
PA

Path Acceleration

Type	Path Contouring or Motion (Linear Interpolated)	Product	Rev
Syntax	<!>PA<r>	6K	5.0
Units	r = units/sec/sec (scalable by SCLD)		
Range	0.00001-39,999,998 (depending on the scaling factor)		
Default	10.0000		
Response	PA: *PA10.0000		
See Also	GOL, PAA, PAD, PADA, SCLD, SCALE		

The Path Acceleration (PA) command specifies the path acceleration to be used with linearly interpolated moves (GOL), and all contouring moves (PLIN, PARCM, PARCOM, PARCOP, PARCP). For both the linear interpolated and the contouring moves, the path acceleration refers to the acceleration experienced by the load as motion gains speed along the path. For linearly interpolated moves, the acceleration of each individual axis is dependent on the distance it contributes to the total path traveled by the load. In contouring paths, the acceleration of each individual axis is dependent on the direction of travel in the X-Y plane. **NOTE:** *The PA value can be altered between path segments, but not within a path segment.*

Contouring and linear interpolation are discussed in detail in the Custom Profiling chapter of the *Programmer's Guide*.

UNITS OF MEASURE and SCALING: refer to page 16.
--

The path acceleration remains set until you change it with a subsequent path acceleration command. Accelerations outside the valid range are flagged as an error, with a message *INVALID DATA-FIELD x, where x is the field number. When an invalid acceleration is entered the previous acceleration value is retained.

If the path deceleration (PAD) command has not been entered, the path acceleration (PA) command will set the path deceleration rate. Once the path deceleration (PAD) command has been entered, the path acceleration (PA) command no longer affects path deceleration.

Example:

```
PV5           ; Set path velocity to 5 units/sec
PA50          ; Set path acceleration to 50 units/sec/sec
PAD100        ; Set path deceleration to 100 units/sec/sec
DEF prog1     ; Begin definition of path named prog1
PAXES1,2      ; Set axes 1 and 2 as the X and Y contouring axes
PAB0          ; Set to incremental coordinates
PLIN1,1       ; Specify X-Y endpoint position to create a 45 degree
               ; angle line segment
END           ; End definition of path prog1
PCOMP prog1   ; Compile path prog1
PRUN prog1    ; Execute path prog1
```

PAA

Path Average Acceleration

Type	Motion (S-Curve); Motion (Linear Interpolated)	Product	Rev
Syntax	<!>PAA<r>	6K	5.0
Units	r = units/sec/sec (scalable by SCLD)		
Range	0.00001-39,999,998 (depending on the scaling factor)		
Default	10.00 (trapezoidal profiling is default, where PAA tracks PA)		
Response	PAA: *PAA10.0000		
See Also	DRES, PA, PAD, PADA, SCLD, SCALE		

The Path Average Acceleration (PAA) command allows you to specify the average acceleration for an S-curve path profile. S-curve profiling provides smoother motion control by reducing the rate of change in acceleration and deceleration; this accel/decel rate of change is known as *jerk* . S-curve profiling improves position tracking performance in linear interpolation applications (not contouring). S-curve profiling is not available for contouring applications. Refer to page 13 for details on S-curve profiling.

NOTE: Path scaling (SCLD) affects PAA the same as it does for PA. Refer to page 16 for details on scaling.

Example:

```
PV5           ; Set path velocity to 5 units/sec
PA50          ; Set path acceleration to 50 units/sec/sec
PAA40         ; Set path s-curve (average) acceleration to 40 units/sec/sec
PAD100        ; Set path deceleration to 100 units/sec/sec
PADA70        ; Set path s-curve (average) deceleration to 70 units/sec/sec
DEF prog1     ; Begin definition of path named prog1
D10,5,2,11    ; Set distance values, axes 1-4
GOL1111      ; Initiate linear interpolation motion
END           ; End definition of path prog1
```

PAB Path Absolute

Type	Path Contouring	Product	Rev
Syntax	<!*>PAB	6K	5.0
Units	n/a		
Range	b = 0 (incremental) or 1 (absolute)		
Default	0		
Response	No response - Must be defining a path (DEF)		
See Also	PL, PLC, SCLD, PWC, SCALE		

The Path Absolute (PAB) command is used to indicate whether the subsequent segment endpoints are specified in either incremental (Ø) or absolute (1) coordinates. Segment endpoint position specifications may be either absolute with respect to the user-defined coordinate system, or incremental, relative to the start of each individual segment. At any point along a path definition, coordinates may be switched from incremental to absolute.

The absolute coordinate system may be either the *work* coordinate system or the *local* coordinate system (see PL).

PAD Path Deceleration

Type	Path Contouring or Motion (Linear Interpolated)	Product	Rev
Syntax	<!*>PAD<r>	6K	5.0
Units	r = units/sec/sec (scalable by SCLD)		
Range	0.00001-39,999,998 (depending on the scaling factor)		
Default	10.0000 (PAD tracks PA)		
Response	PAD: *PAD10.0000		
See Also	GOL, PA, PAA, PADA, SCLD, SCALE		

The Path Deceleration (PAD) command specifies the path deceleration to be used with linearly interpolated moves (GOL), and all contouring moves (PLIN, PARCM, PARCOM, PARCOP, PARCP). For both the linear interpolated and the contouring moves, the path deceleration refers to the deceleration experienced by the load as motion slows along the path. For linearly interpolated moves, the deceleration of each individual axis is dependent on the distance it contributes to the total path traveled by the load. In contouring paths, the deceleration of each individual axis is dependent on the direction of travel in the X-Y plane.

UNITS OF MEASURE and SCALING: refer to page 16.

The path deceleration remains set until you change it with a subsequent path deceleration command. Decelerations outside the valid range are flagged as an error, with a message *INVALID DATA-FIELD x, where x is the field number. When an invalid deceleration is entered the previous deceleration value is retained.

If the path deceleration (PAD) command has not been entered, the path acceleration (PA) command will set the path deceleration rate. Once the path deceleration (PAD) command has been entered, the path acceleration (PA) command no longer affects path deceleration. If PAD is set to zero (PADØ), then the path deceleration will once again track whatever the PA command is set to.

Example: Refer to the path acceleration (PA) command example.

PADA Path Average Deceleration

Type	Motion (S-Curve); Motion (Linear Interpolated)	Product	Rev
Syntax	<!>PADA<r>	6K	5.0
Units	r = units/sec/sec (scalable by SCLD)		
Range	0.00001-39,999,998 (depending on the scaling factor)		
Default	10.00 (PADA tracks PAA)		
Response	PADA: *PADA10.0000		
See Also	DRES, PA, PAA, PAD, SCLD, SCALE		

Use the Path Average Deceleration (PADA) command to specify the average deceleration for an S-curve path profile. S-curve profiling provides smoother motion control by reducing the rate of change in acceleration and deceleration; this accel/decel rate of change is known as *jerk*. S-curve profiling can improve position tracking performance in linear interpolation applications (not contouring). S-curve profiling is not available for contouring applications. Refer to page 13 for details on S-curve profiling.

NOTE: Path scaling (SCLD) affects PADA the same as it does for PAD. Refer to page 16 for details on scaling.

Example: Refer to the path average acceleration (PAA) command example.

[PANI] Position of ANI Inputs

Type	Assignment or comparison	Product	Rev
Syntax	See below	6K	5.0
Units	n/a		
Range	n/a		
Default	n/a		
Response	n/a		
See Also	[ANI], ANIRNG, [FB], [CA], CMDDIR, PSET, SCALE, SCLD, SFB, TANI, TPANI, TFB		

This command is available only to servo axes, and only if you have an analog input SIM in an extended I/O brick.

The PANI operator is used to assign the analog input's position information to a variable, or to make a comparison against another value. The PANI value represents the analog input position after the affects of distance scaling (SCLD), offset (PSET), and commanded direction polarity (CMDDIR).

The TPANI and PANI commands are designed for applications in which analog input is scaled and/or used as position feedback. If you are using analog input to monitor an analog signal, the TANI and ANI commands would be more appropriate (TANI and ANI values are measured in volts and are unaffected by scaling, offset, or command direction polarity).

The PANI value is represented in analog-to-digital converter (ADC) units if scaling is disabled (SCALE0). The ADC has a 12-bit resolution, giving a range of +2047 to -2048 counts when using the full $\pm 10V$ range of the analog input (205 counts/volt). If scaling is enabled (SCALE1), an SCLD scale factor of 205 (the default value when analog input feedback is selected) allows units of volts to be used.

NOTE: If you change the voltage range of the analog input (with the ANIRNG command), the resolution of the PANI response will change accordingly. The default is $\pm 10V$ (+2047 to -2048 counts).

Syntax: VARn=PANI.i where "n" is the variable number, "" is the number of the I/O brick, and "i" is I/O brick address where the analog input resides; or PANI can be used in an expression such as IF(1PANI.10=2.3). If no brick identifier () is provided, it defaults to 1. To understand the I/O brick addressing convention, refer to page 6.

Example:

```

SCLD205          ; Set distance scaling to accommodate values in volts
                 ; (205 counts/volt)
SCALE1          ; Enable scaling
DEL proge        ; Delete program called proge
DEF proge        ; Begin definition of program called proge
VAR4=1PANI.10    ; Position of the 2nd analog input on SIM2 of I/O brick 1
                 ; is assigned to variable 4
IF(1PANI.9<8.2) ; If position of 1st analog input on SIM2 of I/O brick 1
                 ; is < 8.2 volts, do the commands between IF and NIF
TREV            ; Transfer revision level
NIF             ; End if statement
END             ; End definition of proge

```

PARCM Radius Specified CCW Arc Segment

Type	Path Contouring	Product	Rev
Syntax	<!>PARCM<r>,<r>,<r>	6K	5.0
Units	r = units (scalable by the SCLD value)		
Range	0.00000 - ±999,999,999		
Default	n/a		
Response	No response - Must be defining a path (DEF)		
See Also	PARCP, PARCOM, PARCOP, PRTOL, SCLD, SCALE		

The Radius Specified CCW Arc Segment (PARCM) command is used to specify the endpoints and the radius of a counter-clockwise arc segment. The placement, length, radius of curvature, and orientation of the arc are completely specified by the endpoint and radius specifications of the arc segment and the endpoint of the previous segment (current position). The direction of rotation in the X-Y plane will be counter-clockwise.

A complete circle cannot be specified with a PARCM command, because the center is arbitrary. Use the PARCOM command for circles.

Command Syntax: PARCM<Xend>,<Yend>,<Radius>

Segment endpoint position specifications may be either absolute (PAB1) with respect to user defined segment start coordinates, or incremental (PABØ), relative to the start of each individual segment. The first two numbers following the PARCM command specify the X endpoint and the Y endpoint, respectively.

Radius specifications are signed values. A positive radius specifies an arc which is 180 degrees or less. A negative radius specifies an arc which is 180 degrees or more. The last number of the PARCM command specifies the radius.

UNITS OF MEASURE and SCALING: refer to page 16 or to the SCLD description.

Example

```

PV5              ; Set path velocity to 5 units/sec
PA50             ; Set path acceleration to 50 units/sec/sec
PAD100          ; Set path deceleration to 100 units/sec/sec
PSET0,0         ; Set absolute position to 0,0
DEF prog1       ; Begin definition of path named prog1
PAXES1,2        ; Set axes 1 and 2 as the X and Y contouring axes
PAB0            ; Set to incremental coordinates
POUT1001        ; Output pattern during first arc: onboard outputs 1 & 4 are
                 ; on and outputs 2 & 3 are off
PARCM5,5,5      ; Specify incremental X-Y endpoint position and radius arc
                 ; <180 degrees for 1/4 circle counter-clockwise arc
POUT1100        ; Output pattern during second arc: onboard outputs 1 & 2 are
                 ; on and outputs 3 & 4 are off
PARCP5,-5,-5    ; Specify incremental X-Y endpoint position and radius arc
                 ; >180 degrees for 3/4 circle clockwise arc
END             ; End definition of path prog1
PCOMP prog1     ; Compile path prog1
PRUN prog1      ; Execute path prog1
OUT0000         ; Turn off the first four onboard outputs

```

PARCOM Origin Specified CCW Arc Segment

Type	Path Contouring	Product	Rev
Syntax	<!>PARCOM<r>,<r>,<r>,<r>	6K	5.0
Units	r = units (scalable with the SCLD value)		
Range	0.00000 - ±999,999,999		
Default	n/a		
Response	No response - Must be defining a path (DEF)		
See Also	PARCOP, PARCM, PARCP, PRTOL, SCLD, SCALE		

The Origin Specified CCW Arc Segment (PARCOM) command is used to specify the coordinates necessary to create a counter-clockwise arc segment. The placement, length, radius of curvature, and orientation of the arc are completely specified by the endpoint and center specifications of the arc segment and the endpoint of the previous segment (current position). The direction of rotation in the X-Y plane will be counter-clockwise.

Command Syntax: PARCOM<Xend>,<Yend>,<Xcenter>,<Ycenter>

Segment endpoint position specifications may be either absolute (PAB1) with respect to user defined segment start coordinates, or incremental (PAB0), relative to the start of each individual segment. The first two numbers following the PARCOM command specify the X endpoint and the Y endpoint, respectively.

Center position specifications are always incremental, relative to the start of the arc segment. The last two numbers following the PARCOM command specify the X center point and Y center point coordinates, respectively.

UNITS OF MEASURE and SCALING: refer to page 16 or to the SCLD description.

Example:

```
PV5                   ; Set path velocity to 5 units/sec
PA50                  ; Set path acceleration to 50 units/sec/sec
PAD100                ; Set path deceleration to 100 units/sec/sec
PSET0,0               ; Set absolute position to 0,0
DEF prog1             ; Begin definition of path named prog1
PAXES1,2              ; Set axes 1 and 2 as the X and Y contouring axes
PAB0                  ; Set to incremental coordinates
POUT1001              ; Output pattern during first arc: onboard outputs 1 & 4 are
                      ; on and outputs 2 & 3 are off
PARCOM5,5,0,5         ; Specify incremental X-Y endpoint position and X-Y center
                      ; position for quarter circle counter-clockwise arc
POUT1100              ; Output pattern during second arc: onboard outputs 1 & 2 are
                      ; on and outputs 3 & 4 are off
PARCOP0,0,5,0         ; Specify incremental X-Y endpoint position and X-Y center
                      ; position for full circle clockwise arc
END                   ; End definition of path prog1
PCOMP prog1           ; Compile path prog1
PRUN prog1            ; Execute path prog1
OUT0000               ; Turn off the first four onboard outputs
```

PARCOP Origin Specified CW Arc Segment

Type	Path Contouring	Product	Rev
Syntax	<!>PARCOP<r>,<r>,<r>,<r>	6K	5.0
Units	r = units (scalable by the SCLD value)		
Range	0.00000 - ±999,999,999		
Default	n/a		
Response	No response - Must be defining a path (DEF)		
See Also	PARCOM, PARCM, PARCP, PRTOL, SCLD, SCALE		

The Origin Specified CW Arc Segment (PARCOP) command is used to specify the coordinates necessary to create a clockwise arc segment. The placement, length, radius of curvature, and orientation of the arc are completely specified by the endpoint and center specifications of the arc segment and the endpoint of the previous segment (current position). The direction of rotation in the X-Y plane will be clockwise.

Command Syntax: PARCOP<Xend>,<Yend>,<Xcenter>,<Ycenter>

Segment endpoint position specifications may be either absolute (PAB1) with respect to user defined segment start coordinates, or incremental (PABØ), relative to the start of each individual segment. The first two numbers following the PARCOP command specify the X endpoint and the Y endpoint, respectively.

Center position specifications are always incremental, relative to the start of the arc segment. The last two numbers following the PARCOP command specify the X center point and Y center point coordinates, respectively.

UNITS OF MEASURE and SCALING: refer to page 16 or to the SCLD description.

Example: Refer to the PARCOM command example.

PARCP Radius Specified CW Arc Segment

Type	Path Contouring	Product	Rev
Syntax	<!>PARCP<r>,<r>,<r>	6K	5.0
Units	r = units (scalable by the SCLD value)		
Range	0.00000 - ±999,999,999		
Default	n/a		
Response	No response - Must be defining a path (DEF)		
See Also	PARCM, PARCOM, PARCOP, PRTOL, SCLD, SCALE		

The Radius Specified CW Arc Segment (PARCP) command is used to specify the endpoints and the radius of a clockwise arc segment. The placement, length, radius of curvature, and orientation of the arc are completely specified by the endpoint and radius specifications of the arc segment and the endpoint of the previous segment (current position). The direction of rotation in the X-Y plane will be clockwise.

A complete circle cannot be specified with a PARCP command, because the center is arbitrary. Use the PARCOP command for circles.

Command Syntax: PARCP<Xend>,<Yend>,<Radius>

Segment endpoint position specifications may be either absolute (PAB1)with respect to user defined segment start coordinates, or incremental (PABØ), relative to the start of each individual segment. The first two numbers following the PARCP command specify the X endpoint and the Y endpoint, respectively.

Radius specifications are signed values. A positive radius specifies an arc which is 180 degrees or less. A negative radius specifies an arc which is 180 degrees or more. The last number of the PARCP command specifies the radius.

UNITS OF MEASURE and SCALING: refer to page 16 or to the SCLD description.

Example: Refer to the PARCM command example.

PAXES Set Contouring Axes

Type	Path Contouring	Product	Rev
Syntax	<!>PAXES<i>,<i>,<i>,<i>	6K	5.0
Units	Each <i>: X axis, Y axis, Tangent axis, Proportional axis		
Range	i = 1-8 (product dependent)		
Default	1,2,0,0		
Response	No response - Must be defining a path (DEF)		
See Also	DEF, DRES, END, ERES, PCOMP, PPRO, PRUN, SCLD, TSKAX		

The Set Contouring Axes (PAXES) command defines the axes to be used in the current path definition (syntax: PAXES<Xaxis>,<Yaxis>,<Tangent>,<Proportional>). The X and Y axes must be specified, but the Tangent and Proportional axes are optional.

If no axis number is specified for the Tangent or Proportional axes, it signifies that the Tangent or Proportional axes are not included in that path definition. The axis specification for the entire path is done with this command. The PAXES command should be given prior to any contour segments.

NOTES

- For products that control only 2 axes of motion, the Tangent and Proportional axes are not available.
- When using scaling (SCALE1), the units used for path distance, acceleration, and velocity is determined by the SCLD value. For example, suppose you have 2 servo axes (axes 1 & 2) involved in contouring, both axes use encoder feedback with a resolution of 4000 counts/rev, axis 1 uses a 10:1 (10 turns per inch) leadscrew and axis 2 uses a 5:1 (5 turns per inch) lead screw, and you want to program in inches. For this application you would use the SCLD40000,20000 command to establish path motion units in inches: distance is inches, acceleration is inches/sec/sec, and velocity is inches/sec.
- When not using scaling (SCALE0), path motion units are based on the resolution (DRES for steppers, ERES for servos) of axis 1. If multi-tasking is used, path motion units are based on the resolution of the first (lowest number) axis associated with the task (TSKAX).

Example: (see PCOMP)

[PC] Position Commanded

Type	Assignment or Comparison	Product	Rev
Syntax	See below	6K	5.0
Units	n/a		
Range	n/a		
Default	n/a		
Response	n/a		
See Also	ERES, [FB], GOWHEN, [PCC], [PE], [PER], PSET, SCALE, SCLD, SMPER, TAS, TFB, TPC, TPCC, TPE, TPER		

Use PC operator to assign the current *commanded position* (scalable by SCLD) of each axis to a variable, or to make a comparison against another value. If you issue a PSET command, the commanded position value will be offset by the PSET command value.

Servo Axes: The PC value is measured in encoder or analog input (ANI) counts. The commanded position (PC) and the actual position (FB) are used in the control algorithm to calculate the position error (PC - FB = PER) and thereby determine the corrective control signal.

Stepper Axes: The PC value is measured in commanded counts (“motor counts”).

UNITS OF MEASURE and SCALING: refer to page 16.

Syntax: VARn=aPC where “n” is the variable number, and “a” is the axis, or PC can be used in an expression such as IF(1PC>50) . The PC command must be used with an axis specifier or it will default to axis 1 (e.g., 1PC, 2PC, etc.).

Example:

```
VAR1=1PC           ; Commanded position for axis 1 is assigned to variable 1
IF(2PC<50)        ; If the commanded position for axis 2 is <50, do the IF statement
VAR2=2PC+500     ; Commanded position for axis 2 plus 500 is assigned to variable 2
NIF               ; End IF statement
```

[PCC] Captured Commanded Position

Type	Assignment or Comparison	Product	Rev
Syntax	See below	6K	5.0
Units	n/a		
Range	n/a		
Default	n/a		
Response	n/a		
See Also	CMDDIR, ENCCNT, INFNC, [PC], [PCMS], PSET, SCALE, SCLD, SFB, [TRIG], TRGLOT, TPC, TPCC, TTRIG		

Use the PCC operator to assign the captured commanded position of a specific axis to a variable, or to make a comparison against another value.

Syntax: VARn=aPCCc where “n” is the variable number, “a” is the axis, and “c” designates trigger A or B for the axis, or M for the **MASTER TRIG** input (see table below); or PCC can be used in an expression such as IF(1PCCB>23450). The PCC operator must be used with an axis specifier or it will default to axis 1 (e.g., 1PCCA, 2PCCB, 5PCCM, etc.).

Trigger Input (Axis 1-4 “TRIGGERS/OUTPUTS” connector) * Axis			Dedicated PCC Syntax	Trigger Input (Axis 5-8 “TRIGGERS/OUTPUTS” connector) * Axis			Dedicated PCC Syntax
Pin 23,	Trigger 1A	1	1PCCA	Pin 23,	Trigger 5A	5	5PCCA
Pin 21,	Trigger 1B	1	1PCCB	Pin 21,	Trigger 5B	5	5PCCB
Pin 19,	Trigger 2A	2	2PCCA	Pin 19,	Trigger 6A	6	6PCCA
Pin 17,	Trigger 2B	2	2PCCB	Pin 17,	Trigger 6B	6	6PCCB
Pin 15,	Trigger 3A	3	3PCCA	Pin 15,	Trigger 7A	7	7PCCA
Pin 13,	Trigger 3B	3	3PCCB	Pin 13,	Trigger 7B	7	7PCCB
Pin 11,	Trigger 4A	4	4PCCA	Pin 11,	Trigger 8A	8	8PCCA
Pin 9,	Trigger 4B	4	4PCCB	Pin 9,	Trigger 8B	8	8PCCB

* The number of trigger inputs available varies by product (refer to your product's *Installation Guide*).

To use an axis position captured with the MASTER TRIG input, use aPCCM, where “a” can be any axis number.

About Position Capture: The commanded position can be captured only by a trigger input that is defined as “trigger interrupt” input with the INFNCi-H command (see INFNC for details). Each trigger input, when configured as a “trigger interrupt” input, is dedicated to capture the position of a specific axis (see table above). When a “trigger interrupt” input is activated, the commanded position of the dedicated axis is captured and the position is available through the use of the PCC operator and the TPCC display command.

Note for Stepper Axes: By default, stepper axes capture only the commanded position. However, if the axis has Encoder Capture Mode enabled with the ENCCNT command, only the encoder position is captured.

Position Capture Status, Longevity of Captured Position: Use the TTRIG and TRIG commands to ascertain if a trigger interrupt input has been activated. TTRIG displays the status as a binary report, and TRIG is an assignment/comparison operator for using the status information in a conditional expression (e.g., in an IF statement). Once the captured commanded position value is assigned/compared with the PCC operator, the TTRIG/TRIG status bit for that trigger input is cleared; but the position information remains available until it is overwritten by a subsequent position capture from the same trigger input.

Position Capture Accuracy: The commanded position capture accuracy is ± 1 count.

Scaling and Position Offset: If scaling is enabled (SCALE1), the commanded position is scaled by the distance scaling factor (SCLD). If scaling is not enabled (SCALE0), the value assigned will be actual commanded counts. If you issue a PSET (establish absolute position reference) command, any previously captured commanded positions will be offset by the PSET command value.

Example:

```

INFNC1-H      ; Assign trigger input 1A as trigger interrupt input for axis 1
INFNC3-H      ; Assign trigger input 2A as trigger interrupt input for axis 2
VAR1=1PCCA    ; Assign captured commanded position of axis 1 to variable 1
              ; (position was captured when trigger input 1A became active)
IF(2PCCA<40)  ; If the captured commanded position on axis 2
              ; (captured when trigger input 2A became active) is
              ; less than 40, do the IF statement
VAR2=1PCCA+10 ; Add 10 to the captured commanded position on axis 1
              ; (captured when trigger input 1A became active) and
              ; assign the sum to variable #2
NIF          ; End IF statement

```

[PCE] Position of Captured Encoder

Type	Assignment or Comparison	Product	Rev
Syntax	See below	6K	5.0
Units	n/a		
Range	n/a		
Default	n/a		
Response	n/a		
See Also	CMDDIR, ENCCNT, ENCPOL, INFNC, [PCMS], [PE], PSET, SCALE, SCLD, SFB, TPCE, [TRIG], TRGLOT, TTRIG		

Use the PCE operator to assign the captured encoder position of a specific axis to a variable, or to make a comparison against another value.

Syntax: VARn=aPCEc where “n” is the variable number, “a” is the axis, and “c” designates trigger A or B for the axis, or M for the MASTER TRIG input (see table below); or PCE can be used in an expression such as IF(1PCEB>23450). The PCE operator must be used with an axis specifier or it will default to axis 1 (e.g., 1PCEA, 2PCEB, 5PCEM, etc.).

Trigger Input (Axis 1-4 “TRIGGERS/OUTPUTS” connector) * Axis			Dedicated	PCE	Trigger Input (Axis 5-8 “TRIGGERS/OUTPUTS” connector) * Axis			Dedicated	PCE
			Axis	Syntax				Axis	Syntax
Pin 23,	Trigger 1A		1	1PCEA	Pin 23,	Trigger 5A		5	5PCEA
Pin 21,	Trigger 1B		1	1PCEB	Pin 21,	Trigger 5B		5	5PCEB
Pin 19,	Trigger 2A		2	2PCEA	Pin 19,	Trigger 6A		6	6PCEA
Pin 17,	Trigger 2B		2	2PCEB	Pin 17,	Trigger 6B		6	6PCEB
Pin 15,	Trigger 3A		3	3PCEA	Pin 15,	Trigger 7A		7	7PCEA
Pin 13,	Trigger 3B		3	3PCEB	Pin 13,	Trigger 7B		7	7PCEB
Pin 11,	Trigger 4A		4	4PCEA	Pin 11,	Trigger 8A		8	8PCEA
Pin 9,	Trigger 4B		4	4PCEB	Pin 9,	Trigger 8B		8	8PCEB

* The number of trigger inputs available varies by product (refer to your product's *Installation Guide*).

To use an axis position captured with the MASTER TRIG input, use aPCEM, where “a” can be any axis number.

About Position Capture: The encoder position can be captured only by a trigger input that is defined as “trigger interrupt” input with the INFNCi-H command (see INFNC command). Each trigger input, when configured as a “trigger interrupt” input, is dedicated to capture the position of a specific axis (see table above). When a “trigger interrupt” input is activated, the encoder position of the dedicated axis is captured and the position is available through the use of the PCE operator and the TPCE display command. **Stepper Axes:** By default, stepper axes capture only the commanded position. To capture the encoder position, the axis must be in the Encoder Capture Mode (see ENCCNT command).

Position Capture Status, Longevity of Captured Position: Use the TTRIG and TRIG commands to ascertain if a trigger interrupt input has been activated. TTRIG displays the status as a binary report, and TRIG is an assignment/comparison operator for using the status information in a conditional expression (e.g., in an IF statement). Once the captured encoder position value is assigned/compared with the PCE operator, the TTRIG/TRIG status bit for that trigger input is cleared; but the position information remains available until it is overwritten by a subsequent position capture from the same trigger input.

Position Capture Accuracy: The encoder position capture accuracy is ± 1 encoder count.

Scaling and Position Offset: If scaling is enabled (SCALE1), the encoder position is scaled by the distance scaling factor (SCLD). If scaling is not enabled (SCALE0), the value assigned will be actual encoder counts. If you issue a PSET (establish absolute position reference) command, any previously captured encoder positions will be offset by the PSET command value.

Example:

```

INFNC1-H      ; Assign trigger input 1A as trigger interrupt input for axis 1
INFNC3-H      ; Assign trigger input 2A as trigger interrupt input for axis 2
VAR1=1PCEA    ; Assign captured encoder position of axis 1 to variable 1
              ; (position was captured when trigger input 1A became active)
IF(2PCEA<4000) ; If the captured encoder position on axis 2
              ; (captured when trigger input 2A became active) is
              ; less than 4000, do the IF statement
VAR2=1PCEA+10 ; Add 10 to the captured encoder position on axis 1
              ; (captured when trigger input 1A became active) and
              ; assign the sum to variable #2
NIF          ; End IF statement

```

[PCME] Position of Captured Master Encoder

Type	Assignment or Comparison	Product	Rev
Syntax	See below	6K	5.0
Units	n/a		
Range	n/a		
Default	n/a		
Response	n/a		
See Also	INFNC, MEPOL, MESND, [PME], [PCMS], PMECLR, PMESET, TPCME, TPME, TPCMS		

Use the PCME operator to assign the captured master encoder position to a variable, or to make a comparison against another value. The master encoder is connected to the connector labeled “Master Encoder.”

Syntax: VARn=PCME where “n” is the variable number; or PCME can be used in an expression such as IF(PCME>23450).

About Position Capture: The master encoder position can be captured only by the Master Trigger input (labeled “MASTER TRIG”), and only when that input is defined as a “trigger interrupt” input with the INFNC17-H command (see INFNC command). When the “trigger interrupt” input is activated (active edge), the master encoder position is captured and the position is available through the use of the PCME operator and the TPCME display command.

Position Capture Status, Longevity of Captured Position: Use the TTRIG and TRIG commands to ascertain if a trigger interrupt input has been activated. TTRIG displays the status as a binary report, and TRIG is an assignment/comparison operator for using the status information in a conditional expression (e.g., in an IF statement). Once the captured master encoder position value is assigned/compared with the PCME operator, TTRIG/TRIG status bit #17 is cleared; but the position information remains available until it is overwritten by a subsequent position capture from the master trigger input.

Position Capture Accuracy: The master encoder position capture accuracy is ± 1 encoder count.

Scaling and Position Offset: The PCME value is always in master encoder counts; it is never scaled. If you issue a PMESET (establish absolute position reference) command, any previously captured master encoder positions will be offset by the PMESET command value.

Example:

```
INFNC17-H      ; Assign master trigger as trigger interrupt input for the
                ; master encoder
VAR1=PCME      ; Assign captured master encoder position to variable 1
                ; (position was captured when master trigger became active)
IF(PCME<4000)  ; If the captured master encoder position
                ; (captured when master trigger input became active) is
                ; less than 4000, do the IF statement
VAR2=PCME+10   ; Add 10 to the captured master encoder position
                ; (captured when master trigger input became active) and
                ; assign the sum to variable #2
NIF            ; End IF statement
```

[PCMS] Captured Master Cycle Position

Type	Assignment or Comparison	Product	Rev
Syntax	See below	6K	5.0
Units	n/a		
Range	n/a		
Default	n/a		
Response	n/a		
See Also	CMDDIR, ENCCNT, ENCPOL, FOLMAS, INFNC, [PCC], [PCE], [PE], PSET, SCALE, SCLMAS, SFB, TPCMS, [TRIG], TRGLOT, TTRIG		

Use the PCMS operator to assign the captured master cycle position for a specific follower axis to a variable, or to make a comparison against another value.

PCMS (like PMAS) is unique among position assignment variables, because its value rolls over to zero each time the entire master cycle length (FMCLLEN) has been traveled. Thus, the captured PCMS value is essentially a snap-shot of the position relative to the master cycle at the time of the capture.

The master must be assigned first (FOLMAS command) before this operator will be useful.

Syntax: VARn=aPCMSc where “n” is the variable number, “a” is the axis, and “c” designates trigger A or B for the axis, or M for the MASTER TRIG input (see table below); or PCMS can be used in an expression such as IF (1PCMSB>2311). The PCMS operator must be used with an axis specifier or it will default to axis 1 (e.g., 1PCMSA, 2PCMSB, 5PCMSM, etc.).

Trigger Input (Axis 1-4 “TRIGGERS/OUTPUTS” connector) * Axis	Dedicated Axis	PCMS Syntax	Trigger Input (Axis 5-8 “TRIGGERS/OUTPUTS” connector) * Axis	Dedicated Axis	PCMS Syntax
Pin 23, Trigger 1A	1	1PCMSA	Pin 23, Trigger 5A	5	5PCMSA
Pin 21, Trigger 1B	1	1PCMSB	Pin 21, Trigger 5B	5	5PCMSB
Pin 19, Trigger 2A	2	2PCMSA	Pin 19, Trigger 6A	6	6PCMSA
Pin 17, Trigger 2B	2	2PCMSB	Pin 17, Trigger 6B	6	6PCMSB
Pin 15, Trigger 3A	3	3PCMSA	Pin 15, Trigger 7A	7	7PCMSA
Pin 13, Trigger 3B	3	3PCMSB	Pin 13, Trigger 7B	7	7PCMSB
Pin 11, Trigger 4A	4	4PCMSA	Pin 11, Trigger 8A	8	8PCMSA
Pin 9, Trigger 4B	4	4PCMSB	Pin 9, Trigger 8B	8	8PCMSB

* The number of trigger inputs available varies by product (refer to your product's *Installation Guide*).

To use a position captured with the MASTER TRIG input, use aPCMSM, where “a” can be any axis number.

About Position Capture: The master cycle position can be captured only by a trigger input that is defined as “trigger interrupt” input with the INFNCi-H command (see INFNC command). Each trigger input, when configured as a “trigger interrupt” input, is dedicated to capture the position of a specific axis (see table above). When a “trigger interrupt” input is activated, the master cycle position of the dedicated axis is captured and the position is available through the use of the PCMS operator and the TPCMS display command.

Position Capture Status, Longevity of Captured Position: Use the TTRIG and TRIG commands to ascertain if a trigger interrupt input has been activated. TTRIG displays the status as a binary report, and TRIG is an assignment/comparison operator for using the status information in a conditional expression (e.g., in an IF statement). Once the captured master cycle position value is assigned/compared with the PCMS operator, the TTRIG/TRIG status bit for that trigger input is cleared; but the position information remains available until it is overwritten by a subsequent position capture from the same trigger input.

Position Capture Accuracy: The master cycle position is interpolated; the capture accuracy is 50 µs multiplied by the velocity of the axis at the time the trigger input was activated.

Scaling and Position Offset: If scaling is enabled (SCALE1), the master source position is scaled by the distance scaling factor (SCLMAS). If scaling is not enabled (SCALEØ), the value assigned will be actual counts from the commanded or encoder master source as selected with the FOLMAS command. If you issue a PSET (establish absolute position reference) command, any previously captured master cycle positions will be offset by the PSET command value.

PCOMP Compile a Profile or Program

Type	Compiled Motion; Path Contouring; PLC Program	Product	Rev
Syntax	<!>PCOMP<t>	6K	5.0
Units	t = text (name of program/path)		
Range	Text name of 6 characters or less		
Default	n/a		
Response	n/a		
See Also	DEF, DRES, END, GOBUF, GOWHEN, MEMORY, PA, PAA, PAD, PADA, PAB, PARCOM, PARCOP, PARCM, PARCP, PAXES, PEXE, PLOOP, PL, PLC, PLCP, PLIN, PLN, POUTn, PRUN, SCLD, PUCOMP, PULSE, SCANP, [SEG], [SS], TDIR, TMEM, TRGFN, TSEG, TSS		

Use the PCOMP command to compile multi-axis contours, compiled (GOBUF) profiles for individual axes, and compiled PLCP programs for PLC Scan Mode. (For additional detail on contouring and compiled motion, refer to the Custom Profiling chapter in the *Programmer's Guide*.)

“Programs” vs. “Compiled Profiles & Programs”:

- Programs are defined with the DEF and END commands, as demonstrated in the Program Development Scenario in the *Programmer's Guide*.
- Compiled Profiles are defined like programs (using the DEF and END commands), but are compiled with the PCOMP command and executed with the PRUN command. A compiled profile could be a multi-axis contour (a series of arcs and lines), an individual axis profile (a series of GOBUF commands), or a compound profile (combination of multi-axis contours and individual axis profiles).
- Compiled PLC programs are defined with DEF PLCPi and END, compiled with PCOMP, and are normally executed in the PLC Scan Mode with the SCANP.

Compiling and Storing Compiled Paths & Programs:

Your controller's memory has two partitions: one for storing programs (“program” memory) and one for storing profiles & program segments compiled with the PCOMP command (“compiled” memory). The allocation of memory to these two areas is controlled with the MEMORY command.

Programs intended to be compiled are stored in program memory. After they are compiled with the PCOMP command, they remain in program memory and the segments (see segment command list below) from the compiled profile are stored in compiled memory.

- Contouring segments: PARCM, PARCOM, PARCOP, PARCP, PLIN
- Compiled Motion segments: GOBUF, PLOOP, GOWHEN, TRGFN, POUTA, POUTB, POUTC, POUTD
- PLC Program segments: IF, ELSE, NIF, L, LN, OUT, EXE, PEXE, VARI, VARB

The TDIR command uses “COMPILED AS A PATH” to denote the programs compiled as a compiled profile, and “COMPILED AS A PLC PROGRAM” to denote the programs compiled as a PLC programs. TDIR also reports the amount of program storage available, as does the TSEG command. System status bit #29 indicates that compiled memory is 75% full, and system status bit #30 indicates that compiled memory is completely full. (Use TSSF, TSS and SS to work with system status bits.)

If a compile (PCOMP) fails, system status bit #31 (see TSSF, TSS and SS) will be set. This status bit is cleared on power-up, reset, or after a successful compile. Possible causes for a failed compile are:

- Errors in profile design (e.g., change direction while at non-zero velocity; distance and velocity equate to < 1 count/system update; preset move profile ends in non-zero velocity).
- Profile will cause a Following error (see TFSF, TFS and [FS] commands).
- Out of memory (see system status bit #30).
- Axis already in motion at the time of a PCOMP command.
- Loop programming errors (e.g., no matching PLOOP or PLN; more than four embedded PLOOP/END loops).
- PLCP program contains invalid commands or command parameters.

Conditions That Require a Re-Compile (Contouring and Compiled Motion only):

- If it is desired to change a compiled path's velocity, acceleration, or deceleration, the values must be changed and then the path must be re-compiled.
- If the scaling factors are changed, the program must be downloaded again.
- Compiled Motion ONLY: After compiling (PCOMP) and running (PRUN) a compiled profile, the profile segments will be deleted from compiled memory if you cycle power or issue a RESET command.

COMPILED MOTION

When using compiled loops (PLOOP and PLN), the last segment within the loop must end at zero velocity or there must be a final GOBUF segment placed outside the loop. Otherwise an error will result when the profile is compiled. The error is "ERROR: MOTION ENDS IN NON-ZERO VELOCITY-AXIS n".

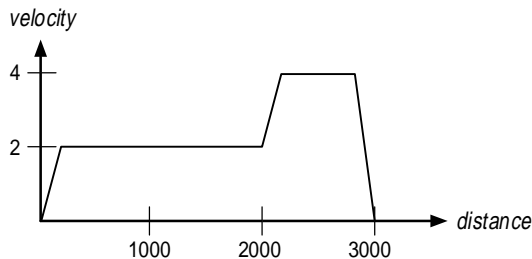
PLC PROGRAM EXAMPLE: see PLCP command description.

CONTOURING EXAMPLE

```
DEF prog1          ; Begin definition of program named prog1
PAXES1,2,3,4      ; Set axes 1, 2, 3, & 4 as the X, Y, Tangent, &
                  ; Proportional axes, respectively
PPO2.25          ; Proportional axis path ratio = 2.25
; *****
; * Put          *
; * MULTIPLE MOTION SEGMENT DEFINITIONS *
; * Here        *
; *****
END              ; End definition of path prog1
PCOMP prog1      ; Compile path prog1
PRUN prog1       ; Execute path prog1
```

COMPILED MOTION EXAMPLE (see profile below)

```
DEF prog2          ; Begin definition of program named prog2
A10,10            ; Set A, V, and D values for axes 1 and 2
V2,2
D2000,2000
GOBUF11          ; First segment of motion for axes 1 and 2
V4,4             ; New A,V, and D values
AD50,50
D1000,1000
GOBUF11          ; Second segment
END              ; End definition of prog2
PCOMP prog2      ; Compile prog2
PRUN prog2       ; Execute prog2
```



[PE] Position of Encoder

Type	Assignment or Comparison	Product	Rev
Syntax	See below	6K	5.0
Units	Encoder counts, or scaled by SCLD		
Range	n/a		
Default	n/a		
Response	n/a		
See Also	CMDDIR, ENCCNT, ENCPOL, ENCSND, [FB], GOWHEN, INFNC, [PC], [PCE], [PER], PESET, PSET, SCALE, SCLD, SFB, TFB, TPE		

The Position of Encoder (PE) operator is used to assign one of the encoder register values to a variable, or to make a comparison against another value. If the encoder has been configured to receive step and direction input (ENCSND), the PE operator will report the position as counted from the step and direction signal.

Stepper axes: If the ENCCNT1 mode is enabled PE reports the encoder position, but in ENCCNT0 mode (the factory default setting) the PE report represents the commanded position.

UNITS OF MEASURE and SCALING: refer to page 16 or to the SCLD command.

If you issue a PSET command, the encoder position value will be offset by the PSET command value. If you are using a stepper axis in the ENCCNT1 mode, use the PESET command instead.

Syntax: VARn=aPE where “n” is the variable number, and “a” is the axis, or PE can be used in an expression such as IF(1PE>23450). The PE command must be used with an axis specifier or it will default to axis 1 (e.g., 1PE, 2PE, etc.).

Example:

```
VAR1=1PE          ; Encoder position for axis 1 is assigned to variable 1
IF(2PE<4000)     ; If the encoder count for axis 2 is less than 4000,
                  ; do the IF statement
VAR2=3PE+4000    ; Encoder position for axis 3 plus 4000 is assigned
                  ; to variable 2
NIF              ; End IF statement
```

[PER] Position Error

Type	Assignment or Comparison	Product	Rev
Syntax	See below	6K	5.0
Units	n/a		
Range	n/a		(applicable to servo axes only)
Default	n/a		
Response	n/a		
See Also	CMDDIR, DRES, ENCPOL, ERES, SCLD, SFB, SMPER, TAS, TPER, TPE, TPC		

The Position Error (PER) operator is used to assign the current position error of each axis to a variable, or to make a comparison against another value. The value assigned to the variable or the value against which the comparison is made is measured in feedback device counts and is scaled by the distance scaling factor (SCLD), if scaling is enabled with the SCALE1 command.

The position error is the difference between the commanded position and the actual position read by the feedback device. This error is calculated every sample period and can be displayed at any time using the TPER command.

Syntax: VARn=aPER where “n” is the variable number, and “a” is the axis, or PER can be used in an expression such as IF(1PER>50). The PER command must be used with an axis specifier or it will default to axis 1 (e.g., 1PER, 2PER, etc.).

Example:

```
VAR1=1PER          ; Position error for axis 1 is assigned to variable 1
IF(2PER>2000)     ; If the position error for axis 2 is >2000 encoder counts,
                  ; do the IF statement (enable output #4)
OUTXXX1          ; Enable onboard output #4
NIF              ; End IF statement
```

PESET Encoder Absolute Position Reference - Stepper Axes

Type	Motion	Product	Rev
Syntax	<!><@>PESET<r>,<r>,<r>,<r>,<r>,<r>,<r>,<r>	6K	5.0
Units	r = units (absolute position of encoder)		
Range	±999,999,999.99999		(applicable to stepper axes only)
Default	n/a		
Response	n/a		
See Also	AXSDEF, ENCCNT, ENCPOL, INFNC, [PCE], [PE], PMESET, PSET, SCALE, SCLD, TPCE, TPE		

Use the PESET command to offset the current absolute encoder position to establish an *absolute position reference* for the encoder reports (TPE, PE, TPCE, PCE). **NOTE:** PESET can only be used for axes that are defined as stepper axes with the AXSDEF command. All PESET values entered are in encoder steps, scalable by the SCLD value if scaling is enabled.

NOTE: If you issue a PESET command, any previously captured encoder positions (INFNCi-H or LIMFNCi-H function) will be offset by the PESET value.

Example:

```
AXSDEF0000      ; Define axes 1-4 as stepper axes
ENCCNT1111     ; Place axes 1-4 in the encoder count referencing mode
PESET0,0,0,1000 ; Set absolute position on axes 1, 2, and 3 to zero,
                ; and axis 4 to 1000 units
TPE            ; Display the new positions. The new encoder position
                ; report should be: *TPE0,0,0,1000
```

PEXE Execute a Compiled Program

Type	PLC Mode Compiled Program Execution	Product	Rev
Syntax	i%PEXEt	6K	5.0
Units	i = Task Number t = Program Name (6 characters or less)		
Range	i = 1-10		
Default	n/a		
Response	n/a		
See Also	EXE, GOBUF, PCOMP, PLCP, SCANP		

Use the PEXE command to start a compiled PLCP program, compiled contouring path, or compiled GOBUF profile from within a compiled PLCP program. The PEXE command specifies the name of the compiled program, and the task in which it will be launched. The program named in the PEXE command need not be defined or compiled at the time the PLCP program is compiled; however, the program must be defined and compiled before the SCANP or PRUN is issued. If no task number is assigned with a % prefix, then the task in which the PLCP program is compiled (PCOMP) will be the task that runs the compiled program. Note, however, that the PEXE program cannot be executed in the Task Supervisor (task 0).

The PLCP program will ignore the PEXE command if a currently running program is detected within the specified task; therefore, the PEXE command can essentially only be used to initiate a new task with the program it is launching. Like the INSELP command, the program launched by the PEXE command will not interrupt a currently running program, nor will it interrupt a WAIT or T command. Also, if launching a compiled contouring path or GOBUF profile, the PEXE will not interrupt motion already in progress.

CAUTION: Using the SCANP command to run a PLCP program in Scan mode will cause the PLCP program to execute as often as every system update period (2 ms). A PEXE command used within a PLCP program running in Scan mode could therefore attempt to launch a program in the specified task as often as every 2 ms. This may not allow enough time for the program launched in the specified task by the PEXE command to complete before the same PEXE command is issued again. As stated, the PLCP program will ignore the PEXE command if a currently running program is detected or motion is in progress on the participating axes, so timing must be considered when launching programs with the PEXE command.

To execute a non-compiled program from within a compiled PLCP program, use the EXE command.

Example:

```
DEF PLCP1      ; Define PLC program PLCP1
1%PEXE PLCP2   ; Launch compiled program PLCP2 in task 1
END
DEF PLCP2      ; Define PLC program PLCP2
OUT(VARB1)     ; Modify outputs
END
```

```

PCOMP PLCP1      ; Compile PLCP1
PCOMP PLCP2      ; Compile PLCP2
SCANP PLCP1      ; Scan with program PLCP1
VARB1=h0000      ; Set VARB1
TOUT              ; Check outputs (response is *TOUT0000_0000_0000_0000)
VARB1=b1010      ; Reassign VARB1
TOUT              ; Check outputs again (response is *TOUT1010_0000_0000_0000)

```

[PI]

PI (π)

Type	Operator (Trigonometric)	Product	Rev
Syntax	See examples below	6K	5.0
Units	n/a		
Range	n/a		
Default	n/a		
Response	n/a		
See Also	[=], [+], [-], [*], [/], [&], [], [^], [~], [ATAN], [COS], IF, [SIN], [SQRT], [TAN], VAR		

The (PI) command is assigned the value 3.14159265. There are 2π radians in 360° . This command is useful for doing trigonometric functions in radian units (RADIAN command).

Example:

```

VAR1=PI          ; 3.14159265 is assigned to variable 1
VAR2=2 * PI      ; 2 pi is assigned to variable 2

```

PL

Define Path Local Mode

Type	Path Contouring	Product	Rev
Syntax	<!>PL	6K	5.0
Units	n/a		
Range	b = 0 (work coordinates) or 1 (local coordinates)		
Default	0		
Response	No response - Must be defining a path (DEF)		
See Also	PAB, PLC, PWC		

The Define Path Local Mode (PL) command is used to specify the use of either the Local coordinate system or the Work coordinate system. Endpoints are allowed to be specified as absolute positions, and these positions may either be in the Work or the Local coordinate system. Programming may switch between Local and Work coordinates before any segment or group of segments.

When switching to Local coordinates, the starting coordinates of the next segment in the Local coordinate system must be specified with the PLC command before the PL1 command is issued.

When using the Work coordinate system (PL0), the starting coordinates of the next segment in the Work coordinate system may be specified with the PWC command for the purpose of shifting the Work coordinate system. If the PWC command is not given, the previous Work coordinate system is used.

Example:

```

PV5              ; Set path velocity to 5 units/sec
PA50             ; Set path acceleration to 50 units/sec/sec
PAD100           ; Set path deceleration to 100 units/sec/sec
DEF prog1        ; Begin definition of path named prog1
PAXES1,2         ; Set axes 1 and 2 as the X and Y contouring axes
PAB1             ; Set to absolute coordinates
PWC0,0           ; Specify X and Y data, work coordinates
PL0             ; Specify work coordinate system
PLIN1,1          ; Specify X-Y endpoint position to create a 45 degree angle line segment
PLC0,0           ; Specify X and Y data, local coordinates
PL1             ; Specify local coordinate system
PARCOP0,0,5,0    ; Specify incremental X-Y endpoint position and X-Y center
                  ; position for full circle clockwise arc
PLIN0,11         ; Specify X-Y endpoint position to create a 90 degree angle line segment
PLC0,0           ; Specify X and Y data, local coordinates
PL1             ; Specify local coordinate system
PARCOP0,0,5,0    ; Specify incremental X-Y endpoint position and X-Y center
                  ; position for full circle clockwise arc
PL0             ; Specify work coordinate system
PLIN0,0          ; Specify X-Y endpoint position to create a line segment back to 0,0
END              ; End definition of path prog1
PCOMP prog1      ; Compile path prog1
PRUN prog1       ; Execute path prog1

```

PLC

Define Path Local Coordinates

Type	Path Contouring	Product	Rev
Syntax	<!>PLC<r>,<r>	6K	5.0
Units	1 st r = X coordinate units (scalable by the SCLD value) 2 nd r = Y coordinate units (scalable by the SCLD value)		
Range	0.00000 - ±999,999,999		
Default	n/a		
Response	No response - Must be defining a path (DEF)		
See Also	PAB, PL, SCLD, PWC, SCALE		

The Define Path Local Coordinates (PLC) command is used to specify the Local X -Y coordinate data required for subsequent segment definition in the Local coordinate system. This command places the X -Y coordinate value of the Local coordinate system at the beginning of the next segment. (The first <r> is the X coordinate, the second <r> is the Y coordinate.) This command must be used before the PL1 command is given.

UNITS OF MEASURE and SCALING: refer to page 16 or to the SCLD description.

Example: Refer to Define Path Local Mode (PL) command example.

PLCP

Compiled PLC Program

Type	PLC Scan Program	Product	Rev
Syntax	<!>PLCPi	6K	5.0
Units	i = number of PLC program		
Range	1-99		
Default	n/a		
Response	n/a		
See Also	DEF, ELSE, EXE, IF, L, LN, MEMORY, NIF, OUT, PCOMP, PEEXE, PRUN, PUCOMP, SCANP, TSCAN, VARI, VARB		

PLCP is not a command; it is used to identify a PLCP program to be defined (e.g., DEF PLCP2), compiled (e.g., PCOMP PLCP2), and executed (e.g., SCANP PLCP2 or PRUN PLCP2). Up to 99 PLCP programs may be defined, identified as PLCP1, PLCP2, PLCP3, and so on. The purpose of PLCP programs is to facilitate fast I/O scanning.

The process of creating and executing a PLCP program is:

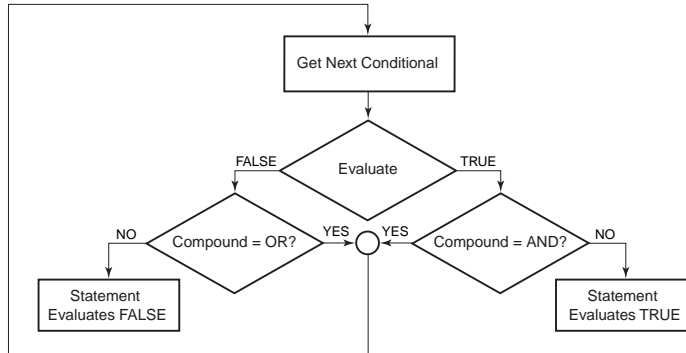
1. Define the PLCP program (DEF PLCPi statement, followed by commands from the list below, followed by END). Only these commands are allowed in a PLCP program:
 - IF, ELSE, and NIF (conditional branching) — see note below for limitations
 - L and LN (loops)
 - OUT (turn on a digital output)
 - EXE (execute a program in a specific task — e.g., 2%EXE MOVE)
 - PEEXE (execute a program in a specific task — e.g., 3%PEEXE PLCP4)
 - VARI (integer variables).
 - VARB (binary variables). Bitwise operations are limited to Boolean And (&), Boolean Inclusive Or (|), and Boolean Exclusive Or (^).
2. Compile the PLCP program (PCOMP PLCPi). A compiled program runs much faster than a standard program.
3. Execute the PLCP program (SCANP PLCPi). When the PLCP program is launched with the SCANP command, it is executed in the “PLC Scan Mode”. The advantage of the PLC Scan Mode is that the PLCP program is executed within a dedicated 0.5 ms time slot during every 2 ms system update period. This gives the PLCP program faster throughput for monitoring and manipulating I/O. *For more information on how the PLCP program is executed with SCANP, refer to the SCANP command description.*

An alternative execution method is to use the PRUN command (PRUN PLCPi). This method is similar to the SCANP PLCPi method, but will only run through the PLCP program once.

Memory Requirements: Most commands allowed in a PLCP program consume one segment of compiled memory after the program is compiled with PCOMP; the exceptions are VAR1 and VARB (each consume 2 segments) and IF statements. Each IF conditional evaluation compounded with either an AND or an OR operator consumes an additional segment (e.g., IF (IN.1=b1 AND 1AS.1=b0) consumes three segments of compiled memory). The number of compounds is limited only by the memory available.

Conditional Expressions:

- **Order of Evaluation.** Because only one level of parenthesis is allowed, the order of evaluation of IF conditionals is from left to right. Refer to the flowchart for the evaluation logic.
- Conditional expressions in a PLC program use the non-scaled integer (“raw”) operand values. Examples of the “raw” operand values are:



- The PE operator reports encoder counts not scaled by SCLD and not scaled by ERES.
- The ANI operator reports ADC counts from an analog input, not scaled by SCLD. Assuming the default ANIRNG4 setting (+/-10V voltage range), 205 ADC counts = 1 volt.
- The DAC operator reports DAC counts (commanded position) not scaled by SCLD.

The only operands that are not allowed are: SIN, COS, TAN, ATAN, VCVT, SQRT, VAR, TW, READ, DREAD, DREADF, DAT, DPTR, and PI.

Programming Example: Refer to the detailed, illustrated example in the SCANP command description.

PLIN		Move in a Line	
Type	Path Contouring	Product	Rev
Syntax	<!><@>PLIN<r>, <r>	6K	5.0
Units	1 st r = X endpoint coordinate (scalable by the SCLD value) 2 nd r = Y endpoint coordinate (scalable by the SCLD value)		
Range	0.00000 - ±999,999,999		
Default	n/a		
Response	No response - Must be defining a path (DEF)		
See Also	PAB, PL, PLC, SCLD, PWC, SCALE		

The Define Line Segment (PLIN) command is used to specify a line segment. The placement, length, and orientation of the line are completely specified by the endpoint of the line segment and the endpoint of the previous segment (current position). Segment endpoint position specifications may be either absolute (PAB1) with respect to the user defined coordinate system, or incremental (PABØ), relative to the start of each individual segment.

When the PLIN command is received, the first value is taken as the X endpoint coordinate and the second value is taken as the Y endpoint coordinate.

UNITS OF MEASURE and SCALING: refer to page 16 or to the SCLD description.

Example: Refer to Define Path Local Mode (PL) command example.

PLN

Loop End, Compiled Motion

Type	Compiled Motion	Product	Rev
Syntax	<@>PLN	6K	5.0
Units	n/a		
Range	b = 1 (end loop), 0 or X (don't end loop)		
Default	n/a		
Response	No response; instead ends loop for compiled motion		
See Also	GOBUF, PCOMP, PLOOP, PRUN, PUCOMP		

The Loop End, Compiled Motion (PLN) command specifies the end of an axis-specific compiled motion profile loop, as initiated with the PLOOP command.

Programming Example: see PLOOP.

PLOOP

Loop Start, Compiled Motion

Type	Compiled Motion	Product	Rev
Syntax	<@>PLOOP<i>,<i>,<i>,<i>,<i>,<i>,<i>,<i>	6K	5.0
Units	i = designated number of loops for specified axis		
Range	0-2,147,483,647 ($2^{31}-1$) 0 = infinite loop		
Default	n/a		
Response	No response; instead starts loop for compiled motion		
See Also	GOBUF, PCOMP, PLN, PRUN, PUCOMP		

The PLOOP command specifies the beginning of an axis-specific profile loop. All subsequent segments defined before the PLN command are included within that loop. The number in a given axis field specifies the number of loops to be executed for that axis. If that number is a zero or blank, then the loop will be executed infinitely. The PLOOP command can be nested up to four levels deep within a program.

When using compiled loops (PLOOP and PLN), the last segment within the loop must end at zero velocity or there must be a final GOBUF segment placed outside (after) the loop. Otherwise an error will result when the profile is compiled. The error is "ERROR: MOTION ENDS IN NON-ZERO VELOCITY-AXIS n".

The PLOOP command will consume one segment of compiled space.

Example:

```
DEF prog1      ; Begin definition of prog1
V1             ; Set velocity to 1 unit/sec
D1000         ; Set distance to 1000 units
GOBUF1        ; Segment of motion sent to buffer

PLOOP3        ; Start loop of the subsequent move profile

V10           ; Set velocity to 10 units/sec
D25000        ; Set distance to 25000 units
GOBUF1        ; First segment within loop sent to buffer

V2            ; Set velocity to 2 units/sec
D1000         ; Set distance to 1000 units
GOBUF1        ; Second segment of motion within loop sent to buffer

V1            ; Set velocity to 1 unit/sec
D25000        ; Set distance to 25000 units
GOBUF1        ; Third segment within loop sent to buffer

PLN1          ; Close loop

V.5           ; Set velocity to 0.5 units/sec
D100          ; Set distance to 100 units
GOBUF1        ; Segment of motion sent to buffer (outside loop)

END           ; End definition of prog1

PCOMP prog1   ; Compile prog1
PRUN prog1    ; Execute prog1
```

[PMAS] Current Master Cycle Position

Type	Following and Assignment or Comparison	Product	Rev
Syntax	See below	6K	5.0
Units	n/a		
Range	n/a		
Default	n/a		
Response	n/a		
See Also	FMCNEW, FMCP, FOLMAS, FOLMD, [FS], GOWHEN, [PCMS], SCALE, SCLMAS, TPMAS, TFS		

The PMAS operator is used to assign the master position register value to a variable, or to make a comparison against another value. This value may be used for subsequent decision making, or for recording the cycle position corresponding to some other event.

PMAS is unique among position assignment variables, because its value rolls over to zero each time the entire master cycle length (FMCLLEN value) has been traveled. If it is desired to WAIT or GOWHEN on a master cycle position of the next master cycle, one master cycle length (value of FMCLLEN) should be added to the master cycle position specified in the argument. This allows commands that sequence follower events through a master cycle to be placed in a loop. The WAIT or GOWHEN command at the top of the loop could execute, even though the actual master travel had not finished the previous cycle. This is done to allow a PMAS value which is equal to the master cycle length to be specified and reliably detected. When using PMAS with IF, UNTIL, or WHILE arguments, the instantaneous PMAS value is used. Be careful to avoid specifying PMAS values that are nearly equal to the master cycle length (FMCLLEN), because rollover may occur before a PMAS sample is read.

The master must be assigned first (FOLMAS command) before this command will be useful.

If scaling is enabled (SCALE1), the PMAS value is scaled by the master scaling factor (SCLMAS). If scaling is disabled (SCALE0), the PMAS value is in counts.

Syntax: VARn=aPMAS where “n” is the variable number and “a” is the axis number, or PMAS can be used in an expression such as IF(2PMAS>23450). The PMAS command must be used with an axis specifier, or it will default to axis 1 (e.g., VAR1=1PMAS, IF(2PMAS>500), etc.).

Example: (refer also to FOLEN example #2)

```
IF(2PMAS>4.3) ; If the master for axis 2 has traveled more than 4.3
               ; master user units then do the IF statement
OUT.2=b1      ; Set onboard output #2 to 1
NIF           ; End of IF statement
VAR14=1PMAS   ; Set VAR14 to axis 1's master cycle position
```

[PME] Position of Master Encoder

Type	Assignment or Comparison	Product	Rev
Syntax	See below	6K	5.0
Units	Master Encoder counts		
Range	n/a		
Default	n/a		
Response	n/a		
See Also	MEPOL, MESND, [PCME], [PE], PMECLR, PASET, TPCME, TPME		

Use the PME operator to assign the current master encoder position to a variable, or to make a comparison against another value. The master encoder is connected to the connector labeled “Master Encoder”. If you issue a PASET command, the encoder position value will be offset by the PASET command value. The PME value is always in encoder counts, it is never scaled.

Syntax: VARn=PME where “n” is the variable number, or PME can be used in an expression such as IF(PME>16000).

Example:

```
VAR1=PME           ; Master encoder position is assigned to variable 1
IF(PME<4000)      ; If the master encoder count is less than 4000,
                  ; do the IF statement
VAR2=PME+4000     ; Master encoder position plus 4000 is assigned to variable 2
NIF               ; End IF statement
```

PMECLR Clear Master Encoder Absolute Position Reference

Type	Motion	Product	Rev
Syntax	<!>PMECLR<r>	6K	5.0
Units	r = master encoder counts (not scalable)		
Range	±999,999,999.99999		
Default	n/a		
Response	n/a		
See Also	MEPOL, MESND, [PCME], [PME], PASET, PSET, TPCME, TPME		

Use the PMECLR command to remove any offset on the master encoder position reports (offset to master encoder position is established with the PASET command).

Example:

```
TPME              ; Report master encoder position. For the sake of this example,
                  ; let's say the response is *TPME10000 (master encoder is at
                  ; absolute position 10,000).
PASET20000        ; Change relative position of master encoder with offset
TPME              ; Report new master encoder position with offset. New position
                  ; response should now be: *TPME20000 (what was considered
                  ; position 10,000 is now considered position 20,000).
PMECLR            ; Clear any offset applied to the master encoder position
TPME              ; Report master encoder position with no offsets.
                  ; Response should be: *TPME10000.
```

PASET Establish Master Encoder Absolute Position Reference

Type	Motion	Product	Rev
Syntax	<!>PASET<r>	6K	5.0
Units	r = master encoder counts (not scalable)		
Range	±999,999,999.99999		
Default	n/a		
Response	n/a		
See Also	MEPOL, MESND, [PCME], [PME], PMECLR, PSET, TPCME, TPME		

Use the PASET command to offset the current absolute position of the master encoder (connected to the connector labeled “Master Encoder”) to establish an *absolute position reference*. To remove the offset, issue the PMECLR command.

All PMESET values entered are in master encoder counts; this value is never scaled.

Example:

```

TPME           ; Report master encoder position. For the sake of this example,
               ; let's say the response is *TPME10000 (master encoder is at
               ; absolute position 10,000).
PMESET20000   ; Change relative position of master encoder with offset
TPME           ; Report new master encoder position with offset. New position
               ; response should now be: *TPME20000 (what was considered
               ; position 10,000 is now considered position 20,000).
PMECLR        ; Clear any offset applied to the master encoder position
TPME           ; Report master encoder position with no offsets.
               ; Response should be: *TPME10000.

```

PORT

Designate Destination Communication ("COM") Port

Type	Communication Interface	Product	Rev
Syntax	<!>PORT<i>	6K	5.0
Units	i = port number		
Range	1 (COM1), 2 (COM2)		
	NOTE: "COM1" is the "RS-232" or "ETHERNET" connector. "COM2" is the "RS-232/485" connector.		
Default	1		
Response	n/a		
See Also], [, BOT, DRPCHK, E, EOL, EOT, ERRDEF, ERRLVL, ERROK, ERBAD, LOCK, [READ], WRITE, XONXOFF		

The Designate Destination Port (PORT) command is used to determine which COM port is affected by the DRPCHK, E, ECHO, BOT, EOL, EOT, ERROK, ERBAD, ERRDEF, ERRLVL, and XONOFF commands. It also specifies the port to which responses and prompts from stored programs should be sent.

The PORT command also selects the target port through which the WRITE and READ commands transmit ASCII text strings. The DWRITE command (as well as all other RP240 commands) will affect the RP240 regardless of the PORT command setting. If no RP240 is detected, the commands are sent to the COM2 port. DWRITE text strings are always terminated with a carriage return.

Example (The PORT command can be used to designate EOT parameters for both ports. Assume that port COM1 is being used to communicate to the controller.)

```

PORT1         ; Select COM1 for EOT setup
EOT45,49,10   ; EOT for COM1 is -1<lf>
TPE           ; Send "Transfer Position of Encoder" response to COM1
               ; using EOT 45,49,10
PORT2         ; Select COM2 for EOT setup
EOT45,50,10   ; EOT for COM2 is -2<lf>
TPC           ; Send "Transfer Commanded Position" response to COM2
               ; using EOT 45,50,10

```

Example (The PORT command specifies both port setups and response destinations in a stored program.)

```

DEF qwe       ; Begin definition of qwe
PORT1
EOT45,49,10   ; EOT for COM1 is -1<lf>
TPE           ; Send "Transfer Position of Encoder" response to COM1
               ; using EOT 45,49,10
PORT2
EOT45,50,10   ; EOT for COM2 is -2<lf>
TPC           ; Send "Transfer Commanded Position" response to COM2
               ; using EOT 45,50,10
END           ; End definition of qwe

```

POUT

Compiled Output

Type	Path Contouring; Compiled Motion	Product	Rev
Syntax	<!>POUT<n> ...	6K	5.0
Units	n = axis identifier letter (for compiled motion only); b = enable bit specific outputs (see page 6)		
Range	n = A-H for axes 1-8, respectively (for compiled motion only); b = 0 (off), 1 (on), or X (don't change)		
Default	0		
Response	n/a		
See Also	GOBUF, OUT, OUTEN, OUTFNC, OUTLVL, PCOMP, PRUN, PUCOMP		

Use the POUT command to control outputs during Contouring Motion or Compiled Motion. The syntax for the POUT command depends on whether you are using it for Contouring or Compiled Motion:

Contouring: POUT

Compiled Motion: POUTA (apply output pattern to the profile for axis #1) POUTB (apply output pattern to the profile for axis #2) POUTC (apply output pattern to the profile for axis #3) POUTD (apply output pattern to the profile for axis #4) POUTE (apply output pattern to the profile for axis #5) POUTF (apply output pattern to the profile for axis #6) POUTG (apply output pattern to the profile for axis #7) POUTH (apply output pattern to the profile for axis #8)
--

You may use the POUT command to control any of the onboard outputs, as well as any outputs on external I/O bricks, as long as they are left in the default function (OUTFNCi-A). Refer to page 6 to understand how to address the outputs (onboard and on optional expansion I/O bricks) available on your 6K product.

If you attempt to change the state of an output that is not defined as an OUTFNCi-A (general-purpose) output, the controller will respond with an error message (“OUTPUT BIT USED AS OUTFNC”) and the POUT command will not be executed (but command processing will continue).

If you wish to set only one output value, instead of all outputs, use the bit select (.) operator, followed by the number of the specified output. Contouring example: 2POUT.12-1 turns on only output 12 on I/O brick 2. Compiled Motion example: 2POUTA.12-1 turns on only output 12 on I/O brick 2 for the axis 1 profile.

The POUT command consumes one segment of compiled memory.

The programmable outputs are sampled once per “system update” (2 ms).

Contouring ONLY:

The POUT command specifies the programmable output bit pattern to be applied to the outputs at the beginning of the next segment and remain throughout that segment. The POUT command may be issued before any segment definition command, and will affect all subsequent segments until a new POUT command is issued. A POUT command will not take affect if there is no segment definition command following it. To change the programmable outputs at the end of a path, the standard output (OUT) command must be used after the path is executed. These segment-defined output patterns are stored as part of the compiled path definition.

CONTOURING EXAMPLE: Refer to the PARCOM command example.

COMPILED MOTION EXAMPLES: (see next page)

COMPILED MOTION EXAMPLES:

```
OUTFNC3-A      ; Default output function for onboard output 3
OUTFNC6-A      ; Default output function for onboard output 6
DEF P1         ; Define program P1
D1000,25000    ; Set distance to travel
GOBUF11        ; Motion segments for axes 1 and 2
POUTA.3-1      ; Turn on onboard output 3 when axis 1 travels to 1000 steps
D2000,50000    ; New distance commanded
GOBUF11        ; Motion segments for axes 1 and 2
POUTA.3-0      ; Turn off onboard output 3 when axis 1 travels 2000
                ; additional steps
POUTB.6-1      ; Turn on onboard output 6 when axis 2 travels to 75000 steps
D1000,25000    ; New distance commanded
GOBUF11        ; Motion segment for axes 1 and 2
POUTB.6-0      ; Turn off onboard output 6 when axis 2 travels 25000
                ; additional steps
END            ; End program definition

PCOMP P1       ; Compiled program P1
PRUN P1        ; Execute program P1
```

When executing a Compiled Following profile, the POUTn statement is always executed as programmed. Therefore, in order to make sure an output is on for a given motion segment no matter what direction the master is traveling, you should use two POUTn statements (see example below).

```
POUTA.3-0      ; Turn off onboard output 3 for axis 1 - master going backwards
POUTA.3-1      ; Turn on onboard output 3 for axis 1 - master going forwards
GOBUF11        ; Motion segments for axes 1
POUTA.3-1      ; Turn on onboard output 3 for axis 1 - master going backwards
POUTA.3-0      ; Turn off onboard output 3 for axis 1 - master going forwards
```

If you desire to “pulse” an output (turn on for a given amount of time), then use the POUTn command along with the GOWHEN(T=n) command. For example:

```
POUTA.1-1      ; Turn on onboard output 1
GOWHEN(T=120)  ; Wait for 120 milliseconds
POUTA.1-0      ; Turn off onboard output 1
```

PPRO

Path Proportional Axis

Type	Path Contouring	Product	Rev
Syntax	<!>PPRO<r>	6K	5.0
Units	r = ratio value		
Range	±0.001 - 1000.000		
Default	n/a		
Response	No response - Must be defining a path (DEF)		
See Also	PAXES		

The Path Proportional Axis (PPRO) command is used to specify the proportional axis to path travel ratio. The proportional axis will keep a position that is proportional to the distance traveled along the X-Y path as the path is executed. This allows the proportional axis to act as the Z axis in helical interpolation or to control the motion of any object which moves with distance and velocity proportional to the path.

The PPRO command should be given prior to any contour segments during a path definition. A negative value for the proportional axis ratio simply causes motion in the negative direction as path travel in the X-Y plane gets larger.

Example: (see contouring programming example in the PRUN command description)

PRTOL **Path Radius Tolerance**

Type	Path Contouring	Product	Rev
Syntax	<!>PRTOL<r>	6K	5.0
Units	r = allowable radius error (scalable by the SCLD value)		
Range	±999,999,999.99999		
Default	1		
Response	No response - Must be defining a path (DEF)		
See Also	PARCM, PARCOM, PARCOP, PARCP, SCLD, SCALE		

The Path Radius Tolerance (PRTOL) command is used to specify the allowable radius error that is encountered when contouring.

The radius error is encountered in one of two ways. The first way is through use of the PARCM or PARCP commands. This error is the difference between the radius value specified in the PARCM or PARCP command and the minimum radius implied by the starting point and endpoint. If the radius provided in the command is smaller than the minimum radius implied by the distance from starting to endpoints and the error is within the radius tolerance then just enough is added to the radius to make a half circle.

A second way to encounter a radius tolerance error is with the PARCOM or PARCOP commands. This error is the difference between the radius implied by the start point and center point and the radius implied by the end point and center point. If the difference in the two radius values is within the radius tolerance specified, then the center point is moved such that an arc can be traveled through the start point and endpoint. The PRTOL command can be executed many times within a path definition allowing some arcs to be exactly known and others to be approximated.

If the radius error exceeds the PRTOL value, an error message is sent.

UNITS OF MEASURE and SCALING: refer to page 16 or to the SCLD description.

Example:

```
PV5           ; Set path velocity to 5 units/sec
PA50          ; Set path acceleration to 50 units/sec/sec
PAD100        ; Set path deceleration to 100 units/sec/sec
DEF prog1     ; Begin definition of path named prog1
PAXES1,2      ; Set axes 1 and 2 as the X and Y contouring axes
PAB0          ; Set to incremental coordinates
PRTOL0.001    ; Allow 25 steps (0.001 x 25000) radius error
PARCM5,5,5    ; Specify incremental X-Y endpoint position and radius
               ; arc <180 degree for quarter circle counter-clockwise arc
PARCP5,-5,-5  ; Specify incremental X-Y endpoint position and radius
               ; arc >180 degree for three quarter circle clockwise arc
END           ; End definition of path prog1
PCOMP prog1   ; Compile path prog1
PRUN prog1    ; Execute path prog1
```

PS Pause Program Execution

Type	Program Flow Control	Product	Rev
Syntax	<!>PS	6K	5.0
Units	n/a		
Range	n/a		
Default	n/a		
Response	n/a		
See Also	C, COMEXR, COMEXS, K, S, [SS], TSS		

The Pause Program Execution (PS) command pauses execution of commands in the command buffer. If a PS command is executed, no commands after the PS will be executed until a !C command is received. However, additional commands may still be placed in the command buffer.

The PS command does not pause motion. In order for motion to be paused, the S and the COMEXS commands should be used.

Example:

```
PS                ; Stop execution of command buffer until !C command
MA0XXX           ; Incremental mode for axis 1
D10000           ; Set distance to 10000 units on axis 1
GO1000           ; Initiate motion on axis 1
D,20000          ; Set distance to 20000 units on axis 2
GO0100           ; Initiate motion on axis 2
; *****
; * NOTE:
; * No commands after the PS command will be executed until a !C
; * command is received.
; *****
```

PSET Establish Absolute Position

Type	Motion	Product	Rev
Syntax	<!><@>PSET<r>,<r>,<r>,<r>,<r>,<r>,<r>,<r>	6K	5.0
Units	r = units (absolute position)		
Range	±999,999,999.99999		
Default	n/a		
Response	n/a		
See Also	CMDDIR, D, ENCPOL, [FB], GO, HOM, INFNC, MA, MC, [PANI], [PC], [PCC], [PCE], [PCMS], [PE], PESET, PMESET, SCALE, SCLD, SFB, TFB, TPANI, TPC, TPCC, TPCE, TPCMS, TPE		

Use the PSET command to offset the current absolute position to establish an *absolute position reference*. To remove the offset, issue the PSET CLR command. All PSET values entered are in steps, unless scaling is enabled (SCALE1), in which case (PSET) is multiplied by the distance scale factor (SCLD).

Steppers – without scaling: The PSET command will define the current commanded position to be the absolute position entered. To set an absolute encoder position, use the PESET command.

Servos – without scaling: The PSET command defines a new absolute position reference. If the drive is enabled (DRIVE), the current commanded position is used as the reference point. If the drive is disabled, the current feedback device position (selected with the SFB command) is used as the reference point.

SERVO AXES

The PSET offset value (per axis) is specific only to the feedback source (per axis) selected with the last SFB command.

If your application requires switching between feedback sources for the same axis, then you must select the feedback source with the appropriate SFB command and issue a PSET value specific to that feedback source. (Each feedback source can have a separate offset.)

NOTE: If you issue a PSET command, any previously captured positions (INFNCi-H or LIMFNCi-H function) will be offset by the PSET value.

If a software end-of-travel limit has been hit, the PSET command will not remove the error condition. The error condition is removed by commanding motion in the opposite direction.

Example:

```
PSET0,0,0,1000 ; Set absolute position on axes 1, 2, and 3 to zero,
                ; and axis 4 to 1000 units
```

[PSHF]		Net Position Shift		
Type	Following; Assignment or Comparison		Product	Rev
Syntax	See below		6K	5.0
Units	n/a			
Range	n/a			
Default	n/a			
Response	n/a			
See Also	FOLEN, FOLMAS, FSHFC, FSHFD, SCALE, SCLD, TPSHF			

The PSHF operator is used to assign to a numeric variable the value of the net (absolute) follower axis position shift that has occurred since that last FOLEN1 command. The position value will be the sum of all shifts performed on that axis, or axes, including decelerations due to limits, kill, or stop. The shift value is set to zero each time a new FOLEN1 command or a FOLMAS command (with a value other than zero) is issued.

If scaling is enabled (SCALE1), the PSHF value is scaled by the distance scaling factor (SCLD). If scaling is not enabled, the value is in commanded counts.

Syntax: VARn=aPSHF where “n” is the variable number and “a” is the axis number, or PSHF can be used in an expression such as IF(2PSHF>2345Ø). The PSHF command must be used with an axis specifier, or it will default to axis 1 (e.g., VAR1=1PSHF, IF(2PSHF>5ØØ), etc.).

Example:

```
IF(2PSHF>4.3) ; If axis 2 has shifted more than 4.3 user units in the
                ; positive direction, then do the IF statement
OUT.2=b1      ; Turn on onboard output #2
NIF           ; End of IF statement
VAR14=3PSHF   ; Set VAR14 to follower axis 3's position shift
```

[PSLV]		Current Commanded Position of Follower Axis		
Type	Following; Assignment or Comparison		Product	Rev
Syntax	See below		6K	5.0
Units	n/a			
Range	n/a			
Default	n/a			
Response	n/a			
See Also	FMCNEW, FMCP, SCLD, SCALE, TPSLV			

Use the PSLV operator to assign the follower axis commanded position register value to a variable, or to make a comparison against another value.

If scaling is enabled (SCALE1), the PSLV value is scaled by the distance scaling factor (SCLD). If scaling is not enabled, the value is in commanded counts.

Syntax: VARn=aPSLV where “n” is the variable number and “a” is the axis number, or PSLV can be used in an expression such as IF(2PSLV>2345Ø). The PSLV command must be used with an axis specifier, or it will default to axis 1 (e.g., VAR1=1PSLV, IF(2PSLV>5ØØ), etc.).

Example:

```
IF(2PSLV>4.3) ; If axis 2 has traveled more than 4.3 user units then do
                ; the IF statement
OUT.2=b1      ; Turn on onboard output #2
NIF           ; End of IF statement
VAR14=3PSLV   ; Set VAR14 to follower axis #3's position
```

PTAN Path Tangent Axis Resolution

Type	Path Contouring	Product	Rev
Syntax	<! <i>>PTAN<i></i>	6K	5.0
Units	<i>i</i> = counts (commanded counts for stepper axes, encoder or analog input counts for servo axes)		
Range	±1 - 999,999,999		
Default	4000		
Response	No response - Must be defining a path (DEF)		
See Also	PAXES		

The Path Tangent Axis Resolution (PTAN) command is used to specify the Tangent axis resolution. The Tangent axis will keep an angular position which changes linearly with the direction of travel implied by X and Y. This allows the Tangent axis to control an object which must stay tangent (or normal) to the direction of travel.

The Tangent axis resolution is the number of counts (motor steps for steppers; encoder or analog input counts) in 360 degrees of arc. The Tangent axis resolution does not necessarily equal axis resolution (DRES for steppers; ERES or analog input counts/volt for servos), but if the motor directly drove the rotating piece, then these numbers would be the same.

The PTAN command should be given prior to any contour segments during a path definition. A negative value for the Tangent axis resolution causes rotation in the negative direction as the angle in the X-Y plane gets larger.

Example:

```
PV5           ; Set path velocity to 5 units/sec
PA50          ; Set path acceleration to 50 units/sec/sec
PAD100        ; Set path deceleration to 100 units/sec/sec
DEF prog1     ; Begin definition of path named prog1
PAXES1,2,3    ; Set axes 1 and 2 as the X and Y contouring axes,
              ; 3 as the tangent axis
PTAN25000     ; Specify Tangent axis resolution
PAB0          ; Set to incremental coordinates
POUT1001      ; Output pattern during first arc (onboard outputs)
PARCM5,5,5    ; Specify incremental X-Y endpoint position and radius
              ; arc <180 degree for quarter circle counter-clockwise arc
POUT1100      ; Output pattern during second arc (onboard outputs)
PARCP5,-5,-5  ; Specify incremental X-Y endpoint position and radius
              ; arc >180 degree for three quarter circle clockwise arc
END           ; End definition of path prog1
PCOMP prog1   ; Compile path prog1
PRUN prog1    ; Execute path prog1
OUT0000       ; Turn off the first four onboard outputs
```

PUCOMP Un-Compile a Compiled Profile (includes Path Uncompile)

Type	Compiled Motion; Path Contouring; PLC Program	Product	Rev
Syntax	<!>PUCOMP<t>	6K	5.0
Units	t = text (name of path)		
Range	Text name of 6 characters or less		
Default	n/a		
Response	n/a		
See Also	DEF, END, GOBUF, MEMORY, PCOMP, PLCP, PRUN, SCANP, TDIR, TMEM, TSEG, GOBUF, PLOOP, PLN		

The Un-Compile (PUCOMP) command is used to delete a previously compiled (PCOMP) program from the compiled memory. The PUCOMP command does not delete the program from program memory.

Example:

```
PUCOMP prog1      ; Delete compiled motion segments for prog1
DEL prog1         ; Delete prog1
DEF prog1         ; Begin definition of path named prog1
PAXES1,2,3,4     ; Set axes 1,2,3,4 as the X, Y, Tangent, and
                  ; Proportional axes respectively
; *****
; * Add multiple path segment          *
; * definitions in this                *
; * portion of the                    *
; * program                            *
; *****
END               ; End definition of path prog1
PCOMP prog1      ; Compile path prog1
PRUN prog1       ; Execute path prog1
```

PULSE Pulse Width

Type	Controller Configuration	Product	Rev
Syntax	<!>@><a>PULSE<r>, <r>, <r>, <r>, <r>, <r>, <r>	6K	5.0
Units	r = microseconds (µs)		
Range	0.3, 0.5, 1.0, 2.0, 4.0, 8.0, or 16.0		(applicable to stepper axes only)
Default	0.3		
Response	PULSE: *PULSE0.3,0.3,0.3,0.3,0.3,0.3,0.3,0.3 1PULSE: *1PULSE0.3		
See Also	AXSDEF, DRES, V		

The Pulse Width (PULSE) command sets the step output pulse width. The pulse width is described as the time the pulse is active, or *on*. The value for the pulse width command is specified in microseconds.

When the pulse width is changed from the default value of 0.3 µs, the maximum velocity range is reduced. The amount of reduction is directly proportional to the change in pulse width (see table below).

Pulse Width (PULSE) Setting	Actual Pulse Width	Maximum Velocity
DEFAULT → 0.3 µs	0.244 µs	2.048 MHz
0.5 µs	0.484 µs	1.024 MHz
1.0 µs	0.976 µs	512 KHz
2.0 µs	1.953 µs	256 KHz
4.0 µs	3.906 µs	128 KHz
8.0 µs	7.812 µs	64 KHz
16.0 µs	15.624 µs	32 KHz

PV**Path Velocity**

Type	Path Contouring or Motion (Linear Interpolated)	Product	Rev
Syntax	<!>PV<r>	6K	5.0
Units	r = units/sec (scalable with SCLD)		
Range	Stepper Axes: 0.00000-2,048,000 (max. depends on SCLD & PULSE) Servo Axes: 0.00000-6,500,000 (max. depends on SCLD)		
Default	1.0000		
Response	PV: *PV1.0000		
See Also	GOL, SCLD, SCALE		

The Path Velocity (PV) command specifies the path velocity to be used in linearly interpolated moves (GOL), and in all contouring moves. In linearly interpolated moves, a path may involve one to four axes, each with its own distance of travel. In contouring paths, only the X and Y axis are included in the calculation of the path.

For both types of moves, the path velocity refers to the velocity of the load as motion proceeds along the path. For linearly interpolated moves, the velocity of each individual axis is dependent on the distance it contributes to the total path traveled by the load. In contouring paths, the velocity of each individual axis is dependent on the direction of travel in the X- Y plane. **NOTE:** *The PV value can be altered between path segments, but not within a path segment.*

UNITS OF MEASURE and SCALING: refer to page 16.

Example: Refer to Define Path Local Mode (PL) command example.

PWC**Path Work Coordinates**

Type	Path Contouring	Product	Rev
Syntax	<!>PWC<r>,<r>	6K	5.0
Units	1 st r = X coordinate units (scalable by the SCLD value) 2 nd r = Y coordinate units (scalable by the SCLD value)		
Range	±999,999,999.99999		
Default	0,0		
Response	No response - Must be defining a path (DEF)		
See Also	PAB, PL, PLC, SCLD, SCALE		

The Path Work Coordinates (PWC) command is used to specify the Work X -Y coordinate data required for subsequent segment definition in the Work coordinate system. This command places the X -Y coordinate value of the Work coordinate system at the beginning of the next segment. (The first <r> is the X coordinate, the second <r> is the Y coordinate.)

This command may be used before the PLØ command is given for the purpose of shifting the Work coordinate system. If the PWC command is not given before a PLØ command, but was previously set, the original work coordinate system is used for the subsequent segments.

UNITS OF MEASURE and SCALING: refer to page 16 or to the SCLD description.

Example: Refer to Define Path Local Mode (PL) command example.

RADIAN Radian Enable

Type	Operators (Trigonometric)	Product	Rev
Syntax	<!>RADIAN	6K	5.0
Units	n/a		
Range	b = 0 (Disable), 1 (Enable) or X (don't care)		
Default	0		
Response	RADIAN: *RADIAN0		
See Also	[ATAN], [COS], [PI], [SIN], [TAN], VAR		

This operator is used to switch between radians and degrees. The command RADIAN1 specifies units in radians for SIN, COS, TAN, and ATAN. The command RADIANØ specifies units in degrees for SIN, COS, TAN, and ATAN.

If a value is given in radians and a conversion is needed to degrees, use the formula: $360^\circ = 2\pi$ radians.

Example:

```
RADIAN1                    ; Set trigonometric functions to radian mode
```

RE Registration Enable

Type	Registration	Product	Rev
Syntax	<!><@><a>RE	6K	5.0
Units	b = 0 (disable), 1 (enable), or X (don't care)		
Range	n/a		
Default	0		
Response	RE: *RE0000_0000 1RE: *1RE0		
See Also	[AS], COMEXC, ENCCNT, [ER], INFNC, [PCC], [PCE], [PCMS], REG, REGLOD, REGSS, TAS, TER, TPCC, TPCE, TPCMS, TRGLOT, [TRIG], TTRIG		

The Registration Enable (RE) command enables the registration function for the specified axes.

When a registration input (a trigger input assigned the “Trigger Interrupt” function) is activated, the motion profile currently being executed is replaced by a *registration profile* with its own distance (REG), acceleration (A & AA), deceleration (AD & ADA), and velocity (V) values. The registration move may interrupt any preset, continuous, or registration move in progress.

The registration move does not alter the rest of the program being executed when registration occurs, nor does it affect commands being executed in the background if the controller is operating in the continuous command execution mode (COMEXC1).

Registration moves will not be executed while the motor is not performing a move, while in the joystick mode (JOY1), or while decelerating due to a stop, kill, soft limit, or hard limit.

How to Set up a Registration Move

1. Configure one of the trigger inputs (TRG-nA or TRG-nB per axis, or TRG-M) to function as a trigger interrupt input; this is done with the INFNCi-H command, where i is the input bit number representing the targeted trigger input.
2. Specify the distance of the registration move with the REG command. For servo axes, the distance refers to the encoder position (not functional with ANI feedback). For stepper axes, the distance refers to commanded position.
3. Enable the registration function with the RE command. Registration is performed only on the axis or axes with the registration function enabled, and with a non-zero distance specified in the respective axis-designation field of the REG command; the other axes will not be affected. Each trigger has a distinct move defined for its dedicated axis.

NOTE: The registration move is executed using the A, AA, AD, ADA, and V values that were in effect when the REG command was entered.

Registration Move Accuracy (see also Registration Move Status below)

The accuracy of the registration move distance specified with the REG command is ± 1 count (servo axes: encoder count; stepper axes: commanded count if ENCCNT0 or encoder count if ENCCNT1).

RULE OF THUMB: To prevent position overshoot, make sure the REG distance is greater than 4 ms multiplied by the incoming velocity.

The lapse between activating the registration input and commencing the registration move (this does not affect the move accuracy) is less than one position sample period (2 ms).

The REG distance will be scaled by the distance scale factor (SCLD value) if scaling is enabled (SCALE1). See page 16 for details on scaling.

Preventing Unwanted Registration Moves (methods)

- **Registration Input Debounce:** Registration Input Debounce: By default, the registration inputs are debounced for 24 ms before another input on the same trigger is recognized. (The debounce time is the time required between a trigger's initial active transition and its secondary active transition.) Therefore, the maximum rate that a registration input can initiate registration moves is 500 times per second. If your application requires a shorter debounce time, you can change it with the TRGLOT command.
- **Registration Single-Shot:** The REGSS command allows you to program the 6K controller to ignore any registration commands after the first registration move has been initiated. Refer to the REGSS command description for further details and an application example.
- **Registration Lockout Distance:** The REGLD command specifies what distance an axis must travel before any trigger assigned as a registration input will be recognized. Refer to the REGLD command description for further details and an application example.

Registration Move Status & Error Handling

Axis Status — Bit #28: This status bit is set when a registration move has been initiated by any registration input (trigger). This status bit is cleared with the next GO command.

AS.28.....Assignment & comparison operator — use in a conditional expression.
TASF.....Full text description of each status bit. (see “Reg Move Commanded” line item)
TAS.....Binary report of each status bit (bits 1-32 from left to right). See bit #28.

Axis Status — Bit #30: If, when the registration input is activated, the registration move profile cannot be performed with the specified motion parameters, the 6K controller will kill the move in progress and set axis status bit #30. This status bit is cleared with the next GO command.

AS.30.....Assignment & comparison operator — use in a conditional expression.
TASF.....Full text description of each status bit. (see “Preset Move Overshot” line item)
TAS.....Binary report of each status bit (bits 1-32 from left to right). See bit #30.

Error Status — Bit #10: This status bit may be set if axis status bit #30 is set. The error status is monitored and reported only if you enable error-checking bit #10 with the ERROR command (e.g., ERROR.10-1). NOTE: When the error occurs, the controller will branch to the error program (assigned with the ERRORP command). This status bit is cleared with the next GO command.

ER.10.....Assignment & comparison operator — use in a conditional expression.
TERF.....Full text description of each status bit. (see “Preset Move Overshot” line item)
TER.....Binary report of each status bit (bits 1-32 from left to right). See bit #10.

Trigger Status — Bits #1-17: Trigger status bits are set when a registration move has been initiated by trigger inputs A or B for each axis, or with the TRIG-M (master trigger) input. This also indicates that the positions of all axes has been captured. As soon as the captured information is transferred or assigned/compared, the respective trigger status bit is cleared (set to \emptyset).

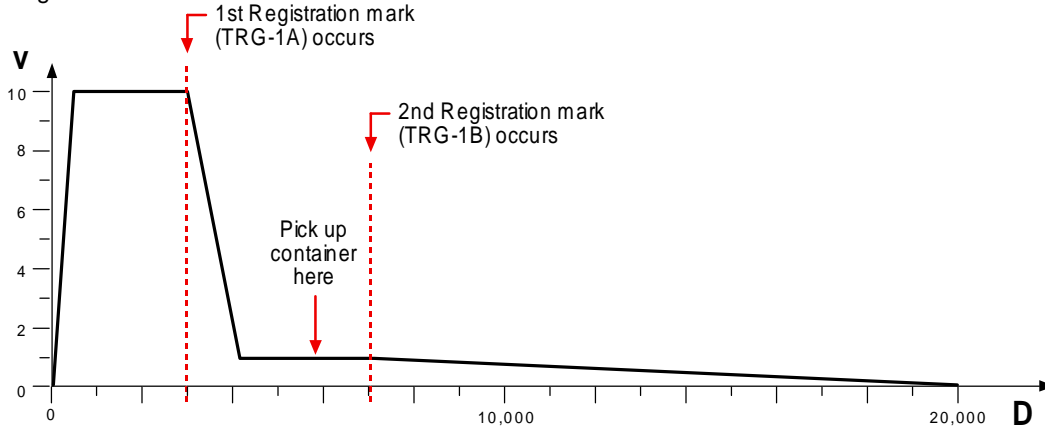
TRIG.....Assignment & comparison operator — use in a conditional expression.
TTRIG.....Binary report of each status bit (bits 1-17 from left to right). From left to right the bits represent trigger A and B for axes 1-8, the 17th bit is master trigger M (the “MASTER TRIG” input terminal) — see page 7.

Example:

In this example (using axis 1), two-tiered registration is achieved. While axis 1 is executing it's 50,000-unit move, trigger input 1A is activated and executes registration move A to slow the load's movement. An open container of volatile liquid is then placed on the conveyor belt. After picking up the liquid and while registration move A is still in progress, trigger input 1B is activated and executes registration move B to slow the load to gentle stop.

```
DEL REGI1      ; Delete program (assume program already resides in memory)
DEF REGI1      ; Begin program definition
INFNC1-H      ; Define trigger input 1A (axis 1) as a trigger interrupt input
INFNC2-H      ; Define trigger input 1B (axis 1) as a trigger interrupt input
A20           ; Set acceleration on axis 1 to 20 units/sec/sec
AD40          ; Set deceleration on axis 1 to 40 units/sec/sec
V1            ; Set velocity on axis 1 to 1 unit/sec
1REGA4000     ; Set trigger 1A's registration distance on axis 1 to 4000 units
              ; (registration A move will use the A, AD, & V values above)
A5            ; Set acceleration on axis 1 to 5 units/sec/sec
AD2           ; Set deceleration on axis 1 to 2 units/sec/sec
V.5           ; Set velocity on axis 1 to 0.5 units/sec
1REGB13000    ; Set trigger 1B's registration distance on axis 1 to 13,000 units
              ; (registration B move will use the A, AD, & V values above)
RE1           ; Enable registration on axis 1 only
A50           ; Set acceleration to 50 units/sec/sec on axis 1
AD50          ; Set deceleration to 50 units/sec/sec on axis 1
V10          ; Set velocity to 10 unit/sec on axis 1
D50000       ; Set distance to 50000 units on axis 1
GO1           ; Initiate motion on axis 1
END           ; End program definition
```

Registration Profile:



[READ] Read a Value

Type	Communication Interface or Assignment	Product	Rev
Syntax	... READi ... (See below)	6K	5.0
Units	i = string variable number		
Range	1-50		
Default	n/a		
Response	n/a		
See Also	' , PORT, [SS], TSS, VAR, VARS, WRITE		

The Read a Value (READ) command provides the user with an efficient way of storing numeric data read from the input buffer into a variable. The READ command can be used as part of a numeric variable assignment statement (e.g., VAR1=READ1) or in another command (A1Ø, (READ1), 12, 1). However, the READ command cannot be used in an expression such as VAR5=1+READ1 or IF (READ1=1).

Syntax: VARx=READi where x is the variable number and i is the string variable to be sent out to prompt the user for the numeric information.

Syntax: Command(READi) where Command is any command that has a separate field (e.g., A, AD, V, D, etc.), and i is the string variable number.

The number attached to the end of the READ command corresponds to the string variable to be sent out the Ethernet port or the RS-232 or RS-485port, at the time this command is executed. The 6K Series controller will then wait for numeric data to be sent to its input buffer. **The numeric data must be preceded with an immediate command identifier and a single quote (! ')**. The information read in can be either integer, or real, and must be terminated by a command delimiter (:, <cr>, <lf>).

Rule of Thumb for command value substitutions: If the command syntax shows that the command field requires a real number (denoted by <r>) or an integer value (denoted by <i>), you can use the READ substitution (e.g., V2, (READ)).

Example:

```
VAR1="Enter the count >" ; Place message in string variable #1
VAR2=READ1                ; Prompt with string variable #1, and read data
                           ; into variable #2
;
; The controller will send this message (string variable #1) to the screen:
;                               "Enter the count >"
; The user must enter the numeric data preceded by the characters !'.
; For example, !'82.5 assigns the value 82.5 to numeric variable 2
```

REG		Registration Distance		Product	Rev
Type	Registration			6K	5.0
Syntax	<!><@>aREGc<r>				
Units	a = axis # c = letter of trigger input r = distance units (scalable by the SCLD value) servo axes: always encoder counts (ANI input not allowed) stepper axes: commanded counts				
Range	a = 1-8 (depending on product) c = A or B r = 0.00000 to 419,430,000.00000 (positive direction only)				
Default	0 (do not make a registration move)				
Response	1REGA: *1REGA0				
See Also	[AS], ENCCNT, [ER], [PCC], [PCE], [PCMS], RE, REGLD, REGSS, SCALE, SCLD, [SS], TAS, TER, TPCE, TRGLOT, [TRIG], TTRIG				

The Registration Distance (REG) command specifies the distance the corresponding axis will travel after receiving a registration input (trigger A or B). Example: 1REGA4000 sets up a 4000-count registration move on axis 1 to be initiated when trigger input 1A is activated.

Servo Axes: REG value always represents encoder counts (registration cannot be used with analog input feedback).

Stepper Axes: REG value represents commanded counts.

Trigger Input (Axis 1-4 "TRIGGERS/OUTPUTS" connector) * Axis			Dedicated	REG	Trigger Input (Axis 5-8 "TRIGGERS/OUTPUTS" connector) * Axis			Dedicated	REG
			Axis	Syntax				Axis	Syntax
Pin 23,	Trigger 1A		1	1REGA	Pin 23,	Trigger 5A		5	5REGA
Pin 21,	Trigger 1B		1	1REGB	Pin 21,	Trigger 5B		5	5REGB
Pin 19,	Trigger 2A		2	2REGA	Pin 19,	Trigger 6A		6	6REGA
Pin 17,	Trigger 2B		2	2REGB	Pin 17,	Trigger 6B		6	6REGB
Pin 15,	Trigger 3A		3	3REGA	Pin 15,	Trigger 7A		7	7REGA
Pin 13,	Trigger 3B		3	3REGB	Pin 13,	Trigger 7B		7	7REGB
Pin 11,	Trigger 4A		4	4REGA	Pin 11,	Trigger 8A		8	8REGA
Pin 9,	Trigger 4B		4	4REGB	Pin 9,	Trigger 8B		8	8REGB

* The number of trigger inputs available varies by product (refer to your product's *Installation Guide*).

The registration move is executed using the A, AA, AD, ADA, and V values that were in effect when the REG command was entered.

RULE OF THUMB: To prevent position overshoot, make sure the REG distance is greater than 4 ms multiplied by the incoming velocity.

The registration distance remains set until you change it with a subsequent REG command. Registration distances outside the valid range are flagged as an error, returning the message *INVALID DATA-FIELD x, where x is the field number.

UNITS OF MEASURE and SCALING: refer to page 16.
--

For additional details on Registration (including programming examples), refer to the RE command description and to the Registration section in the *Programmer's Guide*.

REGLOD Registration Lock-Out Distance

Type	Registration	Product	Rev
Syntax	<!><@>REGLOD<r>, <r>, <r>, <r>, <r>, <r>, <r>, <r>	6K	5.0
Units	r = distance units (scalable by SCLD)		
Range	0.00000 to +999,999,999.99999		
Default	0		
Response	REGLOD: *REGLOD0,0,0,0,0,0,0,0 1REGLOD: *1REGLOD0		
See Also	INFNC, RE, REG, REGSS, SCLD, TRGLOT		

The REGLOD command specifies the distance an axis must travel before its registration input will be recognized. If scaling is enabled (SCALE1), the lock-out distance is scaled by the SCLD value.

Stepper axes: The lock-out distances are measured incrementally from the start of motion to the commanded position.

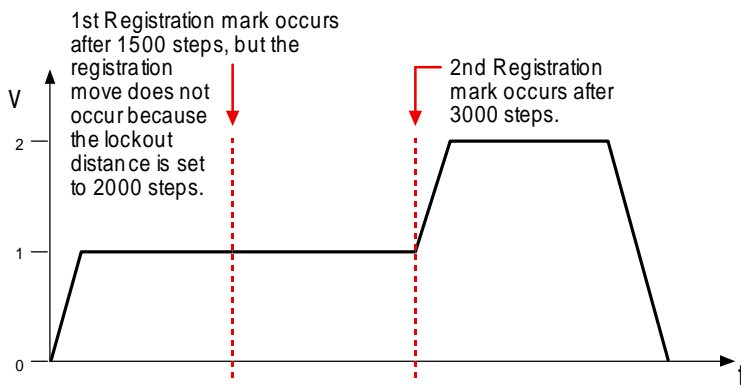
Servo axes: The lock-out distances are measured incrementally from the start of motion to the actual position (as measured by the position feedback device), not the commanded position.

Example (single axis, stepper):

A print wheel uses registration to initiate each print cycle. From the beginning of motion, the controller should ignore all registration marks before traveling 2000 steps. This is to ensure that the unit is up to speed and that the registration mark is a valid one.

```
DEL REGI3      ; Delete program (in case program already resides in memory)
DEF REGI3      ; Begin program definition
INFNC1-H      ; Trigger capture mode for trigger 1A on axis 1
RE1           ; Enable registration
V2            ; Set registration move to a velocity of 2 revs/sec
1REGA2500     ; and a distance of 2500 steps
REGLOD2000    ; Set registration lockout distance to 2000 steps
MC1           ; Start a mode continuous move
V1            ; move at a velocity of 1 rps
GO1           ; Initiate motion
END           ; End program definition
```

Registration Profile:



To check the status of the registration input:

```
> !TTRIG.1    Check to see if trigger interrupt input 1A has been activated
*0            Indicates registration move has not happened on any axis
```

REGSS Registration Single-Shot

Type	Registration	Product	Rev
Syntax	<!><@>REGSS	6K	5.0
Units	n/a		
Range	b=0 (Disable), 1 (Enable), or x (don't care)		
Default	0		
Response	REGSS: *REGSS0000_0000 1REGSS: *1REGSS0		
See Also	RE, REG, REGLOD		

The Registration Single Shot (REGSS) command sets the registration such that only one registration move will take place for the specified axis. This allows the user to prevent any other trigger from interrupting the registration move in progress. A GO command will reset the “one shot” condition.

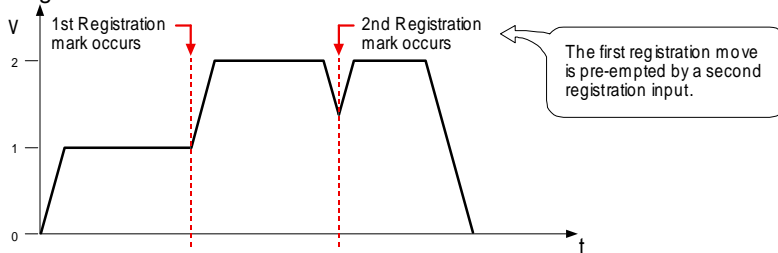
Example – Option A:

A user has a line of material with randomly spaced registration marks. It is known that the first mark must initiate a registration move, and that each registration move cannot be interrupted or the end product will be destroyed. Since the distance between marks is random, it is impossible to predict if a second registration mark will occur before the first registration move has finished.

```

DEL REGI2      ; Delete program (in case program already resides in memory)
DEF REGI2      ; Begin program definition
INFNC1-H      ; Trigger capture mode for trigger 1A on axis 1
RE1           ; Enable registration
V2            ; Set registration move to a velocity of 2 rps
AD.5         ; a deceleration of 0.5 rev/sec/sec
1REGA20000    ; and a distance of 20000 steps
MC1          ; Start a mode continuous
V1           ; move at a velocity of 1 rps
GO1          ; Initiate motion
END          ; End program definition
  
```

Registration Profile:



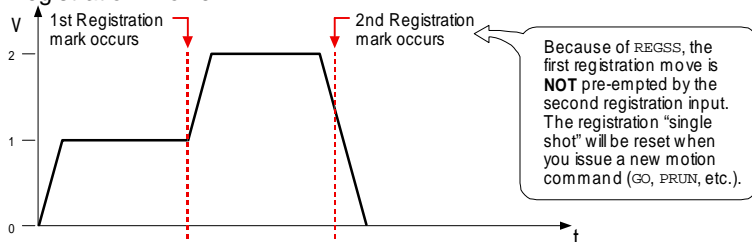
Example – Option B (introducing “single-shot” registration):

In order to stop the second registration from occurring, REGSS can be used:

```

DEL REGI2b    ; Delete program (in case program already resides in memory)
DEF REGI2b    ; Begin program definition
INFNC1-H      ; Trigger capture mode for trigger 1A on axis 1
RE1           ; Enable registration
V2            ; Set registration move to a velocity of 2 rps
1REGA20000    ; and a distance of 20000 steps
REGSS1       ; Enable registration single shot mode
MC1          ; Start a mode continuous
V1           ; move at a velocity of 1 rps
GO1          ; Initiate motion
END          ; End program definition
  
```

Registration Profile:



REPEAT Repeat Statement

Type	Program Flow Control or Conditional Branching	Product	Rev
Syntax	<!>REPEAT	6K	5.0
Units	n/a		
Range	n/a		
Default	n/a		
Response	n/a		
See Also	JUMP, UNTIL		

The Repeat Statement (REPEAT) command, in conjunction with the UNTIL command, provide a means of conditional program flow. The REPEAT command marks the beginning of the conditional statement. The commands between the REPEAT and the UNTIL command are executed at least once. Upon reaching the UNTIL command, the expression contained within the UNTIL command is evaluated. If the expression is false, the program flow is redirected to the first command after the REPEAT command. If the expression is true, the first command after the UNTIL command is executed.

Up to 16 levels of REPEAT . . . UNTIL() commands may be nested.

NOTE: Be careful about performing a GOTO between REPEAT and UNTIL. Branching to a different location within the same program will cause the next REPEAT statement encountered to be nested within the previous REPEAT statement, unless an UNTIL command has already been encountered. The JUMP command should be used in this case.

All logical operators (AND, OR, NOT), and all relational operators (=, >, >=, <, <=, <>) can be used within the UNTIL expression. There is no limit on the number of logical operators, or on the number of relational operators allowed within a single UNTIL expression.

The limiting factor for the UNTIL expression is the command length. The total character count for the UNTIL command and expression cannot exceed 80 characters. For example, if you add all the letters in the UNTIL command and the letters within the () expression, including the parentheses and excluding the spaces, this count must be less than or equal to 80.

All assignment operators (A, AD, AS, D, ER, IN, INO, LIM, MOV, OUT, PC, PCE, PCME, PCMS, PE, PER, SS, TIM, US, V, VEL, etc.) can be used within the UNTIL() expression.

Example:

```
REPEAT          ; Beginning of REPEAT ... UNTIL( ) loop
GO1110         ; Initiate motion on axes 1, 2, and 3
VAR1=VAR1+1    ; Increment variable 1 by 1
UNTIL(VAR1=12) ; Repeat loop until variable 1 = 12
```

RESET

Reset

Type	Communication Interface	Product	Rev
Syntax	<!>RESET	6K	5.0
Units	n/a		
Range	n/a		
Default	n/a		
Response	RESET: ((power-up message is displayed))		
See Also	STARTP, TSTAT		

The Reset (RESET) command affects the 6K controller the same as cycling power. The controller's programs and variables are retained in non-volatile memory; however, all previously entered command values (not saved in programs or variables) will be reset to factory default values.

NOTE: After sending the RESET or !RESET command to the 6K product, you must wait until you see the power-up message (actual time varies by product) before communicating with the product.

CAUTION: The RESET command will disconnect an Ethernet connection.

RUN

Begin Executing a Program

Type	Program or Subroutine Definition	Product	Rev
Syntax	<!>RUN<t>	6K	5.0
Units	t = text (name of program)		
Range	Text name of 6 characters or less		
Default	n/a		
Response	n/a		
See Also	\$, DEF, DEL, END, GOSUB, GOTO		

The Begin Executing a Program (RUN) command executes a program defined with the DEF command. A program name consists of 6 or fewer alpha-numeric characters. The RUN command can be used inside a program or subroutine. The program can also be run by specifying the name of the program without the RUN command. The RUN command functions similar to a GOSUB command in that control returns to the original program when the called program finishes.

Example:

```
DEF pick          ; Begin definition of program named pick
GO1100           ; Initiate motion on axes 1 and 2
END              ; End program definition
RUN pick         ; Executes program named pick
pick             ; Executes program named pick
```