINDLE MAIN_SPINDLE =1 SUB_SPINDLE =2
<b>POST_PATH</b>	CHARACTER	Stores the posted output path.
<b>OPR_LATHE_TAPPING</b>	INTEGER	ProCAM 2D Lathe: Stores whether you need to do a lathe tapping cycle while in the lathe drilling operation. OPR_LATHE_TAPPING=FALSE OPR_LATHE_TAPPING=TRUE
<b>LASER_MATERIAL</b>	CHARACTER	Stores the material name from the setup information and updates the material name in the laser, plasma database, so it matches.

<b>Variable</b>	<b>Type</b>	<b>Usage</b>
<b>LASER_THICKNESS</b>	CHARACTER	Stores the sheet thickness from the setup information and updates the sheet thickness in the laser, plasma database, so it matches.
<b>ARM_SORTER_DBPATH</b>	CHARACTER	Stores Punch system sorter arm database path. Is used with the OPENDB command. It stores either the Metric or English database path depending on the current system selected when posting.
<b>ARM_ENGLISH_LASERDB</b>	CHARACTER	Stores the Punch system sorter arm English database path. Is used with the OPENDB command.
<b>ARM_METRIC_LASERDB</b>	CHARACTER	Stores the Punch system sorter arm Metric database path. Is used with the OPENDB command.
<b>OPR_LATHE_RETRACT_TYPE</b>	INTEGER	Stores the retract type when using Lathe canned cycles: (1) Z then X (2) X then Z (3) Direct
<b>TOOL_SPEC_NAME</b>	CHARACTER	ProCAM 2D Punch: Stores the special tool name for the current tool station. There will be a name for each station that uses a special tool. This variable can also be used in the GETTOOLS(???) command.
<b>TOOL_SPEC_PATH</b>	CHARACTER	ProCAM 2D Punch: Stores the special tool path for the current tool station. There will be a path for each station that uses a special tool. This variable can also be used in the GETTOOLS(???) command.
<b>ABS_PICK_LOCATION</b>	DECIMAL	CAMWorks: Stores the absolute value of the Lathe Sub Spindle transfer location along the Z axis.
<b>INC_PICK_LOCATION</b>	DECIMAL	CAMWorks: Stores the incremental value of the Lathe Sub Spindle transfer location along the Z axis.
<b>TRANSFER_DISTANCE</b>	DECIMAL	CAMWorks: Stores the distance traveled when a transfer takes place between the lathe main and sub spindle along the Z axis.

<b>Variable</b>	<b>Type</b>	<b>Usage</b>
<b>OPR_LATHE_APPROACH_STRATEGY</b>	INTEGER	<p>CAMWorks 2007 and later: Stores the current lathe operation's approach strategy to the start of the cut.</p> <p>OPR_LATHE_APPROACH_STRATEGY=Z_THEN_X  OPR_LATHE_APPROACH_STRATEGY=X_THEN_Z  OPR_LATHE_APPROACH_STRATEGY=DIRECT  OPR_LATHE_APPROACH_STRATEGY=AUTO</p> <p>Z_THEN_X=1  X_THEN_Z=2  DIRECT=3</p> <p>AUTO=no value (CAMWorks attempts to maintain the Approach strategy where possible. For approach moves, CAMWorks will attempt to avoid the WIP model by the specified Clearance. This option allows you to set the Approach type to automatically avoid collisions with the WIP model.)</p>
<b>OPR_LATHE_RETRACT_STRATEGY</b>	INTEGER	<p>CAMWorks 2007 and later: Stores the current lathe operation's retract strategy at the end of the cut.</p> <p>OPR_LATHE_RETRACT_STRATEGY=Z_THEN_X  OPR_LATHE_RETRACT_STRATEGY=X_THEN_Z  OPR_LATHE_RETRACT_STRATEGY=DIRECT  OPR_LATHE_RETRACT_STRATEGY=AUTO</p> <p>Z_THEN_X=1  X_THEN_Z=2  DIRECT=3</p> <p>AUTO=no value (CAMWorks attempts to maintain the Retract strategy where possible. For retract moves, CAMWorks will attempt to avoid the WIP model by the specified Clearance. This option allows you to set the Retract type to automatically avoid collisions with the WIP model.)</p>
<b>CAM_MOVE_FLAG</b>	INTEGER	<p>CAMWorks 2007 and later: Shows what type of lathe move you are doing.</p> <pre> :C: IF FLAGGED (CAM_MOVE_FLAG, CAM_APPROACH) =TRUE THEN :C:   IF FLAGGED (CAM_MOVE_FLAG, CAM_MOVE_X) =TRUE AND :C:   FLAGGED (CAM_MOVE_FLAG, CAM_MOVE_Z) =TRUE THEN :C:   CALL (RAPID_MOVE_LATHE) :C:   RETURN :C:   ELSE :C:     IF FLAGGED (CAM_MOVE_FLAG, CAM_MOVE_X) =TRUE THEN :C:       CALL (RAPID_MOVE_LATHE_X) :C:       RETURN :C:     ELSE :C:       IF FLAGGED (CAM_MOVE_FLAG, CAM_MOVE_Z) =TRUE THEN </pre>

**Variable****Type****Usage**

```
:C:          CALL (RAPID_MOVE_LATHE_Z)
:C:          RETURN
:C:          ENDIF
:C:          ENDIF
:C:          ENDIF
:C:          ENDIF
:C:          ENDIF
```

```
CAM_MOVE_FLAG=CAM_RAPID
CAM_MOVE_FLAG=CAM_APPROACH
CAM_MOVE_FLAG=CAM_RETRACT
CAM_MOVE_FLAG=CAM_MOVE_X
CAM_MOVE_FLAG=CAM_MOVE_Y
CAM_MOVE_FLAG=CAM_MOVE_Z
CAM_MOVE_FLAG=CAM_LEADIN
CAM_MOVE_FLAG=CAM_LEADOUT
```

## Bit Values

```
CAM_RAPID=1
CAM_APPROACH =2
CAM_RETRACT=4
CAM_MOVE_X=8
CAM_MOVE_Y=16
CAM_MOVE_Z=32
CAM_LEADIN=64
CAM_LEADOUT=128
```

## Options

```
N_CAM_MOVE_FLAG      N_ = next move
P_CAM_MOVE_FLAG      P_ = previous move
```

**TOOL\_ORIENT**

INTEGER

CAMWorks 2007 and later: Stores the current tool orientation.

```
TOOL_ORIENT=UP_RIGHT
TOOL_ORIENT=UP_LEFT
TOOL_ORIENT=DOWN_RIGHT
TOOL_ORIENT=DOWN_LEFT
TOOL_ORIENT=LEFT_UP
TOOL_ORIENT=LEFT_DOWN
TOOL_ORIENT=RIGHT_UP
TOOL_ORIENT=RIGHT_DOWN
```

```
UP_RIGHT=1
UP_LEFT=2
DOWN_RIGHT=3
DOWN_LEFT=4
LEFT_UP=5
LEFT_DOWN=6
RIGHT_UP=7
RIGHT_DOWN=8
```

**CAMWORKS\_VER**

INTEGER

CAMWorks: Stores the version you are running. Any version below CW2006 will register as zero. As new commands are inserted into the posting system, they will be available when new CAMWorks versions are released. If a post was to run on two different versions of CAMWorks, then you may need to use this variable depending on the commands used in the post.

## System Variables

### Variable

### Type

### Usage

CAMWORKS\_VER= CAM\_REV2006  
 CAMWORKS\_VER= CAM\_REV2006EX  
 CAMWORKS\_VER= CAM\_REV2007  
 CAMWORKS\_VER= CAM\_REV2008  
 CAMWORKS\_VER= CAM\_REV2009

CAM\_REV2006=0  
 CAM\_REV2006EX=0  
 CAM\_REV2007=7  
 CAM\_REV2007\_SP1=71  
 CAM\_REV2007\_SP2=72  
 CAM\_REV2007\_SP2\_2=73  
 CAM\_REV2007\_SP3=74  
 CAM\_REV2007\_SP3\_1=75  
 CAM\_REV2007\_SP4\_1=76  
 CAM\_REV2008=80  
 CAM\_REV2008\_SP1=81  
 CAM\_REV2009=90  
 CAM\_REV2009\_SP1=91

```
:C: IF CAMWORKS_VER>CAM_REV2006EX THEN
:C:   IF
FLAGGED(CAM_MOVE_FLAG,CAM_APPROACH)=TRUE THEN
:C:     IF FLAGGED(CAM_MOVE_FLAG,
CAM_MOVE_X)=TRUE AND
:C:       FLAGGED(CAM_MOVE_FLAG, CAM_MOVE_Z)=TRUE
THEN
:C:       CALL(RAPID_MOVE_LATHE)
:C:       RETURN
:C:     ELSE
:C:       IF
FLAGGED(CAM_MOVE_FLAG,CAM_MOVE_X)=TRUE THEN
:C:         CALL(RAPID_MOVE_LATHE_X)
:C:         RETURN
:C:       ELSE
:C:         IF
FLAGGED(CAM_MOVE_FLAG,CAM_MOVE_Z)=TRUE THEN
:C:           CALL(RAPID_MOVE_LATHE_Z)
:C:           RETURN
:C:         ENDIF
:C:       ENDIF
:C:     ENDIF
:C:   ENDIF
:C: ENDIF
```

### OPR\_SHIFT\_TYPE

### INTEGER

Stores the shift orientation selected in the Lathe finish grooving cycle. This will be the called section at the time of a shift "CALC\_SHIFT\_TOOL\_LATHE." This section will be called for selecting cutter comp direction "CALC\_CUTTER\_COMP\_LATHE". Supported in CAMWorks 2007 and later. Not supported in any ProCAM product.

<b>Variable</b>	<b>Type</b>	<b>Usage</b>
<b>DRIVE_POINT_TYPE</b> <b>P_DRIVE_POINT_TYPE</b> <b>N_DRIVE_POINT_TYPE</b>	INTEGER	<p>This variable stores the drive point of a finish grooving cycle tool during posting. This variable will be updated in a called section CALC_SHIFT_TOOL_LATHE.</p> <p>DRIVE_CENTER=0  DRIVE_RIGHT=1  DRIVE_LEFT=2  DRIVE_TOOL_NOSE=3</p> <p>Supported in CAMWorks 2007 and later. Not supported in any ProCAM product.</p>
<b>TOUCHOFF_POINT_TYPE</b> <b>P_TOUCHOFF_POINT_TYPE</b> <b>N_TOUCHOFF_POINT_TYPE</b>	INTEGER	<p>Stores the current touch off point of a finish grooving cycle tool during posting. This is the current active edge when you are in a called section CALC_SHIFT_TOOL_LATHE.</p> <p>PROG_POINT_CENTER=0  PROG_POINT_PRIMARY=1  PROG_POINT_SECONDARY=2</p> <p>Supported in CAMWorks 2007 and later. Not supported in any ProCAM product.</p>
<b>RAPID_DURING_DRILL_CYCLE</b>	INTEGER	<p>Tells the system whether your post supports a rapid move in a drilling cycle via editing the toolpath. This section will be called to check for its value</p> <p>CALC_ALLOW_RAPID_DURING_DRILL. If this variable or section is not in the post, then it will assume the post does not support this option.</p> <p>RAPID_DURING_DRILL_CYCLE=TRUE  RAPID_DURING_DRILL_CYCLE=FALSE</p> <p>Supported in CAMWorks 2007 and later. Not supported in any ProCAM product.</p>
<b>OPR_THREAD_CHAMFER_ANG</b>	DECIMAL	<p>This variable stores the lathe threading cycles chamfer angle from entered angle.</p> <p>Supported in CAMWorks 2007 and later. Not supported in any ProCAM product.</p>

<b>Variable</b>	<b>Type</b>	<b>Usage</b>
<b>CAMWORKS_MATERIAL</b>	CHARACTER	Stores the selected material from CAMWorks Stock Manager. Currently, the post used a variable called “material” which was asked in the posting setup tab. If you are loading old parts that used the “material” and it was set, then we will use that string. If you are creating new parts, you can ignore that question and CAMWorks will set “material” to the same value as CAMWORKS_MATERIAL. You can also remove the “material” question from the post and just use CAMWORKS_MATERIAL. It will still set “material” because that variable is used in the post's setup output. Supported in CAMWorks 2007 and later. Not supported in any ProCAM product.
<b>LATHE_X_TOOL_OFFSET</b> <b>P_LATHE_X_TOOL_OFFSET</b> <b>N_LATHE_X_TOOL_OFFSET</b>	DECIMAL	Stores the lathe finish grooving cycle's shift amount. This is an incremental signed distance from the center of the groove tool to the driven touch off point in the X direction. Supported in CAMWorks 2007 and later. Not supported in any ProCAM product.
<b>LATHE_Z_TOOL_OFFSET</b> <b>P_LATHE_Z_TOOL_OFFSET</b> <b>N_LATHE_Z_TOOL_OFFSET</b>	DECIMAL	Stores the lathe finish grooving cycle's shift amount. This is an incremental signed distance from the center of the groove tool to the driven touch off point in the Z direction. Supported in CAMWorks 2007 and later. Not supported in any ProCAM product.
<b>MACH_CX</b> <b>P_MACH_CX</b> <b>N_MACH_CX</b>	DECIMAL	The X axis contact points of the current, previous or next movement. This is available only in Multiaxis operations. Supported in CAMWorks 2008 and later. Not supported in any ProCAM product.
<b>MACH_CY</b> <b>P_MACH_CY</b> <b>N_MACH_CY</b>	DECIMAL	The Y axis contact points of the current, previous or next movement. This is available only in Multiaxis operations. Supported in CAMWorks 2008 and later. Not supported in any ProCAM product.
<b>MACH_CZ</b> <b>P_MACH_CZ</b> <b>N_MACH_CZ</b>	DECIMAL	The Z axis contact points of the current, previous or next movement. This is only available in Multiaxis operations. Supported in CAMWorks 2008 and later. Not supported in any ProCAM product.



<b>Variable</b>	<b>Type</b>	<b>Usage</b>
<b>MACH_CJ</b> <b>P_MACH_CJ</b> <b>N_MACH_CJ</b>	DECIMAL	The J axis vector contact points of the current, previous or next movement. This is available only in Multiaxis operations. Supported in CAMWorks 2008 and later. Not supported in any ProCAM product.
<b>MACH_CK</b> <b>P_MACH_CK</b> <b>N_MACH_CK</b>	DECIMAL	The Z axis vector contact points of the current, previous or next movement. This is available only in Multiaxis operations. Supported in CAMWorks 2008 and later. Not supported in any ProCAM product.
<b>TOOL_COOLANT</b> <b>N_TOOL_COOLANT</b> <b>NC_TOOL_COOLANT</b>	INTEGER	<p>Passes the coolant information from the CAMWorks tool definition.</p> <p>MILL_COOLANT_OFF=1 MILL_COOLANT_FLOOD=2 MILL_COOLANT_MIST=3 MILL_COOLANT_THROUGH_TOOL=4 MILL_COOLANT_AIR_BLAST=5 MILL_COOLANT_HIGH_PRESSURE=6 MILL_COOLANT_SPECIAL1=7 MILL_COOLANT_SPECIAL2=8 LATHE_COOLANT_FLOOD=1 LATHE_COOLANT_OFF=2 LATHE_COOLANT_MIST=3 LATHE_COOLANT_THROUGH_TOOL=4 LATHE_COOLANT_AIR_BLAST=5 LATHE_COOLANT_HIGH_PRESSURE=6 LATHE_COOLANT_SPECIAL1=7 LATHE_COOLANT_SPECIAL2=8</p> <p>Supported in CAMWorks 2008 and later. Not supported in any ProCAM product.</p>
<b>TOOL_DIAM_OFFSET</b> <b>N_TOOL_DIAM_OFFSET</b> <b>NC_TOOL_DIAM_OFFSET</b>	INTEGER	<p>Passes the tool diameter offset number from the CAMWorks tool definition. Supported in CAMWorks 2008 and later. Not supported in any ProCAM product.</p>
<b>TOOL_LENGTH_OFFSET</b> <b>N_TOOL_LENGTH_OFFSET</b> <b>NC_TOOL_LENGTH_OFFSET</b>	INTEGER	<p>Passes the tool length offset number from the CAMWorks tool definition. Supported in CAMWorks 2008 and later. Not supported in any ProCAM product.</p>
<b>TOOL_LENGTH_DIAM_OFFSET_METHOD</b>	INTEGER	<p>Passes whether you selected the length and diameter offset numbers from the tool definition or the post.</p> <p>METHOD_FROM_TOOL=0 METHOD_FROM_POST=1</p> <p>Supported in CAMWorks 2008 and later. Not supported in any ProCAM product.</p>

<b>Variable</b>	<b>Type</b>	<b>Usage</b>
<b>HAVE_SURFACE_CP</b> <b>P_HAVE_SURFACE_CP</b> <b>N_HAVE_SURFACE_CP</b>	INTEGER	Tells the post if the current, previous or next movement has a contact point. HAVE_SURFACE_CP=TRUE or FALSE P_HAVE_SURFACE_CP=TRUE or FALSE N_HAVE_SURFACE_CP=TRUE or FALSE This is available only in multiaxis operations. Supported in CAMWorks 2008 and later. Not supported in any ProCAM product.
<b>ABS_X_PART_END</b> <b>N_ABS_X_PART_END</b> <b>P_ABS_X_PART_END</b>	DECIMAL	Stores the current X end value, next X end value and previous X end value as world coordinates in a multiaxis operation. Available in Mill only. Supported in CAMWorks 2008 and later. Not supported in any ProCAM product.
<b>ABS_Y_PART_END</b> <b>N_ABS_Y_PART_END</b> <b>P_ABS_Y_PART_END</b>	DECIMAL	Stores the current Y end value, next Y end value and previous Y end value as world coordinates in a multiaxis operation. Available in Mill only. Supported in CAMWorks 2008 and later. Not supported in any ProCAM product.
<b>ABS_Z_PART_END</b> <b>N_ABS_Z_PART_END</b> <b>P_ABS_Z_PART_END</b>	DECIMAL	Stores the current Z end value, next Z end value and previous Z end value as world coordinates in a multiaxis operation. Available in Mill only. Supported in CAMWorks 2008 and later. Not supported in any ProCAM product.
<b>CW_DWELL</b>	DECIMAL	Stores the dwell amount that is entered in a Turn Finish operation. The dwell amount can be set on the Finish Turn tab when you set the Cut Type pattern to “Diameter and Length.” To get the post to output this value, you need to have a CALC section called CALC_OUTPUT_DWELL added to your post source LIB file. When this is called, you can then call a template section to output the dwell amount. Available in Turn and Mill/Turn. Support for additional operations may be available in future versions. Supported in CAMWorks 2008 SP3.0 and later. Not supported in any ProCAM product.

<b>Variable</b>	<b>Type</b>	<b>Usage</b>
<b>TOOL_EDGE</b>	INTEGER	Stores the selected tool edge type from the turn tool definition. Available in Turn and Mill/Turn. Supported in CAMWorks 2008 SP3.0 and later. Not supported in any ProCAM product. TOOL_EDGE=SIDE_EDGE or END_EDGE SIDE_EDGE=0 END_EDGE=1
<b>TLP_PART_NAME</b>	CHARACTER	Stores the toolpath part name. In the APT CL output, keywords and associated values are output for each of the node names. Available in Mill, Turn and Mill-Turn. Supported in CAMWorks 2009 and later. Not supported in any ProCAM product.
<b>TLP_PART_DESC</b>	CHARACTER	Stores the toolpath part description. In the APT CL output, keywords and associated values are output for each of the node names. Available in Mill, Turn and Mill-Turn. Supported in CAMWorks 2009 and later. Not supported in any ProCAM product.
<b>TLP_SETUP_NAME</b>	CHARACTER	Stores the toolpath setup name. In the APT CL output, keywords and associated values are output for each of the node names. Available in Mill, Turn and Mill-Turn. Supported in CAMWorks 2009 and later. Not supported in any ProCAM product.
<b>TLP_SETUP_DESC</b>	CHARACTER	Stores the toolpath setup description. In the APT CL output, keywords and associated values are output for each of the node names. Available in Mill, Turn and Mill-Turn. Supported in CAMWorks 2009 and later. Not supported in any ProCAM product.
<b>TLP_FEAT_NAME</b>	CHARACTER	Stores the toolpath feature name. In the APT CL output, keywords and associated values are output for each of the node names. Available in Mill, Turn and Mill-Turn. Supported in CAMWorks 2009 and later. Not supported in any ProCAM product.
<b>TLP_FEAT_DESC</b>	CHARACTER	Stores the toolpath feature description. In the APT CL output, keywords and associated values are output for each of the node names. Available in Mill, Turn and Mill-Turn. Supported in CAMWorks 2009 and later. Not supported in any ProCAM product.

## System Variables

<b>Variable</b>	<b>Type</b>	<b>Usage</b>
<b>TLP_OPER_NAME</b>	CHARACTER	Stores the toolpath operation name. In the APT CL output, keywords and associated values are output for each of the node names. Available in Mill, Turn and Mill-Turn. Supported in CAMWorks 2009 and later. Not supported in any ProCAM product.
<b>TLP_OPER_DESC</b>	CHARACTER	Stores the toolpath operation description. In the APT CL output, keywords and associated values are output for each of the node names. Available in Mill, Turn and Mill-Turn. Supported in CAMWorks 2009 and later. Not supported in any ProCAM product.
<b>SETUP_ID</b> <b>P_SETUP_ID</b> <b>N_SETUP_ID</b>	INTEGER	Stores current, next and previous setup ID number. Available in Mill, Turn and Mill-Turn. Supported in CAMWorks 2009 and later. Not supported in any ProCAM product.
<b>TOOLPATH_ID</b> <b>P_TOOLPATH_ID</b> <b>N_TOOLPATH_ID</b>	INTEGER	Stores current, next and previous toolpath ID number. Available in Mill, Turn and Mill-Turn. Supported in CAMWorks 2009 and later. Not supported in any ProCAM product.
<b>NEXT_SETUP_4TH_ANGLE</b>	DECIMAL	Stores the next 4th axis preposition setup angle. Available in Mill only. Supported in CAMWorks 2009 and later. Not supported in any ProCAM product.
<b>NEXT_SETUP_5TH_ANGLE</b>	DECIMAL	Stores the next 5th axis preposition setup angle. Available in Mill only. Supported in CAMWorks 2009 and later. Not supported in any ProCAM product.
<b>PART_STOCK_DIAM</b>	DECIMAL	Stores the stock diameter from the stock definition. Available in Turn and Mill-Turn. Supported in CAMWorks 2009 and later. Not supported in any ProCAM product.
<b>PART_STOCK_LENGTH</b>	DECIMAL	Stores the stock length from the stock definition. Available in Turn and Mill-Turn. Supported in CAMWorks 2009 and later. Not supported in any ProCAM product.
<b>NEXT_OPR_TYPE</b>	INTEGER	Stores the next operation type. Available in Mill, Turn and Mill-Turn. Supported in CAMWorks 2009 and later. Not supported in any ProCAM product.
<b>NEXT_OPR_SUB_TYPE</b>	INTEGER	Stores the next operation drill cycle type. Available in Mill and Mill-Turn. Supported in CAMWorks 2009 and later. Not supported in any ProCAM product.

<b>Variable</b>	<b>Type</b>	<b>Usage</b>
<b>NEXT_IS_LATHE</b>	INTEGER	Stores if the next operation is lathe or not. Available in Mill-Turn. Supported in CAMWorks 2009 and later. Not supported in any ProCAM product. NEXT_IS_LATHE=TRUE or FALSE
<b>QUERY_ITEM_ID</b>	INTEGER	Stores the ID of the object you are trying to get information from. Available in Mill, Turn and Mill-Turn. Supported in CAMWorks 2009 and later. Not supported in any ProCAM product.
<b>QUERY_RESULT</b>	INTEGER	Stores the return value of the QUERY_SYSTEM() command. Available in Mill, Turn and Mill-Turn. Supported in CAMWorks 2009 and later. Not supported in any ProCAM product. QUERY_RESULT=1 if successful.
<b>QUERY_ERROR</b>	INTEGER	Stores the error value of the QUERY_SYSTEM() command. Available in Mill, Turn and Mill-Turn. Supported in CAMWorks 2009 and later. Not supported in any ProCAM product. QUERY_ERROR=0 for no error. If greater than zero, then it could be different types of errors currently not returned.
<b>QUERY_INT_VALUE</b>	INTEGER	Stores the integer value of the QUERY_SYSTEM() command. Available in Mill, Turn and Mill-Turn. Supported in CAMWorks 2009 and later. Not supported in any ProCAM product. QUERY_INT_VALUE will equal any integer information retrieved from the QUERY_SYSTEM() command.
<b>QUERY_DEC_VALUE</b>	INTEGER	Stores the decimal value of the QUERY_SYSTEM() command. Available in Mill, Turn and Mill-Turn. Supported in CAMWorks 2009 and later. Not supported in any ProCAM product. QUERY_DEC_VALUE will equal any integer information retrieved from the QUERY_SYSTEM() command.
<b>MULTIAXIS_CNC_COMP</b>	INTEGER	Allows the post to detect whether 3D+ CNC cutter comp has been selected in a multiaxis operation. There are specific machines that can use 3D+ CNC cutter comp. This is available only in CAMWorks Multiaxis operations. Supported in CAMWorks 2008 and later. Not supported in any ProCAM product. MULTIAXIS_CNC_COMP=TRUE or FALSE

<b>Variable</b>	<b>Type</b>	<b>Usage</b>
<b>IS_5AXIS</b>	INTEGER	IS_5AXIS = 0 or IS_5AXIS = FALSE IS_5AXIS = 1 or IS_5AXIS = TRUE If True, then it is either a Mill or Mill/Turn multiaxis operation.
<b>IS_MILL_FACE</b>	INTEGER	IS_MILL_FACE = 0 or IS_MILL_FACE = FALSE IS_MILL_FACE = 1 or IS_MILL_FACE = TRUE If True, then it is a Mill/Turn operation on the face.
<b>IS_MILL_OD</b>	INTEGER	IS_MILL_OD = 0 or IS_MILL_OD = FALSE IS_MILL_OD = 1 or IS_MILL_OD = TRUE If True, then it is a Mill/Turn operation on the OD.
<b>IS_WRAPPED</b>	INTEGER	IS_WRAPPED = 0 or IS_WRAPPED = FALSE IS_WRAPPED = 1 or IS_WRAPPED = TRUE If True, then it is a Mill operation on the OD.
<b>TLP_FEAT_SETUP_NAME</b>	CHARACTER	Stores the setup name for the current toolpath feature. Available in Mill, Turn and Mill/Turn.
<b>SYNC_CODE_COMMENT</b>	CHARACTER	Stores the sync code comment. Available in Turn and Mill/Turn. These variables are set in CAMWorks 2010, however the option of syncing turrets is not implemented in CW2010.
<b>REAR_SYNC_CODE</b> <b>P_REAR_SINCE_CODE</b>	INTEGER	Stores the current and previous rear turret sync code number. Available in Turn and Mill/Turn. These variables are set in CAMWorks 2010, however the option of syncing turrets is not implemented in CW2010.
<b>REAR_SYNC_CODE_TYPE</b>	INTEGER	Stores the rear turret sync code type. Available in Turn and Mill/Turn. These variables are set in CAMWorks 2010, however the option of syncing turrets is not implemented in CW2010. REAR_SYNC_CODE_TYPE= SYNC_CODE_UNKNOWN REAR_SYNC_CODE_TYPE= SYNC_CODE_BEFORE_START REAR_SYNC_CODE_TYPE= SYNC_CODE_BEFORE_FIRST_MOVE REAR_SYNC_CODE_TYPE= SYNC_CODE_AFTER_END  SYNC_CODE_UNKNOWN = 0 SYNC_CODE_BEFORE_START = 1 SYNC_CODE_BEFORE_FIRST_MOVE = 2 SYNC_CODE_AFTER_END = 3
<b>FRONT_SYNC_CODE</b> <b>P_FRONT_SYNC_CODE</b>	INTEGER	This variable stores the current and previous front turret sync code number. Available in Turn and Mill/Turn. These variables are set in CW2010, however the option of syncing turrets is not implemented in CW2010.

<b>Variable</b>	<b>Type</b>	<b>Usage</b>
<b>FRONT_SYNC_CODE_TYPE</b>	INTEGER	<p>Stores the front turret sync code type. Available in Turn and Mill/Turn. These variables are set in CAMWorks 2010, however the option of syncing turrets is not implemented in CW2010.</p> <pre> FRONT_SYNC_CODE_TYPE= SYNC_CODE_UNKNOWN FRONT_SYNC_CODE_TYPE= SYNC_CODE_BEFORE_START FRONT_SYNC_CODE_TYPE= SYNC_CODE_BEFORE_FIRST_MOVE FRONT_SYNC_CODE_TYPE= SYNC_CODE_AFTER_END  SYNC_CODE_UNKNOWN = 0 SYNC_CODE_BEFORE_START = 1 SYNC_CODE_BEFORE_FIRST_MOVE = 2 SYNC_CODE_AFTER_END = 3 </pre>
<b>ARC_NORM_X</b> <b>P_ARC_NORM_X</b> <b>N_ARC_NORM_X</b>	DECIMAL	Stores the current, previous and next X axis arc normal vector. Available in Mill, Turn and Mill/Turn.
<b>ARC_NORM_Y</b> <b>P_ARC_NORM_Y</b> <b>N_ARC_NORM_Y</b>	DECIMAL	Stores the current, previous and next Y axis arc normal vector. Available in Mill, Turn and Mill/Turn.
<b>ARC_NORM_Z</b> <b>P_ARC_NORM_Z</b> <b>N_ARC_NORM_Z</b>	DECIMAL	Stores the current, previous and next Z axis arc normal vector. Available in Mill, Turn and Mill/Turn.





---

## **Chapter 13 Additional System Commands**

---

This chapter contains system commands not documented previously.

---

## SYS\_CANNED

### Purpose

To break an entity not supported by the post into a series of entities that are supported by the post. Typically, this command is used the explode line, grid, arc, bolt hole patterns and pecking canned cycles into single points

### Syntax

**SYS\_CANNED(type, section)**

### Comments

<i>Parameter</i>	<i>Description</i>
<code>type</code>	the type of breakup and is a constant <ol style="list-style-type: none"> <li>1 Single points</li> <li>2 Lines, arcs and bolt holes (use only on grids and big hole patterns)</li> <li>3 Breaks a thread cycle into diameters (use only on threading cycles)</li> <li>4 Breaks a Live C post Mill OD or Mill Face line move into increments defined by post variable MILL_FACE_INC as a chord length(use only in ProCAM 2D with post set to :SYSTEM=LATHE/MILL and on Mill line moves only. It will also break the angle rotation of the C axis in the line move).</li> <li>5 Breaks an arc that is not on the top plane by ARC_DEVIATION set in the arc calc section (use only in 4 and 5 axis posts that have arcs that are not on the top plane. This command can only be used in ProCAM II 2004 and CAMWorks 2005 or newer versions).</li> <li>6 Breaks a canned peck drilling cycle into operation defined increments (use only in CAMWorks 2005 or newer version for mill pecking canned cycles).</li> </ol>
<code>section</code>	section that will handle the exploded entity The word <code>SYSTEM</code> instructs the system to call the appropriate sections.

### Example

```

SYS_CANNED(1,?????)
:SECTION=CALC_ARC_PATTERN_PUNCH
:C: SYS_CANNED(1,CALC_SINGLE_HIT_PUNCH)
:SECTION=CALC_GRID_PATTERN_PUNCH
:C: SYS_CANNED(2,SYSTEM)
:SECTION=CALC_MACHINE_THREAD_LATHE
:C: SYS_CANNED(3,CALC_MULTIPLE_THREAD_LATHE)

```

---

Warning: A SYS\_CANNED command cannot be executed while inside of another SYS\_CANNED cycle. In the example below, grids are broken into line patterns, but since the post does not support line patterns, another SYS\_CANNED command is executed. This is an error.

---

**SYS\_CANNED(2,?????)**

:SECTION=CALC\_GRID\_PATTERN\_PUNCH

:C: SYS\_CANNED(2,SYSTEM)

:SECTION=CALC\_LINE\_PATTERN\_PUNCH

:C: SYS\_CANNED(1,CALC\_SINGLE\_HIT\_PUNCH)

The post should have been written:

:SECTION=CALC\_GRID\_PATTERN\_PUNCH

:C: SYS\_CANNED(1,CALC\_SINGLE\_HIT\_PUNCH)

**SYS\_CANNED(4,?????)**

:SECTION=CALC\_LINE\_MOVE\_OD\_FREE

:C: MILL\_FACE\_INC=chord\_length

:C: SYS\_CANNED(4,CALC\_BREAK\_LINE\_OD)

**SYS\_CANNED(5,?????)**

:SECTION=CALC\_ARC\_MOVE\_MILL\_YZ

:C: ARC\_DEVIATION=max\_arc\_dev

:C: SYS\_CANNED(5,CALC\_BREAK\_ARC)

**SYS\_CANNED(6,?????)**

:SECTION=CALC\_HIGH\_SPEED\_PECKING\_CYCLE

:C: SYS\_CANNED(6,CALC\_HIGH\_SPEED\_PECKING)

---

# GETID

## ***Purpose***

To allow a string, which is the attribute name that defines the record list in the OPENDB or LOOKUPDB commands.

## ***Syntax***

**GETID(cutdata\_fields)**

## ***Comments***

Basically to use a POST string attribute that will store the Record list and Lookup list attributes. The "cutdata\_fields" is the attrname that defines the RECORD\_LIST. Since you can have up to 20 different databases open at anytime, you would need to have 20 different OPENDB and LOOKUPDB calc sections. With this function you can put a string in place of the "cutdata\_fields" attrname. As an example, you would use GETID(cutdata\_fields) in a calc section and then call the calc section that does the OPENDB or LOOKUPDB command and all you need is one calc section to do the OPENDB or LOOKUPDB commands for as many databases that are open.

(RECORD\_LIST is an integer variable)

RECORD\_LIST=**GETID(cutdata\_fields)**

OPENDB (1 , LASER\_DBPATH , {FABDB} , RECORD\_LIST , DATABASE\_STATUS)

---

# OPENTXT

## ***Purpose***

Allows the post to open external files and write to them while posting.

## ***Syntax***

**OPENTXT (FileNumber, FileName, Status)**

## ***Comments***

<i>Parameter</i>	<i>Description</i>
FileNumber	Alternate text file ID number - range (0 to 20) – 0 reserved for Post Text file and cannot be used in OPENTXT or CLOSETXT
FileName	Alternate text filename – character string or character variable with full path.
Status	Integer variable to return status of the command – 1 = Success, 0 = Fail.

---

# CLOSETXT

## ***Purpose***

Allows the post to close external files and write to them while posting.

## ***Syntax***

**CLOSETXT (FileNumber)**

## ***Comments***

<i>Parameter</i>	<i>Description</i>
FileNumber	Alternate text file ID number - range (0 to 20) – 0 reserved for Post Text file and cannot be used in OPENTXT or CLOSETXT.

---

# SETTXT

## ***Purpose***

To set the current FileNumber as the file to receive output from the posts template lines.

## ***Syntax***

**SETTXT (FileNumber)**

## ***Comments***

<i>Parameter</i>	<i>Description</i>
FileNumber	Alternate text file ID number - range (0 to 20) – 0 reserved for Post Text file and cannot be used in OPENTXT or CLOSETXT. The FileNumber must be opened before you can write to it.

---

# UPPERTXT

## ***Purpose***

To set the output from the post's template lines as all upper case letters in the current FileNumber.

## ***Syntax***

**UPPERTXT (FileNumber)**

## ***Comments***

<i>Parameter</i>	<i>Description</i>
FileNumber	Alternate text file ID number - range (0 to 20) – 0 reserved for Post Text file and cannot be used in OPENTXT or CLOSETXT. The FileNumber must be opened before you can write to it. You can do a SETTXT(?) command and then do the UPPERTXT(?) command before you output any post template lines.



---

# LOWERTXT

## ***Purpose***

To set the output from the post's template lines as all lower case letters in the current FileNumber.

## ***Syntax***

**LOWERTXT (FileNumber)**

## ***Comments***

<i>Parameter</i>	<i>Description</i>
FileNumber	Alternate text file ID number - range (0 to 20) – 0 reserved for Post Text file and cannot be used in OPENTXT or CLOSETXT. The FileNumber must be opened before you can write to it. You can do a SETTXT(?) command and then do the LOWERTXT(?) command before you output any post template lines.

---

# ORIGINALTXT

## ***Purpose***

To set the output from the post's template lines as either upper or lower case letters in the current FileNumber.

## ***Syntax***

**ORIGINALTXT (FileNumber)**

## ***Comments***

<i>Parameter</i>	<i>Description</i>
FileNumber	Alternate text file ID number - range (0 to 20) – 0 reserved for Post Text file and cannot be used in OPENTXT or CLOSETXT. The FileNumber must be opened before you can write to it. You can do a SETTXT(?) command and then do the ORIGINALTXT(?). This is the standard format if not specified to be upper or lower case.

---

# OPENRWTXT

## **Purpose**

Allows the post to open external files to read and write line by line while posting.

## **Syntax**

**OPENRWTXT (FileNumber, FileName, Status)**

## **Comments**

When reading line by line, you will use GETTXT(FileNumber, StringVar, Status) until Status = 0 then you are at the end of the file. If you want to append to an existing file, you can use the GETTXT command to get to the end of the file, then use SETTXT command to append. To open a new file, you use either OPENTXT or OPENRWTXT command. You can use this command with these commands: CLOSETXT , SETTXT, UPPERTXT, LOWERTXT and ORIGINALTXT.

<i>Parameter</i>	<i>Description</i>
------------------	--------------------

FileNumber	Alternate text file ID number - range (0 to 20) – 0 reserved for Post Text file and cannot be used in OPENTXT or CLOSETXT.
------------	--

FileName	Alternate text filename – character string or character variable with full path.
----------	--

Status	Integer variable to return status of the command – 1 = Success, 0 = Fail.
--------	---

---

# GETTXT

## ***Purpose***

To set the current FileNumber to read line by line.

## ***Syntax***

**GETTXT (FileNumber, StringVar, Status)**

## ***Comments***

This command will work only if the current file has been opened by OPENRWTEXT only. This will not work with OPENTEXT.

<i>Parameter</i>	<i>Description</i>
------------------	--------------------

FileNumber	Alternate text file ID number - range (0 to 20) – 0 reserved for Post Text file and cannot be used in OPENTEXT or CLOSETXT. The FileNumber must be opened before you can read it.
------------	---

StringVar	Stores the read string variable.
-----------	----------------------------------

Status	Integer variable to return status of the command – 1 = Success, 0 = Fail or at end of line.
--------	---

---

# GETMCS

## ***Purpose***

To get all the different MCS offsets inserted in the current part file using ProCAM II or CAMWorks.

## ***Syntax***

**GETMCS(2,calc\_section)**

## ***Comments***

This command will do a search through the whole part file and gather all the different MCS offsets along with the Tool, Tool Comment, Rotation angles and the Work Coordinate Offset values. The calc\_section is any calc section you want to use for getting that information. The GETMCS will call that calc section every time it finds a different MCS offset. This can be used for machines that need to call out all the MCS offsets at the start of the program like Fanuc's G10 command. Variables that will have information stored during this command are MCS\_X\_OFFSET, MCS\_Y\_OFFSET, MCS\_Z\_OFFSET, TOOL, TOOL\_COMMENT, ROT\_TILT\_A, ROT\_TILT\_B, work\_coord, sub\_work\_coord and fixture\_offset.

---

# STRGLEN

## ***Purpose***

Allows the post to get the string length of any string.

## ***Syntax***

**STRGLEN(string\_var)**

## ***Comments***

<i>Parameter</i>	<i>Description</i>
------------------	--------------------

string_var	Is the string variable or hard coded. Hard coded means using the {} braces and putting characters between them.
------------	---

```
:C: STRG={TEST}
```

```
:C: STRG_LENGTH=STRGLEN(STRG)
```

```
The STRG_LENGTH would equal 4
```

---

# LEFTSTRG

## **Purpose**

Allows the post to capture a string starting from the left of the string to the right by a character count (length).

## **Syntax**

**LEFTSTRG(target\_string,string\_var,length)**

## **Comments**

<i>Parameter</i>	<i>Description</i>
target_string	The receiving string variable. This can only be a defined POST character variable.
string_var	The string variable or hard coded. Hard coded means using the {} braces and putting characters between them.

```
:C: STRG={TEST}
:C: LEFTSTRG(STRGA , STRG, 2)
The STRGA would equal "TE"

:C: LENGTH=2
:C: STRG={TEST}
:C: LEFTSTRG(STRGA , STRG, LENGTH)
The STRGA would equal "TE"
```

---

# RIGHTSTRG

## Purpose

Allows the post to capture a string starting from the right of the string to the left by a character count (length).

## Syntax

**RIGHTSTRG(target\_string,string\_var,length)**

## Comments

<i>Parameter</i>	<i>Description</i>
target_string	The receiving string variable. This can only be a defined POST character variable.
string_var	The string variable or hard coded. Hard coded means using the {} braces and putting characters between them.

```
:C: STRG={TEST}
:C: RIGHTSTRG(STRGA , STRG, 2)
  The STRGA would equal "ST"

:C: LENGTH=2
:C: STRG={TEST}
:C: RIGHTSTRG(STRGA , STRG, LENGTH)
The STRGA would equal "ST"
```



---

# MIDSTRG

## Purpose

Allows the post to capture a string starting from a given character count from the left of the string to the right by a character count (length).

## Syntax

**MIDSTRG(target\_string,string\_var,start,length)**

## Comments

<i>Parameter</i>	<i>Description</i>
target_string	The receiving string variable. This can only be a defined POST character variable.
string_var	The string variable or hard coded. Hard coded means using the {} braces and putting characters between them.
start	An integer or integer variable that defines the starting character to capture.
length	An integer or integer variable that stores the given character count from the left of the string starting character to the right.

```
:C: STRG={JOHN DOE}
:C: MIDSTRG(STRGA , STRG, 6, 3)
The STRGA would equal "DOE"

:C: START=6
:C: LENGTH=3
:C: STRG={TEST}
:C: MIDSTRG(STRGA , STRG, START, LENGTH)
The STRGA would equal "DOE"
```

---

# STRGUPPER

## ***Purpose***

Allows the post to change the given string to have all upper case characters.

## ***Syntax***

**STRGUPPER(target\_string,string\_var)**

## ***Comments***

<i>Parameter</i>	<i>Description</i>
target_string	The receiving string variable. This can only be a defined POST character variable.
string_var	The string variable or hard coded. Hard coded means using the {} braces and putting characters between them.

```
:C: STRG={test}  
:C: STRGUPPER(STRGA , STRG)  
The STRGA would equal "TEST"
```

---

# STRGLOWER

## ***Purpose***

Allows the post to change the given string to have all lower case characters.

## ***Syntax***

**STRGLOWER(target\_string,string\_var)**

## ***Comments***

<i>Parameter</i>	<i>Description</i>
target_string	The receiving string variable. This can only be a defined POST character variable.
string_var	The string variable or hard coded. Hard coded means using the {} braces and putting characters between them.

```
:C: STRG={TEST}
```

```
:C: STRGLOWER(STRGA , STRG)
```

```
The STRGA would equal "test"
```

---

## GET\_SELECT STRING

### **Purpose**

To gather a string list from a file that will update the list selection from the Setup Info. It lets the user put more materials in the file and not have to recompile the post. The file that will store the string lists corresponds to the name of the compiled post. If the compiled post is FANUC6M.CTL, then the file used will be FANUC6M.CNF.

### **Syntax**

These commands are associated with this command, which will be in the \*.CNF file.

<i>Command</i>	<i>Description</i>
<b>CONFIG_SELECT_NAME</b>	The name of the lowercase attribute that will show the list in the Setup Info.
<b>CONFIG_SELECT_OPTION</b>	The string that you want to select from. You can have as many of these in a row as needed. It basically creates the list.
<b>CONFIG_SELECT_DEFAULT</b>	Sets the default selection from the list. If you wanted the 3 <sup>rd</sup> string in the list as the default, then this would equal 3.
<b>CONFIG_SELECT_END</b>	Lets the post know you are done with the list.

This is what would be in the \*.CNF file.

```
:CONFIG_SELECT_NAME=material type
:CONFIG_SELECT_OPTION=STEEL
:CONFIG_SELECT_OPTION=STAINLESS STEEL
:CONFIG_SELECT_OPTION=ALUMINUM
:CONFIG_SELECT_OPTION=COPPER
:CONFIG_SELECT_DEFAULT=3
:CONFIG_SELECT_END
```

### **Comments**

Below is an example of using this for gathering a list of materials.

```
*-----
:ATTRNAME=material type
:ATTRTYPE=SELECT
:ATTREMARK=Material type
:ATTRSEL=N
:ATTRTITLE=Material Type
:ATTRSELSTR=Rolled Steel
:ATTRSELSTR=Aluminum
:ATTRSELSTR=Stainless Steel
:ATTRDEFAULT=1
:ATTRUSED=1
:ATTREND
*-----
:ATTRNAME=S MATERIAL TYPE
```

```

:ATTRTYPE=POST
:ATTRVTYPE=CHARACTER
:ATTREMARK=
:CODETYPE=FORMAT
:WORD_ADDRESS_BEf= | (
:WORD_ADDRESS_AfT=)
:LEFT_PLACES=0
:RIGHT_PLACES=0
:UNITFLAG=NON_CONVERT
:ATTRSPACES=YES
:MODAL=YES
:ATTRUSED=1
:ATTREND

```

```

*-----
:SECTION=CALC_INIT_TOOL_CHANGE_MILL
:C: IF SECTIONEXIST(DEBUG) THEN
:C: DEBUG=4 CALL (DEBUG)
:C: ENDIF
*

```

```

:C: GET_SELECT_STRING(material_type,S_MATERIAL_TYPE)
:C: CALL (OUPUT_MATERIAL)

```

```

*-----
:SECTION=OUTPUT_MATERIAL
:T:<S_MATERIAL_TYPE>

```

### **GET\_SELECT\_STRING(material\_type,S\_MATERIAL\_TYPE)**

Material\_type is an attribute that is defined and is lower case, plus it needs to be defined in the master.atr file. You can insert a few :ATTRSELSTR= in this definition as shown above.

The next attribute defined is the output post attribute S\_MATERIAL\_CODE, which can be any name you give it.

The section called :SECTION=CALC\_INIT\_TOOL\_CHANGE\_MILL is where I will start the command. GET\_SELECT\_STRING(material\_type,S\_MATERIAL\_TYPE), then call a template section to test out the selection :C: CALL (OUTPUT\_MATERIAL) . Make sure you have a template section created for the call. You do not need to do this step if you do not need to output the selected string.

---

# FASTLINE

## Purpose

To output lines of code at a faster rate then normal. This is only available in the 3 axis milling operations.

## Syntax

**FASTLINE(template\_section)**

## Comments

Parameter	Description
template_section	The template section that will output the lines of code. You will need to define the ATTRNAMES listed below.

```
*-----
:ATTRNAME=FX
:ATTRTYPE=POST
:ATTRVTYPE=DECIMAL
:ATTREMARK=X End
:CODETYPE=FORMAT
:WORD_ADDRESS_BEf= | X
:VAR=ABS_X_END
:MODAL=YES
:ATTREND
*-----
:ATTRNAME=FY
:ATTRTYPE=POST
:ATTRVTYPE=DECIMAL
:ATTREMARK=Y End
:CODETYPE=FORMAT
:WORD_ADDRESS_BEf= | Y
:VAR=ABS_Y_END
:MODAL=YES
:ATTREND
*-----
:ATTRNAME=FZ
:ATTRTYPE=POST
:ATTRVTYPE=DECIMAL
:ATTREMARK=Z End
:CODETYPE=FORMAT
:WORD_ADDRESS_BEf= | Z
:VAR=ABS_Z_END
:MODAL=YES
:ATTREND
```

Normally you will need to start this command in the CALC\_START\_OPERATION section as shown below.

```

:C: IF SECTIONEXIST (FASTLINE) THEN
:C:   IF OPR_TYPE=MILL_UV_CUT
:C:   OR OPR_TYPE=MILL_SLICE_CUT
:C:   OR OPR_TYPE=MILL_ROUGH_CUT
:C:   OR OPR_TYPE=MILL_CURVE_CUT
:C:   OR OPR_TYPE=MILL_TOPO_CUT
:C:   OR OPR_TYPE=MILL_FREEFORM_CUT THEN
:C:   FASTLINE (FASTLINE)
:C:   ENDIF
:C: ENDIF

```

The template section should look like this.

```

:SECTION=FASTLINE
:T:<N><G:01><FX><FY><FZ><EOL>

```

FASTLINE



---

## **Chapter 14 Add'l. Calc Sections & Operids**

---

This chapter contains calc sections for CAMWorks and ProCAM and operation ID's for EDM in ProCAM 2D that were not documented previously.

---

## CALC\_ARC\_MOVE\_ZX

### ***Purpose***

Mill calc section for arc moves on ZX plane using CAMWorks 2005 or ProCAM II 2004 or newer versions.

### ***Syntax***

:SECTION=CALC\_ARC\_MOVE\_ZX

### ***Comments***

The post will only get to this section when you are doing an arc movement that is on the ZX plane. If section does not exist then post outputs line moves.

---

## CALC\_ARC\_MOVE\_YZ

### ***Purpose***

Mill calc section for arc moves on YZ plane using CAMWorks 2005 or ProCAM II 2004 or newer versions.

### ***Syntax***

:SECTION= CALC\_ARC\_MOVE\_YZ

### ***Comments***

The post will only get to this section when you are doing an arc movement that is on the YZ plane. If section does not exist then post outputs line moves.

---

## CALC\_ARC\_MOVE\_ANYPLANE

### ***Purpose***

Mill calc section for arc moves on none standard planes using CAMWorks 2005 or ProCAM II 2004 or newer versions.

### ***Syntax***

:SECTION=CALC\_ARC\_MOVE\_ANYPLANE

### ***Comments***

The post will only get to this section when you are doing an arc movement that is on none standard planes. If section does not exist then post outputs line moves.

---

## CALC\_POST\_INITIALIZE

### ***Purpose***

Mill calc section for setting 4 and 5 axis parameters used in ProCAM II only.

### ***Syntax***

```
:SECTION=CALC_POST_INITIALIZE  
:C: IF SECTIONEXIST(FIVE_AXIS_LINE_MOVE_MILL) THEN  
:C: CALL(CALC_RESET_REGISTERS)  
:C: CALL(CALC_RESET_FIVE_AXIS_REGISTERS)  
:C: ENDIF
```

### ***Comments***

The post will only get to this section when you have either :4AXIS\_X\_MILLING=TRUE, :4AXIS\_Y\_MILLING=TRUE or :5AXIS\_MILLING=TRUE, It will get to this section before it gets to CALC\_START\_OPERATION for the first time.

---

## CALC\_TOOL\_INITIALIZE

### ***Purpose***

Mill calc section for setting 4 and 5 axis HEAD\_LEN tool parameters. When you have a machine that has the Head that rotates or tilts and you need to add the tool length on to the posted output. Used in ProCAM II 2004 or CAMWorks 2005 or newer versions only.

### ***Syntax***

```
:SECTION=CALC_TOOL_INITIALIZE  
:C: HEAD_LEN=(INIT_TOOL_LENGTH+head_length)
```

### ***Comments***

The post will only get to this section when you have either :4AXIS\_X\_MILLING=TRUE, :4AXIS\_Y\_MILLING=TRUE or :5AXIS\_MILLING=TRUE, It will get to this section before it gets to CALC\_START\_OPERATION for the first time.

INIT\_TOOL\_LENGTH is a system post variable

INIT\_TOOL\_LENGTH holds the tool length from tool definition.

Head\_length = is a post question and it can be added or subtracted

---

## CALC\_RAPID\_MOVE\_SHEAR

### ***Purpose***

Shear system rapid move calc section.

### ***Syntax***

:SECTION=CALC\_RAPID\_MOVE\_SHEAR

### ***Comments***

This section will be called for every rapid move that occurs while you are shearing.

---

## CALC\_INIT\_TOOL\_CHANGE\_SHEAR

### ***Purpose***

Shear system initial tool change calc section.

### ***Syntax***

:SECTION=CALC\_INIT\_TOOL\_CHANGE\_SHEAR

### ***Comments***

This section will be called for the first tool change that occurs while you are shearing.



---

## CALC\_SUB\_TOOL\_CHANGE\_SHEAR

### ***Purpose***

Shear system sub tool change calc section.

### ***Syntax***

:SECTION=CALC\_SUB\_TOOL\_CHANGE\_SHEAR

### ***Comments***

This section will be called for every tool change that occurs after the first tool change while you are shearing.

---

## CALC\_EVERY\_MOVE\_SHEAR

### ***Purpose***

Shear system every move calc section.

### ***Syntax***

:SECTION=CALC\_EVERY\_MOVE\_SHEAR

### ***Comments***

This section will be called for every move type that occurs while you are shearing.

---

## CALC\_FULL\_SHEAR

### ***Purpose***

Shear system full shear calc section.

### ***Syntax***

:SECTION=CALC\_FULL\_SHEAR

### ***Comments***

This section will be called for every stroke that uses a full shear while you are shearing.

---

## CALC\_HALF\_SHEAR\_X

### ***Purpose***

Shear system half shear in X calc section.

### ***Syntax***

:SECTION=CALC\_HALF\_SHEAR\_X

### ***Comments***

This section will be called for every stroke that only uses half the shear tool in the X direction while you are shearing.

---

## CALC\_HALF\_SHEAR\_Y

### ***Purpose***

Shear system half shear in Y calc section.

### ***Syntax***

:SECTION=CALC\_HALF\_SHEAR\_Y

### ***Comments***

This section will be called for every stroke that only uses half the shear tool in the Y direction while you are shearing.

---

## CALC\_FULL\_SHEAR\_DIAGONAL

### ***Purpose***

Shear system full shear diagonally calc section.

### ***Syntax***

:SECTION=CALC\_FULL\_SHEAR\_DIAGONAL

### ***Comments***

This section will be called for every stroke that uses the full shear tool in a diagonal direction while you are shearing.

---

## CALC\_HALF\_SHEAR\_DIAGONAL

### ***Purpose***

Shear system half shear diagonally calc section.

### ***Syntax***

:SECTION=CALC\_HALF\_SHEAR\_DIAGONAL

### ***Comments***

This section will be called for every stroke that uses half of the shear tool in a diagonal direction while you are shearing.

---

## CALC\_REPOSITION\_SHEAR

### ***Purpose***

Shear system reposition calc section.

### ***Syntax***

:SECTION=CALC\_REPOSITION\_SHEAR

### ***Comments***

This section will be called for every reposition created while you are shearing.



---

## CALC\_RAPID\_TO\_TRAP DOOR

### ***Purpose***

Laser or Plasma system rapid move to trap door calc section.

### ***Syntax***

:SECTION=CALC\_RAPID\_TO\_TRAP\_DOOR

### ***Comments***

This section will be called when a trap door is attached to any entity. The header command :TRAPDOOR must equal either DROP or TILT. If this header command is not in the post or is set to FALSE, then you will not get to this calc section.

---

## CALC\_PROFILE\_DRILL\_LASER

### ***Purpose***

Used when you have a PUNCH/PLASMA or a PUNCH/LASER combination machine that uses a prepunched hole for starting a plasma or laser profile.

### ***Syntax***

:SECTION=CALC\_PROFILE\_DRILL\_LASER

### ***Comments***

This section will be called when you do prepunch hole for the plasma or laser profile. Normally, you would do a punch single hit and this, then give the plasma or laser a place to start the cut if not on the edge of the sheet.

---

# CALC\_GET\_TAPER\_EDM

## ***Purpose***

Used when you need to get all the different taper angles used in the current part and output them at the start of the program.

## ***Syntax***

**:SECTION=**CALC\_GET\_TAPER\_EDM

## ***Comments***

This section will be called only when it is inserted into the post source. When you start to post out, the current EDM part file the system will go through the complete tool paths of the part file to gather all the different taper changes, then it will start calling this section for each different taper angle and you can store this information in post arrays. Now it will start the normal post output.

:OPERID

---

## :OPERID

### ***Purpose***

EDM operations ID names.

### ***Syntax***

#### *Syntax*

**:OPERID=EDM\_PROFILE**

**:OPERID=EDM\_SKIM**

**:OPERID=EDM\_CORE**

**:OPERID=EDM\_MACRO**

#### *Purpose*

EDM operation ID for creating an EDM Profile operation.

EDM operation ID for creating an EDM Skimcut operation. This operid was added when operations were added to the ProCAM 2d EDM system.

EDM operation ID for creating an EDM core removal operation. This operid was added when operations were added to the ProCAM 2d EDM system.

EDM operation ID for creating an EDM macro call operation. This operid was added when operations were added to the ProCAM 2d EDM system.

---

## CALC\_SLOWDOWN\_SPEED

### ***Purpose***

Called in a lathe cutoff cycle if a slowdown is selected in the cutoff operation.

### ***Syntax***

**:SECTION=**CALC\_SLOWDOWN\_SPEED

### ***Comments***

Supported in CAMWorks 2007 and later. Not supported in any ProCAM product.

The section needs to be in the \*.src file: CHANGE\_SPEED.

:SECTION=

:SECTION:=CHANGE\_SPEED

---

## CALC\_SHIFT\_TOOL\_LATHE

### ***Purpose***

Called in a lathe finish grooving cycle if a shift from one side of the groove tool to the opposite side is detected.

### ***Syntax***

**:SECTION=**CALC\_SHIFT\_TOOL\_LATHE

### ***Comments***

Supported in CAMWorks 2007 and later. Not supported in any ProCAM product.

These sections need to be added to the \*src file: SHIFT\_OFFSET\_PRIMARY and SHIFT\_OFFSET\_SECONDARY.

```
:SECTION=CALC_SHIFT_TOOL_LATHE
:SECTION: SHIFT_OFFSET_PRIMARY
:SECTION: SHIFT_OFFSET_SECONDARY
```

---

## CALC\_CUTTER\_COMP\_LATHE

### ***Purpose***

Called in a lathe finish grooving cycle if a shift from one side of the groove tool to the opposite side is detected. This will support machine cutter comp values.

### ***Syntax***

**:SECTION=**CALC\_CUTTER\_COMP\_LATHE

### ***Comments***

Supported in CAMWorks 2007 and later. Not supported in any ProCAM product.

These sections need to be added to the \*.src file: LINE\_LEADIN\_MOVE\_LATHE and LINE\_LEADOUT\_MOVE\_LATHE.

```
:SECTION=CALC_CUTTER_COMP_LATHE
:SECTION: LINE_LEADIN_MOVE_LATHE
:SECTION: LINE_LEADOUT_MOVE_LATHE
```

# **CALC\_ALLOW\_RAPID\_DURING\_DRILL**

## **Purpose**

This system section will be called in a mill drilling cycle and handles posted output of toolpath editing in drilling cycles. This section is for allowing rapids in drilling cycles.

## **Syntax**

**:SECTION=CALC\_ALLOW\_RAPID\_DURING\_DRILL**

## **Comments**

This section will look for a post variable called RAPID\_DURING\_DRILL\_CYCLE. The post will set this to a TRUE or FALSE value. If the post does not have this section or variable, then the posting system assumes a FALSE value. When the value is set to TRUE, then the system assumes that the post can support rapids in a drilling cycle that are added in the Edit toolpath area of CAMWorks.

The drilling cycles were designed in a specific way and when you add rapids to the toolpath, you may get unwanted results. You can adjust the drilling cycles in new posts if needed and set RAPID\_DURING=1. The default is handled by not having or setting RAPID\_DURING=0. Older posts do not have RAPID\_DURING compiled in the source, so in that case the default will be zero.

Supported in CAMWorks 2007 and later. Not supported in any ProCAM product.

## **Example**

```
*****
:SECTION=CALC_ALLOW_RAPID_DURING_DRILL
:C: IF CAMWORKS_VER>CAM_REV2006EX THEN
:C: IF RAPID_DURING=1 THEN
:C: RAPID_DURING_DRILL_CYCLE=TRUE
:C: ENDIF
:C: ENDIF
*****
```



---

## CALC\_SET\_PRE\_POSITION\_ROTARY\_TYPE

### **Purpose**

This system section handles the 5th axis preposition and simultaneous output. It will be called automatically in a Mill 4th or 5th axis Preposition operation to set the preferred 4th axis preposition axis output values, either in ROT\_TILT\_A or ROT\_TILT\_B.

### **Syntax**

Integer variable: PRE\_POSITION\_ROTARY\_TYPE

System Constants

ROTARY\_TYPE\_A=4

ROTARY\_TYPE\_B=5

### **Comments**

Currently, posting will assume ROT\_TILT\_B for a 4th axis preposition move. A wrapped feature will always be ROT\_TILT\_A.

PRE\_POSITION\_ROTARY\_TYPE can be set by two constants ROTARY\_TYPE\_A and ROTARY\_TYPE\_B. Setting this to ROTARY\_TYPE\_B supports old preposition posts.

Supported in CAMWorks 2007 and later. Not supported in any ProCAM product.

### **Example**

```
*-----
:SECTION=CALC_SET_PRE_POSITION_ROTARY_TYPE
:C: FOURTH_AXIS_SELECTION=ROTARY_TYPE_B
:C: IF FOURTH_AXIS_SELECTION>3 AND FOURTH_AXIS_SELECTION<6 THEN
:C: IF CAMWORKS_VER>CAM_REV2006EX THEN
:C: PRE_POSITION_ROTARY_TYPE=FOURTH_AXIS_SELECTION
:C: ENDIF
:C: ENDIF
```

---

## **CALC\_ADD\_REAR\_SYNC\_CODE**

### ***Purpose***

CALC\_ADD\_REAR\_SYNC\_CODE gets called when a command to output rear sync code is encountered and the post variables SYNC\_CODE\_COMMENT, REAR\_SYNC\_CODE and REAR\_SYNC\_CODE\_TYPE are set.

---

## **CALC\_ADD\_FRONT\_SYNC\_CODE**

### ***Purpose***

CALC\_ADD\_FRONT\_SYNC\_CODE gets called when a command to output rear sync code is encountered and the post variables SYNC\_CODE\_COMMENT, FRONT\_SYNC\_CODE and FRONT\_SYNC\_CODE\_TYPE are set.

CALC\_ADD\_FRONT\_SYNC\_CODE

---

## **Appendix A Using an Access Database During Posting**

---

# Commands

---

## OPENDB

### *Purpose*

Opens database and defines record variables.

### *Syntax*

**OPENDB**(FileNumber, FileName, TableName, RecordList, Status)

## CLOSEDB

### *Purpose*

Closes database.

### *Syntax*

**CLOSEDB**(FileNumber)

## LOOKUPDB

### *Purpose*

Lookup record based on variables in KeyList.

### *Syntax*

**LOOKUPDB**(FileNumber, KeyList, Status)

### *Comments*

<i>Parameter</i>	<i>Description</i>
FileNumber	Access database file ID number - range (0 to 19)
FileName	Access database filename – character string or character variable with full path
TableName	Access database table name – character string or character variable
RecordList	Attribute list that describes database fields 1 to 1
KeyList	Attribute list that describes key fields to be used for lookup – all members of This list must also be members of the RecordList
Status	Integer variable to return status of the command – 1 = Success, 0 = Fail

## Example

For this example, the database has three fields **Material**, **Thickness** and **Feedrate**. In this demo post, you are going to use the database to lookup values in the fld1 and fld2 attributes to find a match and set the posts **feedrate=fld3**.

1. Unzip **Demo.zip** on any drive and in any folder.
2. Edit **Demo.lib**.

Look at the attributes that were created for the use of this database.

For this example, the database has three fields **Material**, **Thickness** and **Feedrate**. You will open the database later in this example.

The attribute **fld1** represents the material and it is a character type, **fld2** represents the thickness and it is a decimal type, **fld3** represents the feedrate and it is a decimal type.

Below is the list of attributes needed for this example. In this demo post, you are going to use the database to lookup values in fld1 and fld2 to find a match and set the posts **feedrate=fld3**.

```
*-----
* Define Database Attributes
*-----
:ATTRNAME=fld1
:ATTRTYPE=VALUE
:ATTRVTYPE=CHARACTER
:ATTREMARK=Material
:ATTRSEL=N
:ATTRINLEN=25
:ATTRSHORT=Material
:ATTRLONG=ENTER Material Type
:ATTRHIGH=~
:ATTRLOW=
:ATTRDEFAULT=
:ATTRUSED=1
:ATTREND

*-----
:ATTRNAME=fld2
:ATTRTYPE=VALUE
:ATTRVTYPE=DECIMAL
:ATTREMARK=Thickness
:ATTRSEL=N
:ATTRSHORT=Thickness
:ATTRLONG=ENTER Thickness
:ATTRHIGH=9999
:ATTRLOW=0
:ATTRDEFAULT=0
:ATTRUSED=1
:ATTREND
```

## Example

```
*-----  
:ATTRNAME=fld3  
:ATTRTYPE=VALUE  
:ATTRVTYPE=DECIMAL  
:ATTREMARK=Feedrate  
:ATTRSEL=N  
:ATTRSHORT=Feedrate  
:ATTRLONG=ENTER Feedrate  
:ATTRHIGH=9999  
:ATTRLOW=0  
:ATTRDEFAULT=0  
:ATTRUSED=1  
:ATTREND
```

3. Along with defining the fields, you must also define the field attributes in a List type attribute. In this post it is called **demo fields**. You must place these attributes in the position that matches the database. See below.

```
*-----  
:ATTRNAME=demo fields  
:ATTRTYPE=LIST  
:ATTRSEL=N  
:ATTRTITLE=Demo Database Fields  
:ATTRLIST=fld1  
:ATTRLIST=fld2  
:ATTRLIST=fld3  
:ATTRUSED=1  
:ATTRDEFAULT=1  
:ATTREND
```

4. You must also define the lookup attributes in a list type attribute. In this post, it is called **demo lookup**. Since you are going to use fields 1 and 2 for the lookup, then **fld1** & **fld2** are placed in the lookup attribute list as shown below.

```
*-----  
:ATTRNAME=demo lookup  
:ATTRTYPE=LIST  
:ATTRSEL=N  
:ATTRTITLE=Demo Database Lookup  
:ATTRLIST=fld1  
:ATTRLIST=fld2  
:ATTRUSED=1  
:ATTRDEFAULT=1  
:ATTREND
```



5. In this demo post, you are allowing the user to enter the path and filename of the database from the Setup Information command on the CAM menu. As shown below, the default path and name are C:\DEMO\DEMO.MDB.

```
*-----
:ATTRNAME=comment 1
:ATTRTYPE=VALUE
:ATTRVTYPE=CHARACTER
:ATTREMARK=Database Name & Location
:ATTRSEL=N
:ATTRINLEN=25
:ATTRSHORT=Database Name & Location
:ATTRLONG=ENTER Database Name & Location
:ATTRHIGH=~
:ATTRLOW=
:ATTRDEFAULT=C:\DEMO\DEMO.MDB
:ATTRUSED=1
:ATTREND
```

6. You also need to define an attribute to represent the status of opening the database and the lookup of the database as shown below.

```
*-----
:ATTRNAME=DATABASE_STATUS
:ATTRTYPE=POST
:ATTRVTYPE=INTEGER
:ATTREMARK=
:ATTREND
*-----
```

7. Edit **Demo.src** and search for **:ATTRNAME=attachable**, as shown below.

You also need to place all the field attributes and list attributes that were defined in **demo.lib** in the attachable list. Since all these attributes have the **:ATTRSEL=N**, none of these will show up in the attachable list in CAM.

```
-----
* Define Attachable Questions
*-----
:ATTRNAME=attachable
:ATTRTYPE=LIST
:ATTRSEL=N
:ATTRTITLE=Attachable
:ATTRLIST=program stop
:ATTRLISTDEF=1
:ATTRLIST=optional stop
:ATTRLISTDEF=
:ATTRLIST=machine compensation
:ATTRLISTDEF=1
:ATTRLIST=feedrate
:ATTRLISTDEF=10
```

## Example

```
:ATTRLIST=abs inc
:ATTRLISTDEF=1
***** Add the database attributes to the Attachable list
:ATTRLIST=fld1
:ATTRLIST=fld2
:ATTRLIST=fld3
:ATTRLIST=demo fields
:ATTRLIST=demo lookup
*****
:ATTRUSED=1
:ATTRDEFAULT=1
:ATTREND
```

8. In **demo.src**, search for **:ATTRNAME=setup**, as shown below.

Notice that the **material**, and **thickness** are in most posts already, but the attribute **comment 1** has been added that will ask the database path and filename.

```
*-----
* Define Setup Questions
*-----
:ATTRNAME=setup
:ATTRTYPE=LIST
:ATTRSEL=N
:ATTRTITLE=Setup
:ATTRLIST=program number
:ATTRLISTDEF=1
:ATTRLIST=x sheet width
:ATTRLISTDEF=0
:ATTRLIST=y sheet height
:ATTRLISTDEF=0
:ATTRLIST=material ←-----
:ATTRLISTDEF=
:ATTRLIST=thickness ←-----
:ATTRLISTDEF=.125
:ATTRLIST=init abs inc
:ATTRLISTDEF=1
:ATTRLIST=init feedrate
:ATTRLISTDEF=10
:ATTRLIST=i machine compensation
:ATTRLISTDEF=1
:ATTRLIST=d offset reg
:ATTRLISTDEF=1
***** Add this for the database name and path
:ATTRLIST=comment 1
:ATTRLISTDEF=
*****
:ATTRUSED=1
:ATTRDEFAULT=1
:ATTREND
```

9. In **Demo.src**, search for **:SECTION=CALC\_INIT\_CODES**, as shown below.  
Now look for the **CALL(CALC\_OPEN\_DATABASE)** command.

```
*-----
:SECTION=CALC_INIT_CODES
:C: DEFINING_MACRO=NO
:C: OFFSET_RESIDENT=NO
*
*   Sequence number configuration
*
:C: SEQ=10
:C: SEQ_INCREMENT=10
:C: MAX_SEQUENCE=9999
:C: LASER_ON=NO
*
*   Sequence Number configuration
*
*   SEQ_CONFIG = 0   - Floating Sequence  N1, N2 etc.
*   SEQ_CONFIG = 1   - Four Place Sequence  N0001, N0002 etc.
*   SEQ_CONFIG = 2   - Three Place Sequence  N001, N002 etc.
*   SEQ_CONFIG > 2   - No Sequence Numbers.
*
:C: SEQ_CONFIG=3
*
*   Arc Center configuration
*
*   AIC = 0   - Absolute Center
*   AIC = 1   - Incremental distance from Start to Center
*   AIC = 2   - Absolute or Incremental distance from Start to
Center
*   AIC = 3   - Incremental distance from Center to Start
:C: AIC = 1
*
*   D Offset Register Number
*
*   If COMP_OFFSET=20 and TOOL=1 then
COMP_NUMBER=(TOOL+COMP_OFFSET)
*   COMP_NUMBER=21 - <COMP_NUMBER>
*
:C: COMP_OFFSET=0
*
*   Open the database and call the lookup
*
```

10. Now I called a calc section to do the open database. Six lines down in the src file shows the **OPENDB** command line.
- ```
:C: CALL(CALC_OPEN_DATABASE)
```

## Example

11. Once I have opened the database, then I can do a lookup or multiple lookups until I close the database.

```
:C: CALL(CALC_LOOKUP_DATABASE)
```

12. Now we will close the database after the lookup.

```
:C: CLOSEDB(1)
```

```
*
```

```
*-----
```

13. In the example below, **comment\_1** stores the path and file name of the database, **DEMO** is the databases table name, **demo\_fields** stores a list of the fields in the database and **DATABASE\_STATUS** stores the status of the open database command.

Notice that if it cannot open the database then, we call an error message close the database. In this case, you might want to set a flag to not do a lookup, because the database is not open and might error out.

```
:SECTION=CALC_OPEN_DATABASE
```

```
:C: OPENDB(1,comment_1,{DEMO},demo_fields,DATABASE_STATUS)
```

```
:C: IF DATABASE_STATUS=0 THEN CALL(OPEN_ERROR) CLOSEDB(1) RETURN  
ENDIF
```

```
*-----
```

```
*
```

```
:SECTION=OPEN_ERROR
```

```
:T: Could Not Open Demo Database<EOL>
```

```
*-----
```

```
*
```

14. In the example below, you can set **fld1=material** and **fld2=thickness** because material and thickness are asked in the Setup info.

In the example below, the lookup command uses **demo\_lookup** list attribute that uses **fld1** and **fld2** to find a match. If it finds a match, then **feedrate** is set to **fld3**. If it cannot find a match, then we will default the **feedrate** to 999.

```
:SECTION=CALC_LOOKUP_DATABASE
```

```
:C: fld1=material
```

```
:C: fld2=thickness
```

```
:C: LOOKUPDB(1,demo_lookup,DATABASE_STATUS)
```

```
:C: IF DATABASE_STATUS=0 THEN
```

```
:C: CALL(LOOKUP_ERROR)
```

```
:C: fld1={0}
```

```
:C: fld2=0
```

```
:C: fld3=999
```

```
:C: ENDIF
```

```
:C: feedrate=fld3
```

```
*
```

```
:SECTION=LOOKUP_ERROR
```

```
:T: Error in Lookup In Demo Database or<EOL>
```

```
:T: Lookup found no matches in Demo Database<EOL>
```

```
*
```

15. Open the **Demo.mdb** and select the **design** button.

Notice that all the fields are set to text and the field length is set to 255. **All fields must always be set to text.** You do not need to set an index field. Access will ask you that, but you do not have to.

| Field Name | Data Type | Description |
|------------|-----------|-------------|
| Material   | Text      |             |
| Thickness  | Text      |             |
| Feedrate   | Text      |             |

Field Properties

General    Lookup

Field Size: 255

Format:

Input Mask:

Caption:

Default Value:

Validation Rule:

Validation Text:

Required: No

Allow Zero Length: No

Indexed: No

A field name can be up to 64 characters long, including spaces. Press F1 for help on field names.

16. Close the design window in the **Demo.mdb** and select the **open** button.

The form below shows the three fields with information filled in. This example only has 6 records. You can have as many fields and records as you want.

| Material  | Thickness | Feedrate |
|-----------|-----------|----------|
| STEEL     | .05       | 10       |
| STEEL     | .1        | 20       |
| STEEL     | .15       | 30       |
| STAINLESS | .05       | 5        |
| STAINLESS | .1        | 10       |
| STAINLESS | .15       | 15       |

Record: 1 of 6

Example

---

## **Appendix B CALC Sections**

---

# CALC Sections

The SECTION command determines the name of a section and what type of section it is. If the :SECTION= CALC\_? section equals starts with a CALC, then it is a calculation section; otherwise, it is a template section.

Calc sections:

## Mill

## Description

CALC\_LINE\_MOVE\_MILL  
 CALC\_ARC\_MOVE\_MILL  
 CALC\_RAPID\_MOVE\_MILL  
 CALC\_SINGLE\_DRILL\_MILL  
 CALC\_INIT\_TOOL\_CHANGE\_MILL  
 CALC\_SUB\_TOOL\_CHANGE\_MILL  
 CALC\_EVERY\_MOVE\_MILL  
 CALC\_GRID\_PATTERN\_DRILL  
 CALC\_DRILL\_INCREMENT\_DRILL  
 CALC\_BOLT\_HOLE\_CIR\_DRILL  
 CALC\_ARC\_PATTERN\_DRILL  
 CALC\_RAPID\_Z\_UP\_MILL  
 CALC\_RAPID\_Z\_DOWN\_MILL  
 CALC\_FEED\_Z\_MILL  
 CALC\_ROTATE\_X  
 CALC\_ROTATE\_Y  
 CALC\_ROTATE\_Z  
 CALC\_ARC\_MOVE\_MILL\_ZX

:SECTION= CALC\_ARC\_MOVE\_MILL\_ZX

Mill calc section for arc moves on ZX plane using CAMWorks 2005 or later. The post will only get to this section when you are doing an arc movement that is on the ZX plane. If section does not exist, then the post outputs line moves.

If you have an old post that does not have this section compiled in the source, then the post will break the arcs into line moves by a default deviation. If you are creating a new post and your machine does not support arcs on different planes, then you need to add the question "max\_arc\_dev" to either the Setup Info or the operation questions to set the deviation amount. This value will then set the post variable ARC\_DEVIATION to this value.

:C: ARC\_DEVIATION=max\_arc\_dev  
 :C: SYS\_CANNED(5,CALC\_BREAK\_ARC)



CALC\_ARC\_MOVE\_MILL\_YZ

:SECTION=CALC\_ARC\_MOVE\_MILL\_YZ

Mill calc section for arc moves on YZ plane using CAMWorks 2005 or later. The post will only get to this section when you are doing an arc movement that is on the YZ plane. If section does not exist, then the post outputs line moves.

If you have an old post that does not have this section compiled in the source, then the post will break the arcs into line moves by a default deviation. If you are creating a new post and your machine does not support arcs on different planes, then you need to add the question "max\_arc\_dev" to either the Setup Info or the operation questions to set the deviation amount. This value will then set the post variable ARC\_DEVIATION to this value.

:C: ARC\_DEVIATION=max\_arc\_dev

:C: SYS\_CANNED(5,CALC\_BREAK\_ARC)

CALC\_ARC\_MOVE\_MILL\_ANYPLANE

:SECTION=CALC\_ARC\_MOVE\_MILL\_ANYPLANE

Mill calc section for arc moves on non-standard planes using CAMWorks 2005 or later. The post will only get to this section when you are doing an arc movement that is on non-standard planes. If section does not exist, then the post outputs line moves.

If you have an old post that does not have this section compiled in the source, then the post will break the arcs into line moves by a default deviation. If you are creating a new post and your machine does not support arcs on different planes, then you need to add the question "max\_arc\_dev" to either the Setup Info or the operation questions to set the deviation amount. This value will then set the post variable ARC\_DEVIATION to this value.

:C: ARC\_DEVIATION=max\_arc\_dev

:C: SYS\_CANNED(5,CALC\_BREAK\_ARC)

CALC\_POST\_INITIALIZE

:SECTION= CALC\_POST\_INITIALIZE

:C: IF SECTIONEXIST(FIVE\_AXIS\_LINE\_MOVE\_MILL) THEN

:C: CALL(CALC\_RESET\_REGISTERS)

:C: CALL(CALC\_RESET\_FIVE\_AXIS\_REGISTERS)

:C: ENDIF

Mill calc section for setting 4 and 5 axis parameters used in ProCAM II only. The post will only get to this section when you have either

:4AXIS\_X\_MILLING=TRUE, :4AXIS\_Y\_MILLING=TRUE

or :5AXIS\_MILLING=TRUE, It will get to this section before it gets to CALC\_START\_OPERATION for the first time.

## CALC Sections

CALC\_TOOL\_INITIALIZE

:SECTION= CALC\_TOOL\_INITIALIZE

:C: HEAD\_LEN=(INIT\_TOOL\_LENGTH+head\_length)

Mill calc section for setting 4 and 5 axis HEAD\_LEN tool parameters. When you have a machine that has the Head that rotates or tilts and you need to add the tool length on to the posted output. Used in ProCAM II 2004 or CAMWorks 2005 or newer versions only.

The post will only get to this section when you have either :4AXIS\_X\_MILLING=TRUE,

:4AXIS\_Y\_MILLING=TRUE or

:5AXIS\_MILLING=TRUE, It will get to this section before it gets to CALC\_START\_OPERATION for the first time.

INIT\_TOOL\_LENGTH is a system post variable.

INIT\_TOOL\_LENGTH holds the tool length from tool definition.

Head\_length = is a post question and it can be added or subtracted.

### *Lathe*

CALC\_LINE\_MOVE\_LATHE  
CALC\_ARC\_MOVE\_LATHE  
CALC\_RAPID\_MOVE\_LATHE  
CALC\_SYSTEM\_THREAD\_LATHE  
CALC\_INIT\_TOOL\_CHANGE\_LATHE  
CALC\_SUB\_TOOL\_CHANGE\_LATHE  
CALC\_EVERY\_MOVE\_LATHE  
CALC\_MACHINE\_THREAD\_LATHE  
CALC\_SYSTEM\_DRILL\_LATHE  
CALC\_MACHINE\_DRILL\_LATHE  
CALC\_START\_BOUNDARY\_LATHE  
CALC\_END\_BOUNDARY\_LATHE

### *Punch*

CALC\_LINE\_MOVE\_PUNCH  
CALC\_ARC\_MOVE\_PUNCH  
CALC\_RAPID\_MOVE\_PUNCH  
CALC\_SINGLE\_HIT\_PUNCH  
CALC\_INIT\_TOOL\_CHANGE\_PUNCH  
CALC\_SUB\_TOOL\_CHANGE\_PUNCH  
CALC\_EVERY\_MOVE\_PUNCH  
CALC\_START\_OF\_TAPE\_PUNCH  
CALC\_END\_OF\_TAPE\_PUNCH  
CALC\_GRID\_PATTERN\_PUNCH  
CALC\_PUNCH\_INCREMENT\_PUNCH  
CALC\_BOLT\_HOLE\_CIR\_PUNCH  
CALC\_ARC\_PATTERN\_PUNCH  
CALC\_WINDOW\_PUNCH  
CALC\_WINDOW\_FRAME\_PUNCH  
CALC\_REPOSITION\_PUNCH  
CALC\_OFFSET\_PART\_PUNCH  
CALC\_BEG\_MACRO\_PUNCH  
CALC\_END\_MACRO\_PUNCH  
CALC\_MULTIPLE\_MACRO\_CALL\_PUNCH  
CALC\_MIRROR\_MACRO\_CALL\_PUNCH

### **Punch**

CALC\_MACRO\_CALL\_PUNCH  
 CALC\_MULTIPLE\_MACRO\_DEFINE\_PUNCH  
 CALC\_SETUP\_SHEET\_PUNCH

### **Plasma**

### **Description**

CALC\_LINE\_MOVE\_PLASMA  
 CALC\_ARC\_MOVE\_PLASMA  
 CALC\_RAPID\_MOVE\_PLASMA  
 CALC\_INIT\_TOOL\_CHANGE\_PLASMA  
 CALC\_SUB\_TOOL\_CHANGE\_PLASMA  
 CALC\_EVERY\_MOVE\_PLASMA  
 CALC\_GRID\_PATTERN\_PLASMA  
 CALC\_PLASMA\_INCREMENT\_PLASMA  
 CALC\_BOLT\_HOLE\_CIR\_PLASMA  
 CALC\_ARC\_PATTERN\_PLASMA  
 CALC\_REPOSITION\_PLASMA  
 CALC\_RAPID\_TO\_TRAPDOOR\_PLASMA

:SECTION= CALC\_RAPID\_TO\_TRAP\_DOOR  
 Plasma system rapid move to trap door calc section. This section will be called when a trap door is attached to any entity. The header command :TRAPDOOR must equal either DROP or TILT. If this header command is not in the post or is set to FALSE, then you will not get to this calc section.

CALC\_PROFILE\_DRILL\_PLASMA

:SECTION= CALC\_PROFILE\_DRILL\_PLASMA  
 Used when you have a PUNCH/PLASMA or a PUNCH/LASER combination machine that uses a prepunched hole for starting a plasma or laser profile. This section will be called when you do prepunch hole for the plasma or laser profile. Normally, you would do a punch single hit and this, then give the plasma or laser a place to start the cut if not on the edge of the sheet.

## CALC Sections

### *Laser*

CALC\_LINE\_MOVE\_LASER  
CALC\_ARC\_MOVE\_LASER  
CALC\_RAPID\_MOVE\_LASER  
CALC\_PROFILE\_DRILL\_LASER  
CALC\_INIT\_TOOL\_CHANGE\_LASER  
CALC\_SUB\_TOOL\_CHANGE\_LASER  
CALC\_EVERY\_MOVE\_LASER  
CALC\_GRID\_PATTERN\_LASER  
CALC\_LASER\_INCREMENT\_LASER  
CALC\_BOLT\_HOLE\_CIR\_LASER  
CALC\_ARC\_PATTERN\_LASER  
CALC\_REPOSITION\_LASER  
CALC\_RAPID\_TO\_TRAPDOOR\_LASER

### *Description*

:SECTION= CALC\_RAPID\_TO\_TRAP\_DOOR

Laser system rapid move to trap door calc section. This section will be called when a trap door is attached to any entity. The header command :TRAPDOOR must equal either DROP or TILT. If this header command is not in the post or is set to FALSE, then you will not get to this calc section.

CALC\_PROFILE\_DRILL\_LASER

:SECTION= CALC\_PROFILE\_DRILL\_LASER

Used when you have a PUNCH/PLASMA or a PUNCH/LASER combination machine that uses a prepunched hole for starting a plasma or laser profile. This section will be called when you do prepunch hole for the plasma or laser profile. Normally, you would do a punch single hit and this, then give the plasma or laser a place to start the cut if not on the edge of the sheet.

### *Shear*

CALC\_RAPID\_MOVE\_SHEAR  
CALC\_INIT\_TOOL\_CHANGE\_SHEAR  
CALC\_SUB\_TOOL\_CHANGE\_SHEAR  
CALC\_EVERY\_MOVE\_SHEAR  
CALC\_FULL\_SHEAR  
CALC\_HALF\_SHEAR\_X  
CALC\_HALF\_SHEAR\_Y  
CALC\_FULL\_SHEAR\_DIAGONAL  
CALC\_HALF\_SHEAR\_DIAGONAL  
CALC\_REPOSITION\_SHEAR

### **EDM**

```
CALC_LINE_MOVE_EDM
CALC_ARC_MOVE_EDM
CALC_POINT_MOVE_EDM
CALC_RAPID_MOVE_EDM
CALC_INIT_TOOL_CHANGE_EDM
CALC_SUB_TOOL_CHANGE_EDM
CALC_EVERY_MOVE_EDM
CALC_START_HOLE_EDM
CALC_END_HOLE_EDM
CALC_GET_TAPER_EDM
```

### **Description**

```
:SECTION= CALC_GET_TAPER_EDM
```

Used when you need to get all the different taper angles used in the current part and output them at the start of the program. This section will be called only when it is inserted into the post source. When you start to post out, the current EDM part file the system will go through the complete tool paths of the part file to gather all the different taper changes, then it will start calling this section for each different taper angle and you can store this information in post arrays. Now it will start the normal post output.

### **Misc**

```
CALC_BEFORE_ATTRIBUTES
CALC_DURING_ATTRIBUTES
CALC_AFTER_ATTRIBUTES
CALC_SWITCH_TO_PLASMA
CALC_SWITCH_TO_PUNCH
CALC_START_OF_TAPE
CALC_END_OF_TAPE
CALC_OFFSET_PART
CALC_BEG_MACRO
CALC_END_MACRO
CALC_MULTIPLE_MACRO_CALL
CALC_MIRROR_MACRO_CALL
CALC_MACRO_CALL
CALC_MULTIPLE_MACRO_DEFINE
CALC_SETUP_SHEET
CALC_START_OPERATION
CALC_END_OPERATION
```

