CAMWorks®
A Geometric Product

PROCAM™ II

POST PROCESSOR

WRITER'S REFERENCE

Disclaimer

Geometric Americas, Inc. makes no warranties, either express or implied with respect to this manual. Geometric Americas, Inc. reserves the right to revise and improve products as it sees fit, and to revise the specifications and information contained herein without prior notice. Due to continuing product development, specifications and capabilities described in this manual are subject to change without notice.

Trademarks

CAMWorks® is a registered trademark of Geometric Americas, Inc.

ProCAM II ® is a registered trademark of TekSoft, Inc.

Product Name: **CAMWorks**
Release: **2015**
Version: **Service Pack 1.0**

Table of Contents

Chapter 1: System Commands

ABS

Purpose

Returns the absolute value of an argument or number.

Syntax

ABS(arg)

Comments

Parameter	*Description*
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

Parameter	*Description*
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

Parameter	*Description*
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

Parameter	*Description*
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

Parameter	*Description*
r	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

Parameter	*Description*
r	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

Parameter	*Description*
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

Parameter	*Description*
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

Parameter	*Description*
`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

Parameter	*Description*
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(STRA,STRB)
```

SETON

Purpose

To set a parameter to output code.

Syntax

SETON(<name>)

Comments

Parameter	*Description*
`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

Parameter	*Description*
`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 1 Single points 2 Lines, arcs and bolt holes (use only on grids and big hole patterns) 3 Breaks a thread cycle into diameters (use only on threading cycles)
section	section that will handle the exploded entity The word SYSTEM 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

GOTOnumber

Comments

Parameter	*Description*
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

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

Parameter	*Description*
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

Parameter	*Description*
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

Parameter	*Description*
`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

Parameter	*Description*
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

Parameter	*Description*
`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)
```

ROUNDOFF

Purpose

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

Syntax

ROUNDOFF(number,bucket,result,places,type)

Comments

Parameter	*Description*
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: ROUNDOFF(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.

ID	*ZGL*	*XGL*	*Tool*	*Comment*
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

Parameter	*Description*
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
```

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
```

QUERY_INT_HAND_OF_TOOL

Purpose

This command will query the hand of the tool. Available in Mill, Turn and Mill-Turn. Supported in CAMWorks 2013 and later. Not supported in any ProCAM product.

Syntax

QUERY_ITEM_ID=QUERY_INT_HAND_OF_TOOL

Comments

Associated commands and variables:
The example code below shows the other variables used with this command. If the Query result was "1" then the value will be in `QUERY_INT_VAL`.

Example

```
:C: QUERY_ITEM_ID=QUERY_INT_HAND_OF_TOOL
:C: QUERY_SYSTEM()
:C: IF QUERY_RESULT=1 THEN
:C:    IF QUERY_INT_VAL=1 THEN
*      Hand of tool = "LEFT"
:C:    ENDIF
:C:    IF QUERY_INT_VAL=2 THEN
*      Hand of tool = "RIGHT"
:C:    ENDIF
:C: ENDIF
```

QUERY_DEC_OPER_TIME

Purpose

This command will query the estimated machine time per operation. Available in Mill, Turn and Mill-Turn. Supported in CAMWorks 2013 and later. Not supported in any ProCAM product.

Syntax

QUERY_ITEM_ID=QUERY_DEC_OPER_TIME

Comments

Associated commands and variables:
The example code below shows the other variables used with this command. If the Query result was "1" then the value will be in `QUERY_DEC_VAL`.

Example

```
:C: QUERY_ITEM_ID=QUERY_DEC_OPER_TIME
:C: QUERY_SYSTEM()
:C: IF QUERY_RESULT=1 THEN
*    Post Variable = QUERY_DEC_VAL
:C: ENDIF
```

QUERY_DEC_MIN_PROTRUSION

Purpose

This command will query the Minimum tool protrusion in 3axis operation. Available in Mill, Turn and Mill-Turn. Supported in CAMWorks 2013 and later. Not supported in any ProCAM product.

Syntax

QUERY_ITEM_ID=QUERY_DEC_MIN_PROTRUSION

Comments

Associated commands and variables:
The example code below shows the other variables used with this command. If the Query result was "1" then the value will be in `QUERY_DEC_VAL`.

Example

```
:C: QUERY_ITEM_ID=QUERY_DEC_MIN_PROTRUSION
:C: QUERY_SYSTEM()
:C: IF QUERY_RESULT=1 THEN
*    Post Variable = QUERY_DEC_VAL
:C: ENDIF
```

QUERY_EXPORT_ERROR

Purpose

This command will query an error for finding sync codes if post is not setup to handle that. The error will be sent to the CAMWorks error message box. Available in Mill, Turn and Mill-Turn. Supported in CAMWorks 2013 and later. Not supported in any ProCAM product.

Syntax

QUERY_ITEM_ID= QUERY_EXPORT_ERROR

Comments

Associated commands and variables:
The example code below shows the other variables used with this command. This logic should be in

```
:SECTION=CALC_ADD_REAR_SYNC_CODE
```
and
```
:SECTION=CALC_ADD_FRONT_SYNC_CODE
```

Example

```
:C: QUERY_ITEM_ID= QUERY_EXPORT_ERROR
:C: QUERY_ERROR= HAVE_SYNC_CODE_ERROR
:C: QUERY_SYSTEM()
```

QUERY_INT_NUM_TURRETS_ABOVE

Purpose

This command will query the number of turrets above Spindle centerline. This is used in conjunction with CAMWorks Virtual Machine. Available in Mill, Turn and Mill-Turn. Supported in CAMWorks 2013 and later. Not supported in any ProCAM product.

Syntax

QUERY_ITEM_ID=QUERY_INT_NUM_TURRETS_ABOVE

Comments

Associated commands and variables:
The example code below shows the other variables used with this command. This logic should be in

```
:SECTION=CALC_QUERY_POST
```

Example

```
:C: IF QUERY_ITEM_ID = QUERY_INT_NUM_TURRETS_ABOVE  THEN
:C: QUERY_INT_VAL=1
:C: RETURN
:C: ENDIF
```

QUERY_INT_NUM_TURRETS_BELOW

Purpose

This command will query the number of turrets below Spindle centerline. This is used in conjunction with CAMWorks Virtual Machine. Available in Mill, Turn and Mill-Turn. Supported in CAMWorks 2013 and later. Not supported in any ProCAM product.

Syntax

QUERY_ITEM_ID=QUERY_INT_NUM_TURRETS_BELOW

Comments

Associated commands and variables:
The example code below shows the other variables used with this command. This logic should be in

```
:SECTION=CALC_QUERY_POST
```

Example

```
:C: IF QUERY_ITEM_ID = QUERY_INT_NUM_TURRETS_BELOW  THEN
:C: QUERY_INT_VAL=1
:C: RETURN
:C: ENDIF
```

QUERY_GCODE_DEFAULT_WCS

Purpose

This command will query the number of default work coordinates in the posted output. This is used in conjunction with CAMWorks Virtual Machine. Available in Mill, Turn and Mill-Turn. Supported in CAMWorks 2013 and later. Not supported in any ProCAM product.

Syntax

QUERY_ITEM_ID=QUERY_GCODE_DEFAULT_WCS

Comments

Associated commands and variables:
The example code below shows the other variables used with this command. This logic should be in

```
:SECTION=CALC_QUERY_POST
```

Example

```
:C: IF QUERY_ITEM_ID = QUERY_GCODE_DEFAULT_WCS THEN
:C:    IF SECTIONEXIST(QUERY_WCS) THEN
:C:    CALL(QUERY_WCS)  -  Have this section in your post if you
       need to do something special.
:C:    RETURN
:C:    ELSE
:C:    CALL(AUTO_QUERY_WCS) - This section will be in the General
       library files.
:C:    RETURN
:C:    ENDIF
:C: ENDIF
*
:SECTION=AUTO_QUERY_WCS
:T: IF MCS_TYPE=2 OR COORD_TYPE=0 THEN <G!:work_coord><EOL>ENDIF
:T: IF MCS_TYPE=3 AND work_coord=54 THEN
    <G!:work_coord><POINT_ONE><EOL>ENDIF
:T: IF MCS_TYPE=3 AND work_coord>54 THEN <G!:work_coord><EOL>ENDIF
```

QUERY_GCODE_WCS

Purpose

This command will query the number of work coordinates in the posted output. This is used in conjunction with CAMWorks Virtual Machine. Available in Mill, Turn and Mill-Turn. Supported in CAMWorks 2013 and later. Not supported in any ProCAM product.

Syntax

QUERY_ITEM_ID=QUERY_GCODE_WCS

Comments

Associated commands and variables:
The example code below shows the other variables used with this command. This logic should be in

```
:SECTION=CALC_QUERY_POST
```

Example

```
:C: IF QUERY_ITEM_ID = QUERY_GCODE_WCS THEN
:C:    IF SECTIONEXIST(QUERY_WCS) THEN
:C:    CALL(QUERY_WCS)  -  Have this section in your post if you
       need to do something special.
:C:    RETURN
:C:    ELSE
:C:    CALL(AUTO_QUERY_WCS) - This section will be in the General
       library files.
:C:    RETURN
:C:    ENDIF
:C: ENDIF
*
:SECTION=AUTO_QUERY_WCS
:T: IF MCS_TYPE=2 OR COORD_TYPE=0 THEN <G!:work_coord><EOL>ENDIF
:T: IF MCS_TYPE=3 AND work_coord=54 THEN
    <G!:work_coord><POINT_ONE><EOL>ENDIF
:T: IF MCS_TYPE=3 AND work_coord>54 THEN <G!:work_coord><EOL>ENDIF
```

QUERY_INT_LENGTH_REG

Purpose

This command will query the Tool Length offsets in the posted output. This is used in conjunction with CAMWorks Virtual Machine. Available in Mill, Turn and Mill-Turn. Supported in CAMWorks 2013 and later. Not supported in any ProCAM product.

Syntax

QUERY_ITEM_ID=QUERY_INT_LENGTH_REG

Comments

Associated commands and variables:
The example code below shows the other variables used with this command. This logic should be in

```
:SECTION=CALC_QUERY_POST
```

Example

```
:C: IF QUERY_ITEM_ID = QUERY_INT_LENGTH_REG THEN
:C: QUERY_INT_VAL=TOOL
:C: RETURN
:C: ENDIF
```

QUERY_INT_OFFSET_REG

Purpose

This command will query the Tool Diameter offsets in the posted output. This is used in conjunction with CAMWorks Virtual Machine. Available in Mill, Turn and Mill-Turn. Supported in CAMWorks 2013 and later. Not supported in any ProCAM product.

Syntax

QUERY_ITEM_ID=QUERY_INT_OFFSET_REG

Comments

Associated commands and variables:
The example code below shows the other variables used with this command. This logic should be in

```
:SECTION=CALC_QUERY_POST
```

Example

```
:C: IF QUERY_ITEM_ID = QUERY_INT_OFFSET_REG THEN
:C: CALL(CALC_QUERY_DIAM_OFFSET_REG) - This section is in the
    General library files.
:C: RETURN
:C: ENDIF
*
:SECTION=CALC_QUERY_DIAM_OFFSET_REG
:C: QUERY_INT_VAL=(TOOL+COMP_OFFSET)
:C: IF TOOL_LENGTH_DIAM_OFFSET_METHOD=METHOD_FROM_TOOL THEN
:C:    QUERY_INT_VAL=TOOL_DIAM_OFFSET
:C: ENDIF
```

QUERY_GCODE_AXIS_NAMES_T1A

Purpose

This command will define which axis is controlled by work offsets for the 1st turret above centerline in the posted output. This is used in conjunction with CAMWorks Virtual Machine. Available in Mill, Turn and Mill-Turn. Supported in CAMWorks 2013 and later. Not supported in any ProCAM product.

Syntax

QUERY_ITEM_ID=QUERY_GCODE_AXIS_NAMES_T1A

Comments

Associated commands and variables:
The example code below shows the other variables used with this command. This logic should be in

```
:SECTION=CALC_QUERY_POST
```

Example

```
:C: IF QUERY_ITEM_ID = QUERY_GCODE_AXIS_NAMES_T1A THEN
:C: CALL(QUERY_AXIS_NAMES_T1A)- This section is in the General
    library files.
:C: RETURN
:C: ENDIF
*
:SECTION=QUERY_AXIS_NAMES_T1A
:T:X,Y,Z<EOL>
```

QUERY_GCODE_AXIS_NAMES_T2A

Purpose

This command will define which axis is controlled by work offsets for the 1st turret above centerline in the posted output. This is used in conjunction with CAMWorks Virtual Machine. Available in Mill, Turn and Mill-Turn. Supported in CAMWorks 2013 and later. Not supported in any ProCAM product.

Syntax

QUERY_ITEM_ID=QUERY_GCODE_AXIS_NAMES_T2A

Comments

Associated commands and variables:
The example code below shows the other variables used with this command. This logic should be in

```
:SECTION=CALC_QUERY_POST
```

Example

```
:C: IF QUERY_ITEM_ID = QUERY_GCODE_AXIS_NAMES_T2A THEN
:C: CALL(QUERY_AXIS_NAMES_T2A)- This section is in the General
    library files.
:C: RETURN
:C: ENDIF
*
:SECTION=QUERY_AXIS_NAMES_T2A
:T:X,Y,Z<EOL>
```

QUERY_GCODE_AXIS_NAMES_T1B

Purpose

This command will define which axis is controlled by work offsets for the 1st turret above centerline in the posted output. This is used in conjunction with CAMWorks Virtual Machine. Available in Mill, Turn and Mill-Turn. Supported in CAMWorks 2013 and later. Not supported in any ProCAM product.

Syntax

QUERY_ITEM_ID=QUERY_GCODE_AXIS_NAMES_T1B

Comments

Associated commands and variables:
The example code below shows the other variables used with this command. This logic should be in

```
:SECTION=CALC_QUERY_POST
```

Example

```
:C: IF QUERY_ITEM_ID = QUERY_GCODE_AXIS_NAMES_T1B THEN
:C: CALL(QUERY_AXIS_NAMES_T1B)- This section is in the General
    library files.
:C: RETURN
:C: ENDIF
*
:SECTION=QUERY_AXIS_NAMES_T1B
:T:X,Y,Z<EOL>
```

QUERY_GCODE_AXIS_NAMES_T2B

Purpose

This command will define which axis is controlled by work offsets for the 1st turret above centerline in the posted output. This is used in conjunction with CAMWorks Virtual Machine. Available in Mill, Turn and Mill-Turn. Supported in CAMWorks 2013 and later. Not supported in any ProCAM product.

Syntax

QUERY_ITEM_ID=QUERY_GCODE_AXIS_NAMES_T2B

Comments

Associated commands and variables:
The example code below shows the other variables used with this command. This logic should be in

```
:SECTION=CALC_QUERY_POST
```

Example

```
:C: IF QUERY_ITEM_ID = QUERY_GCODE_AXIS_NAMES_T2B THEN
:C: CALL(QUERY_AXIS_NAMES_T2B)- This section is in the General
    library files.
:C: RETURN
:C: ENDIF
*
:SECTION=QUERY_AXIS_NAMES_T2B
:T:X,Y,Z<EOL>
```

QUERY_GCODE_AXIS_NAMES_MILL

Purpose

This command will define which axis is controlled by work offsets for milling in the posted output. This is used in conjunction with CAMWorks Virtual Machine.

Available in Mill, Turn and Mill-Turn.

Supported in CAMWorks 2013 and later. Not supported in any ProCAM product.

Syntax

QUERY_ITEM_ID=QUERY_GCODE_AXIS_NAMES_MILL

Comments

Associated commands and variables:

Note: QUERY_GCODE_AXIS_NAMES_T1A can be used as a default in milling also.

The example code below shows the other variables used with this command. This logic should be in

:SECTION=CALC_QUERY_POST

Example

```
:C: IF QUERY_ITEM_ID = QUERY_GCODE_AXIS_NAMES_MILL THEN
:C: CALL(QUERY_AXIS_NAMES_MILL)- This section is in the General
library files.
:C: RETURN
:C: ENDIF
*
:SECTION=QUERY_AXIS_NAMES_MILL
:T:X,Y,Z<EOL>
```

QUERY_GCODE_POSTED_FILE_NAME

Purpose

This command will define what is the file name for the milling posted output. This is used in conjunction with CAMWorks Virtual Machine.
Available in Mill, Turn and Mill-Turn. Supported in CAMWorks 2013 and later. Not supported in any ProCAM product.
This command is not needed if using the file path and name as in the posting dialog box.

Syntax

QUERY_ITEM_ID=QUERY_GCODE_POSTED_FILE_NAME

Comments

Associated commands and variables:
Note: QUERY_GCODE_FILE_NAME_T1A can be used as a default in milling also.

The example code below shows the other variables used with this command. This logic should be in

:SECTION=CALC_QUERY_POST

Example

```
:C: IF QUERY_ITEM_ID = QUERY_GCODE_POSTED_FILE_NAME THEN
:C: CALL(QUERY_POSTED_FILE_NAME )
:C: RETURN
:C: ENDIF
*
:SECTION=QUERY_POSTED_FILE_NAME
:T:file name<EOL>
```

QUERY_GCODE_FILE_NAME_T1A

Purpose

This command will define what is the file name for 1st turret above centerline. This is used in conjunction with CAMWorks Virtual Machine. Available in Mill, Turn and Mill-Turn. Supported in CAMWorks 2013 and later. Not supported in any ProCAM product. This command is not needed if using the file path and name as in the posting dialog box.

Syntax

QUERY_ITEM_ID=QUERY_GCODE_FILE_NAME_T1A

Comments

Associated commands and variables:
The example code below shows the other variables used with this command. This logic should be in

```
:SECTION=CALC_QUERY_POST
```

Example

```
:C: IF QUERY_ITEM_ID = QUERY_GCODE_FILE_NAME_T1A THEN
:C: CALL(QUERY_FILE_NAME_T1A )
:C: RETURN
:C: ENDIF
*
:SECTION=QUERY_FILE_NAME_T1A
:T:file name<EOL>
```

QUERY_GCODE_FILE_NAME_T2A

Purpose

This command will define what is the file name for 1st turret above centerline. This is used in conjunction with CAMWorks Virtual Machine. Available in Mill, Turn and Mill-Turn. Supported in CAMWorks 2013 and later. Not supported in any ProCAM product. This command is not needed if using the file path and name as in the posting dialog box.

Syntax

QUERY_ITEM_ID=QUERY_GCODE_FILE_NAME_T2A

Comments

Associated commands and variables:
The example code below shows the other variables used with this command. This logic should be in

```
:SECTION=CALC_QUERY_POST
```

Example

```
:C: IF QUERY_ITEM_ID = QUERY_GCODE_FILE_NAME_T2A THEN
:C: CALL(QUERY_FILE_NAME_T2A )
:C: RETURN
:C: ENDIF
*
:SECTION=QUERY_FILE_NAME_T2A
:T:file name<EOL>
```

QUERY_GCODE_FILE_NAME_T1B

Purpose

This command will define what is the file name for 1st turret above centerline. This is used in conjunction with CAMWorks Virtual Machine. Available in Mill, Turn and Mill-Turn. Supported in CAMWorks 2013 and later. Not supported in any ProCAM product. This command is not needed if using the file path and name as in the posting dialog box.

Syntax

QUERY_ITEM_ID=QUERY_GCODE_FILE_NAME_T1B

Comments

Associated commands and variables:
The example code below shows the other variables used with this command. This logic should be in

```
:SECTION=CALC_QUERY_POST
```

Example

```
:C: IF QUERY_ITEM_ID = QUERY_GCODE_FILE_NAME_T1B THEN
:C: CALL(QUERY_FILE_NAME_T1B )
:C: RETURN
:C: ENDIF
*
:SECTION=QUERY_FILE_NAME_T1B
:T:file name<EOL>
```

QUERY_GCODE_FILE_NAME_T2B

Purpose

This command will define what is the file name for 1st turret above centerline. This is used in conjunction with CAMWorks Virtual Machine.
Available in Mill, Turn and Mill-Turn. Supported in CAMWorks 2013 and later. Not supported in any ProCAM product.
This command is not needed if using the file path and name as in the posting dialog box.

Syntax

QUERY_ITEM_ID=QUERY_GCODE_FILE_NAME_T2B

Comments

Associated commands and variables:
The example code below shows the other variables used with this command. This logic should be in

```
:SECTION=CALC_QUERY_POST
```

Example

```
:C: IF QUERY_ITEM_ID = QUERY_GCODE_FILE_NAME_T2B THEN
:C: CALL(QUERY_FILE_NAME_T2B )
:C: RETURN
:C: ENDIF
*
:SECTION=QUERY_FILE_NAME_T2B
:T:file name<EOL>
```

QUERY_INT_MILL_ROUGH_TYPE

Purpose

This command will query the Rough Mill Type.
Available in Mill and Mill-Turn. Supported in CAMWorks 2013 and later. Not supported in any ProCAM product.
This command is not needed if using the file path and name as in the posting dialog box.

Syntax

QUERY_ITEM_ID=QUERY_INT_MILL_ROUGH_TYPE

Comments

Associated commands and variables:
The example code below shows the other variables used with this command. If the Query result was "1" then the value will be in `QUERY_INT_VAL`

Example

```
:C: QUERY_ITEM_ID=QUERY_INT_MILL_ROUGH_TYPE
:C: QUERY_SYSTEM()
:C: IF QUERY_RESULT=1 THEN
:C:    IF QUERY_INT_VAL=0 THEN
*         2 axis "Pocket In" = MILL_ROUGH_TYPE_SPIRALIN
:C:    ENDIF
:C:    IF QUERY_INT_VAL=1 THEN
*         2 axis "Pocket Out" = MILL_ROUGH_TYPE_SPIRALOUT
*         3 axis "Pocket Out" = MILL_ROUGH_TYPE_SPIRALOUT
:C:    ENDIF
:C:    IF QUERY_INT_VAL=2 THEN
*         2 axis "Zig" = MILL_ROUGH_TYPE_ZIG
:C:    ENDIF
:C:    IF QUERY_INT_VAL=3 THEN
*         2 axis "Zigzag" = MILL_ROUGH_TYPE_ZIGZAG
*         3 axis "Lace" = MILL_ROUGH_TYPE_ZIGZAG
:C:    ENDIF
:C:    IF QUERY_INT_VAL=4 THEN
*         2 axis "Spiral In" = MILL_ROUGH_TYPE_TRUESPIRALIN
:C:    ENDIF
:C:    IF QUERY_INT_VAL=5 THEN
*         2 axis "Spiral Out" = MILL_ROUGH_TYPE_TRUESPIRALOUT
:C:    ENDIF
:C:    IF QUERY_INT_VAL=6 THEN
*         2 axis "Plunge Rough" = MILL_ROUGH_TYPE_PLUNGEROUGH
:C:    ENDIF
:C:    IF QUERY_INT_VAL=7 THEN
*         2 axis "Offset Roughing" = MILL_ROUGH_TYPE_POCKETIN_CORE
*         3 axis "Pocket In Core" = MILL_ROUGH_TYPE_POCKETIN_CORE
:C:    ENDIF
```

```
:C:     IF QUERY_INT_VAL=8 THEN
*          2 axis "Volumill" = MILL_ROUGH_TYPE_VOLUMILL
*          3 axis "Volumill" = MILL_ROUGH_TYPE_VOLUMILL
:C:     ENDIF
:C:     IF QUERY_INT_VAL=19 THEN
*          3 axis "Adaptive" = MILL_ROUGH_TYPE_ADAPTIVE
:C:     ENDIF
:C: ENDIF
```

QUERY_INT_PRIMARY_TOUCHOFF_REG

Purpose

This command will query the Primary Tool Length offsets in the posted output. This is used in conjunction with CAMWorks Virtual Machine, Turn and Mill-Turn.
Supported in CAMWorks 2013 SP1 and later. Not supported in any ProCAM product.

Syntax

QUERY_ITEM_ID= QUERY_INT_PRIMARY_TOUCHOFF_REG

Comments

Associated commands and variables:
The example code below shows the other variables used with this command. This logic should be in: `SECTION=CALC_QUERY_POST`

Example

```
:C: IF QUERY_ITEM_ID =QUERY_INT_PRIMARY_TOUCHOFF_REG THEN
:C: QUERY_INT_VAL=TOOL
:C: RETURN
:C: ENDIF
```

QUERY_INT_SECONDARY_TOUCHOFF_REG

Purpose

This command will query the Secondary Tool Length offsets in the posted output. This is used in conjunction with CAMWorks Virtual Machine, Turn and Mill-Turn.
Supported in CAMWorks 2013 SP1 and later. Not supported in any ProCAM product.

Syntax

QUERY_ITEM_ID= QUERY_INT_SECONDARY_TOUCHOFF_REG

Comments

Associated commands and variables:
The example code below shows the other variables used with this command. This logic should be in: `SECTION=CALC_QUERY_POST`

Example

```
:C: IF QUERY_ITEM_ID =QUERY_INT_SECONDARY_TOUCHOFF_REG THEN
:C: QUERY_INT_VAL=TOOL
:C: RETURN
:C: ENDIF
```

QUERY_INT_ZERO_OFFSET_RADIUS

Purpose

If the return value of QUERY_INT_VAL is TRUE, the system will zero out the Radius in the Eureka Offset radius correctors for this tool.
If the return value of QUERY_INT_VAL is FALSE, the system will not zero out the Radius in the Eureka Offset radius correctors for this Tool.
If the Post doesn't know whether to zero out the Radius offset, then don't set it since the system has already pre-set the QUERY_INT_VAL based on whether CNC comp with part geometry or CNC comp for Tool wear only was used.

Syntax

QUERY_ITEM_ID=QUERY_INT_ZERO_OFFSET_RADIUS

Comments

Associated commands and variables:
The example code below shows the other variables used with this command. This logic should be in: `SECTION=CALC_QUERY_POST`

OUTPUT_START_SYNC_CODES()

Purpose

To Find any Auto Start Sync Codes from Sync Manager.

Syntax

This logic should be in CALC_START_OPERATION

```
:C:     IF HAVE_START_OPER_SYNC_CODES=TRUE THEN
:C:     OUTPUT_START_SYNC_CODES()
:C:     ENDIF
```

Example

If rear start sync codes are found then it gets passed to this system section

```
:SECTION=CALC_ADD_REAR_SYNC_CODE
```

If front start sync codes are found then it gets passed to this system section

```
:SECTION=CALC_ADD_FRONT_SYNC_CODE
```

OUTPUT_FIRST_MOVE_SYNC_CODES()

Purpose

To find any Auto after first move Sync Codes from Sync Manager.

Syntax

This logic should be in CALC_EVERY_MOVE_MILL and CALC_EVERY_MOVE_LATHE

```
:C: IF HAVE_FIRST_MOVE_SYNC_CODES=TRUE THEN
:C: OUTPUT_FIRST_MOVE_SYNC_CODES()
:C: ENDIF
```

Example

If rear start sync codes are found then it gets passed to this system section

```
:SECTION=CALC_ADD_REAR_SYNC_CODE
```

If front start sync codes are found then it gets passed to this system section

```
:SECTION=CALC_ADD_FRONT_SYNC_CODE
```

OUTPUT_END_SYNC_CODES()

Purpose

To find any Auto End Sync Codes from Sync Manager.

Syntax

This logic should be in CALC_END_OPERATION

```
:C:     IF HAVE_END_OPER_SYNC_CODES=TRUE THEN
:C:     OUTPUT_END_SYNC_CODES()
:C:     ENDIF
```

Example

If rear start sync codes are found then it gets passed to this system section

```
:SECTION=CALC_ADD_REAR_SYNC_CODE
```

If front start sync codes are found then it gets passed to this system section

```
:SECTION=CALC_ADD_FRONT_SYNC_CODE
```

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()
```

QUERY_INT_KIN_SETUP_POS

Purpose

If the return value of QUERY_INT_VAL is TRUE, the system will look for the Kin file and the result will be passed back to the post as X, Y, Z, I, J, K from the Kinematic Setup Transform in below post variables. To be used in CAMWorks 2014 SP3 or later versions.

Syntax

QUERY_ITEM_ID=QUERY_INT_KIN_SETUP_POS

Comments

Associated commands and variables:

The example code below shows the other variables used with this command.

```
:C: QUERY_ITEM_ID=QUERY_INT_KIN_POS
*    If you need length comp then add below line.
*    QUERY_INT_VAL=ALLOW_TOOL_HEAD_LENGTH_COMP
:C: QUERY_SYSTEM()
:C: IF QUERY_RESULT=TRUE THEN
*    Post Variable = SETUP_MACH_X
*    Post Variable = SETUP_MACH_Y
*    Post Variable = SETUP_MACH_Z
*    Post Variable = SETUP_MACH_I
*    Post Variable = SETUP_MACH_J
*    Post Variable = SETUP_MACH_K
:C: ENDIF
```

QUERY_MILL_TLP_Z_EXTENTS

Purpose

If the return value of QUERY_RESULT is TRUE, the system will pass the Z extents in post variables called QUERY_TLP_Z_MIN and QUERY_TLP_Z_MAX. To be used in CAMWorks 2014 SP3 or later versions.

Syntax

QUERY_ITEM_ID=QUERY_MILL_TLP_Z_EXTENTS

Comments

Associated commands and variables:

The example code below shows the other variables used with this command.

```
:C: QUERY_ITEM_ID=QUERY_MILL_TLP_Z_EXTENTS
:C: QUERY_SYSTEM()
:C: IF QUERY_RESULT=TRUE THEN
*    Post Variable = QUERY_TLP_Z_MIN
*    Post Variable = QUERY_TLP_Z_MAX
:C: ENDIF
```

QUERY_FEARURE_STRATEGY

Purpose

If the return value of QUERY_RESULT is TRUE, then the system will pass the feature strategy information to the post variable called QUERY_CHAR_VAL. To be used in CAMWorks 2015 SP1 or newer version.

Syntax

QUERY_ITEM_ID=QUERY_FEATURE_STRATEGY

Comments

Associated commands and variables:

The example code below shows the other variables used with this command.

```
:C: QUERY_ITEM_ID= QUERY_FEATURE_STRATEGY
:C: QUERY_SYSTEM()
:C: IF QUERY_RESULT=TRUE THEN
**  If you need to output this in the code, then you will need
    to create a post attribute as a character and use
    QUERY_CHAR_VAL as the :VAR=
:C: CALL(section name)
**  You can also save this to a post variable and use it for
    checking for a different feature strategy
:C: ENDIF
```

Chapter 2: System Text Line Commands

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>
```

l

Purpose

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

Syntax

l

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<"#34l":ABS_X_END><EOL>
```

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

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()

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=elipse
:ATTRTYPE=LIST
:ATTRSEL=N
:ATTRTITLE=Elipse
:ATTRLIST=major
:ATTRLISTDEF=10
:ATTRLIST=minor
:ATTRLISTDEF=5
:ATTRLIST=nsegments
:ATTRLISTDEF=120
:ATTRUSED=1
:ATTRDEFAULT=1
:ATTREND

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

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()

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()

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)
```

#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

```
: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
: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()

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()

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()

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=

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=

Purpose

Defines attribute type.

Syntax

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

Comments

Parameter	*Description*
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
:ATTREND

:ATTRNAME=optional stop
:ATTRTYPE=DISPLAY
:ATTREMARK=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
:ATTLISTDEF=0
:ATTRUSED=1
:ATTRDEFAULT=1
:ATTREND
```

: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

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
:ATTLISTDEF=0
:ATTRUSED=1
:ATTRDEFAULT=1
:ATTREND
```

: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
:ATTLISTDEF=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=

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=

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=

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=

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=

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=

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_BEF=|(
: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

Parameter	*Description*
FORMAT	Defines how you want your output to be
SELECT	defines a selectable output.
SELECT_FORMAT	Defines two variables. The first is a select and the second 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_BEF=|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

: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_BEF=|F|
:VARB=F
:LEFT_PLACES=2
:RIGHT_PLACES=4
:CANNOT_BE_SIGNED
:MODAL=YES
:SELECT=2
:WORD_ADDRESS_BEF=|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=

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=

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_BEF=|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

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_BEF=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

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

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_BEF=|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

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_BEF=|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_BEF=|F
:LEFT_PLACES=3
:RIGHT_PLACES=4
:MUST_BE_SIGNED
:MODAL=YES
:ATTREND
```

: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

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_BEF=|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

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=

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_BEF=|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=

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=

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 10th 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=

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=

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

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

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

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	LATHE_OPER_SETUP	
REAMING_DWELL	MILL_ROUGH_CUT		
BORE_DWELL	MILL_CURVE_CUT		
BACK_BORING	MILL_TOPO_CUT		
FINE_BORING	MILL_FREEFORM_CUT		
	MILL_PENCIL_CUT		
	MILL_OPER_SETUP		

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

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

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

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
```

FLUSH_SYNC_CODES

Purpose

Can be used with any post that uses sync codes.

Syntax

```
FLUSH_SYNC_CODES(REAR)
FLUSH_SYNC_CODES(FRONT)
```

Comments

You would use this command in CALC_END_OF_TAPE. To be used in CAMWorks 2014 or newer version.

MILL_OPER_SETUP

Usage

Creates post questions per setup. To be used in CAMWorks 2015 SP0 or newer versions.

```
:OPERID=MILL_OPER_SETUP
:OPERLIST=abs inc
:OPEREND
```

LATHE_OPER_SETUP

Usage

Creates post questions per setup. To be used in CAMWorks 2015 SP0 or newer versions.

```
:OPERID=LATHE_OPER_SETUP
:OPERLIST=abs inc
:OPEREND
```

Chapter 6: System Tool Commands

: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
XHIRANGE=50.000000
YHIRANGE=30.000000
```

: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

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
XHIRANGE=50.000000
YHIRANGE=30.000000
```

: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
XHIRANGE=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
XHIRANGE=50.000000
YHIRANGE=30.000000
```

: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

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
XHIRANGE=50.000000
YHIRANGE=30.000000
```

Chapter 7: System Header Commands

: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

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

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

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

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

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

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

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
Fields2 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

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

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

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

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

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

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

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 OD 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

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

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

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

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

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

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

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

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

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

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

Purpose

Mill 4th axis lathe definition.

Syntax

```
:LATHE=TRUE or FALSE
```

Comments

This command should be set to FALSE if doing a mill 4th 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

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

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

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.

:SORT_BY_TURRET

Purpose

For use with 2 or more turrets in turn or mill turn post.
If set to FALSE operations are posted out as operation tree order.
If set to TRUE operations are posted out with all rear turret operations first then the front turret operations.

Syntax

```
:SORT_BY_TURRET=TRUE or FALSE
```

Comments

This command should be set to FALSE if not using 2 or more turrets.
This command can be set to TRUE if using 2 or more turrets and output needs to be in separate files.

:SINGLE_Y_AXIS_DIRECTION

Purpose

To be used with Mill/Turn posts.

Syntax

```
:SINGLE_Y_AXIS_DIRECTION=TRUE or FALSE
```

Comments

If this is set to “TRUE then this will allow the new system changes to the Main and sub spindle Y axis direction to be transferred to the post as corrected values If this is set to “FALSE” then it assumes the post will make all the corrected changes manually to the posted output. If this header is not in the post then it assumes “FALSE” and the post will work as originally built.

GENERIC_POST

Purpose

Can be used with any post

Syntax

```
:GENERIC_POST=TRUE or FALSE
```

Comments

Set to “TRUE” if post will be used as a generic post for Eureka standard APT output. To be used in CAMWorks 2014 or newer version.

Chapter 8: System Variables

System Variables

Variable	*Type*	*Usage*
NO	CHARACTER	Its value equals 0.
YES	CHARACTER	Its value equals 1.
ABS_X_END **ABS_Y_END** **ABS_Z_END** **ABS_C_END** **ABS_U_END** **ABS_V_END** **ABS_W_END**	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.
N_ABS_X_END **N_ABS_Y_END** **N_ABS_Z_END** **N_ABS_C_END** **N_ABS_U_END** **N_ABS_V_END** **N_ABS_W_END**		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.
P_ABS_X_END **P_ABS_Y_END** **P_ABS_Z_END** **P_ABS_C_END** **P_ABS_U_END** **P_ABS_V_END** **P_ABS_W_END**		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.
O_ABS_X_END **O_ABS_Y_END**		Punch: Original absolute X,Y end point of a move.
ABS_X_START **ABS_Y_START** **ABS_Z_START** **ABS_C_START** **ABS_U_START** **ABS_V_START** **ABS_W_START**	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

Variable	*Type*	*Usage*
N_ABS_X_START **N_ABS_Y_START** **N_ABS_Z_START** **N_ABS_C_START** **N_ABS_U_START** **N_ABS_V_START** **N_ABS_W_START**	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.
P_ABS_X_START **P_ABS_Y_START** **P_ABS_Z_START** **P_ABS_C_START** **P_ABS_U_START** **P_ABS_V_START** **P_ABS_W_START**		Absolute X,Y,Z,C start point of previous move. EDM: Absolute U,V secondary surface start point of previous move. Lathe: Absolute U,W front turret start point of previous move.
O_ABS_X_START **O_ABS_Y_START**		Punch: Original absolute X,Y start point of a move.
INC_X_END **INC_Y_END** **INC_Z_END** **INC_C_END** **INC_U_END** **INC_V_END** **INC_W_END**	METRIC DECIMAL	X,Y,Z,C Signed incremental distance of move. EDM: U,V secondary surface signed incremental distance of move. Lathe: U,W front turret signed incremental distance of move.
N_INC_X_END **N_INC_Y_END** **N_INC_Z_END** **N_INC_C_END** **N_INC_U_END** **N_INC_V_END** **N_INC_W_END**		X,Y,Z,C Signed incremental distance of next move. EDM: U,V secondary surface signed incremental distance of next move. Lathe: U,W front turret signed incremental distance of next move.
P_INC_X_END **P_INC_Y_END** **P_INC_Z_END** **P_INC_C_END** **P_INC_U_END** **P_INC_V_END** **P_INC_W_END**	METRIC DECIMAL	X,Y,Z,C Signed incremental distance of previous move. EDM: U,V secondary surface signed incremental distance of previous move. Lathe: U,W front turret signed incremental distance of previous move.

Variable	*Type*	*Usage*
ABS_I_CENTER **ABS_J_CENTER** **ABS_K_CENTER** **ABS_L_CENTER**	METRIC DECIMAL	I,J absolute center point of arc. EDM: K,L absolute center point of secondary surface arc. Lathe: K,L absolute center point of front turret arc.
N_ABS_I_CENTER **N_ABS_J_CENTER** **N_ABS_K_CENTER** **N_ABS_L_CENTER**		I,J absolute center point of next arc. EDM: K,L absolute center point of next secondary surface arc. Lathe: K,L absolute center point of next front turret arc.
P_ABS_I_CENTER **P_ABS_J_CENTER** **P_ABS_K_CENTER** **P_ABS_L_CENTER**		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.
INC_I_CENTER **INC_J_CENTER** **INC_K_CENTER** **INC_L_CENTER**	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.
N_INC_I_CENTER **N_INC_J_CENTER** **N_INC_K_CENTER** **N_INC_L_CENTER**	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.
P_INC_I_CENTER **P_INC_J_CENTER** **P_INC_K_CENTER** **P_INC_L_CENTER**	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.
REV_INC_I_CENTER **REV_INC_J_CENTER** **REV_INC_K_CENTER** **REV_INC_L_CENTER**	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.
N_REV_INC_I_CENTER **N_REV_INC_J_CENTER** **N_REV_INC_K_CENTER** **N_REV_INC_L_CENTER**	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.

Variable	*Type*	*Usage*
P_REV_INC_I_CENTER **P_REV_INC_J_CENTER** **P_REV_INC_K_CENTER** **P_REV_INC_L_CENTER**	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.
DISPLACED_X **DISPLACED_Y** **DISPLACED_Z**	METRIC DECIMAL	5 axis modified X,Y,Z end point of move.
N_DISPLACED_X **N_DISPLACED_Y** **N_DISPLACED_Z**		5 axis modified X,Y,Z end point of next move.
P_DISPLACED_X **P_DISPLACED_Y** **P_DISPLACED_Z**		5 axis modified X,Y,Z end point of previous move.
ROT_TILT_A	DECIMAL	5 axis "A" absolute rotation move.
N_ROT_TILT_A		5 axis "A" absolute rotation of next move.
P_ROT_TILT_A	DECIMAL	5 axis "A" absolute rotation of previous move.
INC_ROT_TILT_A		5 axis "A" signed incremental rotation move.
N_INC_ROT_TILT_A		5 axis "A" signed incremental rotation of next move.
N_INC_ROT_TILT_A		5 axis "A" signed incremental rotation of previous move.
ROT_TILT_B	DECIMAL	5 axis "B" absolute tilt move.
N_ROT_TILT_B		5 axis "B" absolute tilt of next move.
P_ROT_TILT_B		5 axis "B" absolute tilt of previous move.
INC_ROT_TILT_B		5 axis "B" signed incremental tilt move.
N_INC_ROT_TILT_B		5 axis "B" signed incremental tilt of next move.
N_INC_ROT_TILT_B		5 axis "B" signed incremental tilt of previous move.

Variable	*Type*	*Usage*
VECTOR_I	DECIMAL	5 axis "I" axis vector move.
N_VECTOR_I		5 axis next "I" axis vector move.
P_VECTOR_I		5 axis previous "I" axis vector move.
VECTOR_J	DECIMAL	5 axis "J" axis vector move.
N_VECTOR_J		5 axis next "J" axis vector move.
P_VECTOR_J		5 axis previous "J" axis vector move.
VECTOR_K	DECIMAL	5 axis "K" axis vector move.
N_VECTOR_K	DECIMAL	5 axis next "K" axis vector move.
P_VECTOR_K		5 axis previous "K" axis vector move.
ARC_RADIUS	METRIC DECIMAL	Arc radius.
N_ARC_RADIUS		Next arc radius.
P_ARC_RADIUS		Previous arc radius.
O_ARC_RADIUS		Punch: Original radius.
S_ARC_RADIUS		EDM: Secondary surface arc radius. Lathe: Front turret arc radius.
S_N_ARC_RADIUS		EDM: Next secondary surface arc radius. Lathe: Next front turret arc radius.
S_P_ARC_RADIUS		EDM: Previous secondary surface arc radius. Lathe: Previous front turret arc radius.
DISTANCE	METRIC DECIMAL	Length of a move.
N_DISTANCE		Length of next move.
P_DISTANCE		Length of previous move.
S_DISTANCE		EDM: Length of secondary surface move. Lathe: Length of front turret move.
S_N_DISTANCE		EDM: Length of next secondary surface move. Lathe: Length of next front turret move.

Variable	*Type*	*Usage*
S_P_DISTANCE	METRIC DECIMAL	EDM: Length of previous secondary surface move. Lathe: Length of previous front turret move.
O_DISTANCE		Punch: Original move length.
ARC_START_ANGLE	DECIMAL	Absolute arc start angle.
N_ARC_START_ANGLE		Next absolute arc start angle.
P_ARC_START_ANGLE		Previous absolute arc start angle.
S_ARC_START_ANGLE		EDM: Absolute arc start angle of secondary surface. Lathe: Absolute arc start angle of front turret.
S_N_ARC_START_ANGLE		EDM: Next absolute arc start angle of secondary surface. Lathe: Next absolute arc start angle of front turret.
S_P_ARC_START_ANGLE		EDM: Previous absolute arc start angle of secondary surface. Lathe: Previous absolute arc start angle of front turret.
O_ARC_START_ANGLE		Punch: Original absolute arc start angle.
ARC_END_ANGLE	DECIMAL	Absolute arc end angle.
N_ARC_END_ANGLE		Next absolute arc end angle.
P_ARC_END_ANGLE		Previous absolute arc end angle.
S_ARC_END_ANGLE		EDM: Absolute arc end angle of secondary surface. Lathe: Absolute arc end angle of front turret.
S_N_ARC_END_ANGLE	DECIMAL	EDM: Next absolute arc end angle of secondary surface. Lathe: Next absolute arc end angle of front turret.

Variable	*Type*	*Usage*
S_P_ARC_END_ANGLE		EDM: Previous absolute arc end angle of secondary surface. Lathe: Previous absolute arc end angle of front turret.
O_ARC_END_ANGLE		Punch: Original absolute arc end angle.
ARC_INC_ANGLE	DECIMAL	Absolute incremental arc angle.
N_ARC_INC_ANGLE		Next absolute incremental arc angle.
P_ARC_INC_ANGLE		Previous absolute incremental arc angle.
S_ARC_INC_ANGLE		EDM: Absolute incremental arc angle of secondary surface. Lathe: Absolute incremental arc angle of front turret.
S_N_ARC_INC_ANGLE		EDM: Next absolute incremental arc angle of secondary surface. Lathe: Next absolute incremental arc angle of front turret.
S_P_ARC_INC_ANGLE		EDM: Previous absolute incremental arc angle of secondary surface. Lathe: Previous absolute incremental arc angle of front turret.
O_ARC_INC_ANGLE		Punch: Original absolute incremental arc angle.
ANGLE	DECIMAL	Angle of a Line, Rapid, Grid, Line Pattern or Window move.
N_ANGLE	DECIMAL	Angle of next Line, Rapid, Grid, Line Pattern or Window move.
P_ANGLE		Angle of previous Line, Rapid, Grid, Line Pattern or Window move.
O_ANGLE		Punch: Angle of original Line, Rapid, Grid, Line Pattern or Window move.
TOOL_INDEX_ANGLE	DECIMAL	Punch: Autoindex tool angle.
N_TOOL_INDEX_ANGLE		Punch: Next autoindex tool angle.

Variable	*Type*	*Usage*
P_TOOL_INDEX_ANGLE		Punch: Previous autoindex tool angle.
INC_TOOL_INDEX_ANGLE		Punch: Signed incremental autoindex tool angle.
N_INC_TOOL_INDEX_ANGLE		Punch: Next signed incremental autoindex tool angle.
P_INC_TOOL_INDEX_ANGLE		Punch: Previous signed incremental autoindex tool angle.
PITCH	METRIC DECIMAL	Punch: Nibbling pitch of move.
NPITCH		Punch: Nibbling pitch of next move.
PPITCH		Punch: Nibbling pitch of previous move.
OPITCH		Punch: Original nibbling pitch of move.
N_OPITCH		Punch: Original nibbling pitch of next move.
P_OPITCH	METRIC DECIMAL	Punch: Original nibbling pitch of previous move.
MICRO_START	METRIC DECIMAL	Punch: Signed incremental microjoint start distance.
N_MICRO_START		Punch: Next signed incremental microjoint start distance.
P_MICRO_START		Punch: Previous signed incremental microjoint start distance.
MICRO_END	METRIC DECIMAL	Punch: Signed incremental microjoint end distance.
N_MICRO_END		Punch: Next signed incremental microjoint end distance.
P_MICRO_END		Punch: Previous signed incremental microjoint end distance.
P_MICROJOINT	METRIC DECIMAL	Punch: Previous signed incremental microjoint distance.
N_MICROJOINT		Punch: Next signed incremental microjoint distance.

Variable	*Type*	*Usage*
MACRO_TIME	DECIMAL	Internal time calculation for macros.
MACRO_A **MACRO_B** **MACRO_C** **MACRO_D** **MACRO_E** **MACRO_F** **MACRO_G** **MACRO_H** **MACRO_I** **MACRO_J**	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.
TOOL_ZGL	METRIC DECIMAL	Mill and Lathe: 2nd field of a fixed external file.
TOOL_XGL	METRIC DECIMAL	Mill and Lathe: 3rd field of a fixed external file.
TOOL_QTN	METRIC DECIMAL	Mill and Lathe 4th field of a fixed external file.
TOOL_QT_COMMENT	CHARACTER	Mill and Lathe: 5th field of a fixed external file.
PI	DECIMAL	Mathematical constant = 3.14159265.
BYTE_COUNT	INTEGER	Byte count of the text output file *.txt.
LINE_COUNT	INTEGER	Line count of the text output file *.txt.
METRIC_FLAG	INTEGER	Metric in flag: 0=inch 1=metric.
METRIC_OUT	INTEGER	Metric out flag: 1=inch 2=metric.

Variable	*Type*	*Usage*
SEQ_INCREMENT	INTEGER	Sets the sequence increment.
MOVE_TYPE	INTEGER	Movement type - Line, Arc, Rapid, etc.
N_MOVE_TYPE		Next movement type - Line, Arc, Rapid, etc.
P_MOVE_TYPE		Previous movement type - Line, Arc, Rapid, etc.
S_MOVE_TYPE		EDM and 4 axis Lathe: Secondary or U,V plane.
S_N_MOVE_TYPE	INTEGER	EDM and 4 axis Lathe: Next secondary or U,V plane.
S_P_MOVE_TYPE		EDM and 4 axis Lathe: Previous secondary or U,V plane.
SEQ	INTEGER	Sequence number.
OPR_TYPE	INTEGER	Mill or Lathe operation type - MILL_DRILLING, PROFILING, etc.
ATROVERRIDE	INTEGER	: <X:x_preset> - anything after the colon makes `ATTROVERRIDE=YES`.
ATTRIVALUE	INTEGER	<S:1000> - since "S" is an INTEGER, the value is `ATTRIVALUE`.
ATTRDVALUE	METRIC DECIMAL	<X:x_preset> since "X" is a DECIMAL, the value is `ATTRDVALUE`.
ATTRCVALUE	METRIC DECIMAL	<P:{MILLING}> since "P" is a CHARACTER, the value is `ATTRCVALUE`.
DIST_BET_HOLES_X **DIST_BET_HOLES_Y**	METRIC DECIMAL	Signed incremental X and Y distance between holes in a grid.
N_DIST_BET_HOLES_X **N_DIST_BET_HOLES_Y**		Signed incremental X and Y distance between holes in next grid.
P_DIST_BET_HOLES_X **P_DIST_BET_HOLES_Y**		Signed incremental X and Y distance between holes in previous grid.
DIST_BET_PARTS_X **DIST_BET_PARTS_Y**	METRIC DECIMAL	Signed incremental X and Y distance between parts in a macro.

Variable	*Type*	*Usage*
N_DIST_BET_PARTS_X **N_DIST_BET_PARTS_Y**	METRIC DECIMAL	Signed incremental X and Y distance between parts in next macro.
P_DIST_BET_PARTS_X **P_DIST_BET_PARTS_Y**		Signed incremental X and Y distance between parts in previous macro.
DIST_BETWEEN_CHUCKS	DECIMAL	Gives the distance between chucks in Turn and Mill-Turn posts that use two spindles. To be used in CAMWorks 2014 SP0 or newer versions.
HEIGHT	METRIC DECIMAL	Tool compensated window Y height.
N_HEIGHT		Next tool compensated in window Y height.
P_HEIGHT		Previous tool compensated in window Y height.
WIDTH	METRIC DECIMAL	Tool compensated window X width.
N_WIDTH		Next tool compensated in window X width.
P_WIDTH		Previous tool compensated in window X width.
WINDOW_ORIGIN_X **WINDOW_ORIGIN_Y**	METRIC DECIMAL	Absolute window origin.
N_WINDOW_ORIGIN_X **N_WINDOW_ORIGIN_Y**		Next absolute window origin.
P_WINDOW_ORIGIN_X **P_WINDOW_ORIGIN_Y**		Previous absolute window origin.
WINDOW_HEIGHT	METRIC DECIMAL	CAD entity window Y height.
N_WINDOW_HEIGHT		Next CAD entity window Y height.
P_WINDOW_HEIGHT		Previous CAD entity window Y height.
WINDOW_WIDTH	METRIC DECIMAL	CAD entity window Y height.
N_WINDOW_WIDTH		Next CAD entity window X width.
P_WINDOW_WIDTH		Previous CAD entity window X width.

Variable	*Type*	*Usage*
TOOL_LENGTH **TOOL_WIDTH** **TOOL_DIAMETER**	METRIC DECIMAL	Tool dimensions.
N_TOOL_LENGTH **N_TOOL_WIDTH** **N_TOOL_DIAMETER**		Next entity tool dimensions.
NC_TOOL_LENGTH **NC_TOOL_WIDTH** **NC_TOOL_DIAMETER**		Next tool change tool dimensions.
SCALLOP	METRIC DECIMAL	Scallop height when nibbling.
N_SCALLOP		Next scallop height when nibbling.
P_SCALLOP		Previous scallop height when nibbling.
TOOL_CORNER_RADIUS	METRIC DECIMAL	Corner radius value of a corner radius punch tool.
ISCOLLOP	METRIC DECIMAL	Inside scollop height of an arc or circle when nibbling.
OSCOLLOP	METRIC DECIMAL	Outside scollop height of an arc or circle when nibbling.
TOOL_LOAD_ANGLE	DECIMAL	Load angle of the tool in a punch turret.
TOOL	INTEGER	Tool number.
N_TOOL	INTEGER	Next entity tool number.
NC_TOOL	INTEGER	Next tool change tool number.

Variable	*Type*	*Usage*
TOOL_TYPE	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)
N_TOOL_TYPE		Next entity tool type.
NC_TOOL_TYPE		Next tool change tool type.
NUM_HITS_X **NUM_HITS_Y**	INTEGER	Number of hits in X & Y on a grid or window.
N_NUM_HITS_X **N_NUM_HITS_Y**		Number of hits in X & Y on next grid or window.
P_NUM_HITS_X **P_NUM_HITS_Y**		Number of hits in X & Y on previous grid or window.
NUM_HITS	INTEGER	Total number of hits in a grid or window.

Variable	*Type*	*Usage*
N_NUM_HITS		Total number of hits on next grid or window.
P_NUM_HITS		Total number of hits on previous grid or window.
NUM_PARTS_X **NUM_PARTS_Y**	INTEGER	Number of parts in X & Y in a macro.
N_NUM_PARTS_X **N_NUM_PARTS_Y**		Number of parts in X & Y in next macro.
P_NUM_PARTS_X **P_NUM_PARTS_Y**		Number of parts in X & Y in previous macro.
HORIZ_OR_VERT	INTEGER	Direction of punching in a grid, window or multiple macro call.
N_HORIZ_OR_VERT	INTEGER	Direction of punching in next grid, window or multiple macro call.
P_HORIZ_OR_VERT		Direction of punching in previous grid, window or multiple macro call.
O_DIRECTION	INTEGER	Offset direction of tool: O_DIRECTION= OFFSET_LEFT or OFFSET_RIGHT CENTER_LEFT or CENTER_RIGHT
SYSTEM_COMP	INTEGER	Offset direction of tool: SYSTEM_COMP= OFFSET_LEFT or OFFSET_RIGHT CENTER_LEFT or CENTER_RIGHT
FIXED	INTEGER	Punch: Tool index angle. FIXED=YES or NO.
FRAME	INTEGER	Punch: Grid or window. FRAME=YES or NO
SKIP_1ST_HIT	INTEGER	Punch: Skip first hit. Mill: Skip first hole. - SKIP_1ST_HIT=YES or NO

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

Variable	*Type*	*Usage*
NC_TOOL_MATERIAL	CHARACTER	Next tool change tool material from feed and speed table.
TOOL_SERIAL_NUM	CHARACTER	Tool serial number from feed and speed table.
N_TOOL_SERIAL_NUM		Next entity tool serial number from feed and speed table.
NC_TOOL_SERIAL_NUM		Next tool change tool serial number from feed and speed table.
DIE_SERIAL_NUM	CHARACTER	Die serial number from feed and speed table.
N_DIE_SERIAL_NUM		Next entity die serial number from feed and speed table.
NC_DIE_SERIAL_NUM		Next tool change die serial number from feed and speed table.
TOOL_DIE_CLEAR	METRIC DECIMAL	Tool die clearance amount from feed and speed table.
N_TOOL_DIE_CLEAR		Next entity tool die clearance amount from feed and speed table.
NC_TOOL_DIE_CLEAR		Next tool change tool die clearance amount from feed and speed table.

TOOL_SUB_TYPE	INTEGER	Tool sub type:
		`TOOL_SUB_TYPE=`
		`FLAT_BOTTOM`
		`HEELED`
		`FORM`
		`SHEAR_PROOF`
		`LOUVER`
		`MARKING`

N_TOOL_SUB_TYPE	INTEGER	Next entity tool sub type:
		`N_TOOL_SUB_TYPE=`
		`FLAT_BOTTOM`
		`HEELED`
		`FORM`
		`SHEAR_PROOF`
		`LOUVER`
		`MARKING`

Variable	*Type*	*Usage*
NC_TOOL_SUB_TYPE		Next tool change tool sub type: NC_TOOL_SUB_TYPE= FLAT_BOTTOM HEELED FORM SHEAR_PROOF LOUVER MARKING
TOOL_SPEC_TYPE	INTEGER	Tool special type: TOOL_SPEC_TYPE= AIR_BLOW PRESS_RAISE PRESSURE_IGNORE
N_TOOL_SPEC_TYPE		Next entity tool special type: N_TOOL_SPEC_TYPE= AIR_BLOW PRESS_RAISE PRESSURE_IGNORE
NC_TOOL_SPEC_TYPE		Next tool change tool special type: NC_TOOL_SPEC_TYPE= AIR_BLOW PRESS_RAISE PRESSURE_IGNORE
TOOL_NUM_TEETH	INTEGER	Tool: Number of flutes from feed and speed table.
N_TOOL_NUM_TEETH		Next tool: Number of flutes from feed and speed table.
NC_TOOL_NUM_TEETH		Previous tool: Number of flutes from feed and speed table.
FILLET_RADIUS	METRIC DECIMAL	EDM: Fillet radius for XY surface.
N_FILLET_RADIUS		EDM: Next fillet radius for XY surface.
P_FILLET_RADIUS		EDM: Previous fillet radius for XY surface.
S_FILLET_RADIUS		EDM: Fillet radius for UV surface.
S_N_FILLET_RADIUS		EDM: Next fillet radius for UV surface.
S_P_FILLET_RADIUS		EDM: Previous fillet radius for UV surface.
TAPER_ANGLE_START	DECIMAL	EDM: Absolute taper angle start and end.

Variable	*Type*	*Usage*
N_TAPER_ANGLE_START		EDM: Next absolute taper angle start and end.
P_TAPER_ANGLE_START		EDM: Previous absolute taper angle start and end.
TAPER_ANGLE_END	DECIMAL	EDM: Absolute taper angle end.
N_TAPER_ANGLE_END		EDM: Next absolute taper angle end.
P_TAPER_ANGLE_END		EDM: Previous absolute taper angle end.
PROGRAM_SURF	DECIMAL	EDM: X,Y surface or plane of a part.
OTHER_SURF	DECIMAL	EDM: U,V surface or plane of a part.
XY_GUIDE_OFFSET	DECIMAL	EDM: X,Y surface or plane guide offset of a part.
UV_GUIDE_OFFSET	DECIMAL	EDM: U,V surface or plane guide offset of a part.

CORNER_TYPE	INTEGER	EDM: Taper corner type:
		`CORNER_TYPE=`
		`SHARP_CORNER`
		`EQUAL_CORNER`
		`INDEPENDANT_CORNER`
		`CONIC_CORNER`
		`CHAMFER_CORNER`

N_CORNER_TYPE		EDM: Next taper corner type:
		`N_CORNER_TYPE=`
		`SHARP_CORNER`
		`EQUAL_CORNER`
		`INDEPENDANT_CORNER`
		`CONIC_CORNER`
		`CHAMFER_CORNER`

P_CORNER_TYPE		EDM: Previous taper corner type:
		`P_CORNER_TYPE=`
		`SHARP_CORNER`
		`EQUAL_CORNER`
		`INDEPENDANT_CORNER`
		`CONIC_CORNER`
		`CHAMFER_CORNER`

LEADIN	INTEGER	Profile leadin flag: `LEADIN=YES` or `NO`

Variable	*Type*	*Usage*
N_LEADIN		Profile next leadin flag: `LEADIN=YES` or `NO`.
P_LEADIN		Profile previous leadin flag: `LEADIN=YES` or `NO`.
LEADOUT	INTEGER	Profile leadout flag: `LEADOUT=YES` or `NO`.
N_LEADOUT		Profile next leadout flag: `LEADOUT=YES` or `NO`.
P_LEADOUT		Profile previous leadout flag: `LEADOUT=YES` or `NO`.

WIRE_INCLINATION	INTEGER	EDM: Wire taper direction:
		`WIRE_INCLINATION=`
		`LEFT`
		`RIGHT`

EDM_MODE	INTEGER	EDM mode.
		`EDM_MODE=`
		`EDM4AXIS`
		`TAPER`

G_RIGHT_PLACES	INTEGER	Global number of places to the right of decimal point.
G_LEFT_PLACES	INTEGER	Global number of places to the left of decimal point.
POWER_REG	INTEGER	EDM: Power register.
OFFSET_REG	INTEGER	EDM: Offset tool register.
PATH_TYPE	INTEGER	EDM: Path type. `PATH_TYPE=` `1` (Profiling) `2` (Tab cutting.
TURRET	INTEGER	4-axis Lathe: Turret. `TURRET=FRONT` or `REAR`.

N_TURRET		4-axis Lathe: Turret next operation.
		`N_TURRET=`
		`FRONT`
		`REAR`

Variable	*Type*	*Usage*
P_TURRET		4-axis Lathe: Turret previous operation. P_TURRET= FRONT REAR
FRONT_TOOL	INTEGER	4-axis Lathe: Tool selection. STATION_NUM=F01
REAR_TOOL	INTEGER	4-axis Lathe: Tool selection. STATION_NUM=R01
STOCK_DIAMETER	METRIC DECIMAL	4-axis Lathe: Stock diameter.
OD	INTEGER	Lathe: Outside diameter flag. OD=1 or 0.
ID	INTEGER	Lathe: Inside diameter flag: ID=1 or 0.
OPR_FEED_FPR	METRIC DECIMAL	Lathe: Feed per revolution.
OPR_FEED_FPM	METRIC DECIMAL	Lathe: Feed per minute.
OPR_SPEED_SFPM	METRIC DECIMAL	Lathe: Constant surface feed per minute.
OPR_SPEED_RPM	INTEGER	Lathe: Spindle speed.
OPR_FEED_TYPE	INTEGER	Lathe: Feed type. OPR_FEED_TYPE= FPR FPM.
OPR_SPEED_TYPE	INTEGER	Lathe: Speed type. OPR_SPEED_TYPE= SFPM RPM
OPR_X_POSITION	METRIC DECIMAL	Lathe: X end operation position.
OPR_Z_POSITION	METRIC DECIMAL	Lathe: Z end operation position.
OPR_X_PART_CLEARANCE	METRIC DECIMAL	Lathe: Unsigned incremental roughing X finish allowance.
OPR_Z_PART_CLEARANCE	METRIC DECIMAL	Lathe: Unsigned incremental roughing Z finish allowance.
OPR_CYCLE_CLEARANCE	METRIC DECIMAL	Lathe: Unsigned incremental cycle clearance.

Variable	*Type*	*Usage*
OPR_THREAD_LENGTH	METRIC DECIMAL	Lathe: Unsigned incremental thread length.
OPR_THREAD_MINOR_DIAM	METRIC DECIMAL	Lathe: Thread minor diameter.
OPR_THREAD_FIRST_CUT	METRIC DECIMAL	Lathe: Thread first pass amount.
OPR_THREAD_LEADIN	METRIC DECIMAL	Lathe: Thread unsigned incremental leadin amount.
OPR_THREAD_CHAMFER	METRIC DECIMAL	Lathe: Thread unsigned incremental chamfer amount.
OPR_THREAD_MIN_CUT	METRIC DECIMAL	Lathe: Thread unsigned minimum cut amount.
OPR_THREAD_PITCH	METRIC DECIMAL	Lathe: Thread pitch, feed or lead amount.
OPR_THREAD_ANGLE	DECIMAL	Lathe: Thread angle.
OPR_INFEED_ANGLE	DECIMAL	Lathe: Thread infeed angle.
OPR_THREAD_NUM_SPRING	INTEGER	Lathe: Thread number of spring passes.

OPR_INFEED_TYPE	INTEGER	Lathe thread infeed type.
		`OPR_INFEED_TYPE=`
		`STRAIGHT_INFEED`
		`ANGLE_INFEED`

OPR_FIXED ANGLE	DECIMAL	Lathe/Mill: OD or FACE fixed angle.
OPR_CUT_AMOUNT	METRIC DECIMAL	Lathe: Roughing cycle cut amount.

OPR_POST_CYCLE_TYPE	INTEGER	Lathe: Cycle type.
		`OPR_POST_CYCLE_TYPE=`
		`SYSTEM`
		`MACHINE`

OPR_CYCLE_TYPE	INTEGER	Lathe: Cycle type.
		`OPR_CYCLE_TYPE=`
		`TURNING`
		`FACING`
		Lathe Canned Drilling cycle
		`OPR_CYCLE_TYPE=`
		`SINGLE_DEPTH`
		`CONSTANT`
		`PERCENTAGE`

Variable	*Type*	*Usage*
OPR_RETRACT_TYPE	INTEGER	Lathe: Retract type. OPR_RETRACT_TYPE= SINGLE_RETRACT MULTIPLE_RETRACT
TOOL_TIP_CENTER	INTEGER	Lathe: Tool compensation. TOOL_TIP_CENTER= 1 (center) 2 (tip)
OPR_DEPTH_TYPE	INTEGER	Lathe: Drilling depth type. OPR_DEPTH_TYPE= 0 (tip) 1 (flat)
OPR_END_RETRACT_TYPE	INTEGER	Lathe: Retract type. OPR_END_RETRACT_TYPE= 2 (x,z preset) 1 (x,z position) 0 (none) 4 (both)
OPR_CMODE	INTEGER	Lathe/Mill.
OPR_CFIXED	INTEGER	Lathe/Mill: OPR_CFIXED=CFIXED or CFREE.
MOVE_COUNT	INTEGER	Lathe: Cycle movement count. Used for canned Roughing cycles.
LATHE_TOOL_NAME	CHARACTER	Lathe: Tool name.
LATHE_HOLDER_NAME	CHARACTER	Lathe: Holder name.
OPR_Z_FACE	METRIC DECIMAL	Mill: Absolute Z face of part.
OPR_Z_RAPID_PLANE	METRIC DECIMAL	Mill: Absolute Z rapid plane.
OPR_Z_CLEARANCE	METRIC DECIMAL	Mill: Absolute Z clearance plane.
OPR_Z_DEPTH	METRIC DECIMAL	Mill: Absolute Z depth.
OPR_CLEARANCE	METRIC DECIMAL	Mill: Incremental Z clearance distance.
OPR_LEADIN	METRIC DECIMAL	Mill: Incremental leadin distance.

Variable	*Type*	*Usage*
OPR_LEADOUT	METRIC DECIMAL	Mill: Incremental leadout distance.
OPR_Z_FIRST_PECK	METRIC DECIMAL	Mill/Lathe: Incremental drill first peck amount.
OPR_Z_SUB_PECK	METRIC DECIMAL	Mill/Lathe: Incremental drill subsequent peck amount.
OPR_Z_MIN_PECK	METRIC DECIMAL	Mill/Lathe: Incremental drill minimum peck amount.
OPT_Z_FINISH	METRIC DECIMAL	Mill: Lacing and pocketing Z finish allowance.
OPR_Z_DIST_ALONG	METRIC DECIMAL	Mill: Z distance along a wall.
OPR_Z_FIRST_CUT	METRIC DECIMAL	Mill: Profiling, lacing or pocketing Z first depth.
OPR_Z_SUB_CUT	METRIC DECIMAL	Mill: Profiling, lacing or pocketing Z subsequent depth.
OPR_Z_RETRACT_AMOUNT	METRIC DECIMAL	Mill: Profiling, lacing or pocketing Z retract amount. Lathe: Drilling cycle multiple retract amount.
OPR_LACE_CUT	METRIC DECIMAL	Mill: Lacing cut amount.
OPR_CORNER_CLEAR	METRIC DECIMAL	Mill: Profiling corner clearance amount.
OPR_CORNER_EXT	METRIC DECIMAL	Mill: Profiling corner extension amount.
OPR_Z_FEED	METRIC DECIMAL	Mill: Z feed.
OPR_X_FEED	METRIC DECIMAL	Mill: XY feed.
OPR_Z_PER_PECK	DECIMAL	Mill: variable drill pecking percentage.
OPR_LACE_ANGLE	DECIMAL	Mill: lacing angle.

Variable	*Type*	*Usage*
ROTATE_ANGLE_X	DECIMAL	Mill: 4th axis rotation about the X axis in degrees.
ROTATE_ANGLE_Y	DECIMAL	Mill: 4th axis rotation about the Y axis in degrees.
ROTATE_ANGLE_Z	DECIMAL	Mill: 4th axis rotation about the Z axis in degrees.
OPR_SPEED	INTEGER	Mill: Spindle speed.
OPR_SPEED_DIR	INTEGER	Mill/Lathe: Spindle direction. OPR_SPEED_DIR=1, CW OPR_SPEED_DIR=2, CCW
OPR_DRILL_CYCLE_TYPE	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
OPR_RETRACT_TYPE	INTEGER	Mill: Retract type. OPR_RETRACT_TYPE= CLEARANCE_PLANE RAPID_PLANE
OPR_CORNER_TYPE	INTEGER	Mill: Profiling corner type. OPR_CORNER_TYPE= ROUND_CORNERS SHARP_CORNERS SQUARE_CORNERS TRIANGLE_CORNERS FANUC_CORNERS
OPR_Z_CYCLE_TYPE	INTEGER	Mill: Z cycle type. OPR_Z_CYCLE_TYPE= 1,z_depth 2,z_wall 3,z_extrude

Variable	*Type*	*Usage*
OPR_Z_CUT_METHOD_TYPE	INTEGER	Mill: Z cut type. OPR_Z_CUT_METHOD_TYPE= 1,z_depth 2,distance along wall
OPR_CLEARANCE_TYPE	INTEGER	Mill: Z clearance type. OPR_Z_CLEARANCE_TYPE= 1,single clearance 2,multiple clearance
MACRO_ROTATE_AXIS	INTEGER	Mill: 4th axis rotation. MACRO_ROTATE_AXIS= X_AXIS (about X axis) Y_AXIS (about Y axis) Z_AXIS (about Z axis)
OPR_COMMENT	CHARACTER	Mill/Lathe: Operation comments.
NEW_SFPM	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).
CCL_STATUS	INTEGER	Macro CCL file status.
ROTATE_TILE	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)

Variable	*Type*	*Usage*
OPR_AXIS_TYPE	INTEGER	Mill: Operation type: `OPR_AXIS_TYPE =` `2222` – Any operation that has a preposition rotary movement (4th or 5th Axis) and "None" is not selected in the Indexer tab of the Setup. Normally, this will not be set to 2222 if on the top plane; however, if you rotate to another plane and then go back to the top plane, then it will still be 2222. If multiaxis operations, then IS_5AXIS will be set to TRUE. `3333` – Any 2 and 3 Axis operations on the top plane or any other plane with "None" selected in the Indexer tab of the Setup. Again, if you are rotating to another plane and then go back to the top plane, the value will be 2222. `4444` – This will be set for a Mill 4 Axis wrapped toolpath. In this case, IS_WRAPPED will also be set to TRUE. `5555` – is not defined and not used. `6666` – is not defined and not used.
ARM_LEN	METRIC DECIMAL	5 axis arm extension length.
HEAD_LEN	METRIC DECIMAL	5 axis head length.
MILL_FACE_INC	METRIC DECIMAL	The incremental distance between holes in a line pattern when drilling.

Variable	*Type*	*Usage*
Z_AXIS **X_AXIS** **Y_AXIS** **I_AXIS** **J_AXIS** **K_AXIS** **ABS_PRESET_X** **ABS_PRESET_Y** **ABS_PRESET_Z** **ABS_PRESET_C** **X_START_POSITION** **Z_START_POSITION** **ATTRLVALUE** **OPR_LOOKAHEAD**	METRIC DECIMAL	Reserved for future use.
Q_TAPER_ANGLE_START **P_Q_TAPER_ANGLE_START** **N_Q_TAPER_ANGLE_START** **Q_TAPER_ANGLE_END** **P_Q_TAPER_ANGLE_END** **N_Q_TAPER_ANGLE_END** **R_TAPER_ANGLE_START** **P_R_TAPER_ANGLE_START** **N_R_TAPER_ANGLE_START** **R_TAPER_ANGLE_END** **P_R_TAPER_ANGLE_END** **N_R_TAPER_ANGLE_END**	DECIMAL	Reserved for future use.
DIRECTION **ESC** **RECNUM** **NEXT_OPR_TOOL** **MONTH** **DAY** **YEAR** **CURRENT_MACRO_QUADRANT** **CURRENT_MACRO_DEFINED** **MACRO_COUNT** **MACRO_ERROR** **MACRO_DEFINED** **SYS_COUNT**	INTEGER	Reserved for future use.

Variable	*Type*	*Usage*
TIME **DISK_DRIVE_1** **DISK DRIVE_2** **TAB_CHAR**	CHARACTER	Reserved for future use.
INC_ANGLE	DECIMAL	Punch: Incremental angle of an arc pattern.
N_INC_ANGLE		Punch: Incremental angle of next arc pattern.
P_INC_ANGLE		Punch: Incremental angle of previous arc pattern.
DIST_BET_PARTS_X **DIST_BET_PARTS_Y**	DECIMAL	The signed incremental distance between parts in a multiple macro call.
N_DIST_BET_PARTS_X **N_DIST_BET_PARTS_Y**		The signed incremental distance between parts in the next multiple macro call.
P_DIST_BET_PARTS_X **P_DIST_BET_PARTS_Y**		The signed incremental distance between parts in the previous multiple macro call.
CONVENTIONAL_AXIS_LABELS	INTEGER	Set this variable to either TRUE or FALSE. If variable is set to “TRUE” then this lets the posting system know that the rotary 4th and 5th axis are set to standard axis labels as follows. Rotating about X=“A”, Rotating about Y=“B” and Rotating about Z=“C”. If set to FALSE then the post is handling the axis labels for a special case. Currently this will be set in system calc section called “CALC_PRE_POST_INITIALIZE”. To be used in CAMWorks 2014 or newer version.
NUM_TURRETS_REAR	INTEGER	Lets post know how many turrets are defined for the rear turret. To be used in CAMWorks 2014 or newer version.
NUM_TURRETS_FRONT	INTEGER	Lets post know how many turrets are defined for the front turret. To be used in CAMWorks 2014 or newer version.

Variable	*Type*	*Usage*
REAR_TURRET_TYPE	INTEGER	Lets post know the rear turret type for a single rear turret post. If passed as "TOOL_CHANGER" than this would normally be a B axis mill/turn or turn machine that has a tool changer. If not "TOOL_CHANGER" than it would be a standard turret type turret with not tool changer. To be used in CAMWorks 2014 or newer version.
REAR_TURRET_1_TYPE	INTEGER	Lets post know the 1st rear turret type. If passed as "TOOL_CHANGER" than this would normally be a B axis mill/turn or turn machine that has a tool changer. If not "TOOL_CHANGER" than it would be a standard turret type turret with not tool changer. To be used in CAMWorks 2014 or newer version.
REAR_TURRET_2_TYPE	INTEGER	Lets post know the 2nd rear turret type. If passed as "TOOL_CHANGER" than this would normally be a B axis mill/turn or turn machine that has a tool changer. If not "TOOL_CHANGER" than it would be a standard turret type turret with not tool changer. To be used in CAMWorks 2014 or newer version.
REAR_TURRET_3_TYPE	INTEGER	Lets post know the 3rd rear turret type. If passed as "TOOL_CHANGER" than this would normally be a B axis mill/turn or turn machine that has a tool changer. If not "TOOL_CHANGER" than it would be a standard turret type turret with not tool changer. To be used in CAMWorks 2014 or newer version.

Variable	*Type*	*Usage*
REAR_TURRET_4_TYPE	INTEGER	Lets post know the 4th rear turret type. If passed as "TOOL_CHANGER" than this would normally be a B axis mill/turn or turn machine that has a tool changer. If not "TOOL_CHANGER" than it would be a standard turret type turret with not tool changer. To be used in CAMWorks 2014 or newer version.
FRONT_TURRET_TYPE	INTEGER	Lets post know the front turret type for a single front turret post. If passed as "TOOL_CHANGER" than this would normally be a B axis mill/turn or turn machine that has a tool changer. If not "TOOL_CHANGER" than it would be a standard turret type turret with not tool changer. To be used in CAMWorks 2014 or newer version.
FRONT_TURRET_1_TYPE	INTEGER	Lets post know the 1st front turret type. If passed as "TOOL_CHANGER" than this would normally be a B axis mill/turn or turn machine that has a tool changer. If not "TOOL_CHANGER" than it would be a standard turret type turret with not tool changer. To be used in CAMWorks 2014 or newer version.
FRONT_TURRET_2_TYPE	INTEGER	Lets post know the 2nd front turret type. If passed as "TOOL_CHANGER" than this would normally be a B axis mill/turn or turn machine that has a tool changer. If not "TOOL_CHANGER" than it would be a standard turret type turret with not tool changer. To be used in CAMWorks 2014 or newer version.

Variable	*Type*	*Usage*
FRONT_TURRET_3_TYPE	INTEGER	Lets post know the 3rd front turret type. If passed as "TOOL_CHANGER" than this would normally be a B axis mill/turn or turn machine that has a tool changer. If not "TOOL_CHANGER" than it would be a standard turret type turret with not tool changer. To be used in CAMWorks 2014 or newer version.
FRONT_TURRET_4_TYPE	INTEGER	Lets post know the 4th front turret type. If passed as "TOOL_CHANGER" than this would normally be a B axis mill/turn or turn machine that has a tool changer. If not "TOOL_CHANGER" than it would be a standard turret type turret with not tool changer. To be used in CAMWorks 2014 or newer version.
CHUCK_WIDTH	DECIMAL	Lets post know the chuck width of main spindle. To be used in CAMWorks 2014 or newer version.
SUB_CHUCK_WIDTH	DECIMAL	Lets post know the chuck width of sub spindle. To be used in CAMWorks 2014 or newer version.
MACH_SIM_CHANGER	INTEGER	Returns what the mill LOADTOOL index for simulation is if single rear turret or if turret type is a TOOL_CHANGER, for mill or single turret mill/turn post. To be used in CAMWorks 2014 or newer version.
MACH_SIM_CHANGER_1A	INTEGER	Returns what the mill LOADTOOL index for simulation 1st rear turret if turret type is a TOOL_CHANGER, for mill or multiple turret mill/turn post. To be used in CAMWorks 2014 or newer version.
MACH_SIM_CHANGER_2A	INTEGER	Returns what the mill LOADTOOL index for simulation 2nd rear turret if turret type is a TOOL_CHANGER, for mill or multiple turret mill/turn post. To be used in CAMWorks 2014 or newer version.

Variable	*Type*	*Usage*
MACH_SIM_CHANGER_1B	INTEGER	Returns what the mill LOADTOOL index for simulation 1st front turret if turret type is a TOOL_CHANGER, for mill or multiple turret mill/turn post. To be used in CAMWorks 2014 or newer version.
MACH_SIM_CHANGER_2B	INTEGER	returns what the mill LOADTOOL index for simulation 2nd front turret if turret type is a TOOL_CHANGER, for mill or multiple turret mill/turn post. To be used in CAMWorks 2014 or newer version.
MACH_SIM_CRIB_OFF	INTEGER	Returns a tool crib offset tool number from simulator for single turret. To be used in CAMWorks 2014 or newer version.
MACH_SIM_CRIB_OFF_1A	INTEGER	Returns a tool crib offset tool number from simulator for 1st rear turret. To be used in CAMWorks 2014 or newer version.
MACH_SIM_CRIB_OFF_2A	INTEGER	Returns a tool crib offset tool number from simulator for 2nd rear turret. To be used in CAMWorks 2014 or newer version.
MACH_SIM_CRIB_OFF_1B	INTEGER	Returns a tool crib offset tool number from simulator for 1st front turret. To be used in CAMWorks 2014 or newer version.
MACH_SIM_CRIB_OFF_2B	INTEGER	Returns a tool crib offset tool number from simulator for 2nd front turret. To be used in CAMWorks 2014 or newer version.
MACH_SIM_TOOL_SPINDLE	INTEGER	Returns what the mill tool motor number is for simulation in mill or single turret mill/turn posts. To be used in CAMWorks 2014 or newer version.
MACH_SIM_TOOL_SPINDLE_1A	INTEGER	Returns what the mill tool motor number is for simulation in mill or multiple turret mill/turn posts in 1st rear turret. To be used in CAMWorks 2014 or newer version.

Variable	*Type*	*Usage*
MACH_SIM_TOOL_SPINDLE_2A	INTEGER	Returns what the mill tool motor number is for simulation in mill or multiple turret mill/turn posts in 2nd rear turret. To be used in CAMWorks 2014 or newer version.
MACH_SIM_TOOL_SPINDLE_1B	INTEGER	Returns what the mill tool motor number is for simulation in mill or multiple turret mill/turn posts in 1st front turret. To be used in CAMWorks 2014 or newer version.
MACH_SIM_TOOL_SPINDLE_2B	INTEGER	Returns what the mill tool motor number is for simulation in mill or multiple turret mill/turn posts in 2nd front turret. To be used in CAMWorks 2014 or newer version.
MACH_SIM_MAIN_SPINDLE	INTEGER	Returns the turn main spindle motor number for simulation. To be used in CAMWorks 2014 or newer version.
MACH_SIM_SUB_SPINDLE	INTEGER	Returns the turn sub spindle motor number for simulation. To be used in CAMWorks 2014 or newer version.
OUTPUT_WCS_OFFSETS	INTEGER	Returns whether post should output the SETOFFSET command with offsets or not for simulation. To be used in CAMWorks 2014 or newer version.

Chapter 9: System Constants

System Symbolic Constants

Name	*Value*
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

Name	*Value*
LATHE	LATHE
PUNCH	PUNCH
PLASMA	PLASMA
LASER	LASER
EDM	EDM
WASINO	WASINO
JIG	JIG
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

Name	*Value*
MULTIPLE_RETRACT	0
TURNING	0
FACING	1
CONSTANT_DEPTH	0
CONSTANT_CUT	1
STRAIGHT_INFEED	0
ANGLE_INFEED	1
ROUND_CORNERS	0
SHARP_CORNERS	1
SQUARE_CORNERS	2
TRIANGLE_CORNERS	3
FANUC_CORNERS	4
DRILLING	1
SPOT_DRILLING	2
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

Name	*Value*
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
OPSETUP_AXIS_ANGLE	1
OPSETUP_AXIS_WORKPIECE_AUTOMATIC	2
OPSETUP_AXIS_EDGE	3
OPSETUP_AXIS_SKETCH	4
OPSETUP_AXIS_ASM_FCS	4
SIDE_EDGE	0
END_EDGE	1
HAVE_SYNC_CODE_ERROR	1
SYNC_CODE_UNKNOWN	0
SYNC_CODE_BEFORE_START	1
SYNC_CODE_BEFORE_FIRST_MOVE	2
SYNC_CODE_AFTER_END	3
LOCK	2
SYNC_SPEED	0
SYNC_PHASE	1
CAM_REV2011	110
CAM_REV2011_SP1	111
CAM_REV2011_SP2	112
CAM_REV2012	120
CAM_REV2012_SP1	121
CAM_REV2012_SP2	122
CAM_REV2012_PLUS	125
CAM_REV2013	130
CAM_REV2013_SP1	131
CAM_REV2013_SP2	132

Name	*Value*
CAM_REV2013_SP3	133
CAM_REV2014	140

Mill Operation Symbolic Constants

Name	*Value*
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

Name	*Value*
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

Name	*Value*
LATHE_DRILLING	1011
LATHE_PROFILING	1012
LATHE_ROUGHING	1013
LATHE_GROOVING	1014
LATHE_THREADING	1015
LATHE_MISC	1016
LATHE_SPECIAL	1017

Additional System Constants

Name	*Value*
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
CAM_REV2011_SP1	111
OPSETUP_AXIS_ANGLE	1
OPSETUP_AXIS_WORKPIECE_AUTOMATIC	2
OPSETUP_AXIS_EDGE	3
OPSETUP_AXIS_SKETCH	4
OPSETUP_AXIS_ASM_FCS	4
SIDE_EDGE	0
END_EDGE	1
HAVE_SYNC_CODE_ERROR	1
SYNC_CODE_UNKNOWN	0
SYNC_CODE_BEFORE_START	1
SYNC_CODE_BEFORE_FIRST_MOVE	2
SYNC_CODE_AFTER_END	3
LOCK	2
SYNC_SPEED	0
SYNC_PHASE	1
MILL_ROUGH_TYPE_SPIRALIN	0
MILL_ROUGH_TYPE_SPIRALOUT	1
MILL_ROUGH_TYPE_ZIG	2
MILL_ROUGH_TYPE_ZIGZAG	3
MILL_ROUGH_TYPE_TRUESPIRALIN	4
MILL_ROUGH_TYPE_TRUESPIRALOUT	5
MILL_ROUGH_TYPE_PLUNGEROUGH	6
MILL_ROUGH_TYPE_POCKETIN_CORE	7
MILL_ROUGH_TYPE_VOLUMILL	8
MILL_ROUGH_TYPE_ADAPTIVE	19
CAM_REV2011	110

Name	*Value*
CAM_REV2011_SP1	111
CAM_REV2011_SP2	112
CAM_REV2012	120
CAM_REV2012_SP1	121
CAM_REV2012_SP2	122
CAM_REV2012_PLUS	125
CAM_REV2013	130
CAM_REV2013_SP1	131
CAM_REV2013_SP2	132
CAM_REV2013_SP3	133
CAM_REV2014	140
CAM_REV2014_SP0.1	141
CAM_REV2014_SP1	142
UNLOCK	3
ENGAGE_MILL_MODE	4
DISENGAGE_MILL_MODE	5
SYNC_OFF	2
SYNC_MILL_CAXIS	3
SYNC_MILL_OFF	4
CAM_REV2014_SP2	143
CAM_REV2014_SP3	144
CAM_REV2015	150
CAM_REV2015_SP1	151

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`.

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
```

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 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>
```

6. 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
```

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

   Notice that the attrername 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: 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: 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
*
: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>
```

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 no 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_CLAMPS. 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 MMPM 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 MMPM 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

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.

USE_TOOL_COMMENT_AS_NAME

Purpose

This will instruct Machine Simulation to use the Tool Comment as the Tool Name.

Syntax

```
:USE_TOOL_COMMENT_AS_NAME=TRUE or FALSE
```

Comments

If you are not going to use tool comments for tool names, then this command should be set to FALSE.

If you are going to use tool comments for tool names, then this command can be set to TRUE.

Chapter 12: Additional System Variables

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

System Variables

Variable	*Type*	*Usage*
DRILL_RAPID_X	METRIC DECIMAL	Drilled cycle rapid plane in X, Y and Z. 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.
DRILL_RAPID_Y	METRIC DECIMAL	Drilled cycle rapid plane in X, Y and Z. 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.
DRILL_RAPID_Z	METRIC DECIMAL	Drilled cycle rapid plane in X, Y and Z. 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.
DRILL_CLEAR_X	METRIC DECIMAL	Drilled cycle clearance plane in X, Y and Z. 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.
DRILL_CLEAR_Y	METRIC DECIMAL	Drilled cycle clearance plane in X, Y and Z. 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.

Variable	*Type*	*Usage*
DRILL_CLEAR_Z	METRIC DECIMAL	Drilled cycle clearance plane in X, Y and Z. 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.
DRILL_FACE_X	METRIC DECIMAL	Drilled cycle face plane in X, Y and Z. 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.
DRILL_FACE_Y	METRIC DECIMAL	Drilled cycle face plane in X, Y and Z. 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.
DRILL_FACE_Z	METRIC DECIMAL	Drilled cycle face plane in X, Y and Z. 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.
DRILL_DEPTH_X	METRIC DECIMAL	Drilled cycle drill depth in X, Y and Z. 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.

Variable	*Type*	*Usage*
DRILL_DEPTH_Y	METRIC DECIMAL	Drilled cycle drill depth in X, Y and Z. 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.
DRILL_DEPTH_Z	METRIC DECIMAL	Drilled cycle drill depth in X, Y and Z. 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.
DRILL_SAFE_X	METRIC DECIMAL	Drilled cycle safe retract plane in X, Y and Z. 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.
DRILL_SAFE_Y	METRIC DECIMAL	Drilled cycle safe retract plane in X, Y and Z. 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.
DRILL_SAFE_Z	METRIC DECIMAL	Drilled cycle safe retract plane in X, Y and Z. 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.

Variable	*Type*	*Usage*
DRILL_PECK_DEPTH_X	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.
DRILL_PECK_DEPTH_Y	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.
DRILL_PECK_DEPTH_Z	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.
DRILL_PECK_RAPID_TO_X	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.

Variable	*Type*	*Usage*
DRILL_PECK_RAPID_TO_Y	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.
DRILL_PECK_RAPID_TO_Z	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.
INIT_TOOL_LENGTH	METRIC DECIMAL	Stores the internal tool length that is defined in the tool definition and is used in the system calc section called `: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. Note: Can be used only with CAMWorks 2005 or ProCAM II 2004 or later versions.

Variable	*Type*	*Usage*

ARC_DEVIATION	METRIC DECIMAL	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.
		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.
		`:C: ARC_DEVIATION=max_arc_dev`
		`:C: SYS_CANNED(5,CALC_BREAK_ARC)`

SHAPE_INSIDE	INTEGER	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.

SHAPE_TYPE	INTEGER	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
		`OPEN_SHAPE=0`
		`CLOSED_SHAPE=1`
		`CIRCLE_SHAPE=2`

SHAPE_DIAMETER	METRIC DECIMAL	Stores the diameter of a laser or plasma toolpath if `SHAPE_TYPE` is set to `CIRCLE_SHAPE.`
BOUNDARY_AREA	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`.

Variable	*Type*	*Usage*
V_COMP	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.
XC	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.
YC	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.
ZC	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.
IC	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.
JC	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.
KC	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.
OPR_CLEARANCE	INTEGER	Stores the EDM operation clearance amount.
OPR_GLUE_STOP	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).
OPR_GLUE_DISTANCE	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.

Variable	*Type*	*Usage*
CLAMP1_POSITION	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.
CLAMP2_POSITION	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.
CLAMP3_POSITION	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.
CLAMP4_POSITION	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.
KOMBID	CHARACTER	Stores CAMWorks assembly mode tool ID information per tool change.
NEXT_KOMBID	CHARACTER	Stores CAMWorks assembly mode tool ID information per tool change for the next tool to be output.
NEXT_MOVE_KOMBID	CHARACTER	Stores CAMWorks assembly mode tool ID information for the next movement.
OPR_Z_ROTARY_RETRACT_PLANE	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.
ARM_PICKUP_X	METRIC DECIMAL	Stores the ProCAM 2D sorter arm operations pickup position in the X axis.

Variable	*Type*	*Usage*
ARM_PICKUP_Y	METRIC DECIMAL	Stores the ProCAM 2D sorter arm operations pickup position in the Y axis.
ARM_DESTINATION_X	METRIC DECIMAL	Stores the ProCAM 2D sorter arm operations destination position in the X axis
ARM_DESTINATION_Y	METRIC DECIMAL	Stores the ProCAM 2D sorter arm operations destination position in the Y axis.
ARM_OFFSET	METRIC DECIMAL	Stores the ProCAM 2D sorter arm operations offset distance.
ARM_ACTIVE_CUPS	CHARACTER	Stores the ProCAM 2D sorter arm operations active cups.
OPR_POLAR	INTEGER	`OPR_POLAR=TRUE or FALSE` Set to `FALSE` if your machine does not support special G-code for doing milling on the OD and or FACE. Set this command to `TRUE` if you have a machine that supports the special G-code for milling on the OD and or FACE. 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.
OPR_B_AXIS	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

INC_B_END	METRIC DECIMAL	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.
		Options:
		`N_INC_B_END N_ = next move`
		`P_INC_B_END P_ = previous move`

Variable	*Type*	*Usage*

ABS_B_END	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`

ABS_B_START	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`

MCS_X_OFFSET	METRIC DECIMAL	Stores the Machine Coordinate System as an absolute position in X using ProCAM II or CAMWorks assembly mode.
MCS_Y_OFFSET	METRIC DECIMAL	Stores the Machine coordinate System as an absolute position in Y using ProCAM II or CAMWorks assembly mode.
MCS_Z_OFFSET	METRIC DECIMAL	Stores the Machine coordinate System as an absolute position in Z using ProCAM II or CAMWorks assembly mode.
LASER_DBPATH	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.
ENGLISH_LASERDB	CHARACTER	Laser or Plasma: Stores the system database path. Is used with the `OPENDB` command. It stores the English database path.
METRIC_LASERDB	CHARACTER	Laser or Plasma: Stores the system database path. Is used with the `OPENDB` command. It stores the Metric database path.

Variable	*Type*	*Usage*
LD_METRIC	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 LASER_PLASMA_CUT_DATA must be set to TRUE.
LD_MATERIAL	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 LASER_PLASMA_CUT_DATA must be set to TRUE.
LD_THICKNESS	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 LASER_PLASMA_CUT_DATA must be set to TRUE.

Variable	*Type*	*Usage*
LD_DATA_GROUP	CHARACTER	Laser or Plasma: When using the standard database, this variable stores the Data Group. These are the different data groups: 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 3cut Conditions Hole size 4 cut Conditions Hole size 5 cut Conditions Microjoints 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 `LASER_PLASMA_CUT_DATA` must be set to `TRUE`.
LD_PART_CLEARANCE	DECIMAL	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 `LASER_PLASMA_CUT_DATA` must be set to `TRUE`.

Variable	*Type*	*Usage*
LD_SENSOR_RADIUS	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 LASER_PLASMA_CUT_DATA must be set to TRUE.
LD_GO_AROUND_DISTANCE	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 LASER_PLASMA_CUT_DATA must be set to TRUE.
LD_AUTOBREAK_IN_LENGTH	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 LASER_PLASMA_CUT_DATA must be set to TRUE.
LD_AUTOBREAK_OUT_LENGTH	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 LASER_PLASMA_CUT_DATA must be set to TRUE.

Variable	*Type*	*Usage*
LD_SYSTEM_COMP	INTEGER	Laser or Plasma: When using the standard database, this variable stores what the System comp is: (3) for n/a (2) for Off (1) for On 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 LASER_PLASMA_CUT_DATA must be set to TRUE.
LD_CC_MODE	INTEGER	Laser or Plasma: When using the standard database, this variable stores what the Cutting condition mode is: (1) Area (2) Line Length/Arc Radius 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 LASER_PLASMA_CUT_DATA must be set to TRUE.
LD_DIRECTION	INTEGER	Laser or Plasma: When using the standard database, this variable stores what the Cutting direction is: (1) Clockwise (2) Counter Clockwise (3) Original (4) n/a 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 LASER_PLASMA_CUT_DATA must be set to TRUE.

Variable	*Type*	*Usage*
LD_START_POSITION	INTEGER	Laser or Plasma: When using the standard database, this variable stores what the where cutting is to start from: (1) Auto (2) Long edge midpoint (3) n/a 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 LASER_PLASMA_CUT_DATA must be set to TRUE.
LD_LIFTHEAD	INTEGER	Laser or Plasma: When using the standard database, this variable stores whether you need the Lift Head option or not: (1) On (2) Off (3) n/a 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 LASER_PLASMA_CUT_DATA must be set to TRUE.
LD_START_AT_HOLE_CENTER	INTEGER	Laser or Plasma: When using the standard database, this variable stores if you want to start at hole center or not: (1) On (2) Off (3) n/a 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 LASER_PLASMA_CUT_DATA must be set to TRUE.

Variable	*Type*	*Usage*
LD_LEADIN_MODE	INTEGER	Laser or Plasma: When using the standard database, this variable stores what type of leadin is used. (1) Arc (2) n/a (3) None (4) Parallel (5) Perpendicular 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 LASER_PLASMA_CUT_DATA must be set to TRUE.
LD_LEADIN_LENGTH	DECIMAL	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 LASER_PLASMA_CUT_DATA must be set to TRUE.
LD_LEADIN_ANGLE	DECIMAL	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 LASER_PLASMA_CUT_DATA must be set to TRUE.
LD_LEADIN_ARC_RADIUS	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 LASER_PLASMA_CUT_DATA must be set to TRUE.

Variable	*Type*	*Usage*
LD_LEADIN_ARC_ANGLE	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 LASER_PLASMA_CUT_DATA must be set to TRUE.
LD_OVERLAP	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 LASER_PLASMA_CUT_DATA must be set to TRUE.
LD_LEADOUT_MODE	INTEGER	Laser or Plasma: When using the standard database, this variable stores what type of leadout is used. (1) Clockwise (2) Counter Clockwise (3) Original (4) None 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 LASER_PLASMA_CUT_DATA must be set to TRUE.
LD_LEADOUT_LENGTH	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 LASER_PLASMA_CUT_DATA must be set to TRUE.

Variable	*Type*	*Usage*
LD_LEADOUT_ANGLE	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 LASER_PLASMA_CUT_DATA must be set to TRUE.
LD_LEADOUT_ARC_RADIUS	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 LASER_PLASMA_CUT_DATA must be set to TRUE.
LD_LEADOUT_ARC_ANGLE	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 LASER_PLASMA_CUT_DATA must be set to TRUE.
LD_LEADOUT_OVERLAP	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 LASER_PLASMA_CUT_DATA must be set to TRUE.

Variable	*Type*	*Usage*
LD_SIZE	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 LASER_PLASMA_CUT_DATA must be set to TRUE.
LD_COLOR	INTEGER	Laser or Plasma: When using the standard database, this variable stores the color of the different path sizes and shapes. (1) BLACK (2) BLUE (3) GREEN (4) CYAN (5) RED (6) MAGENTA (7) BROWN (8) GRAY (9) WHITE (10)LTBLUE (11)LTGREEN (12)LTCYAN (13)LTRED (14)LTMAGEN (15)YELLOW (16)HIWHITE 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 LASER_PLASMA_CUT_DATA must be set to TRUE.
LD_SUBROUTINE_NUMBER	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 LASER_PLASMA_CUT_DATA must be set to TRUE.

Variable	*Type*	*Usage*
LD_FEEDRATE	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 LASER_PLASMA_CUT_DATA must be set to TRUE.
LD_FEEDRATE_PERCENT	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 LASER_PLASMA_CUT_DATA must be set to TRUE.
LD_POWER_LEVEL	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 LASER_PLASMA_CUT_DATA must be set to TRUE.
LD_ASSIST_GAS	CHARACTER	Laser or Plasma: When using the standard database, this variable stores the assist gas type. (1)Oxygen (2)Nitrogen (3)Carbon Dioxide (4)n/a 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 LASER_PLASMA_CUT_DATA must be set to TRUE.

Variable	*Type*	*Usage*
LD_GAS_PRESSURE	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 LASER_PLASMA_CUT_DATA must be set to TRUE.
LD_2ND_ASSIST_GAS	CHARACTER	Laser or Plasma: When using the standard database, this variable stores the second assist gas type: (1) Oxygen (2) Nitrogen (3) Carbon Dioxide (4) n/a 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 LASER_PLASMA_CUT_DATA must be set to TRUE.
LD_2ND_GAS_PRESSURE	INTEGER	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 LASER_PLASMA_CUT_DATA must be set to TRUE.
LD_LASER_MODE	CHARACTER	Laser or Plasma: When using the standard database, this variable stores the laser mode type: (3)Continuous Wave (4)Dynamic Power (1)Gated Pulse (5) n/a (2) Super Power 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 LASER_PLASMA_CUT_DATA must be set to TRUE.

Variable	*Type*	*Usage*
LD_FREQUENCY	INTEGER	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 LASER_PLASMA_CUT_DATA must be set to TRUE.
LD_DUTY_CYCLE	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 LASER_PLASMA_CUT_DATA must be set to TRUE.
LD_COOLANT_MODE	CHARACTER	Laser or Plasma: When using the standard database, this variable stores the coolant type used. (1)On (2)Off (3)n/a 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 LASER_PLASMA_CUT_DATA must be set to TRUE.
LD_OFFSET_VALUE	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 LASER_PLASMA_CUT_DATA must be set to TRUE.

Variable	*Type*	*Usage*
LD_DURATION	INTEGER	Laser or Plasma: When using the r 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 LASER_PLASMA_CUT_DATA must be set to TRUE.
LD_START_AT_MICROJOINT	INTEGER	Laser or Plasma: When using the standard database, this variable stores whether you are using microjoints: (1)On (2)Off (3)n/a The file for English is FABDBENGLISH.MDB and for Metric FABDBMETRIC.MDB. These are in the \procad\defaults folder. The post header command called LASER_PLASMA_CUT_DATA must be set to TRUE.
LD_COMMENT	CHARACTER	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 LASER_PLASMA_CUT_DATA must be set to TRUE.
LD_END_AT_HOLE_CENTER	INTEGER	Laser or Plasma: When using the standard database, this variable stores whether you are ending at hole center or not. (1)On (2)Off (3)n/a The file for English is FABDBENGLISH.MDB and for Metric FABDBMETRIC.MDB. These are in the \procad\defaults folder. The post header command called LASER_PLASMA_CUT_DATA must be set to TRUE.

Variable	*Type*	*Usage*
LD_OPTION_START_POSITION	INTEGER	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 LASER_PLASMA_CUT_DATA must be set to TRUE.
LD_CHOICE_START_POSITION	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 LASER_PLASMA_CUT_DATA must be set to TRUE.
LD_USER_INT01 through LD_USER_INT10	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 LASER_PLASMA_CUT_DATA must be set to TRUE.
LD_USER_DBL01 through LD_USER_DBL10	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 LASER_PLASMA_CUT_DATA must be set to TRUE.
RUNNING_SYSTEM	INTEGER	The current running system. RUNNING_SYSTEM=CAMWORKS RUNNING_SYSTEM=PROCAM_2D RUNNING_SYSTEM=PROCAM_3D CAMWORKS=1 PROCAM_2D=2 PROCAM_3D=3

Variable	*Type*	*Usage*
SKIM_CUT	INTEGER	ProCAM 2D EDM: Stores the EDM skim cut. SKIM_CUT=CURRENT SKIM_CUT=PREVIOUS SKIM_CUT=NEXT CURRENT=0 PREVIOUS=1 NEXT=2
TAB_CUT	INTEGER	ProCAM 2D EDM: Stores the EDM tan cut. TAB_CUT=CURRENT TAB_CUT=PREVIOUS TAB_CUT=NEXT
PART_TOTAL_SKIM_CUTS	INTEGER	ProCAM 2D: Stores the total number of skim cuts in a part file.
PART_TOTAL_TAB_CUTS	INTEGER	ProCAM 2D: Stores the total number of tab cuts in a part file.
OPER_TOTAL_SKIM_CUTS	INTEGER	ProCAM 2D: Stores the total number of skim cuts in an operation.
OPER_TOTAL_TAB_CUTS	INTEGER	ProCAM 2D: Stores the total number of tab cuts in an operation.
OPER_CURRENT_SKIM_CUT	INTEGER	ProCAM 2D EDM: Stores the EDM operation type. SKIM_CUT=CURRENT SKIM_CUT=PREVIOUS SKIM_CUT=NEXT CURRENT=0 PREVIOUS=1 NEXT=2
OPER_CURRENT_TAB_CUT	INTEGER	ProCAM 2D EDM: Stores the EDM operation type. TAB_CUT=CURRENT TAB_CUT=PREVIOUS TAB_CUT=NEXT CURRENT=0 PREVIOUS=1 NEXT=2

Variable	*Type*	*Usage*

DSPINDLE	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`

POST_PATH	CHARACTER	Stores the posted output path.
OPR_LATHE_TAPPING	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`
LASER_MATERIAL	CHARACTER	Stores the material name from the setup information and updates the material name in the laser, plasma database, so it matches.
LASER_THICKNESS	CHARACTER	Stores the sheet thickness from the setup information and updates the sheet thickness in the laser, plasma database, so it matches.
ARM_SORTER_DBPATH	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.
ARM_ENGLISH_LASERDB	CHARACTER	Stores the Punch system sorter arm English database path. Is used with the OPENDB command.
ARM_METRIC_LASERDB	CHARACTER	Stores the Punch system sorter arm Metric database path. Is used with the OPENDB command.

OPR_LATHE_RETRACT_TYPE	INTEGER	Stores the retract type when using Lathe canned cycles:
		`(1) Z then X`
		`(2) X then Z`
		`(3) Direct`

Variable	*Type*	*Usage*
TOOL_SPEC_NAME	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.
TOOL_SPEC_PATH	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.
ABS_PICK_LOCATION	DECIMAL	CAMWorks: Stores the absolute value of the Lathe Sub Spindle transfer location along the Z axis.
INC_PICK_LOCATION	DECIMAL	CAMWorks: Stores the incremental value of the Lathe Sub Spindle transfer location along the Z axis.
TRANSFER_DISTANCE	DECIMAL	CAMWorks: Stores the distance traveled when a transfer takes place between the lathe main and sub spindle along the Z axis.

OPR_LATHE_APPROACH_STRATEGY	INTEGER	CAMWorks 2007 and later: Stores the current lathe operation's approach strategy to the start of the cut.
		`OPR_LATHE_APPROACH_STRATEGY=Z_THEN_X`
		`OPR_LATHE_APPROACH_STRATEGY=X_THEN_Z`
		`OPR_LATHE_APPROACH_STRATEGY=DIRECT`
		`OPR_LATHE_APPROACH_STRATEGY=AUTO`
		`Z_THEN_X=1`
		`X_THEN_Z=2`
		`DIRECT=3`
		`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.)

Variable	*Type*	*Usage*
OPR_LATHE_RETRACT_STRATEGY	INTEGER	CAMWorks 2007 and later: Stores the current lathe operation's retract strategy at the end of the cut. OPR_LATHE_RETRACT_STRATEGY=Z_THEN_X OPR_LATHE_RETRACT_STRATEGY=X_THEN_Z OPR_LATHE_RETRACT_STRATEGY=DIRECT OPR_LATHE_RETRACT_STRATEGY=AUTO Z_THEN_X=1 X_THEN_Z=2 DIRECT=3 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.)
CAM_MOVE_FLAG	INTEGER	CAMWorks 2007 and later: Shows what type of lathe move you are doing. :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 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

Variable	Type	Usage
		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. CAMWORKS_VER= CAM_REV2006 CAMWORKS_VER= CAM_REV2006EX CAMWORKS_VER= CAM_REV2007 CAMWORKS_VER= CAM_REV2008 CAMWORKS_VER= CAM_REV2009

Variable	*Type*	*Usage*
		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
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.

Variable	*Type*	*Usage*
DRIVE_POINT_TYPE **P_DRIVE_POINT_TYPE** **N_DRIVE_POINT_TYPE**	INTEGER	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. DRIVE_CENTER=0 DRIVE _RIGHT=1 DRIVE _LEFT=2 DRIVE _TOOL_NOSE=3 Supported in CAMWorks 2007 and later. Not supported in any ProCAM product.
TOUCHOFF_POINT_TYPE **P_TOUCHOFF_POINT_TYPE** **N_TOUCHOFF_POINT_TYPE**	INTEGER	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. PROG_POINT_CENTER=0 PROG_POINT _PRIMARY=1 PROG_POINT _SECONDARY=2 Supported in CAMWorks 2007 and later. Not supported in any ProCAM product.
RAPID_DURING_DRILL_CYCLE	INTEGER	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 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. RAPID_DURING_DRILL_CYCLE=TRUE RAPID_DURING_DRILL_CYCLE=FALSE Supported in CAMWorks 2007 and later. Not supported in any ProCAM product.
OPR_THREAD_CHAMFER_ANG	DECIMAL	This variable stores the lathe threading cycles chamfer angle from entered angle. Supported in CAMWorks 2007 and later. Not supported in any ProCAM product.

Variable	*Type*	*Usage*
CAMWORKS_MATERIAL	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.
LATHE_X_TOOL_OFFSET **P_LATHE_X_TOOL_OFFSET** **N_LATHE_X_TOOL_OFFSET**	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.
LATHE_Z_TOOL_OFFSET **P_LATHE_Z_TOOL_OFFSET** **N_LATHE_Z_TOOL_OFFSET**	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.
MACH_CX **P_MACH_CX** **N_MACH_CX**	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.
MACH_CY **P_MACH_CY** **N_MACH_CY**	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.
MACH_CZ **P_MACH_CZ** **N_MACH_CZ**	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.

Variable	Type	Usage
MACH_CJ **P_MACH_CJ** **N_MACH_CJ**	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.
MACH_CK **P_MACH_CK** **N_MACH_CK**	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.
TOOL_COOLANT **N_TOOL_COOLANT** **NC_TOOL_COOLANT**	INTEGER	Passes the coolant information from the CAMWorks tool definition. 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 Supported in CAMWorks 2008 and later. Not supported in any ProCAM product.
TOOL_DIAM_OFFSET **N_TOOL_DIAM_OFFSET** **NC_TOOL_DIAM_OFFSET**	INTEGER	Passes the tool diameter offset number from the CAMWorks tool definition. Supported in CAMWorks 2008 and later. Not supported in any ProCAM product.
TOOL_LENGTH_OFFSET **N_TOOL_LENGTH_OFFSET** **NC_TOOL_LENGTH_OFFSET**	INTEGER	Passes the tool length offset number from the CAMWorks tool definition. Supported in CAMWorks 2008 and later. Not supported in any ProCAM product.
TOOL_LENGTH_DIAM_OFFSET_METHOD	INTEGER	Passes whether you selected the length and diameter offset numbers from the tool definition or the post. METHOD_FROM_TOOL=0 METHOD_FROM_POST=1 Supported in CAMWorks 2008 and later. Not supported in any ProCAM product.

Variable	*Type*	*Usage*

HAVE_SURFACE_CP
P_HAVE_SURFACE_CP
N_HAVE_SURFACE_CP
	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.

ABS_X_PART_END **N_ABS_X_PART_END** **P_ABS_X_PART_END**	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.
ABS_Y_PART_END **N_ABS_Y_PART_END** **P_ABS_Y_PART_END**	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.
ABS_Z_PART_END **N_ABS_Z_PART_END** **P_ABS_Z_PART_END**	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.
CW_DWELL	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.

Variable	*Type*	*Usage*

TOOL_EDGE	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`

TLP_PART_NAME	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.
TLP_PART_DESC	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.
TLP_SETUP_NAME	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.
TLP_SETUP_DESC	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.
TLP_FEAT_NAME	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.
TLP_FEAT_DESC	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.

Variable	*Type*	*Usage*
TLP_OPER_NAME	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.
TLP_OPER_DESC	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.
SETUP_ID **P_SETUP_ID** **N_SETUP_ID**	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.
TOOLPATH_ID **P_TOOLPATH_ID** **N_TOOLPATH_ID**	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.
NEXT_SETUP_4AX_ANGLE	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.
NEXT_SETUP_5AX_ANGLE	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.
PART_STOCK_DIAM	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.
PART_STOCK_LENGTH	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.
NEXT_OPR_TYPE	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.

Variable	*Type*	*Usage*
NEXT_OPR_SUB_TYPE	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.
NEXT_IS_LATHE	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`
QUERY_ITEM_ID	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.
QUERY_RESULT	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.
QUERY_EERROR	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.
QUERY_INT_VALUE	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.
QUERY_DEC_VALUE	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.

Variable	*Type*	*Usage*
MULTIAXIS_CNC_COMP	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`

IS_5AXIS	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.

IS_MILL_FACE	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.

IS_MILL_OD	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.

IS_WRAPPED	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. OPR_AXIS_TYPE will also equal 4444 in this case.

TLP_FEAT_SETUP_NAME	CHARACTER	Stores the setup name for the current toolpath feature. Available in Mill, Turn and Mill/Turn.
SYNC_CODE_COMMENT	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.
REAR_SYNC_CODE **P_REAR_SINCE_CODE**	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.

Variable	*Type*	*Usage*

REAR_SYNC_CODE_TYPE	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`

FRONT_SYNC_CODE **P_FRONT_SYNC_CODE**	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.

FRONT_SYNC_CODE_TYPE	INTEGER	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.
		`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`

ARC_NORM_X **P_ARC_NORM_X** **N_ARC_NORM_X**	DECIMAL	Stores the current, previous and next X axis arc normal vector. Available in Mill, Turn and Mill/Turn.
ARC_NORM_Y **P_ARC_NORM_Y** **N_ARC_NORM_Y**	DECIMAL	Stores the current, previous and next Y axis arc normal vector. Available in Mill, Turn and Mill/Turn.

Variable	*Type*	*Usage*
ARC_NORM_Z **P_ARC_NORM_Z** **N_ARC_NORM_Z**	DECIMAL	Stores the current, previous and next Z axis arc normal vector. Available in Mill, Turn and Mill/Turn.
CTL_PATH	CHARACTER	Stores the path of the current post being used. Available in Mill, Turn and Mill/Turn. Supported in CAMWorks 2010 and later. Not supported in any ProCAM product.
WORLD_FCS_TO_SETUP_X **WORLD_FCS_TO_SETUP_Y** **WORLD_FCS_TO_SETUP_Z**	DECIMAL	This variable stores the distance in X, Y or Z from the selected World Coordinate System to the current mill Setup. This is a non rotated value. Not supported in any ProCAM product or CAMWorks product earlier then version CW2011 SP1. Available in Mill and Mill/Turn.
WORLD_VECTOR_X **WORLD_VECTOR_Y** **WORLD_VECTOR_Z**	DECIMAL	This variable stores the X, Y or Z signed vector from the selected World Coordinate System to the current mill Setup. This is a non rotated value. Not supported in any ProCAM product or CAMWorks product earlier then version CW2011 SP1. Available in Mill and Mill/Turn.

SETUP_MCS_X_OFFSET
P_SETUP_MCS_X_OFFSET
N_SETUP_MCS_X_OFFSET
	DECIMAL	This variable stores the distance in X from the selected World Coordinate System to the current mill Setup coordinate system. This is a rotated value. Not supported in any ProCAM product or CAMWorks product earlier then version CW2011 SP1. Available in Mill and Mill/Turn.
		Options:
		`N_SETUP_MCS_X_OFFSET N_ = next setup`
		`P_SETUP_MCS_X_OFFSET P_ = previous setup`

SETUP_MCS_Y_OFFSET
P_SETUP_MCS_Y_OFFSET
N_SETUP_MCS_Y_OFFSET
	DECIMAL	This variable stores the distance in Y from the selected World Coordinate System to the current mill Setup coordinate system. This is a rotated value. Not supported in any ProCAM product or CAMWorks product earlier then version CW2011 SP1. Available in Mill and Mill/Turn.
		Options:
		`N_SETUP_MCS_Y_OFFSET N_ = next setup`
		`P_SETUP_MCS_Y_OFFSET P_ = previous setup`

Variable	*Type*	*Usage*

SETUP_MCS_Z_OFFSET
P_SETUP_MCS_Z_OFFSET
N_SETUP_MCS_Z_OFFSET
	DECIMAL	This variable stores the distance in Z from the selected World Coordinate System to the current mill Setup coordinate system. This is a rotated value. Not supported in any ProCAM product or CAMWorks product earlier then version CW2011 SP1. Available in Mill and Mill/Turn.
		Options:
		`N_SETUP_MCS_Z_OFFSET N_ = next setup`
		`P_SETUP_MCS_Z_OFFSET P_ = previous setup`

OPR_LEADIN_FEEDRATE	DECIMAL	This variable stores the operation leadin feedrate in milling. Not supported in any ProCAM product or CAMWorks product earlier then version CW2011 SP1. Available in Mill and Mill/Turn.
OPR_THREAD_DEPTH	DECIMAL	This variable stores the thread depth of a canned lathe threading cycle. Not supported in any ProCAM product or CAMWorks product earlier then version CW2011 SP1. Available in Turn and Mill/Turn.
PRE_POSITION_ROTARY_TYPE	INTEGER	This variable can be set to either have the A axis or the B axis as the 4th axis rotary. `PRE_POSITION_ROTARY_TYPE=ROTARY_TYPE_A or ROTARY_TYPE_B` In ProCAM, the B axis was setup to be the 4th axis rotary and normal software has the A axis as the 4th axis rotary. Not supported in any ProCAM product or CAMWorks product earlier then version CW2010. Available in Mill.
SETUP_REVERSE_Z_DIR	INTEGER	`SETUP_REVERSE_Z_DIR=TRUE or FALSE` This variable can be used to find out if user selected reverse Z axis in turn setup. Not supported in any ProCAM product or CAMWorks product earlier then version CW2013. Available in Lathe and Mill/Turn
IS_COMMENT	INTEGER	`IS_COMMENT=TRUE or FALSE` This variable can be used in a sub spindle operation if user selected to output code or comment to the program. Not supported in any ProCAM product or CAMWorks product earlier then version CW2013. Available in Lathe and Mill/Turn.

Variable	*Type*	*Usage*
SPINDLE_STATE	INTEGER	SPINDLE_STATE=OFF SPINDLE_STATE=ON SPINDLE_STATE=LOCK OFF=0 ON=1 LOCK=2 This variable can be used in a sub spindle operation to see if the spindle is on,off or locked. Not supported in any ProCAM product or CAMWorks product earlier then version CW2013. Available in Lathe and Mill/Turn
SPINDLE_SYNC_TYPE	INTEGER	SPINDLE_SYNC_TYPE= SYNC_SPEED SPINDLE_SYNC_TYPE= SYNC_PHASE SYNC_SPEED=0 SYNC_PHASE=1 This variable can be used in a sub spindle operation to see if the spindle is used for speed or phase. Not supported in any ProCAM product or CAMWorks product earlier then version CW2013. Available in Lathe and Mill/Turn.
SPINDLE_ORIENTATION	DECIMAL	SPINDLE_ORIENTATION=Amount entered in spindle operation This variable can be used in a sub spindle operation to set the spindle orientation before a transfer. Not supported in any ProCAM product or CAMWorks product earlier then version CW2013. Available in Lathe and Mill/Turn.
ABS_SPINDLE_Z_END	METRIC DECIMAL	ABS_SPINDLE_Z_END=Amount entered in spindle operation This variable can be used in a sub spindle operation to move the sub spindle to desired position. Not supported in any ProCAM product or CAMWorks product earlier then version CW2013. Available in Lathe and Mill/Turn.

Variable	*Type*	*Usage*

TLP_FEATURE_ID	INTEGER	This variable can be used find the current, next or previous movement feature ID number.
		Not supported in any ProCAM product or CAMWorks product earlier then version CW2013. Available in Mill, Lathe and Mill/Turn
		Options:
		`N_TLP_FEATURE_ID`
		`P_TLP_FEATURE_ID`

OPR_MOVE_ID	INTEGER	This variable can be used find the current, next or previous movement ID number.
		Not supported in any ProCAM product or CAMWorks product earlier then version CW2013.Available in Mill, Lathe and Mill/Turn
		Options:
		`N_OPR_MOVE_ID`
		`P_OPR_MOVE_ID`

NEXT_OPR_SUB_STATION	INTEGER	This variable can be used find the next operation tool Sub Station number. Not supported in any ProCAM product or CAMWorks product earlier then version CW2013. Available in Mill, Lathe and Mill/Turn.
SUB_STATION	INTEGER	This variable can be used find the current tool Sub Station number. Not supported in any ProCAM product or CAMWorks product earlier then version CW2013. Available in Mill, Lathe and Mill/Turn.
NC_SUB_STATION	INTEGER	This variable can be used find the next tool Sub Station number. Not supported in any ProCAM product or CAMWorks product earlier then version CW2013. Available in Mill, Lathe and Mill/Turn.
N_SUB_STATION	INTEGER	This variable can be used find the next movement tool Sub Station number. Not supported in any ProCAM product or CAMWorks product earlier then version CW2013. Available in Mill, Lathe and Mill/Turn.

Variable	*Type*	*Usage*
HAVE_SUB_STATIONS	INTEGER	HAVE_SUB_STATIONS=TRUE or FALSE This variable can be used to find out if user selected to use sub stations when defining tools. Not supported in any ProCAM product or CAMWorks product earlier then version CW2013. Available in Mill, Lathe and Mill/Turn.
HAVE_START_OPER_SYNC_CODES	INTEGER	HAVE_START_OPER_SYNC_CODES=TRUE or FALSE This variable can be used to find out if CAMWorks has passed sync codes at the start of an operation. Not supported in any ProCAM product or CAMWorks product earlier then version CW2013. Available in Lathe and Mill/Turn.
HAVE_FIRST_MOVE_SYNC_CODES	INTEGER	HAVE_FIRST_MOVE_SYNC_CODES=TRUE or FALSE This variable can be used to find out if CAMWorks has passed sync codes at first rapid move of an operation. Not supported in any ProCAM product or CAMWorks product earlier then version CW2013. Available in Lathe and Mill/Turn.
HAVE_END_OPER_SYNC_CODES	INTEGER	HAVE_END_OPER_SYNC_CODES=TRUE or FALSE This variable can be used to find out if CAMWorks has passed sync codes at end of an operation. Not supported in any ProCAM product or CAMWorks product earlier then version CW2013. Available in Lathe and Mill/Turn.
POST_LIBRARY_VERSION	INTEGER	This variable can be used to find out if when post is compiled what the library version is. Not supported in any ProCAM product or CAMWorks product earlier then version CW2013. Available in Mill, Lathe and Mill/Turn.

Variable	*Type*	*Usage*
POST_LIBRARY_SUBVERSION	INTEGER	This variable can be used to find out if when post is compiled what the library sub version is. Not supported in any ProCAM product or CAMWorks product earlier then version CW2013. Available in Mill, Lathe and Mill/Turn.

MCS_TYPE	INTEGER	`MCS_TYPE=1`
		`MCS_TYPE=2`
		`MCS_TYPE=3`
		`1=Fixture coordinate`
		`2=Work coordinate`
		`3=Work coordinate with Sub work coordinate`
		This variable can be used to find the selected Fixture coordinate type.
		Not supported in any ProCAM product or CAMWorks product earlier then version CW2013. Available in Mill, Lathe and Mill/Turn.

TOOL_ANGLE **N_TOOL_ANGLE** **NC_TOOL_ANGLE**	DECIMAL	This variable stores the taper angle of a tapered mill tool and the drill angle of a drill tool. Not supported in any ProCAM product or CAMWorks product earlier then version CW2010 SP1. Available in Mill and Mill/Turn.
TOOL_PROTRUSION **N_TOOL_PROTRUSION** **NC_TOOL_PROTRUSION**	DECIMAL	This variable stores the tool protrusion from the bottom of the tool holder to the end of a mill tool or a drill tool. Not supported in any ProCAM product or CAMWorks product earlier then version CW2010 SP1. Available in Mill and Mill/Turn.
STOCK_LENGTH	DECIMAL	• This variable stores the stock length in milling. Customer must create a Fixture Coordinate System in order to get value. • Not supported in any ProCAM product or CAMWorks product earlier version than CAMWorks 2010 SP1. • Available in Mill and Mill-Turn.

Variable	*Type*	*Usage*
STOCK_WIDTH	DECIMAL	• This variable stores the stock width in milling. Customer must create a Fixture Coordinate System in order to get value. • Not supported in any ProCAM product or CAMWorks product earlier version than CAMWorks 2010 SP1. • Available in Mill and Mill-Turn.
STOCK_HEIGHT	DECIMAL	• This variable stores the stock height in milling. Customer must create a Fixture Coordinate System in order to get value. • Not supported in any ProCAM product or CAMWorks product earlier version than CAMWorks 2010 SP1. • Available in Mill and Mill-Turn.
OPR_XY_ALLOWANCE	DECIMAL	This variable stores the XY allowance in milling operations. Not supported in any ProCAM product or CAMWorks product earlier then version CW2010 SP2. Available in Mill and Mill/Turn.
TLP_FEAT_SETUP_DESC	CHARACTER	This variable stores the setup description for the current toolpath feature. Not supported in any ProCAM product or CAMWorks product earlier then version CW2010. Available in Mill, Turn and Mill/Turn.

SETUP_WORLD_X_OFFSET	METRIC DECIMAL	This variable can be used find the current, next or previous X world offset none rotated. Not supported in any ProCAM product or CAMWorks product earlier then version CW2013. Available in Mill.
		Options:
		`N_SETUP_WORLD_X_OFFSET`
		`P_SETUP_WORLD_X_OFFSET`

SETUP_WORLD_Y_OFFSET	METRIC DECIMAL	This variable can be used find the current, next or previous Y world offset none rotated. Not supported in any ProCAM product or CAMWorks product earlier then version CW2013. Available in Mill.
		Options:
		`N_SETUP_WORLD_Y_OFFSET`
		`P_SETUP_WORLD_Y_OFFSET`

Variable	*Type*	*Usage*
SETUP_WORLD_Z_OFFSET	METRIC DECIMAL	This variable can be used find the current, next or previous Z world offset none rotated. Not supported in any ProCAM product or CAMWorks product earlier then version CW2013. Available in Mill. Options: N_SETUP_WORLD_Z_OFFSET P_SETUP_WORLD_Z_OFFSET
SUB_STATION_ID	CHARACTER	This variable can be used to find the current tool Sub Station description. Not supported in any ProCAM product or CAMWorks product earlier then version CW2013. Available in Mill, lathe and Mill/Turn.
NC_SUB_STATION_ID	CHARACTER	This variable can be used to find the next tool Sub Station description. Not supported in any ProCAM product or CAMWorks product earlier then version CW2013. Available in Mill, lathe and Mill/Turn.
N_SUB_STATION_ID	CHARACTER	This variable can be used to find the next movement tool Sub Station description. Not supported in any ProCAM product or CAMWorks product earlier then version CW2013. Available in Mill, lathe and Mill/Turn.
TOOL_X_GAGE_OFFSET	METRIC DECIMAL	This variable can be used to find the current tool X, Y and Z gage offset. Not supported in any ProCAM product or CAMWorks product earlier then version CW2013. Available in Mill, lathe and Mill/Turn. Options: TOOL_Y_GAGE_OFFSET TOOL_Z_GAGE_OFFSET
NC_TOOL_X_GAGE_OFFSET	METRIC DECIMAL	This variable can be used to find the next tool X, Y and Z gage offset. Not supported in any ProCAM product or CAMWorks product earlier then version CW2013. Available in Mill, lathe and Mill/Turn. Options: NC_TOOL_Y_GAGE_OFFSET NC_TOOL_Z_GAGE_OFFSET

Variable	*Type*	*Usage*
N_TOOL_X_GAGE_OFFSET	METRIC DECIMAL	This variable can be used to find the next movement tool X,Y and Z gage offset. Not supported in any ProCAM product or CAMWorks product earlier then version CW2013. Available in Mill, lathe and Mill/Turn. Options: `N_TOOL_Y_GAGE_OFFSET` `N_TOOL_Z_GAGE_OFFSET`
SPINDLE_I	DECIMAL	This variable can be used to find the spindle I vector. Not supported in any ProCAM product or CAMWorks product earlier then version CW2013. Available in Mill, lathe and Mill/Turn.
SPINDLE_J	DECIMAL	This variable can be used to find the spindle J vector. Not supported in any ProCAM product or CAMWorks product earlier then version CW2013. Available in Mill, lathe and Mill/Turn.
SPINDLE_K	DECIMAL	This variable can be used to find the spindle K vector. Not supported in any ProCAM product or CAMWorks product earlier then version CW2013. Available in Mill, lathe and Mill/Turn.
OPR_RPM	INTEGER	This variable stores the RPM value from a Turn or Mill/turn speed dialog box. Not supported in any ProCAM product or CAMWorks product earlier then version CW2013 SP1. Available in Turn and Mill/Turn.
MCS_U_NAME	CHARACTER	Tells Simulator what is the axis label for the X axis move of the sub spindle. Can be used in the QUERY_ITEM_ID=QUERY_GCODE_WCS section. To be used in CAMWorks 2014 or newer version.

Variable	*Type*	*Usage*
MCS_V_NAME	CHARACTER	Tells Simulator what is the axis label for the Y axis move of the sub spindle. Can be used in the QUERY_ITEM_ID=QUERY_GCODE_WCS section. To be used in CAMWorks 2014 or newer version.
MCS_W_NAME	CHARACTER	Tells Simulator what is the axis label for the Z axis move of the sub spindle. Can be used in the QUERY_ITEM_ID=QUERY_GCODE_WCS section. To be used in CAMWorks 2014 or newer version.
MCS_U_OFFSET	DECIMAL	Tells Simulator what the X axis offset is for the sub spindle. Can be used in the QUERY_ITEM_ID=QUERY_GCODE_WCS section. To be used in CAMWorks 2014 or newer version.
MCS_V_OFFSET	DECIMAL	Tells Simulator what the Y axis offset is for the sub spindle. Can be used in the QUERY_ITEM_ID=QUERY_GCODE_WCS section. To be used in CAMWorks 2014 or newer version.
MCS_W_OFFSET	DECIMAL	Tells Simulator what the Z axis offset is for the sub spindle. Can be used in the QUERY_ITEM_ID=QUERY_GCODE_WCS section. To be used in CAMWorks 2014 or newer version.
OPR_REVERSE_ARC_DIR=TRUE or FALSE	INTEGER	Tells the post in a Mill-Turn or Mill arc move whether the arcs need to be reversed. Affects MILL_OD, MILL_FACE arcs on Main and Sub spindles. To be used in CAMWorks 2014 SP2 or newer versions.

Variable	*Type*	*Usage*
OPR_REVERSE_ARC_DIR	DECIMAL	Detects if the Mill operations OD or FACE arcs need to be reversed or not in a Mill-Turn post. To be used in CAMWorks 2014 SP2 or newer versions. OPR_REVERSE_ARC_DIR=0 - No reversal is needed. OPR_REVERSE_ARC_DIR=1 - The arc directions should be reversed.
TLP_2AX_FEAT_DEPTH	DECIMAL	Holds the feature machining depth associated to the current 2 Axis toolpath. To be used in CAMWorks 2014 SP3 or later versions.
QUERY_TLP_Z_MIN	DECIMAL	Holds the feature Z min. depth associated to the current toolpath when used in conjunction with QUERY_MILL_TLP_Z_EXTENTS. To be used in CAMWorks 2014 SP3 or later versions.
QUERY_TLP_Z_MAX	DECIMAL	Holds the feature Z max. depth associated to the current toolpath when used in conjunction with QUERY_MILL_TLP_Z_EXTENTS. To be used in CAMWorks 2014 SP3 or later versions.
SETUP_MACH_X	DECIMAL	Gives the distance in X axis between the rotary and tilt axis that is passed from the KIN file. To be used in CAMWorks 2014 SP3 or later versions.
SETUP_MACH_Y	DECIMAL	Gives the distance in Y axis between the rotary and tilt axis that is passed from the KIN file. To be used in CAMWorks 2014 SP3 or later versions.
SETUP_MACH_Z	DECIMAL	Gives the distance in Z axis between the rotary and tilt axis that is passed from the KIN file. To be used in CAMWorks 2014 SP3 or later versions.

Variable	*Type*	*Usage*
SETUP_MACH_I	DECIMAL	Gives the vector distance in X axis between the rotary and tilt axis that is passed from the KIN file. To be used in CAMWorks 2014 SP3 or later versions.
SETUP_MACH_J	DECIMAL	Gives the vector distance in Y axis between the rotary and tilt axis that is passed from the KIN file. To be used in CAMWorks 2014 SP3 or later versions.
SETUP_MACH_K	DECIMAL	Gives the vector distance in Z axis between the rotary and tilt axis that is passed from the KIN file. To be used in CAMWorks 2014 SP3 or later versions.
SOLIDWORKS_FILENAME	CHARACTER	Stores the SOLIDWORKS file name - only 100 characters max. will be stored. To be used in CAMWorks 2014 SP3 or later versions.
OPR_Z_DEPTH	METRIC DECIMAL	Drill: Absolute Z depth. This is used for Drilling only and not available for Milling.
O_CYCLE_START_X	DECIMAL	Passes the X start point of Turn canned OD, ID and Face cycle. This value will be used before the canned cycle is output. To be used in CAMWorks 2015 or newer versions.
O_CYCLE_START_Z	DECIMAL	Passes the Z start point of Turn canned OD, ID and Face cycle. This value will be used before the canned cycle is output. To be used in CAMWorks 2015 or newer versions.
USE_FEATURE_OD_ID	INTEGER	Setting this variable to "1" will set the post system variables OD and ID to the correct value in a Rough Turn Canned Cycle. This variable can be set in a Turn or Mill-Turn post at either CALC_START_OF_TAPE in your post library file or CALC_INIT_CODES in the SRC file. To be used in CAMWorks 2015 SP0 or newer version.

Variable	*Type*	*Usage*
STOCK_MIN_X	DECIMAL	Stores the Stock Minimum value in X axis. To be used in CAMWorks 2015 SP0 or newer version.
STOCK_MIN_Y	DECIMAL	Stores the Stock Minimum value in Y axis. To be used in CAMWorks 2015 SP0 or newer version.
STOCK_MIN_Z	DECIMAL	Stores the Stock Minimum value in Z axis. To be used in CAMWorks 2015 SP0 or newer version.
STOCK_MAX_X	DECIMAL	Stores the Stock Maximum value in X axis. To be used in CAMWorks 2015 SP0 or newer version.
STOCK_MAX_Y	DECIMAL	Stores the Stock Maximum value in Y axis. To be used in CAMWorks 2015 SP0 or newer version.
STOCK_MAX_Z	DECIMAL	Stores the Stock Maximum value in Z axis. To be used in CAMWorks 2015 SP0 or newer version.
5AXIS_FEED_DISTANCE **N_5AXIS_FEED_DISTANCE** **P_5AXIS_FEED_DISTANCE**	DECIMAL	Stores the distance traveled in a 5 Axis toolpath. Can be used instead of DISTANCE, N_DISTANCE and P_DISTANCE. To be used in CAMWorks 2015 SP0 or newer version.
IS_TLP_COMPENSATED	INTEGER	Passes information to the post whether the toolpath is compensated or not. The value will either be TRUE or FALSE. To be used in CAMWorks 2015 SP1 or newer version.
IS_PATTERN_FEATURE	INTEGER	Passes information to the post whether the feature is patterned or not. The value will either be TRUE or FALSE. To be used in CAMWorks 2015 SP1 or newer version.

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

Parameter	*Description*
type	the type of breakup and is a constant 1 Single points 2 Lines, arcs and bolt holes (use only on grids and big hole patterns) 3 Breaks a thread cycle into diameters (use only on threading cycles) 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). 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). 6 Breaks a canned peck drilling cycle into operation defined increments (use only in CAMWorks 2005 or newer version for mill pecking canned cycles).
section	section that will handle the exploded entity The word SYSTEM 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

Parameter	*Description*
`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

Parameter	*Description*
`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

Parameter	*Description*
`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

Parameter	*Description*
`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

Parameter	Description
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

Parameter	*Description*
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.

Parameter	*Description*
`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 OPENRWTXT only. This will not work with OPENTXT.

Parameter	*Description*
`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 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(1,calc_section)

GETMCS(2,calc_section)

GETMCS(3,calc_section)

Comments

- Using GETMCS (1, calc_section) will call all Setups, including duplicate Setups. To be used in CAMWorks 2015 SP1 or newer version.
- Using GETMCS (2, calc_section) will do a search through the whole part file and gather all the different MCS offsets along with the Tool, Tool Comment, Rotation angles, Work Coordinate Offset values and TLP_SETUP_NAME . 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`.
- Using GETMCS (3, calc_section) will call all unique Setups with different angles, thus excluding duplicate Setups. To be used in CAMWorks 2015 SP1 or newer version.

STRGLEN

Purpose

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

Syntax

STRGLEN(string_var)

Comments

Parameter	*Description*
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

Parameter	*Description*
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

Parameter	*Description*
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

Parameter	*Description*
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

Parameter	*Description*
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

Parameter	*Description*
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 life 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.

Command	*Description*
CONFIG_SELECT_NAME	The name of the lowercase attribute that will show the list in the Setup Info.
CONFIG_SELECT_OPTION	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.
CONFIG_SELECT_DEFAULT	Sets the default selection from the list. If you wanted the 3rd string in the list as the default, then this would equal 3.
CONFIG_SELECT_END	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(OUPTUT_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>
```

ATOF

Purpose

To turn a string input into a number. This will work in connection with opening an external file. By reading in an external file line by line, you will be able to transform the string into a number, if required.

Syntax

ATOF()

Comments

The example code below shows you how you could use this command. The post variable TESTSTR could also be a string that was being read in line by line from an external file. MyVal could be either an integer or decimal post value.

```
:C: TESTSTR={6.1957}
:C: MyVal=ATOF(TESTSTR)
:C: CALL(SEE_MY_VAL)
```

The result would be:

```
N1 X6.1957
```

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.

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_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.

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_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.

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

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_TRAPDOOR_LASER

Purpose

Laser system rapid move to trap door calc section.

Syntax

```
:SECTION=CALC_RAPID_TO_TRAP_DOOR_LASER
```

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_RAPID_TO_TRAPDOOR_PLASMA

Purpose

Plasma system rapid move to trap door calc section.

Syntax

```
:SECTION=CALC_RAPID_TO_TRAP_DOOR_PLASMA
```

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/LASER combination machine that uses a pre-punched hole for starting a laser profile.

Syntax

```
:SECTION=CALC_PROFILE_DRILL_LASER
```

Comments

This section will be called when you do pre-punch hole for the laser profile. Normally, you would do a punch single hit and this, then give the laser a place to start the cut if not on the edge of the sheet.

CALC_PROFILE_DRILL_PLASMA

Purpose

Used when you have a PUNCH/PLASMA combination machine that uses a pre-punched hole for starting a plasma profile.

Syntax

```
:SECTION=CALC_PROFILE_DRILL_PLASMA
```

Comments

This section will be called when you do pre-punch hole for the plasma profile. Normally, you would do a punch single hit and this, then give the plasma 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

Purpose

EDM operations ID names.

Syntax

Syntax	Purpose
:OPERID=EDM_PROFILE	EDM operation ID for creating an EDM Profile operation.
:OPERID=EDM_SKIM	EDM operation ID for creating an EDM Skimcut operation. This operid was added when operations were added to the ProCAM 2d EDM system.
:OPERID=EDM_CORE	EDM operation ID for creating an EDM core removal operation. This operid was added when operations were added to the ProCAM 2d EDM system.
:OPERID=EDM_MACRO	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=CALC_SLOWDOWN_SPEED
: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 looks 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_CHANGE_MILL_SPEED

Purpose

This section will be called by CAMWorks automatically when you are in Volumill.

Comments

Supported in CAMWorks 2011 and later. Not supported in any ProCAM product.

Available in Mill, and Mill-Turn.

Example

```
*----------------------------------
:SECTION=CALC_CHANGE_MILL_SPEED
:C: IF SECTIONEXIST(CHANGE_MILL_SPEED) THEN CALL(CHANGE_MILL_SPEED)
ENDIF
*----------------------------------
```

CALC_OUTPUT_POLAR_SPINDLE_ORIENTATION

Purpose

This section will be called by CAMWorks automatically.

Syntax

To activate this option set post header as
:MILL_FACE_POLAR=X+_ONLY.

This will work for Face Polar and Fixed only.
SPINDLE_ORIENTATION post variable will have the value of rotation to get the feature to C zero.

Comments

Supported in CAMWorks 2013 and later. Not supported in any ProCAM product.

Available in Mill-Turn.

Example

```
*-----------------------------------
:SECTION=CALC_OUTPUT_POLAR_SPINDLE_ORIENTATION
:C: IF SECTIONEXIST(OUTPUT_POLAR_SPINDLE_ORIENTATION) THEN
:C: CALL(OUTPUT_POLAR_SPINDLE_ORIENTATION)
:C: ENDIF
*-----------------------------------
```

CALC_OUTPUT_CANCEL_POLAR_SPINDLE_ ORIENTATION

Purpose

This section will be called by CAMWorks automatically. Available in Mill-Turn.

Syntax

To activate this option set post header as
:MILL_FACE_POLAR=X+_ONLY.

Comments

This will work for Face Polar and Fixed only.
Supported in CAMWorks 2013 and later. Not supported in any ProCAM product.

Example

```
*-----------------------------------
:SECTION=CALC_OUTPUT_CANCEL_POLAR_SPINDLE_ORIENTATION
:C: IF SECTIONEXIST(OUTPUT_CANCEL_POLAR_SPINDLE_ORIENTATION) THEN
:C: CALL(OUTPUT_CANCEL_POLAR_SPINDLE_ORIENTATION)
:C: ENDIF
*-----------------------------------
```

CALC_OUTPUT_SPINDLE_STATE

Purpose

This section will be called by CAMWorks automatically when using a sub spindle operation and you insert a Spindle On/Off/Lock step.

Syntax

Post variable `DSPINDLE` will be available with values of either, 1= MAIN_SPINDLE, 2= SUB_SPINDLE
Post variable `SPINDLE_STATE` will be available with values of either:
0 = OFF, 1 =ON, 2 = LOCK

Comments

Supported in CAMWorks 2013 and later. Not supported in any ProCAM product.

Available in Mill-Turn and Turn.

Example

```
*-----------------------------------
:SECTION=CALC_OUTPUT_SPINDLE_STATE
:C: IF SECTIONEXIST(OUTPUT_SPINDLE_STATE) THEN
:C: CALL(OUTPUT_SPINDLE_STATE)
:C: ENDIF
*-----------------------------------
```

CALC_OUTPUT_SPINDLE_ORIENTATION

Purpose

This section will be called by CAMWorks automatically when using a sub spindle operation and you insert a Spindle Orient step.

Syntax

Post variable SPINDLE_ORIENTATION will be available with an Orientation angle.

Comments

Supported in CAMWorks 2013 and later. Not supported in any ProCAM product.

Available in Mill-Turn and Turn.

Example

```
*-----------------------------------
:SECTION=CALC_OUTPUT_SPINDLE_ORIENTATION
:C: IF SECTIONEXIST(OUTPUT_SPINDLE_ORIENTATION) THEN
:C: CALL(OUTPUT_SPINDLE_ORIENTATION)
:C: ENDIF
*-----------------------------------
```

CALC_OUTPUT_SPINDLE_COMMENT

Purpose

This section will be called by CAMWorks automatically when using a sub spindle operation and you insert a Text/with Comment selected step.

Syntax

Post variable SPINDLE_COMMENT will be available with a comment string.
This will output text as a comment "N1 (M05)".

Comments

Supported in CAMWorks 2013 and later. Not supported in any ProCAM product.

Available in Mill-Turn and Turn.

Example

```
*----------------------------------
:SECTION=CALC_OUTPUT_SPINDLE_COMMENT
:C: IF SECTIONEXIST(OUTPUT_SPINDLE_COMMENT) THEN
:C: CALL(OUTPUT_SPINDLE_COMMENT)
:C: ENDIF
*----------------------------------
```

CALC_OUTPUT_SPINDLE_COMMAND

Purpose

This section will be called by CAMWorks automatically when using a sub spindle operation and you insert a Text/with No Comment selected step.

Syntax

Post variable SPINDLE_COMMENT will be available with a comment string.
This will output raw text (hardcoded) output "N1 M05".

Comments

Supported in CAMWorks 2013 and later. Not supported in any ProCAM product.

Available in Mill-Turn and Turn.

Example

```
*-----------------------------------
:SECTION=CALC_OUTPUT_SPINDLE_COMMAND
:C: IF SECTIONEXIST(OUTPUT_SPINDLE_COMMAND) THEN
:C: CALL(OUTPUT_SPINDLE_COMMAND)
:C: ENDIF
*-----------------------------------
```

CALC_OUTPUT_CHUCK_OPEN

Purpose

This section will be called by CAMWorks automatically when using a sub spindle operation and you insert a Spindle Unclamp step.

Syntax

Post variable DSPINDLE will be available with values of either, 1= MAIN_SPINDLE, 2= SUB_SPINDLE

Comments

Supported in CAMWorks 2013 and later. Not supported in any ProCAM product.

Available in Mill-Turn and Turn.

Example

```
*-----------------------------------
:SECTION=CALC_OUTPUT_CHUCK_OPEN
:C: IF SECTIONEXIST(OUTPUT_CHUCK_OPEN) THEN
:C: CALL(OUTPUT_CHUCK_OPEN)
:C: ENDIF
*-----------------------------------
```

CALC_OUTPUT_CHUCK_CLOSE

Purpose

This section will be called by CAMWorks automatically when using a sub spindle operation and you insert a Spindle Clamp step.

Syntax

Post variable `DSPINDLE` will be available with values of either, 1= MAIN_SPINDLE, 2= SUB_SPINDLE

Comments

Supported in CAMWorks 2013 and later. Not supported in any ProCAM product.

Available in Mill-Turn and Turn.

Example

```
*-----------------------------------
:SECTION=CALC_OUTPUT_CHUCK_CLOSE
:C: IF SECTIONEXIST(OUTPUT_CHUCK_CLOSE) THEN
:C: CALL(OUTPUT_CHUCK_CLOSE)
:C: ENDIF
*-----------------------------------
```

CALC_OUTPUT_SPEED

Purpose

This section will be called by CAMWorks automatically when using a sub spindle operation and you insert a Spindle Speed step.

Syntax

Post variable DSPINDLE will be available with values of either, 1= MAIN_SPINDLE, 2= SUB_SPINDLE
Post variable OPR_SPEED_DIR will be available with values of either, 1= CW, 2= CCW
Post variable OPR_SPEED_RPM will be available with RPM value.

Comments

Supported in CAMWorks 2013 and later. Not supported in any ProCAM product.

Available in Mill-Turn and Turn.

Example

```
*----------------------------------
:SECTION=OUTPUT_SPEED
:T: IF DSPINDLE=MAIN_SPINDLE AND OPR_SPEED_DIR=1 THEN
<N><G!:SPEED_TYPE><S!:OPR_SPEED_RPM><M!:MC(M_SPIN_CW)><EOL>ENDIF
:T: IF DSPINDLE=MAIN_SPINDLE AND OPR_SPEED_DIR<>1 THEN
<N><G!:SPEED_TYPE><S!:OPR_SPEED_RPM><M!:MC(M_SPIN_CCW)><EOL>ENDIF
:T: IF DSPINDLE=SUB_SPINDLE AND OPR_SPEED_DIR=1 THEN
<N><G!:SPEED_TYPE><S!:OPR_SPEED_RPM><M!:MC(M_SUB_CW)><EOL>ENDIF
:T: IF DSPINDLE=SUB_SPINDLE AND OPR_SPEED_DIR<>1 THEN
<N><G!:SPEED_TYPE><S!:OPR_SPEED_RPM><M!:MC(M_SUB_CCW)><EOL>ENDIF
*----------------------------------
```

CALC_OUTPUT_RAPID_SPINDLE_TO_POSITION

Purpose

This section will be called by CAMWorks automatically when using a sub spindle operation and you insert a Spindle Rapid step.

Syntax

Post variable `DSPINDLE` will be available with values of either, 1= MAIN_SPINDLE, 2= SUB_SPINDLE
Post variable ABS_SPINDLE_Z_END will be available with Decimal value.

Comments

Supported in CAMWorks 2013 and later. Not supported in any ProCAM product.

Available in Mill-Turn and Turn.

Example

```
*-----------------------------------
:SECTION=CALC_OUTPUT_RAPID_SPINDLE_TO_POSITION
:C: IF SECTIONEXIST(OUTPUT_RAPID_SPINDLE_TO_POSITION) THEN
:C: CALL(OUTPUT_RAPID_SPINDLE_TO_POSITION)
:C: ENDIF
*-----------------------------------
```

CALC_OUTPUT_FEED_SPINDLE_TO_POSITION

Purpose

This section will be called by CAMWorks automatically when using a sub spindle operation and you insert a Spindle Feed step.

Syntax

Post variable DSPINDLE will be available with values of either, 1= MAIN_SPINDLE, 2= SUB_SPINDLE
Post variable OPR_FEED_FPM will be available with feedrate value.
Post variable ABS_SPINDLE_Z_END will be available with Decimal value.

Comments

Supported in CAMWorks 2013 and later. Not supported in any ProCAM product.

Available in Mill-Turn and Turn.

Example

```
*-----------------------------------
:SECTION=CALC_OUTPUT_FEED_SPINDLE_TO_POSITION
:C: IF SECTIONEXIST(OUTPUT_FEED_SPINDLE_TO_POSITION) THEN
:C: OPR_FEED_TYPE=2
:C: FEED_TYPE=GC(G_FPM)
:C: CALL(OUTPUT_FEED_SPINDLE_TO_POSITION)
:C: ENDIF
*-----------------------------------
```

CALC_OUTPUT_SPINDLE_SYNC

Purpose

This section will be called by CAMWorks automatically when using a Sub Spindle operation and you insert a Spindle Feed step.

Syntax

Post variable DSPINDLE will be available with values of either, 1= MAIN_SPINDLE, 2= SUB_SPINDLE
Post variable SPINDLE_SYNC_TYPE will be available with values of either, 0= SPEED, 1= PHASE

Comments

Supported in CAMWorks 2013 and later. Not supported in any ProCAM product.

Available in Mill-Turn and Turn.

Example

```
*-----------------------------------
:C: SECTION=CALC_OUTPUT_SPINDLE_SYNC
:C: IF SECTIONEXIST(OUTPUT_SPINDLE_SYNC) THEN
:C: CALL(OUTPUT_SPINDLE_SYNC)
:C: ENDIF
*-----------------------------------
```

CALC_OUTPUT_REFERENCE_POINT

Purpose

Reserved for future use in Sub Spindle operation.

Comments

Available in Mill-Turn and Turn.

Example

```
*-----------------------------------
:SECTION=CALC_OUTPUT_REFERENCE_POINT
:C: IF SECTIONEXIST(OUTPUT_REFERENCE_POINT) THEN
:C: CALL(OUTPUT_REFERENCE_POINT)
:C: ENDIF
*-----------------------------------
```

CALC_OUTPUT_ARBOR_OPEN

Purpose

Reserved for future use in Sub Spindle operation.

Syntax

Post variable DSPINDLE will be available with values of either, 1= MAIN_SPINDLE, 2= SUB_SPINDLE

Example

```
*-----------------------------------
:SECTION=CALC_OUTPUT_ARBOR_OPEN
:C: IF SECTIONEXIST(OUTPUT_ARBOR_OPEN) THEN
:C: CALL(OUTPUT_ARBOR_OPEN)
:C: ENDIF
*-----------------------------------
```

CALC_OUTPUT_ARBOR_CLOSE

Purpose

Reserved for future use in Sub Spindle operation.

Syntax

Post variable DSPINDLE will be available with values of either, 1= MAIN_SPINDLE, 2= SUB_SPINDLE

Example

```
*-----------------------------------
:SECTION=CALC_OUTPUT_ARBOR_CLOSE
:C: IF SECTIONEXIST(OUTPUT_ARBOR_CLOSE) THEN
:C: CALL(OUTPUT_ARBOR_CLOSE)
:C: ENDIF
*-----------------------------------
```

CALC_OUTPUT_COLLET_OPEN

Purpose

Reserved for future use in Sub Spindle operation.

Syntax

Post variable DSPINDLE will be available with values of either, 1= MAIN_SPINDLE, 2= SUB_SPINDLE

Example

```
*-----------------------------------
:SECTION=CALC_OUTPUT_COLLET_OPEN
:C: IF SECTIONEXIST(OUTPUT_COLLET_OPEN) THEN
:C: CALL(OUTPUT_COLLET_OPEN)
:C: ENDIF
*-----------------------------------
```

CALC_OUTPUT_COLLET_CLOSE

Purpose

Reserved for future use in Sub Spindle operation.

Syntax

Post variable DSPINDLE will be available with values of either, 1= MAIN_SPINDLE, 2= SUB_SPINDLE

Example

```
*-----------------------------------
:SECTION=CALC_OUTPUT_COLLET_CLOSE
:C: IF SECTIONEXIST(OUTPUT_COLLET_CLOSE) THEN
:C: CALL(OUTPUT_COLLET_CLOSE)
:C: ENDIF
*-----------------------------------
```

CALC_OUTPUT_FEEDRATE

Purpose

Reserved for future use in Sub Spindle operation.

Syntax

Post variable OPR_FEED_FPM will be available with feedrate value.

Example

```
*-----------------------------------
:SECTION=CALC_OUTPUT_FEEDRATE
:C: IF SECTIONEXIST(OUTPUT_FEEDRATE) THEN
:C: CALL(OUTPUT_FEEDRATE)
:C: ENDIF
*-----------------------------------
```

CALC_QUERY_POST

Purpose

This is used in conjunction with CAMWorks Virtual Machine.

Comment

Supported in CAMWorks 2013 and later. Not supported in any ProCAM product. Available in Mill, Turn and Mill-Turn.

Example

```
:SECTION=CALC_QUERY_POST
:C:  IF QUERY_ITEM_ID = QUERY_INT_NUM_TURRETS_ABOVE THEN
QUERY_INT_VAL=1 RETURN ENDIF
:C:  IF QUERY_ITEM_ID = QUERY_INT_NUM_TURRETS_BELOW THEN
QUERY_INT_VAL=1 RETURN ENDIF
:C:  IF QUERY_ITEM_ID = QUERY_GCODE_DEFAULT_WCS THEN
:C:  IF SECTIONEXIST(QUERY_WCS) THEN
:C:  CALL(QUERY_WCS)
:C:  RETURN
:C:  ELSE
:C:  CALL(AUTO_QUERY_WCS)
:C:  RETURN
:C:  ENDIF
:C:  ENDIF
:C:  IF QUERY_ITEM_ID = QUERY_GCODE_WCS THEN
:C:  IF SECTIONEXIST(QUERY_WCS) THEN
:C:  CALL(QUERY_WCS)
:C:  RETURN
:C:  ELSE
:C:  CALL(AUTO_QUERY_WCS)
:C:  RETURN
:C:  ENDIF
:C:  ENDIF
:C:  IF QUERY_ITEM_ID = QUERY_INT_LENGTH_REG THEN
QUERY_INT_VAL=TOOL RETURN ENDIF
:C:  IF QUERY_ITEM_ID = QUERY_INT_OFFSET_REG THEN
CALL(CALC_QUERY_DIAM_OFFSET_REG)RETURN ENDIF
:C:  IF QUERY_ITEM_ID = QUERY_GCODE_AXIS_NAMES_T1A THEN
CALL(QUERY_AXIS_NAMES_T1A)RETURN ENDIF
:C:  IF QUERY_ITEM_ID = QUERY_GCODE_AXIS_NAMES_T2A THEN
CALL(QUERY_AXIS_NAMES_T2A)RETURN ENDIF
:C:  IF QUERY_ITEM_ID = QUERY_GCODE_AXIS_NAMES_T1B THEN
CALL(QUERY_AXIS_NAMES_T1B)RETURN ENDIF
:C:  IF QUERY_ITEM_ID = QUERY_GCODE_AXIS_NAMES_T2B THEN
CALL(QUERY_AXIS_NAMES_T2B)RETURN ENDIF
```

CALC_PRE_POST_INITIALIZE

Purpose

Can be used with all posts.

Syntax

```
:SECTION=CALC_PRE_POST_INITIALIZE
```

Comments

This section will get called when posting before any other section gets called. Can be used to set system or post variables before other main sections get called for initializing code. To be used in CAMWorks 2014 or newer version.

CALC_OUTPUT_RAPID_SPINDLE_TO_HOME

Purpose

Can be used with any turn or mill/turn posts that support sub spindle operations.

Syntax

```
:SECTION=CALC_OUTPUT_RAPID_SPINDLE_TO_HOME
```

Comments

This section will get called when you insert a sub spindle operation and you select “Sub Spindle Home” in a rapid step. To be used in CAMWorks 2014 or newer version.

CALC_OUTPUT_FEED_SPINDLE_TO_HOME

Purpose

Can be used with any turn or mill/turn posts that support sub spindle operations.

Syntax

```
:SECTION=CALC_OUTPUT_FEED_SPINDLE_TO_HOME
```

Comments

This section will get called when you insert a sub spindle operation and you select “Sub Spindle Home” in a feed step. To be used in CAMWorks 2014 or newer version.

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

Parameter	*Description*
`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.**

8. Unzip **Demo.zip** on any drive and in any folder.
9. 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

*-----------------------------------
:ATTRNAME=fld3
:ATTRTYPE=VALUE
:ATTRVTYPE=DECIMAL
:ATTREMARK=Feedrate
:ATTRSEL=N
:ATTRSHORT=Feedrate
:ATTRLONG=ENTER Feedrate
:ATTRHIGH=9999
:ATTRLOW=0
:ATTRDEFAULT=0
:ATTRUSED=1
:ATTREND
```

10. 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
```

11. 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
```

12. 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
```

13. 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
*-----------------------------------
```

14. 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



: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
```

15. 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
```

16. 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
*
```

17. 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)
```

18. 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)
```

19. Now we will close the database after the lookup.

```
:C: CLOSEDB(1)
*
*---------------------------------
```

20. 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>
*-----------------------------------
*
```

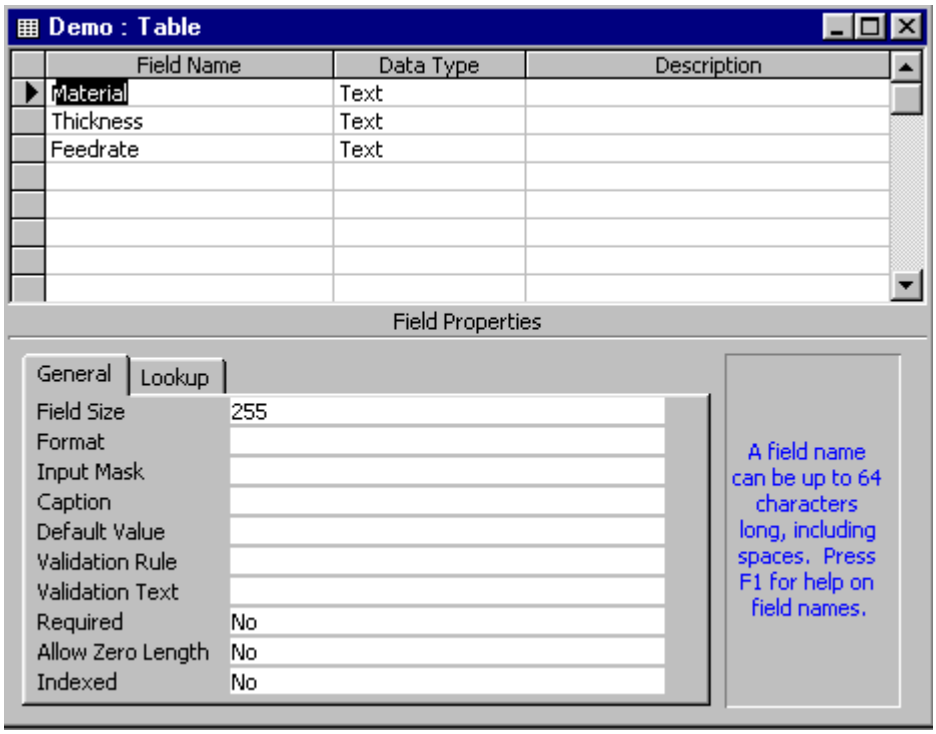
21. In the example below, you can set **fld1=material** and **fld2=thickness** because material and thickness are asked in the Setup info.

    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>
*
```

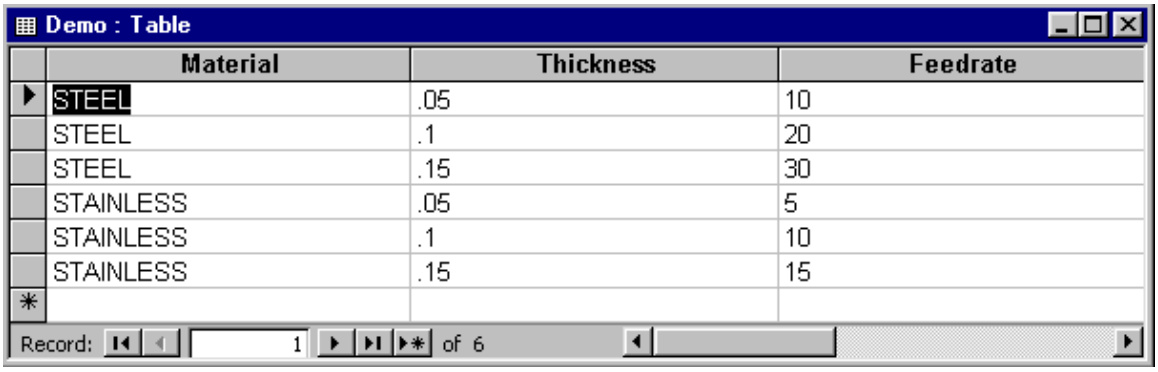
22. Open the **Demo.mdb** and select the **design** button.

    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.



23. 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.



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

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

CALC ARC MOVE MILL YZ

CALC ARC MOVE MILL ANYPLANE

CALC POST INITIALIZE

CALC TOOL INITIALIZ

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

CALC SLOWDOWN SPEED

CALC SHIFT TOOL LATHE

CALC CUTTER COMP LATHE

Plasma

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

CALC PROFILE DRILL PLASMA

Mill-Turn

CALC CHANGE MILL SPEED

CALC OUTPUT POLAR SPINDLE ORIENTATION

CALC OUTPUT CANCEL POLAR SPINDLE ORIENTATION

CALC OUTPUT SPINDLE STATE

CALC OUTPUT SPINDLE ORIENTATION

CALC_OUTPUT_SPINDLE_COMMENT

CALC OUTPUT SPINDLE COMMAND

CALC OUTPUT CHUCK OPEN

CALC OUTPUT CHUCK CLOSE

CALC OUTPUT ARBOR OPEN

CALC OUTPUT ARBOR CLOSE

CALC OUTPUT COLLET OPEN

CALC_OUTPUT_COLLET_CLOSE

CALC OUTPUT FEEDRATE

CALC OUTPUT REFERENCE POINT

CALC OUTPUT SPINDLE SYNC

CALC OUTPUT RAPID SPINDLE TO POSITION

CALC OUTPUT SPEED

CALC OUTPUT FEED SPINDLE TO POSITION

CALC QUERY POST

CALC OUTPUT RAPID SPINDLE TO HOME

CALC OUTPUT FEED SPINDLE TO HOME

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

CALC_MACRO_CALL_PUNCH

CALC_MULTIPLE_MACRO_DEFINE_PUNCH

CALC_SETUP_SHEET_PUNCH

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

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

CALC PROFILE DRILL LASER

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

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
