Motion

Application Specific Function Block Manual

Version 16.1.1

G & L Motion Control Inc.

NOTE

Progress is an on-going commitment at G & L Motion Control Inc. We continually strive to offer the most advanced products in the industry; therefore, information in this document is subject to change without notice. The illustrations and specifications are not binding in detail. G & L Motion Control Inc. shall not be liable for any technical or editorial omissions occurring in this document, nor for any consequential or incidental damages resulting from the use of this document.

DO NOT ATTEMPT to use any G & L Motion Control Inc. product until the use of such product is completely understood. It is the responsibility of the user to make certain proper operation practices are understood. G & L Motion Control Inc. products should be used only by qualified personnel and for the express purpose for which said products were designed.

Should information not covered in this document be required, contact the Customer Service Department, G & L Motion Control Inc., 672 South Military Road, P.O. Box 1960, Fond du Lac, WI 54936-1960. Telephone us at (800) 558-4808 (within the United States only) or (920) 921–7100 or by e-mail at glmotion.support@danahermotion.com.

DISCLAIMER: All programs in this release (application demos, application specific function blocks (ASFB's), etc.), are provided “AS IS, WHERE IS”, WITHOUT ANY WARRANTIES, EXPRESS OR IMPLIED. There may be technical or editorial omissions in the programs and their specifications. These programs are provided solely for user application development and user assumes all responsibility for their use. Programs and their content are subject to change without notice.

Release 2004

Catalog No. (Order No.) M.1300.7573

Printed Version Part No. M.3000.0416

Electronic Version Part No. M. 3000.0415

Table of Contents: Motion ASFB Manual

CHAPTER 1 Application Specific Function Block Guidelines

Installation

The following guidelines are recommended ways of working with Application Specific Function Blocks (i.e. ASFBs) from Danaher Motion.

The Applications CD includes the ASFB package as follows:

- .LIB file(s) containing the ASFB(s)
- source .LDO(s) from which the ASFB(s) was made
- example LDO(s) with the ASFB(s) incorporated into the ladder which you can then use to begin programming from or merge with an existing application ladder

When you install the Applications CD, the ASFB paths default to:

C:\G&L Motion Control Data\Applications **v*xx.x.r***\ASFB

and

C:\G&L Motion Control Data\Applications **v*xx.x.r***\Examples

where **v*xx.x*** *is the PiCPro version number that these ASFBs and examples were built under. The* ***.r*** *is the revision number of the Application software itself.*

The .LIB files and source .LDO files are put in the ASFB subdirectory. The example .LDO files are put in the Examples subdirectory.

Revisions

The first four networks of each ASFB source ladder provide the following information:

Network 1

The first network just informs you that the ASFB is provided to assist your application development.

Network 2

The second network is used to keep a revision history of the ASFB. Revisions can be made by Danaher Motion personnel or by you.

The network identifies the ASFB, lists the requirements for using this ASFB, the name of the library the ASFB is stored in, and the revision history.

The revision history includes the date, ASFB version (see below), the version of PiCPro used while making the ASFB, and comments about what the revision involved.

When an ASFB is revised, the number of the first input (EN_ _ or RQ_ _) to the function block is changed in the software declarations table. The range of numbers available for Danaher Motion personnel is 00 to 49. The range of numbers available for you is 50 to 99. See chart below.

Revision	Danaher Motion revisions	User revisions
1st	EN00	EN50
2nd	EN01	EN51
.	.	.
.	.	.
.	.	.
50th	EN49	EN99

Network 3

The third network describes what you should do if you want to make a revision to the ASFB.

Network 4

The fourth network describes the ASFB and defines all the inputs and outputs to the function block.

Using ASFBs

When you are ready to use the ASFB in your application, there are several approaches you can take as shown below.

- Create a new application LDO starting with the example LDO for the ASFB package. The advantage is that the software declarations table for the ASFB has been entered for you.
- If you already have an application LDO, copy and paste the example LDO into yours. The software declaration tables for both LDOs will also merge.

NOTES

CHAPTER 2 Motion ASFBs

The motion support function blocks are contained in the libraries as shown. They are used to aid in the application of servo and digitizing axes. Included with these library files are other example LDO files as listed. The motion support function blocks are described in alphabetical order.

The SERCOS motion function blocks are also shown. They are used to aid in the application of SERCOS servo and digitizing axes. Their names start with S_. They are written to replace the corresponding M_ motion function block, when the axes use SERCOS control rather than analog control.

Library	Function Block	Description
M_C2M_1		
	M_C2M_1	Translates a M and G code format ASCII file into servo motion.
M_COMMON		
	BYTE2HEX	Places the data type byte into hexadecimal notation.
	DWOR2HEX	Places the data type double word into hexadecimal notation.
	HEX2BYTE	Places the hexadecimal notation into a byte.
	HEX2DWOR	Places the hexadecimal notation into a double word.
	HEX2WORD	Places the hexadecimal notation into a word.
	M_DW2BOO	Places the data type double word into 32 booleans
	WORD2HEX	Places the data type from word into hexadecimal notation.
M_DATA		
	M_DATCAP	Captures axis information on an interrupt basis.
	M_DATCPT	Captures axis information on an interrupt basis to printable text file
	M_ERROR	Returns E-stop, C-stop, and programming errors for a servo axis or E-stop errors for a digitizing axis.
	M_INCPTR	Increment buffer pointers for M_DATCPT (not used in your LDO).
	M_PRTCAM	Creates a text file for the CAM input of RATIOCAM.
	M_PRTREL	Creates a text file for the REAL input of RATIO_RL.
	M_PRTSLP	Creates a text file for the SLOPE input of RATIOSLP.
	M_RATREL	Calculates ending ratio and slope for use in ratio real profile.
	M_RATSLP	Calculates ending ratio and slope for use in ratio slope profile.
	M_RDTUNE	Reads tuning parameters for a closed loop axis.
	M_RGSTAT	Returns registration information for a closed loop or digitizing axis.

M_STATUS	Returns status information (for example, position and following error) for a closed loop, time, or digitizing axis.
M_WTTUNE	Changes tuning parameters on a closed loop axis

M_DEVNET

M_DNJOGC	Jogs a Centurion DeviceNet drive axis
M_DNPOSC	Moves a Centurion DeviceNet drive to a position (either absolute or incremental)
M_DNSTAT	Obtains the DeviceNet module status

M_DRVCOM

ADDCKSUM	Support routine for M_DSMCOM. (Not used in your LDO.)
CHKCKSUM	Support routine for M_DSMCOM. (Not used in your LDO.)
M_DSCOM	Allows interfacing between the PiC and one or more Centurion DS100/200 servo drives.

M_INDEX

INDXFILE	Support routine for INDXFILE. (Not used in your LDO.)
M_INDEX	Single axis indexer / motion sequencer

M_INIT

M_CHK1	Checks to see which servo axes (1 to 8) have been initialized.
M_CHK101	Checks to see which servo axes 101 to 108 (17 to 24) have been initialized.
M_CHK109	Checks to see which servo axes 109 to 116 (25 to 32) have been initialized.
M_CHK49	Checks to see which digitizing axes (49 to 56) have been initialized.
M_CHK57	Checks to see which digitizing axes (57 to 64) have been initialized.
M_CHK65	Checks to see which digitizing axes (65 to 72) have been initialized.
M_CHK73	Checks to see which digitizing axes (73 to 80) have been initialized.
M_CHK9	Checks to see which servo axes (9 to 16) have been initialized.
M_CHKALL	Checks to see which servo axes (1 to 132) have been initialized.
M_CLOS1	Closes the loop on servo axes 1 to 8.
M_CLOS9	Closes the loop on servo axes 9 to 16.
M_CLS101	Closes the loop on servo axes 101 to 108 (17 to 24).
M_CLS109	Closes the loop on servo axes 109 to 116 (25 to 32).
M_CLSALL	Closes the loop on servo axes 1 to 132.
M_RSET49	Resets E-stop errors on digitizing axes 49 to 56.

M_RSET57	Resets E-stop errors on digitizing axes 57 to 64.
M_RSET65	Resets E-stop errors on digitizing axes 65 to 72.
M_RSET73	Resets E-stop errors on digitizing axes 73 to 80.

M_MOVE

M_DISMV1	Performs and monitors distance moves.
M_JOG	Jogs a closed loop axis.
M_LINCIR	Performs linear, circular, and simultaneous endpoint arrival moves on closed loop axes.
M_POSMV1	Performs and monitors position moves.
M_REGMOV	Performs an incremental distance / registration move.
M_SACC	Calculates the ACC and JERK values to be used with the ACC_JERK function.
M_SCRVLC	Provides the interface from the application .LDO to the RATIO_RL function in order to perform linear coordinated, circular, or third axis departure (simultaneous endpoint arrival) moves with S-curve acceleration and deceleration.
M_SUPMV	Add move to geared axis.

M_PROFL

M_CNST_V	Constant velocity segment.
M_PRF1MV	One slave move for master.
M_PRF2MV	Two slave moves for master
M_PRFDWL	Slave dwell in profile.
M_PRFERR	Check for profile errors.
M_PROFL	Make profile for 1 move
M_SC_ACC	Acceleration segment.
M_SC_DEC	Deceleration segment
M_SETVAJ	Set velocity, acceleration, and jerk values.

M_REF

M_CHOME	Performs a reference cycle on an axis using a hard mechanical stop as a location sense.
M_CRSFIN	Implements coarse, medium and fine resolvers.
M_FHOME	Performs a home cycle on a closed loop axis using the fast input as the reference switch.
M_LHOME	Performs a home cycle on a closed loop axis using a discrete input as the reference switch.

M_SERCOS

M_SRCMON	Monitors up to five SERCOS IDNs.
M_SRCPRC	Executes a SERCOS procedure command function.
M_SRCRDL	Reads a list of SERCOS IDNs.
M_SRCWT	Writes and reads up to five SERCOS IDNs.
M_SRCWTL	Writes a list of SERCOS IDNs.

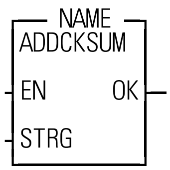
M_XL2CM		
	M_XL2CM	Converts an ASCII CSV file in the correct format to a data structure that can be called directly by RATIOCAM, RATIOSLP or RATIO_RL.
S_ASFB		
	S_CLOS1	Closes the loop on SERCOS servo axes 1 to 8 (to replace M_CLOS1)
	S_CLOS9	Closes the loop on SERCOS servo axes 9 to 16 (to replace M_CLOS9)
	S_ERRORC	Returns e-stop, c-stop, and programming errors for a SERCOS servo axis or e-stop errors for a SERCOS digitizing axis; SERCOS ring and slave errors are also returned (to replace M_ERROR for SERCOS axis).
	S_FHOME	Performs a home cycle on a SERCOS servo axis using the fast input as the reference switch (to replace M_FHOME for SERCOS axis)
	S_IO_C	Allows control of the discrete I/O for a SERCOS servo axis with a Centurion drive.
	S_LHOME	Performs a home cycle on a SERCOS servo axis using a discrete input as the reference switch (to replace M_LHOME for SERCOS axis)
SD_AXIS		
	SD_AXIS	Performs MMC Digital Smart Drive axis and drive status, loop control, home and jog functions.
SD_DATA		
	SD_IO	Reads the current values of the analog and DC inputs and reads/writes the outputs for the MMC Digital Smart Drive.
	SD_STAT	Provides status and fault information using the READ_SV variables and other common Motion functions for an MMC Digital Servo Axis.
SD_STAT1		
	SD_STAT1	Provides status and fault information using the READ_SV variables and other common Motion functions for an MMC Digital Servo Axis.

The following example LDOs are included:

M_CAMREL	An example .LDO that uses the M_RATREL function block to convert a RATIOCAM profile to a RATIO_RL profile.
M_CAMSLP	An example .LDO that uses the M_RATSLP function block to convert a RATIOCAM profile to a RATIO-SLP profile. The M_PRTSLP function block is then used to print the RATIOSLP profile.
M_CAPTUR	An example .LDO that shows how to use the M_DATCAP function block.
M_COORD	An example .LDO that uses the M_LINCIR function block to perform linear and circular coordinated moves on a pair of axes.
M_DSM_EX	An example .LDO that uses the M_DSMCOM function block to communicate with Centurion drives through a serial communications board in rack 0, slot 10, channel 2.
M_EXAMPL	An example .LDO that shows how to use the M_CHK1, M_CHK49, M_CLOS1, M_CRSFIN, M_ERROR, M_FHOME, M_JOG, M_LHOME, M_RGSTAT, M_RSET49, and M_STATUS function blocks.
M_PRF_EX	An example .LDO that shows how to use the M_PRF2MV function block to configure a slave profile for a RATIO_RL move.
M_TUNE	An example .LDO that shows how to use the M_RDTUNE and M_WTTUNE function blocks.
MMC_DND	An example .LDO that controls a Centurion DeviceNet drive axis. The axis is homed, jogged or moved to a position (either an absolute position or a relative distance).

ADDCKSUM

Add checksum to string **USER/M_DRVCOM**



Inputs: EN (BOOL) - enables execution

STRG(STRING) - input string

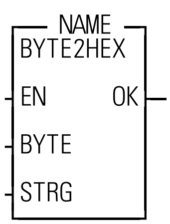
Outputs: OK (BOOL) - execution complete

ADDCKSUM(EN := <<BOOL>>, OK => <<BOOL>>);

This function block appends the one-byte checksum to the end of an input string.This is a support routine and is not used in your LDO.

BYTE2HEX

Converts a byte to a hex value **USER/M_COMMON**



Inputs: EN (BOOL) - enables execution

BYTE (BYTE) - value to convert

STRG(STRING) - converted value

Outputs: OK (BOOL) - execution complete

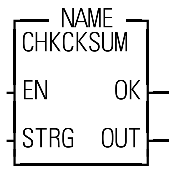
<<INSTANCE NAME>>:BYTE2HEX(EN := <<BOOL>>, BYTE := <<BYTE>>, STRG := <<STRING>>, OK => <<BOOL>>);

This function block places the hexadecimal notation of the value at BYTE into the string at STRG.

Example: If 27 is entered at the BYTE input, 1B will be reported at STRG.

CHKCKSUM

Check checksum in string **USER/M_DRVCOM**



Inputs: EN (BOOL) - enables execution

STRG(STRING) - input string to check

Outputs: OK (BOOL) - execution complete

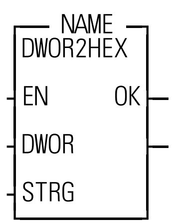
OUT (BOOL) - checksum OK output

CHKCKSUM(EN := <<BOOL>>, STRG := <<STRING>>, OK => <<BOOL>>, OUT => <<BOOL>>);

This function block checks the checksum in an input string. This is a support routine and is not used in your LDO.

DWOR2HEX

Converts a double word to a hex value **USER/M_COMMON**



Inputs: EN (BOOL) - enables execution

DWOR (DWORD) - value to convert

STRG (STRING) - converted value

Outputs: OK (BOOL) - execution complete

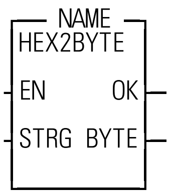
<<INSTANCE NAME>>:DWOR2HEX(EN := <<BOOL>>, DWOR := <<DWORD>>, STRG := <<STRING>>, OK => <<BOOL>>);

This function block places the hexadecimal notation of the value at DWOR into the string at STRG.

Example: If 845,621 is entered at the DWOR input, CE735 will be reported at STRG.

HEX2BYTE

Converts a hex value to a byte **USER/M_COMMON**



Inputs: EN (BOOL) - enables execution

STRG (STRING) - hexadecimal value to convert

Outputs: OK (BOOL) - execution complete

BYTE (BYTE) - converted value

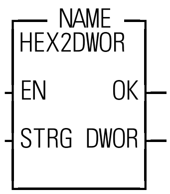
<<INSTANCE NAME>>:HEX2BYTE(EN := <<BOOL>>, STRG := <<STRING>>, OK => <<BOOL>>, BYTE => <<BYTE>);

This function block places the hexadecimal notation of the string at STRG into the output at BYTE.

Example: If 1B is entered at the STRG input, 27 will be reported at the BYTE output.

HEX2DWOR

Converts a hex value to a double word **USER/M_COMMON**



Inputs: EN (BOOL) - enables execution

STRG (STRING) - hexadecimal value to convert

Outputs: OK (BOOL) - execution complete

DWOR (DWORD) - converted value

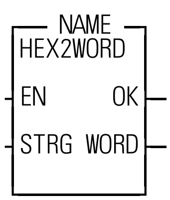
<<INSTANCE NAME>>:HEX2DWOR(EN := <<BOOL>>, STRG := <<STRING>>, OK => <<BOOL>>, DWOR => <<DWORD>);

This function block places the hexadecimal notation at STRG into the output at DWOR.

Example: If CE735 is entered at the STRG input, 845,621 will be reported at the DWOR output.

HEX2WORD

Converts a hex value to a word **USER/M_COMMON**



Inputs: EN (BOOL) - enables execution

STRG (STRING) - hexadecimal value to convert

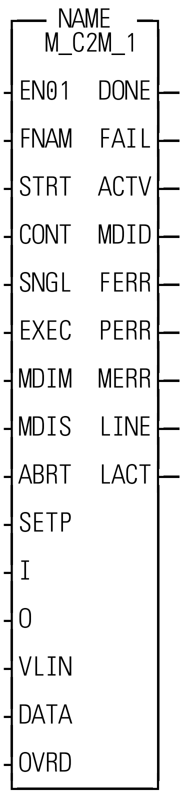
Outputs: OK (BOOL) - execution complete

WORD (WORD) - converted value

<<INSTANCE NAME>>:HEX2WORD(EN := <<BOOL>>, STRG := <<STRING>>, OK => <<BOOL>>, WORD => <<WORD>);

This function block places the hexidecimal notation at STRG into the output at WORD.

Example: If 26,854 is entered at the STRG input, 68E6 will be reported at the WORD output.



Inputs: E**:** EN01 (BOOL) - enables execution
FNAM (STRING[32]) - filename string (name of program to be executed).
STRT (BOOL)) - pulsed to start program execution
CONT (BOOL) - energized to repeat program execution continuously
SNGL (BOOL) - energized to enter single step mode
EXEC (BOOL) - pulsed to execute next instruction in single step mode
MDIM (BOOL) - energize to select Manual Data Input mode
MDIS (STRING[130]) - manual data input string
ABRT (BOOL) - pulse to abort program execution
SETP (STRUCT) - defines the operation of this application of M_C2M_1
I (STRUCT) - user Input structure
O (STRUCT) - user Output structure
VLIN (STRING[64]) - string that shows the current program line being executed.
DATA- (STRUCT) - modal data structure
OVRD (USINT) - Path feedrate override

Outputs: DONE (BOOL) - initialization completed without error or was aborted by an ABRT request
FAIL (BOOL) - indicates that an error occurred while trying to execute the program
ACTV (BOOL) - active indicates that program execution is in progress
MDID(BOOL) -Manual Data Input mode done
FERR (INT) - indicates file read error
PERR (INT) - indicates program error
MERR (INT) - indicates motion error
LINE (DINT) – line number
LACT (BOOL) – line active

```
<<INSTANCE NAME>>:M_C2M_1(EN01 := <<BOOL>>,  FNAM :=
  <<STRING[32]>>, STRT :=  <<BOOL>>, CONT := <<BOOL>>, SNGL :=
  <<BOOL>>, EXEC := <<BOOL>>, MDIM := <<BOOL>>, MDIS :=
  << STRING[130]>>, ABRT := <<BOOL>>, SETP := <<STRUCT>>,
  I := << STRUCT >>, O := << STRUCT >>, VLIN := <<STRING[64]>>, DATA
  := << STRUCT >>, OVRD := <<USINT>>, DONE => <<BOOL>>, FAIL =>
  <<BOOL>>, ACTV => <<BOOL>>, MDID => <<BOOL>>, FERR =>
  <<INT>>, PERR => <<INT>>, MERR => <<INT>>, LINE => <<DINT>>,
  LACT => <<BOOL>>);
```

M_C2M_1, the Cad2Motion ASFB translates an M and G code format ASCII file into servo motion. Many applications require description of their motion path using CAD software. Third party packages (such as Gcode2000) will convert the CAD package DXFoutput to M and G code text files. M_C2M_1 will translate the M and G code file to servomotion. Applications include glue laying and textile cutting, to name a few. This ASFB is not intended for application to metal cutting machine tools such as lathes and mills and as such does not support features required by CNC applications such as cutter radius compensation, tool offsets and constant surface speed.

Code	Type	Use	Description
%		Start of Program	All characters before the % are ignored
/		Start/End Comment	After first / is encountered all code is ignored until another / encountered or the end of the program
N		Line Number	For reference only, ignored
X		X Command	X axis endpoint (G90) or incremental distance (G91)
Y		Y Command	Y axis endpoint (G90) or incremental distance (G91)
Z		Z Command	Z axis endpoint (G90) or incremental distance (G91)
W		W Command	W axis endpoint (G90) or incremental distance (G91)
A		A Command	A axis endpoint (G90) or incremental distance (G91)
B		B Command	B axis endpoint (G90) or incremental distance (G91)
I		X Center Point	Incremental distance from starting position to X Center point
J		Y Center Point	Incremental distance from starting position to Y Center point
K		Z Center Point	Incremental distance from starting position to Z Center point
F	Modal	Rate	Path velocity for G01, G02, G03, time for G04. Reset to 0 by abort or M30 end of program when not continuous
L	Modal	Rate	Value for use by application program to specify use logic seqeunce to execute
S	Modal	Rate	Rate value for use by application program, reset to 0 by abort or end of program when not continuous

G00	Modal Group	Rapid	Move all axes at default Rapid rate to position/increment, G00 is default
G01		Linear	Move all axes linearly at programmed rate to position/increment
G02		Circular Clockwise	Move all axes linearly at programmed rate to position/increment
G03		Circular Counter-clockwise	Move all axes linearly at programmed rate to position/increment
G04		Dwell	Wait length of time specified by F
G09		Decel-to-zero	Wait until all axes are in position with no move queued
G17	Modal Group	XY Plane	XY Plane Select, G17 is default, Z, W, A, B departure
G18		XZ Plane	XZ Plane Select, Y, W, A, B departure
G19		YZ Plane	YZ Plane Select, X, W, A, B departure
G90	Modal Group	Absolute	Select Absolute positioning mode, G90 is default
G91		Incremental	Select Incremental positioning mode
G92		Assign Dimension	Assign dimension specfied to X, Y, Z, W, and B data to the respective axis current position

Code	Type	Use	Description
M03	Modal	Start Cut	Set M03 indicator for use by user application
M05		Stop Cut	Reset M03 indicator for use by user application
M30		End Program	Stop executing program, reset all modal data flags to default
M10x	Modal	Turn On Output	Turn on user output 1 to 9, all outputs cleared by abort or end of program when not continuous
M20x		Turn Off Output	Turn off user output 1 to 9
M50x		Wait for Input On	Wait until specified input (x = 1 to 9) is on M10x and M20x instructions programmed in the same line as motion (G00..G03) will cause the specified output to turn on/off when the queued move becomes active
M60x		Wait for Input Off	Wait until specified input (x = 1 to 9) is off M50x and M60x instructions programmed in the same line as motion (G00..G03) will cause the queuing of the specified move to be delayed until the wait for on/off states are satisfied
M70x		Wait for Input On at end of instruction	At end of instruction, wait until specified input (x = 1 to 9) is on. After all other operations programmed in the instruction have been executed begin waiting until the specified input is On.

Code	Type	Use	Description
%		Start of Program	All characters before the % are ignored

All information in all program lines is ignored until a “%”, Start of Program, is encountered. This allows a detailed description of the program at its beginning. All subsequent “%” characters are ignored and have no effect.

Optionally, the name of the application can be specified in SETP.Application_Name. When this string is not null, the comment field immediately after the % must be the same string, e.g. %.Saw_Type_1/. If SETP.Application_Name is a null then this check is not performed. If it is specified and the name does not match, PERR is set to 7016 and FAIL is set.

Code	Type	Use	Description
/		Start/End Comment	After first / is encountered all code is ignored until another / encountered or the end of the program

All information after the first “/” slash character is encountered is ignored until a second “/” character or the end of the program file is found. The second “/” may be programmed in the same program line or later in the program. A single line can contain multiple sections which surrounded by slashes and ignored. In the line “N1000 /M101/ M201 G90 / M501/” the M101 and M501 commands would both be ignored.

Code	Type	Use	Description
N		Line Number	For reference only, ignored

The N data word is used by the programmer as a reference point to determine which line is being processed. It is not used in any way and is ignored.

Code	Type	Use	Description
X		X Command	X axis endpoint (G90) or incremental distance (G91)
Y		Y Command	Y axis endpoint (G90) or incremental distance (G91)
Z		Z Command	Z axis endpoint (G90) or incremental distance (G91)
W		W Command	W axis endpoint (G90) or incremental distance (G91)
A		A Command	A axis endpoint (G90) or incremental distance (G91)
B		B Command	B axis endpoint (G90) or incremental distance (G91)

Including X,Y, Z, W, A, or B endpoints/distances indicates that the specified axes should be moved according to the modal move mode (G00 to G03) and modal plane select (G17 to G19). Examples of valid X,Y,Z data format include: "X1", "X0", "X-1.23", "X.001".

For more detail see the Instruction Execution section below.

Code	Type	Use	Description
I		X Center point	Incremental distance from starting position to X Center point
J		Y Center point	Incremental distance from starting position to Y Center point
K		Z Center point	Incremental distance from starting position to Z Center point

When circular clockwise (G02) or circular counterclockwise (G03) modal move type is active the circle centerpoints must be specified by I and J when the modal XY plane (G17) is active, or J and K when the modal XZ plane (G18) is active, or I and K when the modal XZ plane (G19) is selected. Circle centerpoints must always be programmed as incremental distances from the circle starting position to the circle centerpoint, independent of whether the G90 absolute or G91 incremental positioning mode is active.

Code	Type	Use	Description
F	Modal	Rate	Path velocity for G01, G02, G03, time for G04. Reset to 0 by abort or M30 end of program when not continuous

The F data word specifies the path feedrate for G01 linear and G02/G03 circular moves. It is modal and does not need to be specified again until a new value is required. It will be reset to zero if a fault or abort occurs or if an M30 end of program occurs and continuous program repeat mode is not selected.

The F data word specifies the delay in seconds when specified with a G04 dwell instruction. To select a one-half second delay “G04 F0.5” would be programmed.

Code	Type	Use	Description
S	Modal	Rate	Rate value for use by application program, reset to 0 by abort or end of program

The value specified by S is presented to the user application by the output SDAT of the M_C2M_1 ASFB. Typically it is used for rate control. It is modal and its value will not change until another occurrence of S. It will be cleared to 0 if an error occurs, after M30 end of program when not in continuous mode or when program execution is aborted.

Code	Type	Use	Description
G00	Modal Group	Rapid	Move all axes at default Rapid rate to position/increment, G00 is default
G01		Linear	Move all axes linearly at programmed rate to position/increment
G02		Circular Clockwise	Move all axes linearly at programmed rate to position/increment
G03		Circular Counterclockwise	Move all axes linearly at programmed rate to position/increment

G00, G01, G02 and G03 are the modal path positioning group. G00 is the default mode. Once selected the position type does not need to be specified again until you wish to change it. It will be reset to G00 if an error occurs, after M30 end of program when not in continuous mode or when program execution is aborted.

Code	Type	Use	Description
G04		Dwell	Wait length of time specified by F

The F data word specifies the delay in seconds when specified with a G04 dwell instruction. To select a one-half second delay “G04 F0.5” would be programmed.

The time delay begins immediately unless a Wait for Input (M50x/M60x) is programmed in the same line, in which case the delay begins after the Wait condition is satisfied.

Code	Type	Use	Description
G09		Decel-to-zero	Wait until all axes are in position with no move queued

Stalls execution until all axes are within the in position bandwidth specified by servo setup and no moves are queued. If programmed in a line with motion the G09 begins after the motion has been queued.

Code	Type	Use	Description
G17	Modal Group	XY Plane	XY Plane Select, G17 is default, Z, W, A, B departure
G18		XZ Plane	XZ Plane Select, Y, W, A, B departure
G19		YZ Plane	YZ Plane Select, X, W, A, B departure

G17, G18 and G19 are the modal plane select group. G17 is the default mode. Once selected the plane does not need to be selected again until you wish to change it. It will be reset to G17 if an error occurs, after M30 end of program when not in continuous mode or when program execution is aborted. The plane specified is the plane which axes can be moved using G02 circular clockwise and G03 circular counterclockwise moves. If endpoints are specified for the departure axes and G02 or G03 is performed, the departure axis will arrive at their programmed endpoints at the same time as the circular interpolation axes.

Code	Type	Use	Description
G90	Modal Group	Absolute	Select Absolute positioning mode, G90 is default
G91		Incremental	Select Incremental positioning mode

G90 and G91 select absolute or incremental position mode. G90 is the default mode. Once selected, the positioning mode does not need to be specified again until you wish to change it. It will be reset to G90 if an error occurs, after M30 end of program when not in continuous mode or when program execution is aborted.

Code	Type	Use	Description
G92		Assign Dimension	Assign dimension specified to X, Y, Z, W, A and B data to the respective axis current position

G92 will wait for all axes to decel-to-zero an be in-position and then will assign the value programmed in the X, Y, Z, W, A and B data words to be the current position for each respective axes.

Code	Type	Use	Description
M03	Modal	Start Cut	Set M03 indicator for use by user application
M04		Spindle CCW	Set Data.M04, reset Data.M03 for use by user application
M05		Stop Cut	Reset M03 indicator for use by user application

M03, M04 and M05 are the modal start/stop indicators. M05 is the default mode. Once selected the start/stop mode does not need to be specified again until you wish to change it. It will be reset to M05 if an error occurs, after M30 end of program when not in continuous mode or when program execution is aborted. Programming an M03/M04 in a line without motion will cause the M03 output of the M_C2M_1 ASFB to energize immediately. Programming an M03/M04 in a line with motion will cause the M03/M04 output of the M_C2M_1 ASFB to energize when the move becomes active. Programming an M05 in a line without motion will cause the M03/M04 indicator to de-energize immediately. Programming an M05 in a line with motion will cause the M03/M04 indicator to de-energize when the move becomes active.

Code	Type	Use	Description
M30		End Program	Stop executing program, reset all modal data flags to default

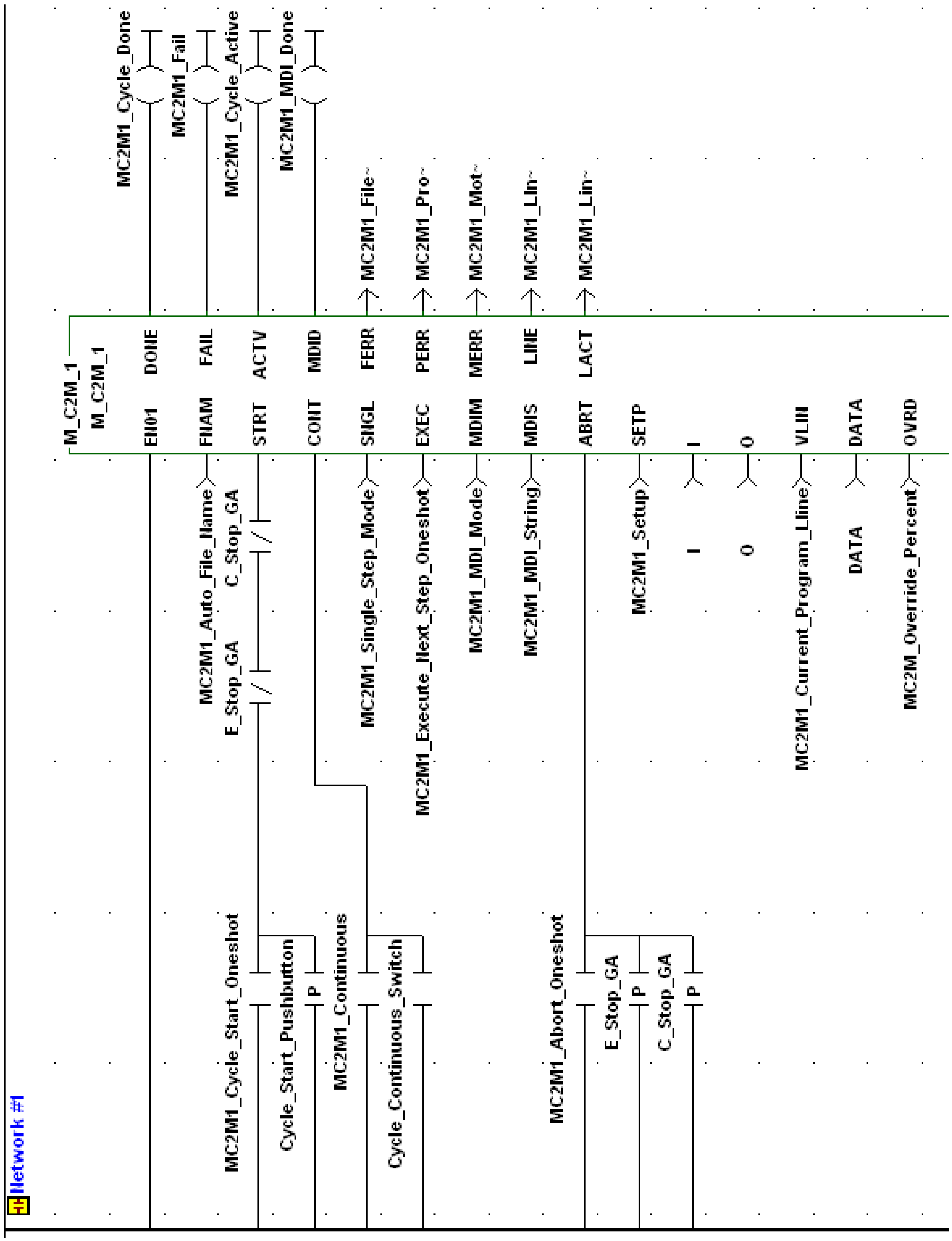
M30 indicates the end of the program. When a line contains M30 all instructions in the line will execute, the equivalent of a G09 decel to zero will execute and then the file will be closed. If the CONT, continuous mode input to M_C2M_1 is energized the program file will be opened and executed again. If CONT is not energized then all modal data will be reset, all User Outputs will be de-energized (O.O1 to O.O9), the program file will be closed and the DONE output of M_C2M_1 will be energized indicating the program has finished executing. All lines following a line with M30 are ignored.

Code	Type	Use	Description
M10x	Modal	Turn On Output	Turn on user output 1 to 9, all outputs cleared by abort or end of program when not continuous
M20x		Turn Off Output	Turn off user output 1 to 9. M10x and M20x instructions programmed in the same line as motion (G00..G03) will cause the specified output to turn on/off when the queued move becomes active

M10x and M20x are the modal user output control group. M20x, outputs off, is the default state. Once selected the output state is maintained and does not need to be specified again until you wish to change it. It will be reset to M20x, outputs off, after M30 end of program when not in continuous mode or when program execution is aborted. If an M10x, Turn On, or an M20x, Turn Off, is specified in a line without motion it will take effect immediately. If specified in a line with motion, it will take effect when the move becomes active. Multiple M10x's and M20x's may be programmed in a single line. Outputs 1 to 9 are presented to the user application via the O input of M_C2M_1.

Code	Type	Use	Description
M50x		Wait for Input On	Wait until specified input (x = 1 to 9) is on
M60x		Wait for Input Off	Wait until specified input (x = 1 to 9) is off
			M50x and M60x instructions programmed in the same line as motion (G00..G03) will cause the queuing of the specified move to be delayed until the wait for on/off states are satisfied

M50x, Wait of Input On, and M60x, Wait for Input Off, are non-modal and effect only the line they are programmed in. If the line contains M50x and M60x instructions, the rest of the line will not be executed until all of the M50x and M60x wait for inputs are satisfied.


Network #1
M_C2M_1
M_C2M_1
EN01
DONE
FNAM
FAIL
STRT
ACTV
CONT
MDID
SNGL
FERR
EXEC
PERR
MDIM
MERR
MDIS
LINE
ABRT
LACT
SETP
I
O
VLIN
DATA
OVRD
MC2M1_Cycle_Done
MC2M1_Fail
MC2M1_Cycle_Active
MC2M1_MDI_Done
MC2M1_File~
MC2M1_Pro~
MC2M1_Mot~
MC2M1_LIn~
MC2M1_Lin~
MC2M1_Auto_File_Name
E_Stop_GA
C_Stop_GA
MC2M1_Single_Step_Mode
MC2M1_Execute_Next_Step_Oneshot
MC2M1_MDI_Mode
MC2M1_MDI_String
MC2M1_Setup
I
O
MC2M1_Current_Program_Line
DATA
MC2M_Override_Percent
MC2M1_Cycle_Start_Oneshot
Cycle_Start_Pushbutton
P
MC2M1_Continuous
Cycle_Continuous_Switch
MC2M1_Abort_Oneshot
E_Stop_GA
P
C_Stop_GA
P

INPUTS

Input	Description
EN01	Enable - must be energized at all times.
FNAM	Filename string – Name of program file to be executed. Typically “RAMDISK:<filename.txt>$00. The string must be terminated by $00.
STRT	Pulse to start program execution - Modal data is reset.
CONT	Energize to repeat program execution continuously.
SNGL	Energize to enter single step mode. After using single step mode deenergizing SNGL will cause execution to continue. When entering single step mode all pre-processed motion (up to three moves) will execute before execution is stalled. When in single step mode the VLIN will show the instruction that will be executed when EXEC is pulsed. When in single step mode, pulse MDIM to activate Manual Data Input Mode.
EXEC	Pulse to execute next instruction when in single step mode. Pulse to execute instruction in MDIS when in MDI (Manual Data Input) mode.
MDIM	Energize to select Manual Data Input mode. While in MDI mode pulse EXEC to execute the instruction specified by MDIS.
MDIS	Manual Data Input String. M and G code instruction to be executed when EXEC is pulsed when in MDI mode.
ABRT	Pulse to abort program execution. When program execution is aborted all user outputs (O.O1 to O.O9) will be deenergized, M03 will turn off, SDAT will be cleared to zero, and all axes motion will be aborted.
SETP	See the table below for a description of the Setup data structure.
I	User Input structure I.I1 to I.I9 corresponding to Wait for Input On M501 to M509 and Wait for Input Off M601 to M609, respectively.
O	User Output structure O.O1 to O.O9 corresponding to turn On Output M101 to M109 and Turn Off Output M201 to M209, respectively.
VLIN	String which will show the current program line being executed. Note that due to preprocessing VLIN can be up to three lines ahead of actual application motion.
DATA	M_C2M_1 Data Structure. Provides the data for the active program line.
OVRD	Path feedrate override. Specify from 0 to 255 percent of programmed (or Rapid) feedrate.

SETUP DATA STRUCTURE - The setup data structure defines the operation of this application of M_C2M_1. Values should be specified as initial values in Software Declarations and not changed while running.

Name	Type	Description
Setp	STRUCT	Setup Data Structure
.X_ACTIVE	BOOL	X, Y, Z, W, and B_ACTIVE are set to 1 to indicate axis is active in this application. X, Y, Z, W, A and B_AXIS_Number are set to 1 to 16 to indicate each axis' servo setup axis number. X, Y, Z, W, A and B_DG2R are set to indicate the number of digits to the right of the implied decimal point. See Implied Decimal Point Data section below.
.X_DG2R	INT	
.X_Axis_Number	USINT	
.Y_ACTIVE	BOOL	
.Y_DG2R	INT	
.Y_Axis_Number	USINT	
.Z_ACTIVE	BOOL	
.Z_DG2R	INT	
.Z_Axis_Number	USINT	
.W_ACTIVE	BOOL	
.W_DG2R	INT	
.W_Axis_Number	USINT	
.A_ACTIVE	BOOL	
.A_DG2R	INT	
.A_Axis_Number	USINT	
.B_ACTIVE	BOOL	
.B_DG2R	INT	
.B_Axis_Number	USINT	
.I_DG2R	INT	Indicate the number of digits to the right of the implied decimal point. See Implied Decimal Point Data section below.
.J_DG2R	INT	
.K_DG2R	INT	
.F_DG2R	INT	
.S_DG2R	INT	
.RAPID	DINT	The feedrate used for G00 Rapid moves
.BNDW	DINT	The Circular Endpoint on circle bandwidth. See PiCPro Function Block Help for M_SCRVLC for further information.
.PATH	USINT	Typically set to 1. Set to 2,3 or 4 for applications running up to four simultaneous M_C2M_1 instances. See Interpolation Paths section below for more information.

.ACCEL	LREAL	Path S-Curve Acceleration/Deceleration definition. See PiCPro Function Block Help for M_SCRVLC for more information.
.JERK	LREAL	
.MAXF	DINT	
.X Offset	DINT	X axis absolute endpoint offset value
.X Offset_Ignore	BOOL	Set to ignore X axis endpoint offset
.Y Offset	DINT	Y axis absolute endpoint offset value
.Y Offset_Ignore	BOOL	Set to ignore Y axis endpoint offset
.Z Offset	DINT	Z axis absolute endpoint offset value
.Z Offset_Ignore	BOOL	Set to ignore Z axis endpoint offset
.W Offset	DINT	W axis absolute endpoint offset value
.W Offset_Ignore	BOOL	Set to ignore W axis endpoint offset
.A Offset	DINT	A axis absolute endpoint offset value
.A Offset_Ignore	BOOL	Set to ignore A axis endpoint offset
.B Offset	DINT	B axis absolute endpoint offset value
.B Offset_Ignore	BOOL	Set to ignore B axis endpoint offset
.Application_Name	STRING[25]	Name of the application; reference only
.Structure_Check_Constant	DINT	Must have initial value of +12345. Used to verify that the application Setup structure matches the M_C2M_1 structure.
END_STRUCT		

The offset data provides the application with the ability to identify offset values for X, Y, Z, W, A, and B which will be added to all absolute (G90) position endpoints. This includes G00, G01, G02 and G03 move endpoints. Incremental mode (G91) distances are not affected. The offset values are applied when the corresponding "ignore" flag is not set.

INPUT DATA STRUCTURE - The input data structure "I" allows integration of user inputs with the execution of the program.

Name	**Type**	**Description**
I	STRUCT	Input Data Structure
.I1	BOOL	Input 1, M501 Wait for Input On , M601 Wait for Input Off
.I2	BOOL	Input 2, M502 Wait for Input On , M602 Wait for Input Off
.I3	BOOL	Input 3, M503 Wait for Input On , M603 Wait for Input Off
.I4	BOOL	Input 4, M504 Wait for Input On , M604 Wait for Input Off
.I5	BOOL	Input 5, M505 Wait for Input On , M605 Wait for Input Off
.I6	BOOL	Input 6, M506 Wait for Input On , M606 Wait for Input Off
.I7	BOOL	Input 7, M507 Wait for Input On , M607 Wait for Input Off
.I8	BOOL	Input 8, M508 Wait for Input On , M608 Wait for Input Off
I9	BOOL	Input 9, M509 Wait for Input On , M609 Wait for Input Off
.M01_Input	BOOL	If ON enter single step mode when M01 programmed
END_STRUCT		

OUPUT DATA STRUCTURE - The input data structure “O” allows integration of user outputs with the execution of the program.

Name	**Type**	**Description**
O	STRUCT	Output Data Structure
.O1	BOOL	Output 1, M101 Turn On Output, M201 Turn Off Output
.O2	BOOL	Output 2, M102 Turn On Output, M202 Turn Off Output
.O3	BOOL	Output 3, M103 Turn On Output, M203 Turn Off Output
.O4	BOOL	Output 4, M104 Turn On Output, M204 Turn Off Output
.O5	BOOL	Output 5, M105 Turn On Output, M205 Turn Off Output
.O6	BOOL	Output 6, M106 Turn On Output, M206 Turn Off Output
.O7	BOOL	Output 7, M107 Turn On Output, M207 Turn Off Output
.O8	BOOL	Output 8, M108 Turn On Output, M208 Turn Off Output
.O9	BOOL	Output 9, M109 Turn On Output, M209 Turn Off Output
END_STRUCT		

DATA STRUCTURE - The M_C2M_1 Data Structure "DATA" provides the application with a view of all of the program data variables and flags.

Name	Type	Description
.X_Dat	DINT	X Command data value.
.Y_Dat	DINT	Y Command data value.
.Z_Dat	DINT	Z Command data value.
.W_Dat	DINT	W Command data value.
.A_Dat	DINT	A Command data value.
.B_Dat	DINT	B Command data value.
.F_Dat	DINT	F Command data value.
.S_Dat	DINT	S Command data value.
.N_Dat	DINT	N Command data value.
.I_Dat	DINT	I Command data value.
.J_Dat	DINT	J Command data value.
.K_Dat	DINT	K Command data value.
.L_Dat	DINT	L Command data value.
.G00	BOOL	G00 Command is active.
.G01	BOOL	G01 Command is active.
.G02	BOOL	G02 Command is active.
.G03	BOOL	G03 Command is active.
.G17	BOOL	G17 Command is active.
.G18	BOOL	G18 Command is active.
.G19	BOOL	G19 Command is active.
.G90	BOOL	G90 Command is active.
.G91	BOOL	G91 Command is active.
.M00_M01_Program_Pause	BOOL	M00/M01 Command is active.
.M03	BOOL	M03 Command is active.
.M04	BOOL	M04 Command is active.
.M05	BOOL	M05 Command is active.
.M101	BOOL	M101 Command is active.
.M102	BOOL	M102 Command is active.
.M103	BOOL	M103 Command is active.
.M104	BOOL	M104 Command is active.
.M105	BOOL	M105 Command is active.
.M106	BOOL	M106 Command is active.
.M107	BOOL	M107 Command is active.
.M108	BOOL	M108 Command is active.
.M109	BOOL	M109 Command is active.
.Wait_for_Input	BOOL	Wait_for_Input Command is active.

.Wait_for_Time_Delay	BOOL	Wait_for_Time_Delay Command is active.
.Time_Left	UDINT	Time left for the Wait_for_Time_Delay Command.
.Wait_for_In_Position	BOOL	Wait_for_In_Position_Command is active.
.Wait_for_Servo_Queue	BOOL	Wait_for_Servo_Queue Command is active.
.Structure_Check_Constant	DINT	Must have initial value of 12345. Used to verify that the Data structure matches the M_C2M_1 structure.

M_C2M_1 OUTPUTS

Outputs	Description
DONE	Indicates that the program execution has completed successfully or was aborted by and ABRT request.
FAIL	Indicates that an error occurred while trying to execute the program. The type of error is indicated by FERR, PERR and MERR as described below. When FAIL occurs all user outputs will be reset, all axes motion aborted and the program file will be closed.
ACTV	Active indicates that program execution is in progress
MDID	MDI mode done - set on entry to MDI mode and when execution of MDIS completes during MDI mode operation.
FERR	File read error - Using PiCPro Online Help, refer to I/O Function Block Error Codes under Error Codes for a description of these errors.
PERR	Program Error - See the Program Error table below for a description of these error codes.
MERR	Motion Error - See PiCPro Function Block Help for M_SCRVLC and refer to the ERR output for a description of error codes.
LINE	Indicates the Line number.
LACT	Indicates that the Line is active.

PERR Program Errors - This table provides a description of errors that will be reported if an improperly formatted program is encountered.

Error Number	Description
7001	CRLF line terminator not found
7002	Unrecognizable Field Code (i.e. not N,X,Y…M)
7003	Unrecognizable G Code
7004	Unrecognizable Mxx Code
7005	Bad data for N Code
7006	Missing CRLF line terminator (line wider than 128 charac-ters
7007	Missing LF line terminator
7008	Missing CRLF line terminator
7009	End of File missing CRLF line terminator
7010	End of File missing CRLF line terminator
7011	Poorly formed data
7012	SETP structure doesn't match M_C2M's SETP structure
7013	Multiple G00, G01, G02, G03, G92, codes in single line
7014	Incomplete line programmed, i.e. "G03 G17X1 Y1 11", J data word is missing so move could not be executed
7015	% Start of program not found
7016	SETP.Application_Name does not match application name specified at beginning of program (e.g. %/Saw Type 1/)

IMPLIED DECIMAL POINT DATA - Implied decimal point data accommodates the fact that position and feedrate information used with PiCPro motion control programming is stored in 32-bit double integer variables. The M and G code program will need to specify position and feedrate information with a decimal point. In the Setup data structure input to the M_C2M_1 ASFB the digits-to-right (i.e. X_DG2R) specified for each program code is used to scale data appropriately to the needs of the PiCPro motion control instructions. The table below shows the effect of setting the DG2R precision to various values.

Digits-to-Right	**Data in Program Line**	**Data Delivered to Motion Function**
3	X123	123000
3	X.1	100
3	X1.2	1200
3	X1	1000
3	X1.0002	1000
4	X123	1230000
4	X.1	1000
4	X1.2	12000
4	X1	10000
4	X1.0002	10002

Scaling from programmed units to machine servo feedback units is defined when programming the application specific servo setup data using PiCPro.

When G00 - Rapid is active the axes specified in the line will be move to the end-point (G90 absolute) or the incremental distance (G91 incremental) specified by the X, Y, Z, W, A and B data words at the rate specified by SETUP.RAPID. Unprogrammed axes will not move.. .

G00 Rapid Mode

G00 - The axes specified in the line will be move to the endpoint (G90 absolute) or the incremental distance (G91 incremental) specified by the X, Y, Z, W, A and B data words at the rate specified by SETUP.RAPID. Unprogrammed axes will not move.

G00 Rapid Mode Example

%/Start of Program/

N1000 G90 G00 X10.0 Y5.0 /Position X to 10.0 and Y to 5.0 at rapid rate /

N1010 X15 Y0 G09 /Position X to 10 and Y to 0 at rapid rate, decel to zero/

N1020 G91 Y1 /Move Y incrementally 1 at rapid rate/

N1030 G09 Y1 /Move Y incrementally 1 at rapid rate, decel to zero /

G01 Linear Interpolation

G01 - The axes specified in the line will be move to the endpoint (G90 absolute) or the incremental distance (G91 incremental) specified by the X, Y, Z, W, A and B data words at the rate specified by F using linear interpolation. Unprogrammed axes will not move.

G01 Linear Interpolation Example

%/Start of Program/

N1000 G90 G01 X10.0 Y5.0 F100.0 /Position X to 10.0 and Y to 5.0 at a path rate of 100 /

N1010 X15 Y0 G09 /Position X to 10 and Y to 0 at a path rate of 100, decel to zero/

N1020 G91 Y1 /Move Y incrementally 1 at a path rate of 100/

N1030 G09 Y1 /Move Y incrementally 1 at path rate of 100, decel to zero /

.

G02 Circular Clockwise and G03 Counter Clockwise Circular Interpolation

Plane	Mode	Description
G17, XY	G02, G03	Use circular interpolation to move to X and Y endpoints (incremental or absolute based on G90/G91), I and J centerpoints (always incremental from start of circle) at F modal path feedrate. If Z, W, A, or B are programmed in the same line they will be moved in a third axis departure move and arrive at their programmed position simultaneously with X and Y.
G18, XZ	G02, G03	Use circular interpolation to move to X and Z endpoints (incremental or absolute based on G90/G91), I and K centerpoints (always incremental from start of circle) at F modal path feedrate. If Y, W, A or B are programmed in the same line they will be moved in a third axis departure move and arrive at their programmed position simultaneously with X and Z.
G19, YZ	G02, G03	Use circular interpolation to move to Y and Z endpoints (incremental or absolute based on G90/G91), J and K centerpoints (always incremental from start of circle) at F modal path feedrate. If X, W, A or B are programmed in the same line they will be moved in a third axis departure move and arrive at their programmed position simultaneously with Y and Z.

G02 Clockwise and G03 Counter Clockwise Circular Interpolation Examples

```
%/Start of Program/
G90 /Select Absolute Positioning Mode
F800 /Specify path feedrate of 800 /
G01 X8.000 Y0.000
G01 X16.472 Y0.000
G03 X17.472 Y1.000 I0.000 J1.000 / Circular Counter Clockwise /
G01 X17.472 Y11.707
G03 X9.472 Y19.707 I-8.000 J0.000
G01 X1.000 Y19.707
G03 X0.000 Y18.707 I0.000 J-1.000
G01 X0.000 Y8.000
G03 X8.000 Y0.000 I8.000 J0.000
G09
```

.

Coordinating User Outputs with Motion

To turn outputs on and off in step with servo axis positioning program M10x and M20x instructions in the same line as the desired motion.

```
%/Start of Program/
N1000 M101 / Immediately Turn on Output 1 /
N1010 G04 F1.5 /Wait for 1.5 seconds
N1020 M102 / Immediately Turn off Output 1 /
N1030 M101 G91 G01 F100.0 X1.00 / Output 1 turns on when this move begins/
N1040 M102 X2.0 / Output 2 turns on when this move begins /
N1050 X3.0 G09
N1060 M103 / Output 3 turns on when the move in N1050 completes /
```

. .

Coordinating Motion with User Inputs

The M50x Wait for Input On and M60x Wait for Input Off instructions are used to coordinate program execution with the state of user application inputs.

%/Start of Program/

N1000 M501 / Program execution stalls until user input 1 is on /

N1010 M502 G91 G01 F100.0 X1.00 / When user input 2 is on start move of 1 /

N1010 M503 G04 F1.5 / When user input 3 is on begin delay of 1.5 seconds /

Effect of Motion Que and Program Execution

To provide continuous path motion a queuing system is used to buffer one move which will blend with the currently active move with no deceleration of the servo axes. This queuing system requires that program lines be read and executed while motion started by previous lines is completed. This will lead to the program line display, VLIN, showing the line currently being parsed and queued and this line may be many lines after the line which started the current motion.

Instruction	**Action**
N1000 G90 G01 X100	Starts move of X to 100
N1010 G90 G01 X200	Queues move of X to 200
N1030 G90 G01 X300	Waits until queue move in N1000 completes which will make room on the queue

Any lines between N1000 and N1030 would execute immediately. Including G09 decel to zero changes the execution as described below.

Instruction	**Action**
N1000 G90 G01 X100 G09	Starts move of X to 100, and wait till in position
N1010 G90 G01 X200 G09	Start move of X to 200 and wait till in position
N1030 G90 G01 X300 G09	Start move of X to 300 and wait till in position

Using the G09 stalls execution until the move in the line completes

Adding user outputs to the same examples also shows the effect of the queue

Instruction	Action
N1000 G90 G01 X100 M101	Starts move of X to 100, turn on output 1
M102	Output 2 turns on while move to 100 is occurring
N1010 G90 G01 X200 M103	Queues move of X to 200, output 3 will turn on when this move becomes active
M104	Output 4 turns on while move to 100 in N1000 is occuring. Output 4 will turn on before output 3
N1030 G90 G01 X300	Waits until queue move in N1000 completes which will make room on the queue.

Line Execution

Programs are executed a line at a time. In a line containing multiple instructions the order of execution is based on the type of instruction, not the order of its occurrence within the line. Line execution is performed in the following order:

1 - If the line contains any Wait for Input On or Off instructions execution will wait until all of the conditions have been satisfied.

2 - If the line contains a G04 dwell instruction execution will delay until the time specified by F passes.

3 - If the line contains a G00 to G03 motion instruction execution will wait until the servo queue is ready to accept the next move.

Lines Containing Incomplete Motion Instructions

Incomplete motion instructions that can be detected will result in FAIL being set with PERR = 7014 and the move will not be executed. An example of a line containing an incomplete motion instruction would be "G03 G17 X1 Y2 I3". In this case the J data word specifying the Y axis centerpoint is missing resulting in the M&G program execution being ignored.

Simultaneous Multiple Paths

M_C2M_1 will support up to four completely independent motion programs on four separate interpolators. To do this the users application must have four separate instances of M_C2M_1. The SETUP.PATH should be set to 1,2,3 and 4 for instance 1 to 4, respectively. The table below describes the servo axis numbering in the applications servo setup data that must be used for each path.

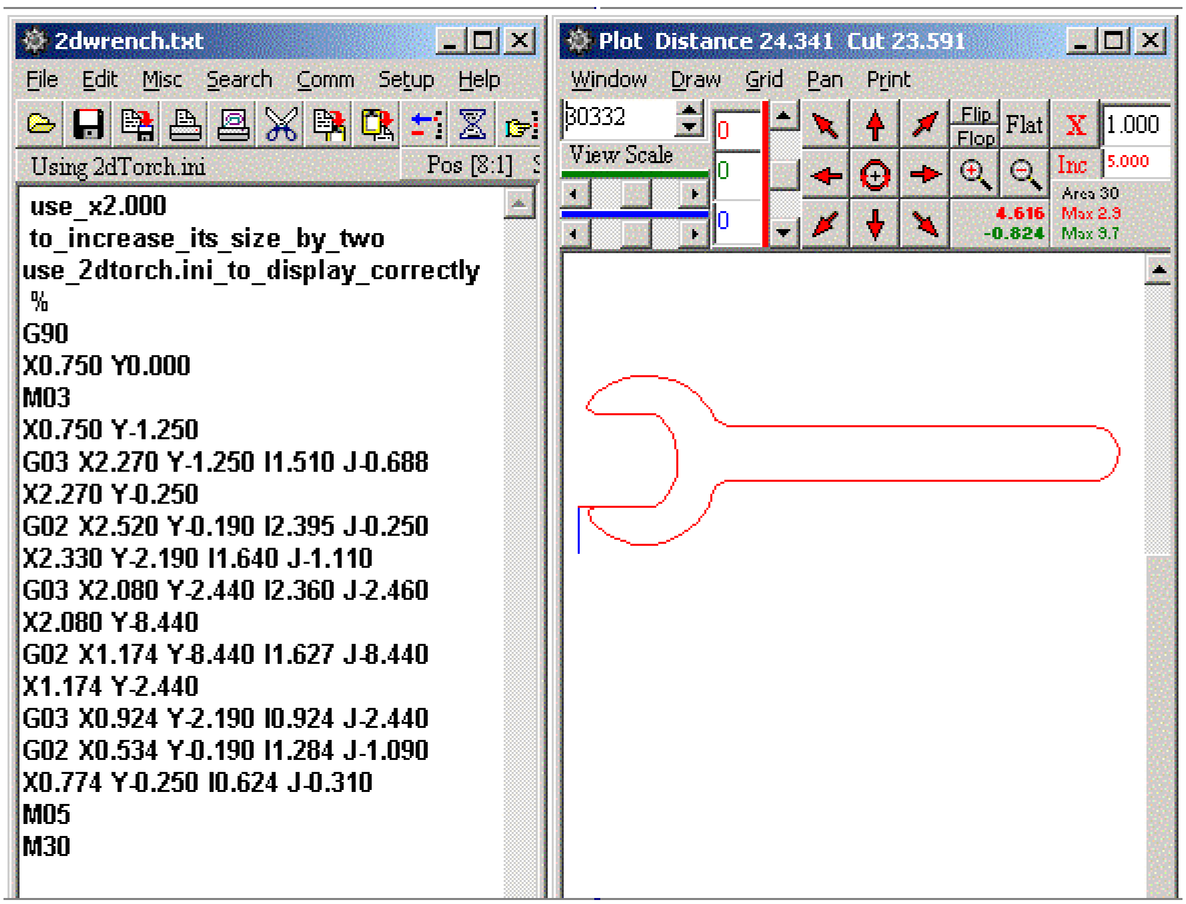
Program File Structure

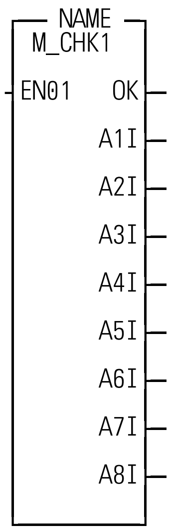
Program lines can contain up to 126 characters and must be terminated with a carriage return ($0D) and line feed ($0A). Fields within the line must be combinations of letters followed by values, i.e. X-123.456, no space can occur between the letter and the value. One or more spaces must occur between fields, i.e. X123 Y245 is valid, X1Y2 is invalid.

The last line of the program must contain a carriage return ($0D) and line feed ($0A).

Using M_C2M_1 with Third Party Cad-to-Motion Tools

M_C2M_1, the Cad2Motion ASFB can be used to translate the output of a third-party DXF to Ascii file conversion program like Gcode2000 to machine and motion control. For more information log onto http://members.aol.com/_ht_a/gcodemcode/index.htm.





Inputs: EN01 (BOOL) - enables execution

Outputs: OK (BOOL) - execution complete

A1I (BOOL) - set if axis number 1 has been initialized

A2I (BOOL) - set if axis number 2 has been initialized

A3I (BOOL) - set if axis number 3 has been initialized

A4I (BOOL) - set if axis number 4 has been initialized

A5I (BOOL) - set if axis number 5 has been initialized

A6I (BOOL) - set if axis number 6 has been initialized

A7I (BOOL) - set if axis number 7 has been initialized

A8I (BOOL) - set if axis number 8 has been initialized

<<INSTANCE NAME>>:M_CHK1(EN01 := <<BOOL>>, OK => <<BOOL>>, A1I => <<BOOL>>, A2I => <<BOOL>>, A3I => <<BOOL>>, A4I => <<BOOL>>, A5I => <<BOOL>>, A6I => <<BOOL>>, A7I => <<BOOL>>, A8I => <<BOOL>>);

This function block checks to see which servo axes numbered from 1 to 8 have been initialized by the user's servo setup function

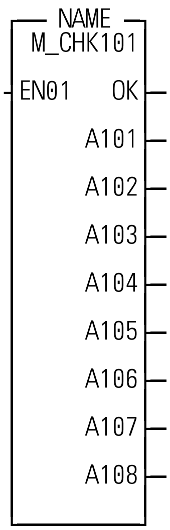
The OK output of the STRTSERV function should be wired directly to the enable (EN01) input of this function.

The outputs of this function will remain set even after the function is no longer enabled.

The outputs of this function can be used to ensure the correct setup information has been used. They can also be used as conditional contacts to qualify other motion functions.

M_CHK101

Check for Servo Axis Initialized **USER/M_INIT**



Inputs: EN01 (BOOL) - enables execution

Outputs: OK (BOOL) - execution complete

A101 (BOOL) - set if axis number 101 has been initialized (servo axis 17).

A102 (BOOL) - set if axis number 102 has been initialized (servo axis 18).

A103 (BOOL) - set if axis number 103 has been initialized (servo axis 19).

A104 (BOOL) - set if axis number 104 has been initialized (servo axis 20).

A105 (BOOL) - set if axis number 105 has been initialized (servo axis 21).

A106 (BOOL) - set if axis number 106 has been initialized (servo axis 22).

A107 (BOOL) - set if axis number 107 has been initialized (servo axis 23).

A108 (BOOL) - set if axis number 108 has been initialized (servo axis 24).

```
<<INSTANCE NAME>>:M_CHK101(EN01 := <<BOOL>>, OK =>
  <<BOOL>>, A101 => <<BOOL>>, A102 => <<BOOL>>, A103 =>
  <<BOOL>>, A104 => <<BOOL>>, A105 => <<BOOL>>, A106 =>
  <<BOOL>>, A107 => <<BOOL>>, A108 => <<BOOL>>);
```

This function block checks to see which servo axes numbered from 101 to 108 (servo axes 17 to 24) have been initialized by the user's servo setup function.

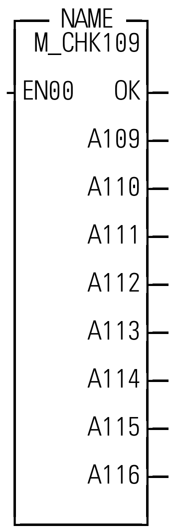
The OK output of the STRTSERV function should be wired directly to the enable (EN01) input of this function.

The outputs of this function will remain set even after the function is no longer enabled.

The outputs of this function can be used to ensure the correct setup information has been used. They can also be used as conditional contacts to qualify other motion functions.

M_CHK109

Check for Servo Axis Initialized **USER/M_INIT**



Inputs: EN01 (BOOL) - enables execution

Outputs: OK (BOOL) - execution complete

A109 (BOOL) - set if axis number 109 has been initialized (servo axis 25).

A110 (BOOL) - set if axis number 110 has been initialized (servo axis 26).

A111 (BOOL) - set if axis number 111 has been initialized (servo axis 27).

A112 (BOOL) - set if axis number 112 has been initialized (servo axis 28).

A113 (BOOL) - set if axis number 113 has been initialized (servo axis 29).

A114 (BOOL) - set if axis number 114 has been initialized (servo axis 30).

A115 (BOOL) - set if axis number 115 has been initialized (servo axis 31).

A116 (BOOL) - set if axis number 116 has been initialized (servo axis 32).

<<INSTANCE NAME>>:M_CHK109(EN01 := <<BOOL>>, OK => <<BOOL>>
A109 => <<BOOL>>, A110 => <<BOOL>>, A111 => <<BOOL>>, A112 =>
<<BOOL>>, A113 => <<BOOL>>, A114 => <<BOOL>>, A115 =>
<<BOOL>>, A116 => <<BOOL>>);

This function block checks to see which servo axes numbered from 109 to 116 (servo axes 25 to 32) have been initialized by the user's servo setup function.

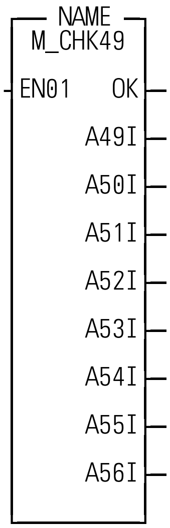
The OK output of the STRTSERV function should be wired directly to the enable (EN01) input of this function.

The outputs of this function will remain set even after the function is no longer enabled.

The outputs of this function can be used to ensure the correct setup information has been used. They can also be used as conditional contacts to qualify other motion functions.

M_CHK49

Check for Digitizing Axis Initialized **USER/M_INIT**



Inputs: EN01 (BOOL) - enables execution

Outputs: OK (BOOL) - execution complete

A49I (BOOL) - set if axis number 49 has been initialized

A50I (BOOL) - set if axis number 50 has been initialized

A51I (BOOL) - set if axis number 51 has been initialized

A52I (BOOL) - set if axis number 52 has been initialized

A53I (BOOL) - set if axis number 53 has been initialized

A54I (BOOL) - set if axis number 54 has been initialized

A55I (BOOL) - set if axis number 55 has been initialized

A56I (BOOL) - set if axis number 56 has been initialized

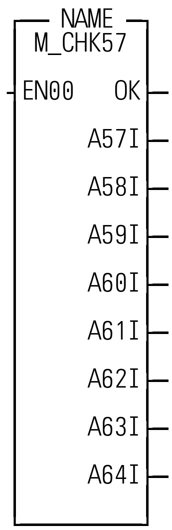
<<INSTANCE NAME>>:M_CHK49(EN01 := <<BOOL>>, OK => <<BOOL>> A49I => <<BOOL>>, A50I => <<BOOL>>, A51I => <<BOOL>>, A52I => <<BOOL>>, A53I => <<BOOL>>, A54I => <<BOOL>>, A55I => <<BOOL>>, A56I => <<BOOL>>);

This function block checks to see which digitizing axes numbered from 49 to 56 have been initialized by the user's servo setup function.

The OK output of the STRTSERV function should be wired directly to the enable (EN01) input of this function.

The outputs of this function will remain set even after the function is no longer enabled.

The outputs of this function can be used to ensure the correct setup information has been used. They can also be used as conditional contacts to qualify other functions.



Inputs: EN00 (BOOL) - enables execution

Outputs: OK (BOOL) - execution complete

A57I (BOOL) - set if axis number 57 has been initialized

A58I (BOOL) - set if axis number 58 has been initialized

A59I (BOOL) - set if axis number 59 has been initialized

A60I (BOOL) - set if axis number 60 has been initialized

A61I (BOOL) - set if axis number 61 has been initialized

A62I (BOOL) - set if axis number 62 has been initialized

A63I (BOOL) - set if axis number 63 has been initialized

A64I (BOOL) - set if axis number 64 has been initialized

<<INSTANCE NAME>>:M_CHK57(EN00 := <<BOOL>>, OK => <<BOOL>> A57I => <<BOOL>>, A58I => <<BOOL>>, A59I => <<BOOL>>, A60I => <<BOOL>>, A61I => <<BOOL>>, A62I => <<BOOL>>, A63I => <<BOOL>>, A64I => <<BOOL>>);

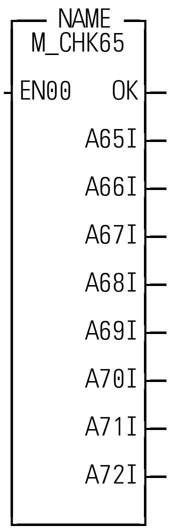
This function block checks to see which digitizing axes numbered from 57 to 64 have been initialized by the user's servo setup function.

The OK output of the STRTSERV function should be wired directly to the enable (EN00) input of this function.

The outputs of this function will remain set even after the function is no longer enabled.

The outputs of this function can be used to ensure the correct setup information has been used. They can also be used as conditional contacts to qualify other functions.

M_CHK65



Inputs: EN00 (BOOL) - enables execution

Outputs: OK (BOOL) - execution complete

A65I (BOOL) - set if axis number 65 has been initialized

A66I (BOOL) - set if axis number 66 has been initialized

A67I (BOOL) - set if axis number 67 has been initialized

A68I (BOOL) - set if axis number 68 has been initialized

A69I (BOOL) - set if axis number 69 has been initialized

A70I (BOOL) - set if axis number 70 has been initialized

A71I (BOOL) - set if axis number 71 has been initialized

A72I (BOOL) - set if axis number 72 has been initialized

<<INSTANCE NAME>>:M_CHK65(EN00 := <<BOOL>>, OK => <<BOOL>> A65I => <<BOOL>>, A66I => <<BOOL>>, A67I => <<BOOL>>, A68I => <<BOOL>>, A69I => <<BOOL>>, A70I => <<BOOL>>, A71I => <<BOOL>>, A72I => <<BOOL>>);

This function block checks to see which digitizing axes numbered from 65 to 72 have been initialized by the user's servo setup function.

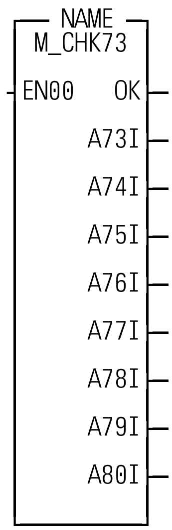
The OK output of the STRTSERV function should be wired directly to the enable (EN00) input of this function.

The outputs of this function will remain set even after the function is no longer enabled.

The outputs of this function can be used to ensure the correct setup information has been used. They can also be used as conditional contacts to qualify other functions.

M_CHK73

Check for Digitizing Axis Initialized **USER/M_INIT**



Inputs: EN00 (BOOL) - enables execution

Outputs: OK (BOOL) - execution complete

A73I (BOOL) - set if axis number 73 has been initialized

A74I (BOOL) - set if axis number 74 has been initialized

A75I (BOOL) - set if axis number 75 has been initialized

A76I (BOOL) - set if axis number 76 has been initialized

A77I (BOOL) - set if axis number 77 has been initialized

A78I (BOOL) - set if axis number 78 has been initialized

A79I (BOOL) - set if axis number 79 has been initialized

A80I (BOOL) - set if axis number 80 has been initialized

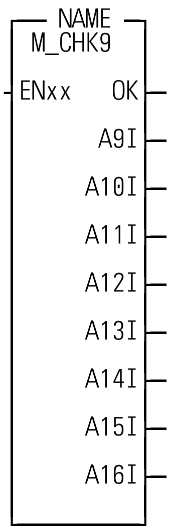
<<INSTANCE NAME>>:M_CHK73(EN00 := <<BOOL>>, OK => <<BOOL>> A73I => <<BOOL>>, A74I => <<BOOL>>, A75I => <<BOOL>>, A76I => <<BOOL>>, A77I => <<BOOL>>, A78I => <<BOOL>>, A79I => <<BOOL>>, A80I => <<BOOL>>);

This function block checks to see which digitizing axes numbered from 73 to 80 have been initialized by the user's servo setup function.

The OK output of the STRTSERV function should be wired directly to the enable (EN00) input of this function.

The outputs of this function will remain set even after the function is no longer enabled.

The outputs of this function can be used to ensure the correct setup information has been used. They can also be used as conditional contacts to qualify other functions.



Inputs: ENxx (BOOL) - enables execution

Outputs: OK (BOOL) - execution complete

A9I (BOOL) - set if axis number 9 has been initialized

A10I (BOOL) - set if axis number 10 has been initialized

A11I (BOOL) - set if axis number 11 has been initialized

A12I (BOOL) - set if axis number 12 has been initialized

A13I (BOOL) - set if axis number 13 has been initialized

A14I (BOOL) - set if axis number 14 has been initialized

A15I (BOOL) - set if axis number 15 has been initialized

A16I (BOOL) - set if axis number 16 has been initialized

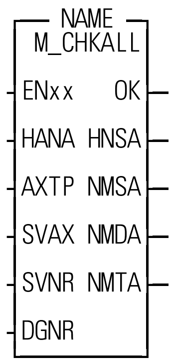
<<INSTANCE NAME>>:M_CHK9(EN00 := <<BOOL>>, OK => <<BOOL>>
A9I => <<BOOL>>, A10I => <<BOOL>>, A11I => <<BOOL>>, A12I =>
<<BOOL>>, A13I => <<BOOL>>, A14I => <<BOOL>>, A15I =>
<<BOOL>>, A16I => <<BOOL>>);

This function block checks to see which servo axes numbered from 9 to 16 have been initialized by the user's servo setup function.

The OK output of the STRTSERV function should be wired directly to the enable (EN00) input of this function.

The outputs of this function will remain set even after the function is no longer enabled.

The outputs of this function can be used to ensure the correct setup information has been used. They can also be used as conditional contacts to qualify other motion functions.



Inputs: ENxx (BOOL) - enable FB - turn on only after the drives have initialized OK (xx indicates revision #)

HANA (USINT) - Highest axis number allowed by PiCPro

AXTP (USINT) - Array sized for HANA + 1. See below.

SVAX (BOOL) - Array sized for HANA + 1. See below.

SVNR (USINT) - Array sized for number of servo axes + 1. See below.

DGNR (USINT) - Array sized for number of digitizing axes + 1. See below.

Outputs: OK (BOOL) - Block executed OK

HNSA (USINT) - Highest numbered servo axis found

NMSA (USINT) - Number of servo axes found

NMDA (USINT) - Number of digitizing axes found

NMTA (USINT) - Number of time axes found

<<INSTANCE NAME>>:M_CHKALL(ENxx := <<BOOL>>, HANA := <<USINT>>, AXTP := <<MEMORY AREA>>, SVAX := <<MEMORY AREA>>, SVNR := <<MEMORY AREA>>, DGNR := <<MEMORY AREA>>, OK => <<BOOL>>, HNSA => <<USINT>>, NMSA => <<USINT>>, NMDA => <<USINT>>, NMTA => <<USINT>>);

This function block checks for information about the axes that have been initialized by the servo setup function and STRTSERV or DSTRTSRV. It provides information in several ways about the number and types of all axes that have been initialized.

Its results can be checked manually to make sure all the axes expected are present. It may also be used in the ladder for automatic configuration of features or some safety checking.

Note on Axis Numbering

PiCPro 16 Pro Edition allows 132 axis numbers, consisting of 64 servo / 64 digitizing and 4 time axes assigned how the user wants. Users can assign any number to any axis and there can be gaps in the axes numbers used. However, for more efficient use of memory, it is better to keep the servo axes numbers as low as possible. Servo axis array memory is assigned for all axes up to the highest numbered one.

Virtual axes are counted in the servo axis quantities.

AXTP[N] will contain a number identifying axis type for that number.

Values are 0= none, 1 = servo / virtual, 2 = digitizing, 3 = time

SVAX [N] will be set if a servo axis of number N is present

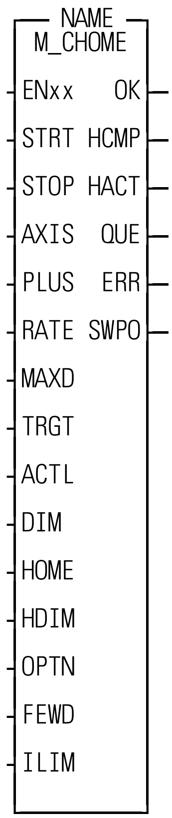
SVNR [1] will contain the axis number of the 1st servo axis found, SVNR [2] the 2nd and so on. If an element is 0, there is no servo axis present.

DGNR [1] will contain the axis number of the 1st digitizing axis found, DGNR [2] the 2nd and so on. If an element is 0, there is no digitizing axes present there.

HNSA should be used in the application ladder to check that the highest numbered servo axis found is not greater than the number specified for the array sizes for the axes data.

M_CHOME

Performs a Reference Cycle Using a Hard Mechanical Stop **USER/M_REF**



Inputs:

ENxx (BOOL) - Enable FB - on at all times (xx indicates revision #)

STRT (BOOL) - Start Home Cycle - one shot on to start

STOP (BOOL) - Stop Home Cycle - turn on to stop

AXIS (USINT) - Axis number to be homed

PLUS (BOOL) - Home initial direction - on for plus, off for minus

RATE (UDINT) - Home rate in LU per min

MAXD (DINT) - Maximum distance to travel looking for current/foll. err home in LU

TRGT (DINT) - Target current / following error setting for home position

ACTL (DINT) - Actual motor current / position loop following error

DIM (DINT) - Value axis position is set to at home target / marker pulse

HOME (BOOL) - On if a move is to be made after home target / marker seen

HDIM (DINT) - Dimension to move to if Home input is on

OPTN (WORD) - Options for home cycle involving use of marker pulse

FEWD (DINT) - %(100-200) to change following error limit to during home cycle

ILIM (DINT) - %(1-100) to set drive direction current limit at while homing

Outputs:

OK (BOOL) - Home cycle command accepted

HCMP (BOOL) - Home cycle completed successfully

HACT (BOOL) - Home cycle is active

QUE (USINT) - Home queue number

ERR (BYTE) - Home command error number - see below

SWPO (DINT) - Distance in feedback units (FU) from the reference switch to the index mark of an encoder or the null of a resolver

<<INSTANCE NAME>>:M_CHOME(ENxx := <<BOOL>>, STRT := <<BOOL>>, STOP := <<BOOL>>, AXIS := << USINT>>, PLUS := <<BOOL>>, RATE := <<UDINT>>, MAXD := <<DINT>>, TRGT := <<DINT>>, ACTL := <<DINT>>, DIM := <<DINT>>, HOME := <<BOOL>>, HDIM := <<DINT>>, OPTN := <<WORD>>, FEWD := <<DINT>>, ILIM := <<DINT>>, OK => <<BOOL>>, HCMP => <<BOOL>>, HACT => <<BOOL>>, QUE => <<USINT>>, ERR => <<BYTE>>, SWPO => <<DINT>>);

This function block performs a reference cycle on an axis using a hard mechanical stop as the target position. This is detected by using motor current or position loop following error setting. Once sensed, it can be optionally followed by a reversal to the motor index pulse and / or a homing (position) move to a designated location.

Before this function block can be used, the axis must be initialized and the position loop closed with no errors present.

The block will remove any axis position rollover value before it runs and restore it afterward. This will help avoid the possibility of driving the axis further on to the hard stop due to the settings of the DIM and HDIM inputs and any rollover setting that may be in use. The block will set the following error limit, to avoid nuisance trips, to the % of the FEWD input - range 100-200%.

The drive current limit in the homing direction can be reduced during the home cycle to the % value at the ILIM input - range 10-100%.

NOTE: If the scan is stopped or the control is turned off while M_CHOME is active, the following error and/or current limit will be set to the adjusted values when the scan is restarted. Thus you should set these to the correct values after the servos are initialized.

The reference cycle will cause the selected axis to move in the designated direction until the ACTL input is greater than the TRGT input. The actual motor current or following error is not measured by this block and must be provided at the ACTL input. It is up to the main ladder program to filter the actual reading if required to avoid false tripping of the home point detection due to acceleration or other axis conditions. The function block takes the absolute values of the ACTL and TRGT inputs, so the programmer need not be concerned about sign.

When the target hard stop is sensed the axis may optionally back off to the next index mark of an encoder or the nearest null of a resolver. It will then reference (assign a value) to the position using the data at the DIM input. After the value is assigned the axis will reposition to the DIM value +/- 1LU if the HOME input is off. The 1 LU moves the axis off the hard stop to allow the current to drop off. Otherwise, it tends to stay high due to the integral action in the drive loops.

If the HOME input is on when the reference done is sensed, a move will automatically be triggered to position the axis at a desired location specified at the HDIM input. This position move must be away from the hard stop or an error will occur and the cycle will not start.

The SWPO output is used to determine if the hard stop location will allow for repeatable referencing if the index mark is to be used. If the hard stop is too close to the index mark of an encoder or the null of a resolver it could possibly intermittently reference a revolution off due to the effect of ladder scan time. To prevent this, the value reported by this output should be as follows:

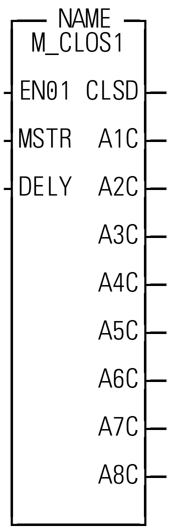
For an encoder system it should be greater than 25% and less than 75% of the total counts (FU) per revolution. Example: For 8000 FU/Rev, the value should be >2000 and <6000. For a resolver system the value of this output should be less than 25% or greater than 75% of the total counts (FU) per revolution. Example: For 4000 FU/Rev, the value should be <1000 or >3000. If the value is out of range either the hard stop will have to be moved or the transducer coupling shifted.

See the table below for a description of valid OPTN input values.

OPTN	Description
0	Do full cycle
1	Ignore index mark search

See the table below for a description of ERR output values.

ERR	Description
1	Final home move point is beyond hard stop
2	Invalid option selected
3	Active queue not available to start home move
4	Failed to read/write rollover, reduced current or expanded following error limit
5	Distance move did not start
6	Target limit sensed - no move in progress
7	Hard stop not reached - distance move ended first
8	Distance move abort failed
9	Part Reference failed
10	Move to index mark did not start
11	Reference End function not OK
12	Move to final position did not start
13	Home cycle stopped from Stop input
14	Axis C stop or P stop occurred during home cycle
15	Restore of original rollover or following error limit failed
16	Following error increase % out of range
17	Current limit % out of range



Inputs: EN01 (BOOL) - enables execution

MSTR (BOOL) - machine start input

DELY (TIME) - amount of time that will elapse after a positive transition of MSTR until the loops will be closed

Outputs: CLSD (BOOL) - one or more of axes 1 to 8 have their position loops closed

A1C (BOOL) - set when the loop is closed on axis 1

A2C (BOOL) - set when the loop is closed on axis 2

A3C (BOOL) - set when the loop is closed on axis 3

A4C (BOOL) - set when the loop is closed on axis 4

A5C (BOOL) - set when the loop is closed on axis 5

A6C (BOOL) - set when the loop is closed on axis 6

A7C (BOOL) - set when the loop is closed on axis 7

A8C (BOOL) - set when the loop is closed on axis 8

<<INSTANCE NAME>>:M_CLOS1(EN01 := <<BOOL>>, MSTR := <<BOOL>>, DELY := <<TIME>>, CLSD => <<BOOL>> A1C => <<BOOL>>, A2C => <<BOOL>>, A3C => <<BOOL>>, A4C => <<BOOL>>, A5C => <<BOOL>>, A6C => <<BOOL>>, A7C => <<BOOL>>, A8C => <<BOOL>>);

This function block is used to reset the E-stop, C-stop, and programming errors on servo axes 1 through 8 when the machine start input is pulsed. It closes the loop on servo axes 1 through 8 after the machine start input is pulsed and a programmable time delay has elapsed. It drops the loop closed flag if an E-stop fault occurs.

This function block can be enabled every scan. If the enable input changes from ON to OFF during the time delay after machine start, the function block will abort the time delay and not close the position loops.

If there are conditions that should abort the sequence to close the position loops (such as an electrical E-stop condition during the time delay), then the enable should include both the positive transition of the machine start input and the current state of the electrical E-stop status as shown below.

```
     ESTOPACT
--| / |----+
           |
     MACHSTRT   |
--| P |----+---- ENxx
```

The reason for these two input conditions is to provide the enable at the start of the time delay (with the P contact of the machine start signal) and to maintain the enable during the time delay as needed (with the NC contact for the electrical E-stop condition).

The MMC example applications located on the Applications CD (in the examples sub-directory) illustrate the recommended ladder logic for the E-stop handling of the M_CLOS1 application. Please refer to MMC2_EX.LDO for an example of M_CLOS1.

The machine start input must go through a positive transition (off to on) to reset the errors and close the loop.

The time at DELY is normally in the range from 500 ms to 2 sec.

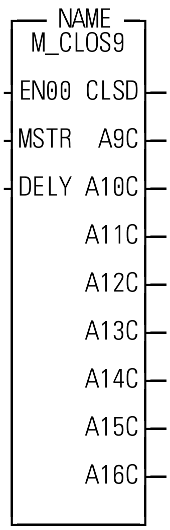
On a positive transition of MSTR, this function will reset all E-stop, C-stop, and programming errors on axes 1 through 8.

The positive transition of MSTR enables a timer with a preset time of DELY. After DELY has elapsed, the loops will be closed on axes 1 to 8. CLSD will be energized if one or more axes 1 to 8 have their position loops closed. The delay allows the drive some time to power up before it starts controlling the axis.

If an E-stop fault occurs on any of axis 1 to 8, its loop closed output (A1 to A8) will be dropped. CLSD is true as long as one or more of axes 1 to 8 have their position loops closed.

M_CLOS9

Close Loop on Servo Axes 9 to 16 **USER/M_INIT**



Inputs: EN00 (BOOL) - enables execution

MSTR (BOOL) - machine start input

DELY (TIME) - amount of time that will elapse after a positive transition of MSTR until the loops will be closed

Outputs: CLSD (BOOL) - one or more of axis 9 to 16 have their position loops closed

A9C (BOOL) - set when the loop is closed on axis 9

A10C (BOOL) - set when the loop is closed on axis 10

A11C (BOOL) - set when the loop is closed on axis 11

A12C (BOOL) - set when the loop is closed on axis 12

A13C (BOOL) - set when the loop is closed on axis 13

A14C (BOOL) - set when the loop is closed on axis 14

A15C (BOOL) - set when the loop is closed on axis 15

A16C (BOOL) - set when the loop is closed on axis 16

<<INSTANCE NAME>>:M_CLOS9(EN00 := <<BOOL>>, MSTR := <<BOOL>>, DELY := <<TIME>>, CLSD => <<BOOL>> A9C => <<BOOL>>, A10C => <<BOOL>>, A11C => <<BOOL>>, A12C => <<BOOL>>, A13C => <<BOOL>>, A14C => <<BOOL>>, A15C => <<BOOL>>, A16C => <<BOOL>>);

This function block is used to reset the E-stop, C-stop, and programming errors on servo axes 9 through 16 when the machine start input is pulsed. It closes the loop on servo axes 9 through 16 after the machine start input is pulsed and a programmable time delay has elapsed. It drops the loop closed flag if an E-stop fault occurs.

This function block can be enabled every scan. If the enable input changes from ON to OFF during the time delay after machine start, the function block will abort the time delay and not close the position loops.

If there are conditions that should abort the sequence to close the position loops (such as an electrical E-stop condition during the time delay), then the enable should include both the positive transition of the machine start input and the current state of the electrical E-stop status as shown below.

```
      ESTOPACT
 - - -| / |- - - - -┐
                    |
      MACHSTRT      |
 - - -| P |- - - - -┴- - - - - ENxx
```

The reason for these two input conditions is to provide the enable at the start of the time delay (with the P contact of the machine start signal) and to maintain the enable during the time delay as needed (with the NC contact for the electrical E-stop condition).

The MMC example applications located on the Applications CD (in the examples sub-directory) illustrate the recommended ladder logic for the E-stop handling of the M_CLOSx application. Please refer to MMC2_EX.LDO for an example of M_CLOS1 usage that applies to M_CLOS9 as well.

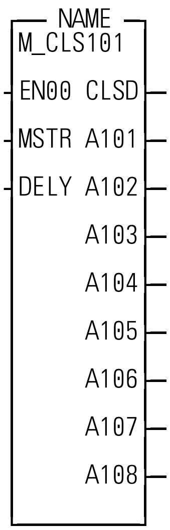
The machine start input must go through a positive transition (off to on) to reset the errors and close the loop.

The time at DELY is normally in the range from 500 ms to 2 sec.

On a positive transition of MSTR, this function will reset all E-stop, C-stop, and programming errors on axes 9 through 16.

The positive transition of MSTR enables a timer with a preset time of DELY. After DELY has elapsed, the loops will be closed on axes 9 to 16. CLSD will be energized if one or more of axes 9 to 16 have their position loops closed. The delay allows the drive some time to power up before it starts controlling the axis.

If an E-stop fault occurs on any of axis 9 to 16, its loop closed output (A9 to A16) will be dropped. CLSD is true as long as one or more of axes 9 to 16 have their position loops closed.



Inputs: EN00 (BOOL) - enables execution

MSTR (BOOL) - machine start input

DELY (TIME) - amount of time that will elapse after a positive transition of MSTR until the loops will be closed

Outputs: CLSD (BOOL) - one or more of axes 101 to 108 have their position loops closed

A101 (BOOL) - set when the loop is closed on axis called 101 (the 17th defined axis)

A102 (BOOL) - set when the loop is closed on axis called 102 (the 18th defined axis)

A103 (BOOL) - set when the loop is closed on axis called 103 (the 19th defined axis)

A104 (BOOL) - set when the loop is closed on axis called 104 (the 20th defined axis)

A105 (BOOL) - set when the loop is closed on axis called 105 (the 21st defined axis)

A106 (BOOL) - set when the loop is closed on axis called 106 (the 22nd defined axis)

A107 (BOOL) - set when the loop is closed on axis called 107 (the 23rd defined axis)

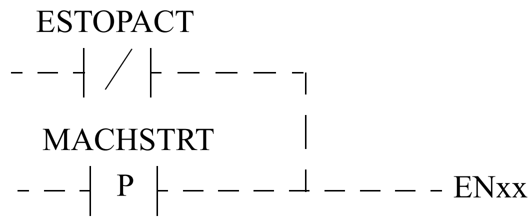
A108 (BOOL) - set when the loop is closed on axis called 108 (the 24th defined axis)

<<INSTANCE NAME>>:M_CLS101(EN00 := <<BOOL>>, MSTR := <<BOOL>>, DELY := <<TIME>>, CLSD => <<BOOL>> A101 => <<BOOL>>, A102 => <<BOOL>>, A103 => <<BOOL>>, A104 => <<BOOL>>, A105 => <<BOOL>>, A106 => <<BOOL>>, A107 => <<BOOL>>, A108 => <<BOOL>>);

This function block is used to reset the E-stop, C-stop, and programming errors on servo axes called 101 through 108 (the 17th to the 24th defined axes) when the machine start input is pulsed. It closes the loop on servo axes 101 through 108 after the machine start input is pulsed and a programmable time delay has elapsed. It drops the loop closed flag if an E-stop fault occurs.

This function block can be enabled every scan. If the enable input changes from ON to OFF during the time delay after machine start, the function block will abort the time delay and not close the position loops.

If there are conditions that should abort the sequence to close the position loops (such as an electrical E-stop condition during the time delay), then the enable should include both the positive transition of the machine start input and the current state of the electrical E-stop status as shown below.



The reason for these two input conditions is to provide the enable at the start of the time delay (with the P contact of the machine start signal) and to maintain the enable during the time delay as needed (with the NC contact for the electrical E-stop condition).

The MMC example applications located on the Applications CD (in the examples sub-directory) illustrate the recommended ladder logic for the E-stop handling of the M_CLOSx application. Please refer to MMC2_EX.LDO for an example of M_CLOS1 usage that applies to M_CLS101 as well.

The machine start input must go through a positive transition (off to on) to reset the errors and close the loop.

The time at DELY is normally in the range from 500 ms to 2 sec.

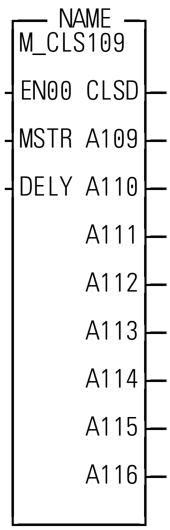
On a positive transition of MSTR, this function will reset all E-stop, C-stop, and programming errors on axes 101 to 108.

The positive transition of MSTR enables a timer with a preset time of DELY. After DELY has elapsed, the loops will be closed on axes 101 to 108. CLSD will be energized if one or more of axes 101 to 108 have their position loops closed. The delay allows the drive some time to power up before it starts controlling the axis.

If an E-stop fault occurs on any of axis 101 to 108, its loop closed output (A101 to A108) will be dropped. CLSD is true as long as one or more of axes 101 to 108 have their position loops closed.

M_CLS109

Close Loop on Servo Axes 109-116 (25th to 32nd **USER/M_INIT**



Inputs: EN00 (BOOL) - enables execution

MSTR (BOOL) - machine start input

DELY (TIME) - amount of time that will elapse after a positive transition of MSTR until the loops will be closed

Outputs: CLSD (BOOL) - one or more of axes 109 to 116 have their position loops closed

A109 (BOOL) - set when the loop is closed on axis called 109 (the 25th defined axis)

A110 (BOOL) - set when the loop is closed on axis called 110 (the 26th defined axis)

A111 (BOOL) - set when the loop is closed on axis called 111 (the 27th defined axis)

A112 (BOOL) - set when the loop is closed on axis called 112 (the 28th defined axis)

A113 (BOOL) - set when the loop is closed on axis called 113 (the 29th defined axis)

A114 (BOOL) - set when the loop is closed on axis called 114 (the 30th defined axis)

A115 (BOOL) - set when the loop is closed on axis called 115 (the 31st defined axis)

A116 (BOOL) - set when the loop is closed on axis called 116 (the 32nd defined axis)

<<INSTANCE NAME>>:M_CLOS9(EN00 := <<BOOL>>, MSTR := <<BOOL>>, DELY := <<TIME>>, CLSD => <<BOOL>> A9C => <<BOOL>>, A10C => <<BOOL>>, A11C => <<BOOL>>, A12C => <<BOOL>>, A13C => <<BOOL>>, A14C => <<BOOL>>, A15C => <<BOOL>>, A16C => <<BOOL>>);

This function block is used to reset the E-stop, C-stop, and programming errors on servo axes called 109 through 116 (the 25th to the 32nd defined axes) when the machine start input is pulsed. It closes the loop on servo axes 109 through 116 after the machine start input is pulsed and a programmable time delay has elapsed. It drops the loop closed flag if an E-stop fault occurs.

This function block can be enabled every scan. If the enable input changes from ON to OFF during the time delay after machine start, the function block will abort the time delay and not close the position loops.

If there are conditions that should abort the sequence to close the position loops (such as an electrical E-stop condition during the time delay), then the enable should include both the positive transition of the machine start input and the current state of the electrical E-stop status as shown below.

```
   ESTOPACT
--┤ / ├----┐
           |
   MACHSTRT|
--┤ P ├----┴---- ENxx
```

The reason for these two input conditions is to provide the enable at the start of the time delay (with the P contact of the machine start signal) and to maintain the enable during the time delay as needed (with the NC contact for the electrical E-stop condition).

The MMC example applications located on the Applications CD (in the examples sub-directory) illustrate the recommended ladder logic for the E-stop handling of the M_CLOSx application. Please refer to MMC2_EX.LDO for an example of M_CLOS1 usage that applies to M_CLS109 as well.

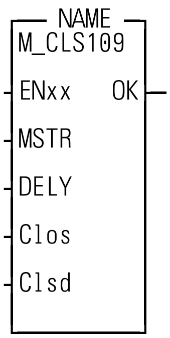
The machine start input must go through a positive transition (off to on) to reset the errors and close the loop.

The time at DELY is normally in the range from 500 ms to 2 sec.

On a positive transition of MSTR, this function will reset all E-stop, C-stop, and programming errors on axes 109 to 116.

The positive transition of MSTR enables a timer with a preset time of DELY. After DELY has elapsed, the loops will be closed on axes 109 to 116. CLSD will be energized if one or more of axes 109 to 116 have their position loops closed. The delay allows the drive some time to power up before it starts controlling the axis.

If an E-stop fault occurs on any of axis 109 to 116, its loop closed output (A109 to A116) will be dropped. CLSD is true as long as one or more of axes 109 to 116 have their position loops closed.



Inputs: ENxx (BOOL) - enables execution, (xx indicates revision #)

MSTR (BOOL) - machine start input

DELY (TIME) - amount of time that will elapse after a positive transition of MSTR until the loops will be closed

Clos (BOOL[0..132]) - array of bools indicating which axes should be closed.

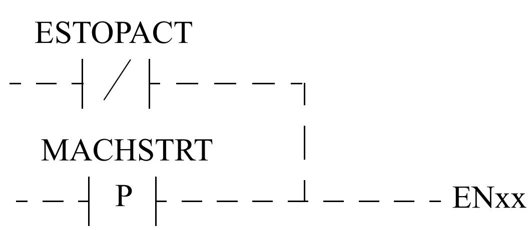
Clsd (BOOL[0..132]) - array of bools indicating which axes have been closed. Clsd[0] indicates if any are closed.

Outputs: OK (BOOL) - execution complete

<<INSTANCE NAME>>:M_CLSALL(ENxx:=1,MSTR:=<<BOOL>>, DELY:=<<TIME>>,Clos:=<<MEMORY AREA>>,Clsd:=<<MEMORY AREA>>,OK=><<BOOL>>);

This function block is used to reset the E-stop, C-stop, and programming errors on the servo axes when the machine start input is pulsed. It closes the loop on servo axes after the machine start input is pulsed and a programmable time delay has elapsed. It drops the loop closed flag if an E-stop fault occurs. This function block can be enabled every scan. If the enable input changes from ON to OFF during the time delay after machine start, the function block will abort the time delay and not close the position loops.

If there are conditions that should abort the sequence to close the position loops (such as an electrical E-stop condition during the time delay), then the enable should include both the positive transition of the machine start input and the current state of the electrical E-stop status as shown below.



The reason for these two input conditions is to provide the enable at the start of the time delay (with the P contact of the machine start signal) and to maintain the enable during the time delay as needed (with the NC contact for the electrical E-stop condition).

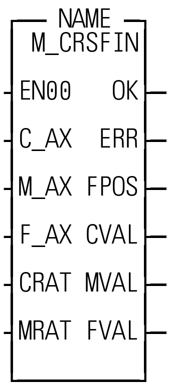
The machine start input must go through a positive transition (off to on) to reset the errors and close the loop.

The time at DELY is normally in the range from 500 ms to 2 sec.

On a positive transition of MSTR, this function will reset all E-stop, C-stop, and programming errors on the axes.

The positive transition of MSTR enables a timer with a preset time of DELY. After DELY has elapsed, the loops will be closed on the axes. CLSD[0] will be energized if one or more axes have their position loops closed. The delay allows the drive some time to power up before it starts controlling the axis.

If an E-stop fault occurs on any axis, its loop closed output (CLSD[Axis]) will be dropped. CLSD[0] is true as long as one or more of the axes have their position loops closed.



Inputs: EN00 (BOOL) - enables execution

C_AX (USINT) - coarse resolver axis number

M_AX (USINT) - medium resolver axis number

F_AX (USINT) - fine resolver axis number

CRAT (DINT) - coarse to medium or coarse to fine ratio

MRAT (DINT) - medium to fine ratio

Outputs: OK (BOOL) - execution complete without error

ERR (INT) - error number

FPOS (DINT) - position that the fine axis has been part referenced to

CVAL (DINT) - value read from the coarse resolver

MVAL (DINT) - value read from the medium resolver

FVAL (BOOL) - value read from the fine resolver

<<INSTANCE NAME>>:M_CLS101(EN00 := <<BOOL>>, C_AX := <<USINT>>, M_AX := <<USINT>>, F_AX:= <<USINT>>, CRAT := <<DINT>>, MRAT := <<DINT>> OK => <<BOOL>> ERR => <<INT>>, FPOS => <<DINT>>, CVAL => <<DINT>>, MVAL => <<DINT>>, FVAL => <<BOOL>>);

This function block reads coarse, medium, and fine resolvers and then part references the fine axis to the value calculated by using coarse, medium and fine. Three separate combinations of resolvers can be used: coarse, medium, and fine; coarse and fine; or medium and fine.

This function block should be one-shot after the axes have been initialized by the user's servo setup function.

The value entered at C_AX is the axis number for the coarse resolver, or 0 if you are not using a coarse resolver.

The value entered at M_AX is the axis number for the medium resolver, or 0 if you are not using a medium resolver.

The value entered at F_AX is the axis number for the fine resolver. This is also the axis that will be part referenced by this function block.

The value entered at CRAT is the coarse to medium ratio if coarse, medium and fine resolvers are being used, or the coarse to fine ratio if only coarse and fine resolvers are being used.

The value entered at MRAT is the medium to fine ratio.

The OK output indicates execution complete without error. If the OK output is not set, then an error has occurred and the error code will be stored in the ERR output. A listing of possible errors is shown below:

ERR	Description
0	No error
1	The OK from the READ_SV function for the fine axis was not set
2	The OK from the READ_SV function for the medium axis was not set
3	The OK from the READ_SV function for the coarse axis was not set
4	M_AX and C_AX inputs are both zero
5	M_AX was non-zero, but MRAT was zero
6	C_AX was non-zero, but CRAT was zero
7	The fine axis is moving or drifting
8	The medium axis is moving or drifting
9	The coarse axis is moving or drifting
10	The fine axis position was not between 0 and 3999
11	The medium axis position was not between 0 and 3999
12	The coarse axis position was not between 0 and 3999
13	An error occurred in the calculations for coarse, medium and fine
14	An error occurred in the calculations for coarse and fine
15	An error occurred in the calculations for medium and fine
16	The OK from the part reference function for F_AX did not get set

The FPOS output will show the final value that the fine axis has been part reference to.

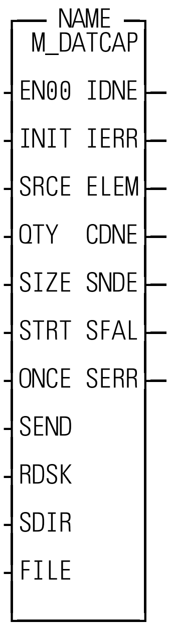
The CVAL output will show the value read from the coarse resolver.

The MVAL output will show the value read from the medium resolver.

The FVAL output will show the value read from the fine resolver.

M_DATCAP

Captures Axis Information **USER/M_DATA**



Inputs: EN00 (BOOL) - enables execution

INIT (BOOL) - initializes data capture memory area

SRCE (STRUCT(0..7)) - defines axis number and variable number to capture

QTY (USINT) - defines the number of variables to capture (This is the same as the number of elements used in the SRCE array.)

SIZE (UINT) - defines the number of samples to be captured

STRT (BOOL) - starts data capture

ONCE (BOOL) - set to capture data once; reset to capture data continuously

SEND (BOOL) - starts save of captured data to RAMDISK or workstation

RDSK (BOOL) - set if data will be saved to the RAMDISK or reset if data will be saved to the workstation

SDIR (STRING) - the subdirectory on the workstation or RAMDISK to send the data to (an eight
character maximum)

FILE (STRING) - the file name that the data will be saved as (a 12 character maximum)

Outputs: IDNE (BOOL) - initialization complete without error

IERR (USINT) - error number that occurred during initialization

ELEM (UINT) - the element number currently being captured

CDNE (BOOL) - capture done

SDNE (BOOL) - file send done

SFAL (BOOL) - file send failed

SERR (INT) - error number that occurred during file send

<<INSTANCE NAME>>:M_DATCAP (EN00 := <<BOOL>>, INIT := <<BOOL>>, SRCE := <<MEMORY AREA>>, QTY:= <<USINT>>, SIZE := <<UINT>>, STRT := <<BOOL>>, ONCE := <<BOOL>> SEND := <<BOOL>>, RDSK := <<BOOL>>, SDIR := <<STRING>>, FILE := <<STRING>>, IDNE => <<BOOL>>, IERR => <<USINT>>, ELEM => <<UINT>>, CDNE => <<BOOL>>, SDNE => <<BOOL>>, SFAL => <<BOOL>>, SERR => <<INT>>);

This function block is considered obsolete. It requires the CAP2ASC.EXE DOS utility to extract the data captured. The M_DATCPT function block performs the same data capture operations as M_DATCAP with the same function block inputs and outputs except M_DATCPT creates an output file that is already in a directly viewable ASCII text format (it is a tab-delimited variable format).

This function block captures axis information on an interrupt basis and stores it in a structure. The structure can then be written out to a binary file on the RAMDISK or the workstation. In order to manipulate the data, convert this binary file to an ASCII text file using the CAP2ASC.exe which is included with the Motion ASFB examples. On your PC, type:

CAP2ASC *filename*

where filename is the name you assigned to the binary file. You can then view and/ or edit this ASCII file using a text editing program or import it into a spreadsheet.

The EN00 input of this function block should be set every scan.

On a positive transition of the INIT input, the values entered at the SRCE input are examined. The SRCE input is an array of structures and must have the following members:

Name	Data Type	Definition
SRCE	STRUCT(0..7)	Defines axis and variables to capture
.AXIS	USINT	Defines the axis to capture data for
.VAR	USINT	Defines the variable to capture

The SRCE(X).VAR input must be one of the following values:

SRCE(X).VAR	Definition
1	Actual Position (variable 1 of READ_SV)*
2	Fast input occurred
3	Commanded position (variable 3 of READ_SV)*
4	Position error (variable 4 of READ_SV)*
5	Filter error (variable 5 of READ_SV)*
6	Command change (variable 6 of READ_SV)*
7	Position change (variable 7 of READ_SV)*
8	Feedback position (variable 8 of READ_SV)*
9	Prefilter commanded
10	Prefilter command change
11	Remaining master offset
12	Remaining slave offset

* The variables in the READ_SV function are reported in ladder units (LU). The variables in DATCAP function block are reported in feedback units (FU).

If an error is found at the SRCE input, then IDNE will not be set and IERR will hold a number describing the error that occurred. If no errors are found at the SRCE input, then IDNE will be set. A listing of possible errors at IERR are shown below:

IERR	Description
0	No error
1	The function block has not stopped capturing data from a previous data capture initialization.
2	An axis number in the structure is invalid
3	The limit of eight variables in the array of structures has been exceeded.
4	Parameter number in the structure is out of range.
5	The initialization was done before the STRTSERV function was called.
6	Reserved
7	Reserved
8	Reserved
9	The total number of bytes to capture exceeds 7992.

The QTY input defines the number of variables that will be captured. This is the same as the number of array elements used in the SRCE input.

The SIZE input defines the number of samples to capture.

When the STRT input is on, if ONCE is also on, the data will be captured once. When the STRT input is on, if ONCE is off, then the data will be captured continuously until the STRT input drops.

While the data is being captured, the ELEM output will show the current element number being captured. When data capture is complete, the CDNE output will be set.

Once the data has been captured, it can be sent to a file on the RAMDISK or workstation. The data will be sent when the SEND input is pulsed. If the RDSK input is ON when the SEND input is pulsed, then the captured data will be sent to the PiC900 RAMDISK. If the RDSK input is OFF when the SEND input is pulsed, then the captured data will be sent to the workstation C: drive.

The file will be saved with the name entered at FILE. This must be of the form FILENAME.EXT.

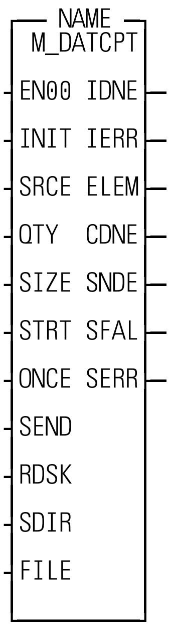
The SDIR input defines the subdirectory where the file will be located. The subdirectory must not exceed eight characters.

When the file has been successfully sent, the SDNE output will be set. If an error occurred in writing the file, then SFAL will be set and SERR will contain a number describing the error that occurred. A list of errors is shown below:

SERR	Description
0	No error
1 to 99	Error occurred on file open
101 to 199	Error occurred on file write
201 to 299	Error occurred on file write
301 to 399	Error occurred on file write
401 to 499	Error occurred on file write
501 to 599	Error occurred on file close

M_DATCPT

Capture Axis data to file **USER/M_DATA**



Inputs: EN00 (BOOL) - enables execution

INIT (BOOL) - initializes data capture memory area

SRCE (STRUCT(0..7)) - defines axis number and variable number for each item to capture

QTY (USINT) - number of variables to capture (This is the same as the number of elements used in the SRCE array.)

SIZE (UINT) - number of samples to be captured

STRT (BOOL) - starts the data capture

ONCE (BOOL) - set to capture data once; reset to capture data continuously

SEND (BOOL) - starts save of captured data to specified file

RDSK (BOOL) - set if data will be saved to the RAMDISK or reset if data will be saved to the PC hard disk.

SDIR (STRING) - name of subdirectory (an eight character maximum)

FILE (STRING) - the file name that the data will be saved as (8.3 format)

Outputs: IDNE (BOOL) - initialization complete without error

IERR (USINT) - initialization error number

ELEM (UINT) - the element number currently being captured

CDNE (BOOL) - capture done

SDNE (BOOL) - file send done

SFAL (BOOL) - file send failed

SERR (INT) - error number that occurred during file send

<<INSTANCE NAME>>:M_DATCPT (EN00 := <<BOOL>>, INIT := <<BOOL>>, SRCE := <<MEMORY AREA>>, QTY:= <<USINT>>, SIZE := <<UINT>>, STRT := <<BOOL>>, ONCE := <<BOOL>> SEND := <<BOOL>>, RDSK := <<BOOL>>, SDIR := <<STRING>>, FILE := <<STRING>>, IDNE => <<BOOL>>, IERR => <<USINT>>, ELEM => <<UINT>>, CDNE => <<BOOL>>, SDNE => <<BOOL>>, SFAL => <<BOOL>>, SERR => <<INT>>);

This function block captures axis information on an interrupt basis and stores it in a structure. The structure can then be written out to a text file on the RAMDISK or the workstation. This text file is directly viewable with any text editor. It is also tab delimited so its possible to import it into some spreadsheet applications. This function block provides simpler control of CAPTINIT and CAPSTAT. Both of these standard functions are documented in the PiCPro Function Block Reference Guide and in on-line Help.

The EN00 input of this function block should be set every scan.

On a positive transition of the INIT input, the values entered at the SRCE input are examined. The SRCE input is an array of structures and must have the following members:

Name	**Data Type**	**Definition**
SRCE	STRUCT(0..7)	Defines axis and variables to capture
.AXIS	USINT	Defines the axis to capture data for
.VAR	USINT	Defines the variable to capture

The SRCE(X).VAR input must be one of the following values:

SRCE(X).VAR	**Definition**
1	Actual Position (variable 1 of READ_SV)*
2	Fast input occurred
3	Commanded position (variable 3 of READ_SV)*
4	Position error (variable 4 of READ_SV)*
5	Filter error (variable 5 of READ_SV)*
6	Command change (variable 6 of READ_SV)*
7	Position change (variable 7 of READ_SV)*
8	Feedback position (variable 8 of READ_SV)*
9	Prefilter commanded position
10	Prefilter command change
11	Remaining master offset
12	Remaining slave offset

13	Command change (variable 6 of READ_SV)
14	Position change (variable 7 of READ_SV)
15	Prefilter command change

* The variables in the READ_SV function are reported in ladder units (LU). The variables in DATCAP function block are reported in feedback units (FU).

If an error is found at the SRCE input, then IDNE will not be set and IERR will hold a number describing the error that occurred. If no errors are found at the SRCE input, then IDNE will be set. A listing of possible errors at IERR are shown below:

IERR	Description
0	No error
1	The function block has not stopped capturing data from a previous data capture initialization.
2	An axis number in the structure is invalid
3	The limit of eight variables in the array of structures has been exceeded.
4	Parameter number in the structure is out of range.
5	The initialization was done before the STRTSERV function was called.
6	Reserved
7	Reserved
8	Reserved
9	The total number of bytes to capture exceeds 7992.

The QTY input defines the number of variables that will be captured. This is the same as the number of array elements used in the SRCE input.

The SIZE input defines the number of samples to capture.

When the STRT input is on, if ONCE is also on, the data will be captured once. When the STRT input is on, if ONCE is off, then the data will be captured continuously until the STRT input drops.

While the data is being captured, the ELEM output will show the current element number being captured. When data capture is complete, the CDNE output will be set.

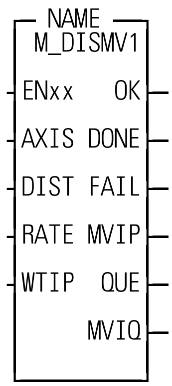
Once the data has been captured, it can be sent to a file on the RAMDISK or workstation. The data will be sent when the SEND input is pulsed. If the RDSK input is ON when the SEND input is pulsed, then the captured data will be sent to the PiC900 RAMDISK. If the RDSK input is OFF when the SEND input is pulsed, then the captured data will be sent to the workstation C: drive.

The file will be saved with the name entered at FILE. This must be of the form FILENAME.EXT.

The SDIR input defines the subdirectory where the file will be located. The subdirectory must not exceed eight characters.

When the file has been successfully sent, the SDNE output will be set. If an error occurred in writing the file, then SFAL will be set and SERR will contain a number describing the error that occurred. A list of errors is shown below:

SERR	Description
0	No error
1 to 99	Error occurred on file open. (See Appendix B in the online help)
101 to 199	Error occurred on file write
201 to 299	Error occurred on file close



Inputs:

ENxx (BOOL) - Enables execution (one shot)

AXIS (USINT) - Axis number

DIST (DINT) - Incremental Distance to move in LU

RATE (UDINT) - Feed rate in LU / minute

WTIP (BOOL) - Wait for In Position signal before completing

Outputs:

OK (BOOL) - Command was accepted

DONE (BOOL) - Distance move is complete

FAIL (BOOL) - No queue available or move is E-stopped or C-stopped

MVIP (BOOL) - Distance move is in progress

QUE (USINT) - Queue number for move

MVIQ (BOOL) - Queued waiting for active move to end

<<INSTANCE NAME>>:M_DISMV1(ENxx := <<BOOL>>, AXIS := <<USINT>>, DIST := <<DINT>>, RATE := <<UDINT>>, WTIP := <<BOOL>>, OK => <<BOOL>>, DONE => <<BOOL>>, FAIL => <<BOOL>>, MVIP => <<BOOL>>, QUE => <<USINT>>, MVIQ => <<BOOL>>);

This block will initiate and track progress on a Distance move for the specified axis. It is mainly intended for use in sequential move applications where each move is followed by another non motion or different axis action. In these situations, the move will typically become active immediately.

However, the block will queue a move as long as a queue is available and it will start when the active move completes.

E-stops and C-stops will cause the motion being monitored by the block to end. In these cases, the OK output will go off and the FAIL output will come on. The DONE output will stay off.

Aborts will also cause the move to terminate. The block cannot tell that an Abort has been done, and the OK and DONE outputs will come on in these cases. There is also one specific case where the MVIQ output will stay on incorrectly. This will occur if a move triggered by this block is in the queue and it is aborted and another Distance move is started before this block runs.

INPUTS:

ENxx - (BOOL) one shot to trigger move

AXIS - (USINT) axis number to position

DIST - (DINT) Incremental Distance to move in LU

RATE - (UDINT) feed rate for move in LU / Minute

WTIP - (BOOL) wait for in position. If this is on, the DONE output will not turn on until the axis is within the in position band. If off, DONE will turn on when the command position stops iterating. If a move is in the queue behind this move, then this setting has no effect and the DONE will be on when the move stops iterating.

OUTPUTS:

OK - (BOOL) turns on if move is accepted and goes off if move does not complete normally

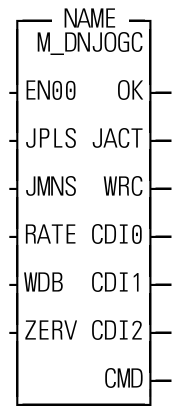
DONE - (BOOL) turns on when move completes, turns off when new move started

FAIL - (BOOL) on if move stopped due to E or C stop or no queue is available when EN turns on

MVIP - (BOOL) distance move is in progress

QUE - (USINT) queue number for distance move

MVIQ - (BOOL) move is in queue waiting for active move to end



Inputs: EN00 (BOOL) - enables execution

JPLS (BOOL) - jog in the PLUS direction (CW)

JMNS (BOOL) - jog in the MINUS direction (CCW)

RATE (DINT) - rate or velocity (programmed as RPM * 65536)

WDB (BOOL) - DeviceNet write data busy flag

ZERV (BOOL) - axis zero velocity - axis has stopped

Outputs: OK (BOOL) - function block is active

JACT (BOOL) - axis jog is active

WRC (BOOL) - write data/command to the drive

CDI0 (BOOL) - command data index - bit 0

CDI1 (BOOL) - command data index - bit 1

CDI2 (BOOL) - command data index - bit 2

CMD (DWORD) - command data value

<<INSTANCE NAME>>:M_DNJOGC (EN00 := <<BOOL>>, JPLS := <<BOOL>>, JMNS := <<BOOL>>, RATE:= <<DINT>>, WDB := <<BOOL>>, ZERV := <<BOOL>>, OK => <<BOOL>>, JACT => <<BOOL>>, WRC => <<BOOL>>, CDI0 => <<BOOL>>, CDI1 => <<BOOL>>, CDI2 => <<BOOL>>, CMD => <<DWORD>>);

This function block is used to allow a manual jog (move at a velocity) of a Centurion DeviceNet Drive axis.

Before this function block can be used, the axis must be enabled and placed into servo lock.

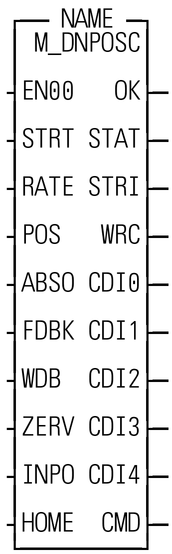
If the enable is active, triggering job plus (JPLS) or jog minus (JMNS) input will cause the specified DeviceNet axis to move at the indicated rate in the corresponding direction. When the input is deactivated, motion will stop.

This function block should be used only to allow an operator to manually move an axis on a machine. It is not designed for any other purpose.

Important - If the enable is disabled while a move is underway the axis will continue to move until the jog switch is deactivated.

The JPLS input enables a move in the positive direction for the selected axis. The JMNS input enables a move in the negative direction for the selected axis.

Rate is programmed in RPM * 65,536. An example: for 100 RPM, Rate = 6553600. If both the JPLS and the JMNS inputs are set; motion will stop until both inputs are dropped and one is again selected.



Inputs: EN00 (BOOL) - enables execution

STRT (BOOL) - start the axis move

RATE (DINT) - rate or velocity (programmed as RPM * 65536)

POS (DINT) - command position in FU

ABSO (BOOL) - absolute or incremental position (set indicates absolute)

FDBK (DWORD) - actual position (feedback) from the drive

WDB (BOOL) - DeviceNet write data busy flag

ZERV (BOOL) - axis is at zero velocity - axis has stopped

INPO (BOOL) - axis is in position - axis is at its commanded
position

HOME (BOOL) - axis is homed

Outputs: OK (BOOL) - function block is active

STAT (INT) - axis status value

STRI (BOOL) - start move indicator

WRC (BOOL) - write data/command to the drive

CDI0 (BOOL) - command data index - bit 0

CDI1 (BOOL) - command data index - bit 1

CDI2 (BOOL) - command data index - bit 2

CDI3 (BOOL) - command data index - bit 3

CDI4 (BOOL) - command data index - bit 4 (not used)

CMD (DWORD) - command data value

<<INSTANCE NAME>>:M_DNPOSC (EN00 := <<BOOL>>, STRT := <<BOOL>>, RATE := <<DINT>>, POS := <<DINT>>, ABSO := <<BOOL>>, ABSO := <<BOOL>>, FDBK := <<DWORD>>, WDB := <<BOOL>>, ZERV := <<BOOL>>, INPPO := <<BOOL>>, HOME := <<BOOL>>, OK => <<BOOL>>, STAT => <<INT>>, STRI => <<BOOL>>, WRC => <<BOOL>>, CDI0 => <<BOOL>>, CDI1 => <<BOOL>>, CDI2 => <<BOOL>>; CDI3 => <<BOOL>>, CDI4 => <<BOOL>>, CMD => <<BOOL>>);

This function block is used to allow a position / index move with a Centurion DeviceNet Drive axis.

Before this function block can be used, the axis must be enabled and placed into servo lock.

If the enable is active, triggering (STRT) input will cause the specified DeviceNet axis to move at the indicated rate to the position endpoint (POS). The axis will travel an incremental distance if the ABSO input is deactivated. The axis will travel to an absolute position if the ABSO input is activated.

Important - If the enable is disabled while a move is underway, the axis will continue to move until it has reached its endpoint.

The Position command (POS) is entered in feedback counts. (Example: for an 8000 counts/rev encoder and an incremental move, Position = 16000 will result in a move of 2 revolutions).

Rate is programmed in RPM * 65,536. For example, for 100 RPM, Rate = 6553600.

The axis status (STAT) will indicate the status of the axis based on the following code:

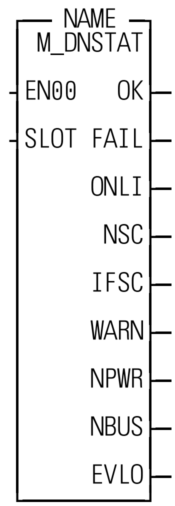
1 = Axis is Positioning

2 = Absolute mode: the command is equal to current position

3 = Incremental mode: the command is equal to zero

4 = Rate is equal to zero

5 = Absolute mode and axis is not Homed



Inputs: EN00 (BOOL) - enables execution

SLOT (USINT) - slot number for the DeviceNet module

Outputs: OK (BOOL) - execution complete

FAIL (BOOL) - failure getting the DeviceNet status

ONLI - (BOOL) - DeviceNet module is online

NSC (BYTE) - DeviceNet Network Status Code

IFSC (BYTE) - DeviceNet Interface Status Code

WARN (BOOL) - DeviceNet communication error warning

NPWR (BOOL) - No DeviceNet bus power

NBUS (BOOL) - No DeviceNet bus connection

EVLO (BOOL) - DeviceNet event was lost due to full event
queue

<<INSTANCE NAME>>:M_DNSTAT (EN00 := <<BOOL>>, SLOT := <<USINT>>, OK => <<BOOL>>, FAIL =>, <<BOOL>>, ONLI => <<BOOL>>, NSC => <<BYTE>>, IFSC => <<BYTE>>, WARN => <<BOOL>>, NPWR => <<BOOL>>, NBUS => <<BOOL>>, EVLO =>);

This function block obtains the DeviceNet network and interface status conditions. Those conditions are presented in outputs as bytes and booleans.

ONLI is set if the DeviceNet module is communicating with nodes.

NSC is the status of the DeviceNet module network interface.

0 = network interface is offline.

1 = network interface is offline due to a network fault.

2 = network interface is offline due to a configuration fault.

3 = network interface is online and no faults are detected.

4 = network interface is online but one or more network services have failed.

5 = network interface is online and is exchanging data; no faults are detected.

6 = network interface is online and is exchanging data; one or more network services is receiving an idle indication; no faults are detected.

7 = network interface is online but one or more previously active network services have been suspended; no faults are detected.

IFC is the status of the DeviceNet module data exchange interface.

0 = data exchange interface is closed.

1 = data exchange interface is open

2 = data exchange interface is faulted due to a "heartbeat" timeout.

WARN is set when the communication warning threshold has been exceeded.

NPWR is set when DeviceNet bus power is not present.

NBUS is set when DeviceNet bus is not connected.

EVLO is set when an event was lost due to a full event queue in the DeviceNet module. This flag is cleared when the DeviceNet interface is closed (FB_CLS).

For more information regarding how this information is gathered or the meaning of any of the outputs, consult the FB_STA function description.

Inputs: EN00 (BOOL) - enables execution

PORT (STRING) - identifies the serial communication port

ADDR (USINT) - identifies the Centurion servo drive address

INIT (BOOL) - (one-shot) initializes M_DSMCOM

SEND (BOOL) - (one-shot) executes read or write command

CMD (UINT) - command to execute

WDAT (memory area) source of data for the write command
memory area is a STRING, ARRAY, or STRUCTURE

WNUM (USINT) - number of bytes of data in WDAT

RDAT (memory area) - destination of data returned by the read command
memory area is a STRING, ARRAY, or STRUCTURE

Outputs: DONE (BOOL) - command executed without error

FAIL (BOOL) - command encountered an error

FERR (UINT) - PiC format error number

OERR (UINT) - operation error number

DERR (UINT) - Centurion drive error number

RNUM (USINT) - number of bytes of data in RDAT

<<INSTANCE NAME>>:M_DSMCOM (EN00 := <<BOOL>>, PORT := <<STRING>>, ADDR := <<USINT>>, INIT := <<BOOL>>, SEND := <<BOOL>>, CMD := <<UINT>>, WDAT := <<MEMORY AREA>>, DONE => <<BOOL>>, FAIL => <<BOOL>>, FERR => <<UINT>>, DERR => <<UINT>>, RNUM => <<USINT>>);

The M_DSMCOM function block allows the PiC to interface with from 1 to 32 Centurion DSM100 servo drives via RS232 or RS422/RS485 serial communication links. With this function block, various drive parameters can be read and written. These parameters are listed in Appendix A.

The EN00 input of this function block should be set every scan.

The PORT input identifies the serial communication port. If the PiC user port is used, the reserved name USER:$00 is entered. If a serial communication module is used, the name assigned to the port by the ASSIGN function block should be entered. The string can be no longer than 10 characters, with up to eight characters for the name followed by a ":" and the null character"$00".

The ADDR input identifies the Centurion servo drive address. The drive address is set using the sixteen position rotary addressing switch on the drive or via software using DSMPro. The range is 0 to 32.

The INIT input initializes the M_DSMCOM function block. The DONE output will be set when the initialization has successfully completed. This initialization must be executed before a read or write is executed.

The SEND input executes a read or write command.

To execute a read command:

1. Move the command number into the CMD input.
2. One-shot the SEND input.

When the DONE output goes high:

- RDAT will hold the data read.
- RNUM will hold the number of bytes of data read.

To execute a write command:

1. Move the command number into the CMD input.
2. Move the data to write into the WDAT input.
3. Move the number of bytes of data into the WNUM input.
4. One-shot the SEND input.

When the DONE output goes high, the command is complete.

NOTE: Never send a new command until any previous command or initialization has completed. Completion is indicated by the DONE (or FAIL) output going high.

The CMD input specifies which read or write command to execute. See Appendix A for a list of all the available commands.

The WDAT input is the data to be written to the drive. The type and number of data depends on the write command being executed. There are two ways to handle the data to this input:

1. If your application will only be writing one specific command or different commands that are all the same data type, use a structure whose member(s) is/are the correct data type(s) to be sent.
   For example, the write command 0DDH Analog Output Write Value expects an unsigned byte value followed by a signed word value. With this command, you could enter a structure at the WDAT input whose first member is an USINT and whose second member is an INT.
2. If your application will be writing different commands that are different data types, use a structure with one member that is the largest data type and use the PiCPro datatype conversion functions to convert any data to the data type of the structure member before sending the data.

The WNUM input is the number of bytes of data in WDAT.

The RDAT input is the data read from the drive. Following the successful completion of a read command, the memory area pointed to by the RDAT input holds the data read from the drive. The RNUM output will indicate the number of bytes of data read. The type and number of data depends on the read command being executed. Again, there are two ways to handle this data.

1. If your application will only be reading one specific command or different commands that are all the same data type, use a structure whose member(s) is/are the correct data type(s) to be sent.
   For example, the read command 042H Gear Ratio reads two signed word values. With this command, you could enter a structure at the RDAT input with two INT members.
2. If your application will be reading different commands that are different data types, use a structure with one member that represents the largest data type and use the PiCPro conversion functions to convert any data to its correct data type after reading it.

Outputs

The DONE output will be set if the initialization or a read or write command is completed successfully. The FAIL output will be set if an error occurs during the execution of the initialization or a read or write command.

The FERR output will identify errors encountered by the M_DSMCOM function block when using the PiC serial communications function blocks. The OERR output will identify errors detected when a read or write command is executed. They are described below.

OERR	Description
0	No error

1	Checksum error - invalid checksum in the drive response
2	Timeout error - drive did not respond in time
3	Read or write attempted before initialization
4	Invalid PORT name
5	CMD input out of range
6	ADDR input out of range
7	WNUM input out of range
8	Invalid address in drive response
9	Invalid function in drive response
10	Invalid data in drive response
11	Invalid drive response

The DERR output will identify errors reported by the Centurion drive in a response to a command. They are described below.

DERR	Description
0	No error
1	Invalid data
2	Command not enabled
3	EEPROM write error
4	Data accepted after limiting to minimum
5	Data accepted after limiting to maximum
6	Command disabled when drive is enabled
7	Flash programming error
8	Invalid function code
9	Command disabled when drive is disabled

The RNUM output indicates the number of bytes of data in RDAT after a read command has executed.

Application Notes

1. The M_DSMCOM function block must only be entered in the LDO once for each serial port being used.
2. A read or write command must not be attempted until the function block initialization is complete.
3. A read or write command must not be attempted until a previous read or write command is complete.
4. If no data is being sent with a command (which is the normal mode for most read commands), the WNUM input must be zero.

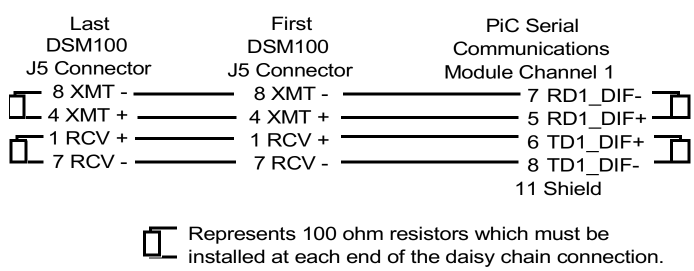
RS232 Connections

In single drive applications where the communications link is less than 50 feet, a three wire RS232 serial communication link may be used. The pinout is shown below.

Drive J5 Serial Port 9-pin D Connector	PiC User Port 10-pin Screw Terminal Connector
2 RCV	10 TD
3 XMT	9 RD
5 COM	8 GRD

RS422/RS485 Connections

Typically, the M_DSMCOM function block will be used with RS422/RS485 serial communication. RS422/RS485 provides superior noise immunity, allows communication links greater than 50 feet, and allows multiple drive connections to one PiC. A four wire daisy chain connection is made between a PiC Serial Communications Module and the DSM100 drives.

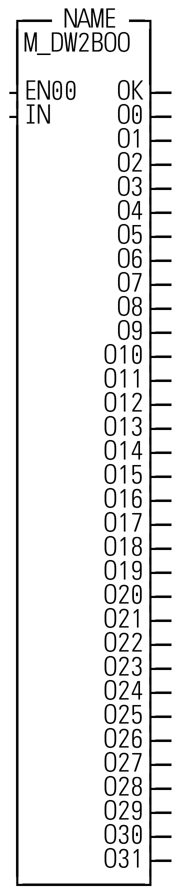


Example LDO with the M_DSMCOM Function Block

Please refer to the example ASFB M_DSM_EX.LDO ladder.

M_DW2BOO

Convert DWORD to BOOLs **USER/M_COMMON**



Inputs: EN01 (BOOL) - enables execution

IN (DWORD) - the data to convert

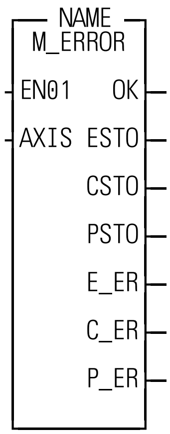
Outputs: OK (BOOL) - execution complete
O0 (BOOL) - bit 0 of IN (least significant bit of IN)
O1 (BOOL) - bit 1 of IN
O2 (BOOL) - bit 2 of IN
O3 (BOOL) - bit 3 of IN
O4 (BOOL) - bit 4 of IN
O5 (BOOL) - bit 5 of IN
O6 (BOOL) - bit 6 of IN
O7 (BOOL) - bit 7 of IN
O8 (BOOL) - bit 8 of IN
O9 (BOOL) - bit 9 of IN
O10 (BOOL) - bit 10 of IN
O11 (BOOL) - bit 11 of IN
O12 (BOOL) - bit 12 of IN
O13 (BOOL) - bit 13 of IN
O14 (BOOL) - bit 14 of IN
O14 (BOOL) - bit 15 of IN
O16 (BOOL) - bit 16 of IN
O17 (BOOL) - bit 17 of IN
O18 (BOOL) - bit 18 of IN
O19 (BOOL) - bit 19 of IN
O20 (BOOL) - bit 20 of IN
O21 (BOOL) - bit 21 of IN
O22 (BOOL) - bit 22 of IN
O23 (BOOL) - bit 23 of IN
O24 (BOOL) - bit 24 of IN
O25 (BOOL) - bit 25 of IN
O26 (BOOL) - bit 26 of IN
O27 (BOOL) - bit 27 of IN
O28 (BOOL) - bit 28 of IN
O29 (BOOL) - bit 29 of IN
O30 (BOOL) - bit 30 of IN
O31 (BOOL) - bit 31 of IN (most significant bit)

```
<<INSTANCE NAME>>:M_DW2BOO(EN00 := <<BOOL>>, IN :=
  <<DWORD>>, O0 => <<BOOL>> O1 => <<BOOL>>, O2 => <<BOOL>>, O3
  => <<BOOL>>, O4 => <<BOOL>>, O5 => <<BOOL>>, O6 => <<BOOL>>,
  O7 => <<BOOL>>, O8 => <<BOOL>>, O8 => <<BOOL>> O9 =>
  <<BOOL>>, O10 => <<BOOL>>, O11 => <<BOOL>>, O12 => <<BOOL>>,
  O13 => <<BOOL>>, O14 => <<BOOL>>, O15 => <<BOOL>>, O16 =>
  <<BOOL>>, O17 => <<BOOL>> O18 => <<BOOL>>, O19 => <<BOOL>>,
  O20 => <<BOOL>>, O21 => <<BOOL>>, O22 => <<BOOL>>, O23 =>
  <<BOOL>>, O24 => <<BOOL>>, O25 => <<BOOL>>, O26 => <<BOOL>>
  O27 => <<BOOL>>, O28 => <<BOOL>>, O29 => <<BOOL>>, O30 =>
  <<BOOL>>, O31 => <<BOOL>>);
```

This function block converts a DWORD to 32 BOOLs.

M_ERROR

Axis Error Checking



Inputs: EN01 (BOOL) - enables execution

AXIS (USINT) - identifies axis

Outputs: OK (BOOL) - execution complete

ESTO (BOOL) - indicates an E-stop is active when set

CSTO (BOOL) - indicates an C-stop is active when set

PSTO (BOOL) - indicates a programming error has occurred when set

E_ER (WORD) - identifies E-stop errors

C_ER (WORD) - identifies C-stop errors

P_ER (WORD) - identifies programming errors

<<INSTANCE NAME>>:M_ERROR(EN01 := <<BOOL>>, AXIS := <<USINT>>, OK => <<BOOL>> ESTO => <<BOOL>>, CSTO => <<BOOL>>, PSTO => <<BOOL>>, E_ER => <<WORD>>, C_ER => <<WORD>>, P_ER => <<WORD>>);

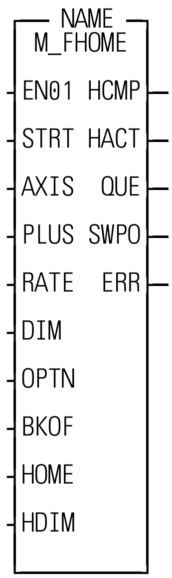
This function block is used to report **servo** E-stop, C-stop and programming error conditions in the ladder. These conditions may be caused by the servo software or defined by the programmer. If defined by the programmer they will be triggered using the E-STOP or C_STOP functions. All of these errors for the defined axis are reported in one location

The enable input of this function should be directly connected to the rail with a wire, causing this function block to be executed each scan.

The boolean outputs can be used as flags in the ladder to report error conditions.

The word outputs can be converted to a HEX display by using the Module Monitor Edit View List command and inserting the variables. An option will be given on the format to display them. The variable's value during animation will be displayed in HEX format if the variable provided has 16#0 for its initial value. The default format during animation is decimal.

After monitoring them in HEX, referring to the tables in the manual of functions E_ERRORS, C_ERRORS and P_ERRORS will help identify the exact problem.



Inputs: EN01 (BOOL) - enables execution

STRT (BOOL) - enables the home cycle

AXIS (USINT) - identifies axis

PLUS (BOOL) - indicates direction of home cycle

RATE (UDINT) - feedrate at which motion occurs (entered in LU/MIN)

DIM (DINT) - reference dimension for the nearest resolver null or the next encoder index mark when the reference switch is set (entered in LUs)

OPTN (WORD) - provides referencing options (0 or 1) 0=No option 1=Ignore index or null

BKOF (BOOL) - selects backoff of reference switch option

HOME (BOOL) - selects homing after referencing option

HDIM (DINT) - home location to move to after reference is complete

Outputs: HCMP (BOOL) - home cycle is complete

HACT (BOOL) - home cycle is being executed

QUE (USINT) - number of move for queue

SWPO (DINT) - distance in feedback units (FUs) from the reference switch to the index mark of an encoder or the null of a resolver

ERR (BYTE) - report an error 1-4 if input data is invalid

<<INSTANCE NAME>>:M_FHOME(EN01 := <<BOOL>>, STRT := <<USINT>>, AXIS := <<USINT>> PLUS := <<BOOL>>, RATE := <<UDINT>>, DIM := <<DINT>>, OPTN := <<WORD>>, BKOF := <<BOOL>>, HOME := <<BOOL>>, HDIM := <<DINT>>, HCMP => <<BOOL>>, HACT => <<BOOL>>, QUE => <<USINT>>, SWPO => <<DINT>>, ERR => <<BYTE>>,);

This function block performs a fast reference cycle on an axis, followed by a homing (position) move to a designated location.

Before this function can be used, the axis must be initialized and the position loop closed.

The reference cycle will cause the selected axis to move in the designated direction until the reference switch is sensed. In a fast reference this reference switch is wired to the fast input of the selected axis on the feedback module in the PiC900. When the fast input occurs, the position of the axis is latched by the hardware on the encoder module independent of the ladder scan. When the reference switch is sensed the axis will reference (assign a value) to the next index mark of an encoder or the nearest null of a resolver. After the value is assigned, the axis will decelerate to a stop and set the reference done flag.

If the HOME input is on when the reference done has been sensed, the home move will automatically be triggered to position the axis at a desired location.

If the BKOF input is on when the reference is requested, and the axis is on the reference switch, the axis will move in the opposite direction until the reference switch opens and will then move back onto the reference switch.   If the BKOF input is not on the axis will move in the specified direction until it sees an off to on transition of the limit switch.

This function block is used to perform a fast reference, immediately followed by a position move to a selected home position. It should be executed every scan unless a home cycle will only be performed when the machine is started. In that case a normally closed contact of the output of HCMP may be used.

The inputs to this function block are basically the same as for the FAST_REF function. There are three additional inputs listed below.

The BKOF input selects the backoff reference switch option.

The HOME input selects the homing after referencing option.

The HDIM input assigns the home dimension to move to.

If the axis is sitting on the limit switch when the home cycle is requested, and the BKOF input is on, the axis will move in the opposite direction of that indicated by the PLUS input until the switch opens and then will complete the home cycle in the normal manner.

The SWPO output is used to determine if the reference switch location will allow for repeatable referencing. If the reference switch is not properly located in relationship to the index marker of an encoder or the null of a resolver it could possibly reference a revolution off. To prevent this, the value reported by this output should be as follows:

- For an encoder system the value of this output should be greater than 25% and less than 75% of the total counts (FUs) per revolution. Example: For 8000 FUs/Rev, the value should be >2000 and <6000.
- For a resolver system the value of this output should be less than 25% or greater than 75% of the total counts (FUs) per revolution. Example: For 4000 FUs/Rev, the value should be <1000 or >3000.

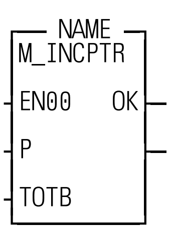
If the value is out of range either the reference switch will have to be moved or the transducer coupling shifted.

The ERR output indicates that invalid data was entered on one of the inputs. The possible errors are listed below:

ERR	Description
0	No error
1	The queue was not empty when the reference was requested
2	An error occurred in backing off of the reference switch
3	An error occurred in referencing
4	An error occurred in homing

M_INCPTR

Increment buffer pointers **USER/M_DATA**



Inputs: EN00 (BOOL) - enable

P (STRUCT) - pointer for data buffer

TOTB (UINT) - total bytes to increment

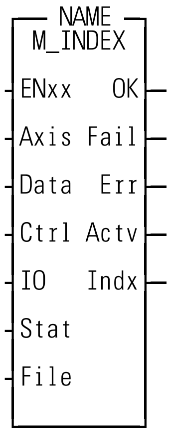
Outputs: OK (BOOL) - increment of pointers ok

This function block increments the buffer pointers for M_DATCPT.

<<INSTANCE NAME>>:M_INCPTR (EN00 := <<BOOL>>, P := <<MEMORY AREA>>, TOTB := <<UINT>> OK => <<BOOL>>);

M_INDEX

Single Axis Indexer / Motion Sequencer **USER/M_INDEX**



Inputs: ENxx (BOOL) - Enables execution, should be enabled all the time. (xx indicates revision #)

Axis (USINT) - Axis Number.

Data (STRUCT) - Data structure that contains the indexer sequence parameters defined below.

Ctrl (STRUCT) - Used to control (start, stop, etc.) the index sequence.

IO (STRUCT) - Used to pass the state of the index inputs and outputs between the main ladder and the ASFB.

Stat (STRUCT) - Status information, faults, active flags, etc.

File (STRUCT) - Used to read/write the index Data from/to RAMDISK.

Outputs: OK (BOOL) - Indicates that the ASFB is enabled and was initialized correctly.

Fail (BOOL) - Indicates that the ASFB failed to initialize correctly.

Err (UINT) - Error code.

Actv (BOOL) - Index Sequence is Active.

Indx (UINT) - Number of active index step.

<<INSTANCE NAME>>:M_INDEX(ENxx := <<BOOL>>, Axis := <<USINT>>, Data := <<MEMORY AREA>>, Ctrl := <<MEMORY AREA>>, IO := <<MEMORY AREA>>, Stat := <<MEMORY AREA>>, File := <<MEMORY AREA>>, OK => <<BOOL>>, Fail => <<BOOL>>, Err => <<UINT>>, Actv => <<BOOL>>, Indx => <<UINT>>);

The M_INDEX ASFB is a single axis motion indexer or sequencer. It can be used to perform the same functions as a standalone positioning type drive. This ASFB executes up to 300 different indexes or program steps. The index can be a used to move an axis, turn on an output, wait for an input, or to pass data to the main ladder for user-defined functions. Indexes can be stringed together to create a motion/IO sequence. This section describes these functions along with the inputs and outputs for the M_INDEX ASFB. There are several different types of commands that can be executed in each index of the program.

Distance Moves:

1. Incremental
2. Registration

Position Moves:

3. Absolute
4. Rotary shortest distance
5. Rotary Forward
6. Rotary Backward

Velocity/Jog Moves:

7. Velocity Forward
8. Velocity Backward
9. Velocity Stop

Motion Support:

10. Set Rollover Position
11. Part Reference
12. Set Ignore Distance for Registration move
13. Set Maximum Number of Missed Registration Marks
14. Set S-curve Acceleration/Deceleration Rates
15. Set Linear Acceleration/Deceleration Rates
16. Wait for Position Greater Than or Equal To
17. Wait for Position Less Than or Equal To
18. Wait for In-Position
19. New Rate

I/O Control:

20. Turn On
21. Turn Off Output
22. Wait for Input On
23. Wait for Input Off

Repeat/Cycle Counters:

24. Increment the Cycle Counter
25. Set the Loop Counter
26. Increment the Loop Counter

User Defined Commands:

27. User Command 1
28. User Command 2

Misc. Commands:

29. NOP. No operation
30. End of Program

Error Codes

The table below describes the different errors that may occur.

Error Number	Description
1-9	**General Index Data Errors.**
1	Invalid Range for Index Complete Mode. Range of values is 0 - 3.
2	Increment Cycle Count failed, cycle count may have exceeded 65,535.
3	Invalid Loop Counter. Range of values is 1 - 2.
4	Loop Count exceeded 65,535.
5	Invalid Index Number. Range of values is 0 - 299.
6	Invalid Index Type, check IndexType string variable.
7-9	Reserved.
8	Invalid Index Number. Range of values is 0 – 299.
9	Invalid Index Type, check IndexType string variable.
10-19	**Motion Support Command Errors.**

10	Set S-Curve Acc/Dec failed. Check parameters and make sure that the “Enable S-Curve” box was checked in the servo setup.
11	Set Linear Acc/Dec failed, check parameters.
12	New Rate failed. Either a move was not active or the Rate was invalid.
13	Set Rollover failed, check Data_1 variable.
14	Invalid Range Index Complete Mode, “Start Next Index Before Move Is Done” was selected for a non-motion command.
15	Part Reference failed. Axis may have been in motion when the index was executed.
16	Set Ignore Distance command failed.
17	Invalid number of missed marks.
18-19	Reserved.
20-29	**Distance Command Errors.**
20	Distance command was called when another move was active.
21	Distance command failed, check parameters.
22-29	Reserved.
30-39	**Registration Command Errors.**
30-31	Registration command was called when another move was active.
32	Distance move failed, check Data_1 variable or WriteSV to enable Fast Queuing failed.
33	A move was queued up outside of the M_INDEX ASFB. No motion commands should be executed in the main ladder when an index is active or started.
34	FAST_QUE function or DISTANCE for RegDist failed.
35	A move was queued up outside of the M_INDEX ASFB. No motion commands should be executed in the main ladder when an index is active or started.

36	The fast input for the Registration Sensor never went off.
37-38	Reserved.
39	The maximum number of consecutive missed registration was reached.
40-49	**Position Command Errors.**
40	Position command was called when another move was active.
41	Calculation for Rotary move failed, check rollover value and Data_1 variable.
42	Position commanded failed, check Data_1 and Rate variables.
43-49	Reserved.
50-59	**Velocity Command Errors.**
50	Velocity command was called when another move was active.
51	VEL_STRT commanded failed, check Rate variable.
52	VEL_END command failed, move may have been aborted in the main ladder.
53	Velocity Stop command was called when there was not an active Velocity move.
60-69	**IO Command Errors.**
60	Invalid Output Number, valid range 1-8.
61	Invalid Input Number, valid range 1-8.
62-69	Reserved.
101-109	**General Initialization Errors.**
101	The structure size of the Data structure is not valid. Check to make sure that the structure matches the description in the manual.
102	The structure size of the Ctrl structure is not valid. Check to make sure that the structure matches the description in the manual.

103	The structure size of the IO structure is not valid. Check to make sure that the structure matches the description in the manual.
104	The structure size of the Stat structure is not valid. Check to make sure that the structure matches the description in the manual.
105	The structure size of the File structure is not valid. Check to make sure that the structure matches the description in the manual.
106	SizeOf function failed when calculating the size of the Data and/or IO Structures, contact factory.
107-109	Reserved.
110	**File Read/Write Error.** An error has occurred while trying to read or writ ethe Indexing Program from or to the RAMDISK. Refer to the File.FileError and the File.Error variables and the error codes below for a more detailed error description.
File.FileError <> 0	**File Operation Failed. An error occurred with one of the file operations, i.e. OPEN, READ, WRITE, etc. Refer to the "PiCPro Software Manual Appendix B - Errors - I/O Function Block Error Codes" for a description of the error.**
File.Error <> 0	**File Read Errors. These errors occurred while reading and converting the text file. Check to make sure the text format matches the Excel Spread Sheet. The Error numbers below will be shown in the File.Error variable.**
1	The structure size of the Data structure is not valid. Check to make sure that the structure matches the description in the manual.
2	SizeOf function failed when calculating the size of the Data Structure, contact factory.
3	Invalid File Name.
4-9	Reserved.
10	Line too long, a line in the text file exceeded 255 characters.

11	Missing StartOfProgram in the file.
12	Missing EndOfData in the file.
13-19	Reserved.
20	Number of lines in the text file exceeded 32,767.
21	A line in the text file contained more than 7 data fields.
22	Invalid format in the line of text.
23-29	Reserved.
30	Index Number field is not valid, valid range is 0-299.
31	Index Type field is not valid.
32	Data_1 field is not valid.
33	Data_2 field is not valid.
34	Data_3 field is not valid.
35	Dwell field is not valid.
36	Next Index Number field is not valid, valid range is 0-299.
37	Index Complete Mode field is not valid.
38	Invalid format in the line of text.
39-40	Reserved.
50-59	**File Write Errors. These errors occurred while trying to write the file, check the Data Structure format in your ladder and make sure it matches the format outlined below.**
50	Invalid Data.Comments[] string, check Data Structure format to make sure that the comment string size is 40.
51	Invalid comment header, check Data Structure format.
52	Unable to convert Data structure to a text string, check Data Structure format.
53	Invalid EndOfData string when writing the file, consult factory.

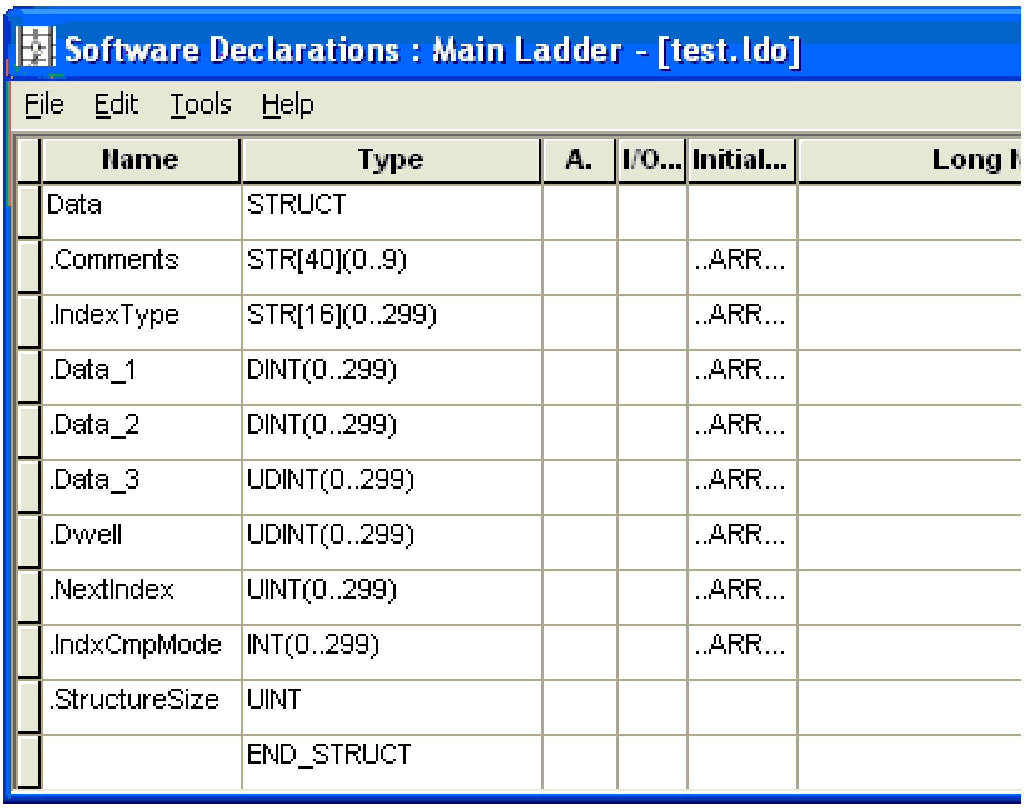
54-59	Reserved.

Inputs/Outputs

This section describes the inputs and outputs for the M_INDEX ASFB. The structures are shown as inputs but in some cases the structure elements are written by the ASFB.

Axis USINT Axis Number.

Data Structure


Software Declarations : Main Ladder - [test.ldo]

File Edit Tools Help

Name	Type	A.	I/O...	Initial...	Long I
Data	STRUCT				
.Comments	STR[40](0..9)			..ARR...	
.IndexType	STR[16](0..299)			..ARR...	
.Data_1	DINT(0..299)			..ARR...	
.Data_2	DINT(0..299)			..ARR...	
.Data_3	UDINT(0..299)			..ARR...	
.Dwell	UDINT(0..299)			..ARR...	
.NextIndex	UINT(0..299)			..ARR...	
.IndxCmpMode	INT(0..299)			..ARR...	
.StructureSize	UINT				
	END_STRUCT				

The Data structure contains the index data or part program. The following is a description of the structure elements. You can have up to 300 different indexes or program steps.

Data.Comments STR[40](0..9) You can have up to 10 lines of program comments. Each line can have up to 40 charaters.

Data.IndexType STR[16](0..299) Type of Index or Program Step, there are 30 different types. See the list above or detailed descriptions below.

Data.Data_1 DINT(0..299) Data for index, description changes with the type of index.

Data.Data_2 DINT(0..299) Data for index, description changes with the type of index.

Data.Data_3 UDINT(0..299) Data for index, description changes with the type of index.

Data.Dwell UDINT(0..299) The amount of time between the end of the move or index and the start of the index or step. The time is entered in milliseconds and has a range of 0 to 4,294,967,295 ms.

Data.NextIndex UINT(0..299) The number of the next index or step to execute when Index Complete Mode is not set to Stop. The value can be between 0 and 299.

Data.IndxCmpMode INT(0..299) Action taken when the index or step is complete. The following are valid values:

0. Stop.
1. Start next index when move is complete and after Dwell.
2. Start next index when Start Index Input is on.
3. Start next index before move is complete.
4. Optional Stop, stops at the end of the index if the Optional Stop input is on.

NOTE
The Data_1, Data_2 and Data_3 elements have different descriptions depending upon the IndexType

Control Structure

Software Declarations : Main Ladder - [test.ldo]

File Edit Tools Help

Name	Type	A.	I/O...	Initial...	Long Name
Ctrl	STRUCT				
.Start	BOOL				Start\Index\\
.IndxSel	UINT				Index\Select\\
.Hold	BOOL				
.Stop	BOOL				
.OptStop	BOOL				
.SingleStep	BOOL				
.FeedRateOv...	UINT				
.StatusSV	WORD				Status Sv\output\\
.StructureSize	UINT				
	END_STRUCT				

The Ctrl Structure is used to control the indexing sequence or part program. All of the elements are inputs to the ASFB. The following is a description of each element.

Ctrl.Start BOOL Starts the index selected by the Ctrl.IndxSel if a sequence is not active. If a sequence is active, it continues or starts the next index step.

Ctrl.IndxSel UINT Number of the index to start executing when that Ctrl.Start input turns on and a sequence is not active.

Ctrl.Hold BOOL When the input turns on, it pauses or holds the execution of the index and issues a feed hold for the axis until the input goes off.

Ctrl.Stop BOOL Aborts any active moves and resets/stops the index sequence.

Ctrl.OptStop BOOL When this input is on and the Index Complete Mode is set to Optional Stop the index sequence will stop at the end of the index and wait for a Start input. The Optional Stop can be used to set break points in the program for debug or to have the operator check the part periodically.

Ctrl.SingleStep BOOL Forces the index sequence to stop at the end of each index or step. The next index will be executed when the Ctrl.Start input goes on. This feature allows the user to step though the program one index at a time.

Ctrl.FeedRateOveride UINT Feed rate override percentage, percent to increase or decrease feed rate for all moves for the specified axis. The range is from 0 to 199% with 100% being the feed rate entered at RATE for distance, position and velocity moves. NOTE: If 200 to 255% is entered, the software handles it as if 199 was entered.

Ctrl.StatusSv WORD STAT output from the STATUSSV function. The ASFB uses the Move Started and Fast Input Occurred status bits. These bits are "read and clear" (one shot) bits. A set bit means the event occurred since the last time the function was called. Therefore, the function can be called only once in the ladder to prevent missing the event.

Ctrl.StructureSize UINT Must be set to the size or number of bytes of the Ctrl Structure. The SIZEOF function can be used to calculate the size of the structure.

IO Structure

Software Declarations : Main Ladder - [test.ldo]

File Edit Tools Help

Name	Type	A.	I/O...	Initial...	Long Name
IO	STRUCT				
.Output	BOOL(0..8)				
.Input	BOOL(0..8)				
.StructureSize	UINT				
	END_STRUCT				

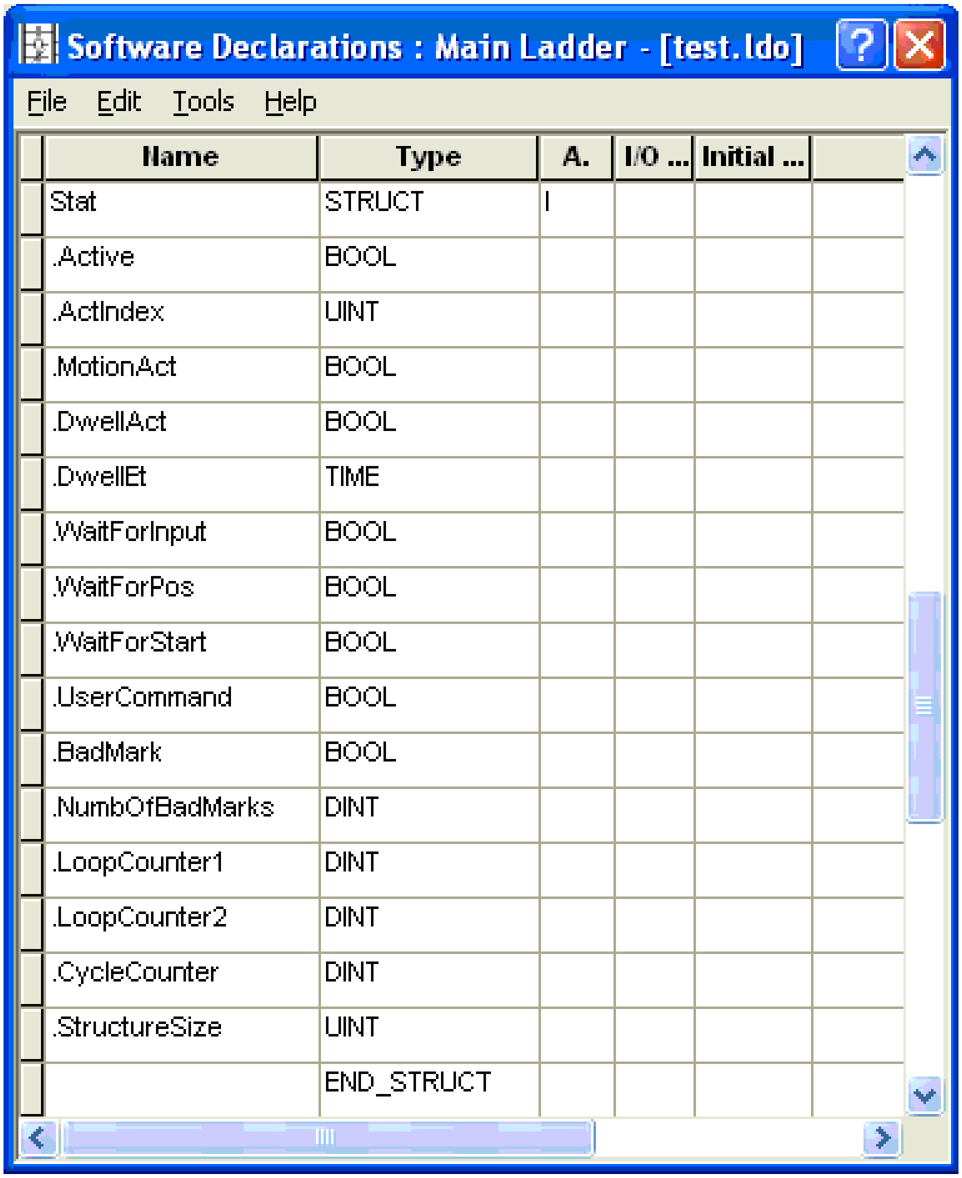
The IO Structure is used to interface between the main ladder and the Turn On/Off and Wait for Input On/Off commands in the indexing program. The outputs are written by the program and the inputs are read. The zero [0] elements are not used in the program. The following is a description of each of elements.

IO.Output BOOL(0..8) The Turn On/Off commands are used to control the 8 user defined outputs, which can be used in the main ladder to control physical outputs or drive other ladder logic.

IO.Input BOOL(0..8) The 8 user defined inputs are used by Wait for Input On/Off commands. These variables are typically turned on/off in the main ladder by energize or set/reset coils. They can be connected to physical input variables or driven by other ladder logic.

IO.StructureSize UINT Must be set to the size or number of bytes of the IO Structure. The SIZEOF function can be used to calculate the size of the structure.

Stat Structure



Name	Type	A.	I/O ...	Initial ...
Stat	STRUCT	I		
.Active	BOOL			
.ActIndex	UINT			
.MotionAct	BOOL			
.DwellAct	BOOL			
.DwellEt	TIME			
.WaitForInput	BOOL			
.WaitForPos	BOOL			
.WaitForStart	BOOL			
.UserCommand	BOOL			
.BadMark	BOOL			
.NumbOfBadMarks	DINT			
.LoopCounter1	DINT			
.LoopCounter2	DINT			
.CycleCounter	DINT			
.StructureSize	UINT			
	END_STRUCT			

The Stat Structure is used to pass status information to the main ladder. The following is a description of each element.

Stat.Active BOOL Indicates that there is an active part program or index.

Stat.ActIndex UINT Number of the current or active index step.

Stat.MotionAct BOOL Indicates that the axis is currently in motion or not in-position.

Stat.DwellAct BOOL Indicates that the dwell is active for the current index step.

Stat.DwellEt TIME Elapsed time for the dwell timer.

Stat.WaitForInput BOOL Indicates that the active index step is either a WaitForInputOn or WaitForInputOff command, which is waiting for the user defined input to change states.

Stat.WaitForPos BOOL Indicates that the active index step is either a WaitForPosnGE or WaitForPosnLE command, which is waiting for the axis position to reach the programmed position.

Stat.WaitForStart BOOL Indicates that the active index step has completed it command and is waiting for the Ctrl.Start input before continuing on to the next index.

Stat.UserCommand BOOL Indicates that the active index step is one of the two user commands, UserCommand1 or UserCommand2.

Stat.BadMark BOOL Indicates that the last Registration command or index did not see a registration mark.

Stat.NumbOfBadMarks DINT Number of consecutive registration marks missed, this variable is reset to zero when we get a good registration mark.

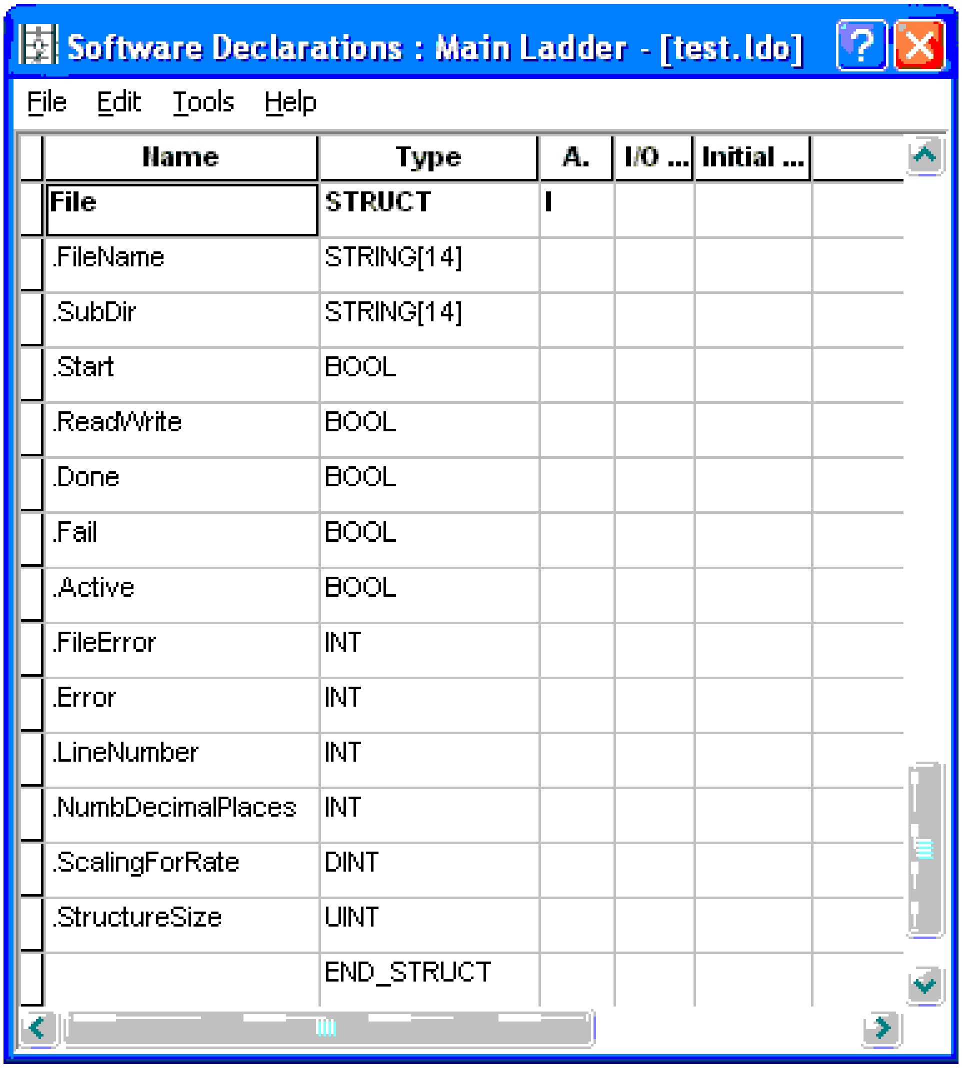
Stat.LoopCounter1 DINT Value of the program Loop Counter 1, which is controlled in the part program by the SetLoopCount and IncLoopCount commands.

Stat.LoopCounter2 DINT Value of the program Loop Counter 2, which is controlled in the part program by the SetLoopCount and IncLoopCount commands.

Stat.CycleCounter DINT Value of the program Cycle Counter, which is controlled in the part program by the IncCycleCount command.

Stat.StructureSize UINT Must be set to the size or number of bytes of the Stat Structure. The SIZEOF function can be used to calculate the size of the structure.

File Structure



Software Declarations : Main Ladder - [test.ldo]

File Edit Tools Help

Name	Type	A.	I/O ...	Initial ...
File	STRUCT	I		
.FileName	STRING[14]			
.SubDir	STRING[14]			
.Start	BOOL			
.ReadWrite	BOOL			
.Done	BOOL			
.Fail	BOOL			
.Active	BOOL			
.FileError	INT			
.Error	INT			
.LineNumber	INT			
.NumbDecimalPlaces	INT			
.ScalingForRate	DINT			
.StructureSize	UINT			
	END_STRUCT			

The File structure is used to control and provide status information for reading and writing the data structure from/to the RAMDISK. The following is a description of each element.

File.FileName STRING[14] Name of the file to be read or written from the RAMDISK. The file name format is "FileName.Ext$00, where the file name can be up to 8 characters and the extension up to 3 characters. The $00 is a null terminator.

File.SubDir STRING[14] Name of the Sub-Directory on the RAMDISK were the file is stored. If it is left blank the file will be stored in the root directory. The subdirectory name format is "SubDir", where the SubDir can be up to 8 characters.

File.Start BOOL Starts the read or write operation.

File.ReadWrite BOOL Off = File Read, On = File Write.

File.Done BOOL Indicates that the file operation is complete.

File.Fail BOOL Indicates that the file operation failed to complete.

File.Active BOOL Indicates that the file operation is active.

File.FileError INT File access Error Number. An error occurred with one of the file operations, i.e. OPEN, READ, WRITE, etc. Refer to the "PiCPro Software Manual Appendix B -Errors - I/O Function Block Error Codes" for a description of the error.

File.Error INT File File Data Format Error Number. An error occurred while converting the text file for the Data Structure. See the File Error Codes above for a detailed description of the error.

File.LineNumber INT Line number in the text file where the File.Error occurred.

File.NumbDecimalPlaces INT The number of decimal places in the file for the axis position/distant data.

File ScalingForRate DINT Scaling factor used to convert the Rate in the File to LU/Min. This would allow you to enter the Rate as RPM vs LU/Min, in this case you would put in the number of LU per Rev. The number of decimal places is also applied. Thus if 1 LU = .001 degrees or 360,000 LU/Revolution, you would set the scaling to 360 or 360.000 degrees per Revolution.

File.StructureSize UINT Must be set to the size or number of bytes of the File Structure. The SIZEOF function can be used to calculate the size of the structure.

Programming Indexes

The following sections describe how to program each type of index.

ABSOLUTE

Moves the axis to an absolute position or endpoint. The axis acceleration and deceleration ramps are defined by the SetScurveAccDec or SetLinearAccDec commands. This command uses the POSITION function to execute the move, reference the PiCPro Function/Function Block Reference Guide for details on the POSITION function.

Data Variable	Description
.IndexType[x]	Absolute
.Data_1[x]	Absolute position to go to. The value is entered in LU.
.Data_2[x]	Not Used.
.Data_3[x]	Maximum velocity. The value is entered in LU/Min.
.Dwell[x]	The amount of time between the end of the move or index and the start of the index or step. The time is entered in milliseconds and has a range of 0 to 4,294,967,295 ms.
.Next Index[x]	The number of the next index or step to execute when Index Complete Mode is not set to Stop. The value can be between 0 and 299.
.Index Complete Mode[x]	Action taken when the index or step is complete. The following are valid values: 0. Stop. 1. Start next index when move is complete and after Dwell. 2. Start next index when Start Index Input is on. 3. Start next index before move is complete. 4. Optional Stop, stops at the end of the index if the Optional Stop input is on.

INCREMENTAL

Moves the axis an incremental distance. The axis acceleration and deceleration ramps are defined by the SetScurveAccDec or SetLinearAccDec commands. This command uses the DISTANCE function to execute the move, reference the PiCPro Function/Function Block Reference Guide for details on the DISTANCE function.

Data Variable	Description
.IndexType[x]	Incremental
.Data_1[x]	Incremental Distance to go, the value is entered in LU.
.Data_2[x]	Not Used.
.Data_3[x]	Maximum velocity, the value is entered in LU/Min.
.Dwell[x]	The amount of time between the end of the move or index and the start of the index or step. The time is entered in milliseconds and has a range of 0 to 4,294,967,295 ms.
.Next Index[x]	The number of the next index or step to execute when Index Complete Mode is not set to Stop. The value ranges from 0 to 299.
.Index Complete Mode[x]	Action taken when the index or step is complete. The following are valid values: 0. Stop. 1. Start next index when move is complete and after Dwell. 2. Start next index when Start Index Input is on. 3. Start next index before move is complete. 4. Optional Stop, stops at the end of the index if the Optional Stop input is on.

INCREMENT CYCLE COUNTER

At the start of an index sequence, the cycle counter is reset. This command increment's the cycle counter by 1. If the value is equal to the (.Data_1) variable, the index sequence stops. Otherwise, the next index is started based on the Index Complete Mode.

Data Variable	Description
.IndexType[x]	IncCycleCount
.Data_1[x]	Number of Times to repeat the cycle. The value can be between 0 and 65,535.
.Data_2[x]	Not Used.
.Data_3[x]	Not Used.
.Dwell[x]	The amount of time between the end of the move or index and the start of the index or step. The time is entered in milliseconds and has a range of 0 to 4,294,967,295 ms.
.Next Index[x]	The number of the next index or step to execute when Index Complete Mode is not set to Stop. The value can be between 0 and 299.
.Index Complete Mode[x]	Action taken when the index or step is complete. The following are valid values: 0. Stop. 1. Start next index when move is complete and after Dwell. 2. Start next index when Start Index Input is on. 3. Start next index before move is complete. 4. Optional Stop, stops at the end of the index if the Optional Stop input is on.

INCREMENT LOOP COUNTER

There are two counters that can be used for looping in the index sequence or program. This command is used to increment the value of one of the Loop Counters by 1. If the new value is less that the loop count value then the Loop To Index will be the next index, otherwise the Next Index will be the next index. The SetLoop-Count command can be used to set the counter.

Data Variable	Description
.IndexType[x]	IncLoopCount
.Data_1[x]	Loop Counter Number, range is 1 or 2.
.Data_2[x]	Number of times to loop, range is 0 to 2,147,483,647
.Data_3[x]	Loop to this Index Number if the loop counter is less than the number of times to loop.
.Dwell[x]	The amount of time between the end of the move or index and the start of the index or step. The time is entered in milliseconds and has a range of 0 to 4,294,967,295 ms.
.Next Index[x]	The number of the next index or step to execute when Index Complete Mode is not set to Stop. The value can be between 0 and 299.
.Index Complete Mode[x]	Action taken when the index or step is complete. The following are valid values: 0. Stop. 1. Start next index when move is complete and after Dwell. 2. Start next index when Start Index Input is on. 3. Start next index before move is complete. 4. Optional Stop, stops at the end of the index if the Optional Stop input is on.

NEW RATE

Allows you to change the rate or velocity of the active move. This command uses the NEW_RATE function to change the rate, reference the PiCPro Function/Function Block Reference Guide for details on the NEW_RATE function.

Data Variable	Description
.IndexType[x]	NewRate
.Data_1[x]	Not Used.
.Data_2[x]	Not Used.
.Data_3[x]	New velocity, the value is entered in LU/Min.
.Dwell[x]	The amount of time between the end of the move or index and the start of the index or step. The time is entered in milliseconds and has a range of 0 to 4,294,967,295 ms.
.Next Index[x]	The number of the next index or step to execute when Index Complete Mode is not set to Stop. The value can be between 0 and 299.
.Index Complete Mode[x]	Action taken when the index or step is complete. The following are valid values: 0. Stop. 1. Start next index when move is complete and after Dwell. 2. Start next index when Start Index Input is on. 3. Start next index before move is complete. 4. Optional Stop, stops at the end of the index if the Optional Stop input is on.

NOP

This command performs No OPeration, it can be used as a blank program step for future use.

Data Variable	Description
.IndexType[x]	NOP
.Data_1[x]	Not Used.
.Data_2[x]	Not Used.
.Data_3[x]	Not Used.
.Dwell[x]	The amount of time between the end of the move or index and the start of the index or step. The time is entered in milliseconds and has a range of 0 to 4,294,967,295 ms.
.Next Index[x]	The number of the next index or step to execute when Index Complete Mode is not set to Stop. The value can be between 0 and 299.
.Index Complete Mode[x]	Action taken when the index or step is complete. The following are valid values: 0. Stop. 1. Start next index when move is complete and after Dwell. 2. Start next index when Start Index Input is on. 3. Start next index before move is complete. 4. Optional Stop, stops at the end of the index if the Optional Stop input is on.

PART REF

This command allows you to change the current position of an axis. No motion occurs when a part reference is performed. The axis must be at rest or not in motion when calling this command. This command uses the PART_REF function to execute the move, reference the PiCPro Function/Function Block Reference Guide for details on the PART_REF function.

Data Variable	Description
.IndexType[x]	PartRef
.Data_1[x]	Reference dimension. The value is entered in LU.
.Data_2[x]	Not Used.
.Data_3[x]	Not Used.
.Dwell[x]	The amount of time between the end of the move or index and the start of the index or step. The time is entered in milliseconds and has a range of 0 to 4,294,967,295 ms.
.Next Index[x]	The number of the next index or step to execute when Index Complete Mode is not set to Stop. The value can be between 0 and 299.
.Index Complete Mode[x]	Action taken when the index or step is complete. The following are valid values: 0. Stop. 1. Start next index when move is complete and after Dwell. 2. Start next index when Start Index Input is on. 3. Start next index before move is complete. 4. Optional Stop, stops at the end of the index if the Optional Stop input is on.

REGISTRATION

This command is used to perform a registration move. It allows you to program the axis to travel an incremental distance past a registration mark. A sensor must be wired to the fast or registration input for the axis to detect the registration mark. This command uses the M_REGMOV function to execute the move, reference the Motion ASFB Manual for details on the M_REGMOV function. If a registration mark is not detected, the axis goes the distance programmed in the Maximum Incremental Distance variable and stops. The BMRK (Bad Mark) output goes on and NMBD (Number of Bad Marks) counter is incremented. If the number of consecutively missed registration marks equals the SetMaxMarkMissed variable, a fault is generated and the indexing sequence is stopped. Otherwise, the index continues as programmed. When a registration mark is detected, the NMBD (Number of Bad Marks) counter is reset. The BMRK output is reset at the start of the next index.

Data Variable	Description
.IndexType[x]	Registration
.Data_1[x]	Maximum Incremental Distance to go. The value is entered in LU.
.Data_2[x]	Distance to go past the registration mark. The value is entered in LU.
.Data_3[x]	Maximum velocity. The value is entered in LU/ Min.
.Dwell[x]	The amount of time between the end of the move or index and the start of the index or step. The time is entered in milliseconds and has a range of 0 to 4,294,967,295 ms.
.Next Index[x]	The number of the next index or step to execute when Index Complete Mode is not set to Stop. The value can be between 0 and 299.
.Index Complete Mode[x]	Action taken when the index or step is complete. The following are valid values: 0. Stop. 1. Start next index when move is complete and after Dwell. 2. Start next index when Start Index Input is on. 3. Start next index before move is complete. 4. Optional Stop, stops at the end of the index if the Optional Stop input is on.

ROTARY BACKWARD

Moves the axis backward to an absolute position or endpoint. The axis rollover position must be set using the SetRollover command, otherwise a fault is generated. The axis acceleration and deceleration ramps are defined by the SetScurveAccDec or SetLinearAccDec commands. This command uses the POSITION function to execute the move, reference the PiCPro Function/Function Block Reference Guide for details on the POSITION function.

Data Variable	Description
.IndexType[x]	RotaryBackward
.Data_1[x]	Absolute position to go to. The value is entered in LU.
.Data_2[x]	Not Used.
.Data_3[x]	Maximum velocity. The value is entered in LU/Min.
.Dwell[x]	The amount of time between the end of the move or index and the start of the index or step. The time is entered in milliseconds and has a range of 0 to 4,294,967,295 ms.
.Next Index[x]	The number of the next index or step to execute when Index Complete Mode is not set to Stop. The value can be between 0 and 299.
.Index Complete Mode[x]	Action taken when the index or step is complete. The following are valid values: 0. Stop. 1. Start next index when move is complete and after Dwell. 2. Start next index when Start Index Input is on. 3. Start next index before move is complete. 4. Optional Stop, stops at the end of the index if the Optional Stop input is on.

ROTARY FORWARD

Moves the axis forward to an absolute position or endpoint. The axis rollover position must be set using the SetRollover command, otherwise a fault is generated. The axis acceleration and deceleration ramps are defined by the SetScurveAccDec or SetLinearAccDec commands. This command uses the POSITION function to execute the move, reference the PiCPro Function/Function Block Reference Guide for details on the POSITION function.

Data Variable	Description
.IndexType[x]	RotaryForward
.Data_1[x]	Absolute position to go to. The value is entered in LU.
.Data_2[x]	Not Used.
.Data_3[x]	Maximum velocity. The value is entered in LU/Min.
.Dwell[x]	The amount of time between the end of the move or index and the start of the index or step. The time is entered in milliseconds and has a range of 0 to 4,294,967,295 ms.
.Next Index[x]	The number of the next index or step to execute when Index Complete Mode is not set to Stop. The value can be between 0 and 299.
.Index Complete Mode[x]	Action taken when the index or step is complete. The following are valid values: 0. Stop. 1. Start next index when move is complete and after Dwell. 2. Start next index when Start Index Input is on. 3. Start next index before move is complete. 4. Optional Stop, stops at the end of the index if the Optional Stop input is on.

ROTARY SHORTEST

Moves the axis forward or backward to an absolute position or endpoint. The direction is based upon the shortest distance from the current actual position to the programmed position. The axis rollover position must be set using the SetRollover command, otherwise a fault is generated. The axis acceleration and deceleration ramps are defined by the SetScurveAccDec or SetLinearAccDec commands. This command uses the POSITION function to execute the move, reference the PiCPro Function/Function Block Reference Guide for details on the POSITION function.

Data Variable	Description
.IndexType[x]	RotaryShortest
.Data_1[x]	Absolute position to go to. The value is entered in LU.
.Data_2[x]	Not Used.
.Data_3[x]	Maximum velocity. The value is entered in LU/Min.
.Dwell[x]	The amount of time between the end of the move or index and the start of the index or step. The time is entered in milliseconds and has a range of 0 to 4,294,967,295 ms.
.Next Index[x]	The number of the next index or step to execute when Index Complete Mode is not set to Stop. The value can be between 0 and 299.
.Index Complete Mode[x]	Action taken when the index or step is complete. The following are valid values: 0. Stop. 1. Start next index when move is complete and after Dwell. 2. Start next index when Start Index Input is on. 3. Start next index before move is complete. 4. Optional Stop, stops at the end of the index if the Optional Stop input is on.

SET IGNORE DISTANCE

Sets the ignore distance for the Registration command. The Registration command will ignore any fast inputs or registration marks that fall within this distance from the start of the move. This command sets the Ignr input of the M_REGMOV ASFB, reference the Motion ASFB Manual for details on the M_REGMOV function.

Data Variable	Description
.IndexType[x]	SetIgnoreDist
.Data_1[x]	Distance in LU from the start of the move that the Registration command will ignore any fast inputs.
.Data_2[x]	Not Used.
.Data_3[x]	Not Used.
.Dwell[x]	The amount of time between the end of the move or index and the start of the index or step. The time is entered in milliseconds and has a range of 0 to 4,294,967,295 ms.
.Next Index[x]	The number of the next index or step to execute when Index Complete Mode is not set to Stop. The value can be between 0 and 299.
.Index Complete Mode[x]	Action taken when the index or step is complete. The following are valid values: 0. Stop. 1. Start next index when move is complete and after Dwell. 2. Start next index when Start Index Input is on. 3. Start next index before move is complete. 4. Optional Stop, stops at the end of the index if the Optional Stop input is on.

SET LINEAR ACCELERATION/DECELERATION

Changes the linear acceleration/deceleration rate for the motion commands. This command uses the ACC_DEC function to set the rate, reference the PiCPro Function/Function Block Reference Guide for details on the ACC_DEC function.

Data Variable	Description
.IndexType[x]	SetLinearAccDec
.Data_1[x]	Total time for the axis to reach the Maximum Velocity Rate. The value is entered in ms.
.Data_2[x]	Total time for the axis to decelerate to a stop from the Maximum Velocity Rate. The value is entered in ms.
.Data_3[x]	Maximum velocity, the value is entered in LU/Min.
.Dwell[x]	The amount of time between the end of the move or index and the start of the index or step. The time is entered in milliseconds and has a range of 0 to 4,294,967,295 ms.
.Next Index[x]	The number of the next index or step to execute when Index Complete Mode is not set to Stop. The value can be between 0 and 299.
.Index Complete Mode[x]	Action taken when the index or step is complete. The following are valid values: 0. Stop. 1. Start next index when move is complete and after Dwell. 2. Start next index when Start Index Input is on. 3. Start next index before move is complete. 4. Optional Stop, stops at the end of the index if the Optional Stop input is on.

SET LOOP COUNTER

There are two counters that can be used for looping in the index sequence or program. This command is used to set the value of one of the Loop Counters. The IncLoopCount command is used to loop to the program.

Data Variable	Description
.IndexType[x]	SetLoopCount
.Data_1[x]	Loop Counter Number, range is 1 or 2.
.Data_2[x]	Value to set loop counter to, range is 0 to 2,147,483,647.
.Data_3[x]	Not Used.
.Dwell[x]	The amount of time between the end of the move or index and the start of the index or step. The time is entered in milliseconds and has a range of 0 to 4,294,967,295 ms.
.Next Index[x]	The number of the next index or step to execute when Index Complete Mode is not set to Stop. The value can be between 0 and 299.
.Index Complete Mode[x]	Action taken when the index or step is complete. The following are valid values: 0. Stop. 1. Start next index when move is complete and after Dwell. 2. Start next index when Start Index Input is on. 3. Start next index before move is complete. 4. Optional Stop, stops at the end of the index if the Optional Stop input is on.

SET MAXIMUM NUMBER OF MISSED REGISTRATION MARKS

The Registration command checks for a missing registration mark. If the number of consecutively missed registration marks equals the SetMaxMarkMissed variable, a fault is generated and the indexing sequence is stopped. This command is used to set the SetMaxMarkMissed variable. If the value is zero the check is disabled.

Data Variable	Description
.IndexType[x]	SetMaxMarkMissed
.Data_1[x]	Value to set the SetMaxMarkMissed variable to, range is 0 to 2,147,483,647, 0 disables the check.
.Data_2[x]	Not Used.
.Data_3[x]	Not Used.
.Dwell[x]	The amount of time between the end of the move or index and the start of the index or step. The time is entered in milliseconds and has a range of 0 to 4,294,967,295 ms.
.Next Index[x]	The number of the next index or step to execute when Index Complete Mode is not set to Stop. The value can be between 0 and 299.
.Index Complete Mode[x]	Action taken when the index or step is complete. The following are valid values: 0. Stop. 1. Start next index when move is complete and after Dwell. 2. Start next index when Start Index Input is on. 3. Start next index before move is complete. 4. Optional Stop, stops at the end of the index if the Optional Stop input is on.

SET ROLLOVER

Sets the axis rollover position value. The actual position of the axis will rollover to zero when it reaches this position when going forward. The actual position of the axis will rollover to this value when it reaches zero when going backwards This command uses the READ_SV/WRITE_SV function variable 12 to set the rollover value, reference the PiCPro Function/Function Block Reference Guide for details on the READ_SV function.

Entering a 0 turns rollover on position feature off. Negative values cannot be entered.

NOTE
Without rollover on position when 2,147,483,647 is reached, the next number is -2,147,483,648. The count continues to zero and back up to 2,147,483,647, etc.

Data Variable	Description
.IndexType[x]	SetRollover
.Data_1[x]	Roll Over Position, the value is entered in LU.
.Data_2[x]	Not Used.
.Data_3[x]	Not Used.
.Dwell[x]	The amount of time between the end of the move or index and the start of the index or step. The time is entered in milliseconds and has a range of 0 to 4,294,967,295 ms.
.Next Index[x]	The number of the next index or step to execute when Index Complete Mode is not set to Stop. The value can be between 0 and 299.
.Index Complete Mode[x]	Action taken when the index or step is complete. The following are valid values: 0. Stop. 1. Start next index when move is complete and after Dwell. 2. Start next index when Start Index Input is on. 3. Start next index before move is complete. 4. Optional Stop, stops at the end of the index if the Optional Stop input is on.

SET S-CURVE ACCELERATION/DECELERATION

Changes the S-Curve acceleration/deceleration rate for the motion commands. This command uses the M_SACC ASFB and ACC_JERK function to set the rate, reference the Motion ASFB Manual for details on the M_SACC function and the PiCPro Function/Function Block Reference Guide for details on the ACC_JERK function.

Data Variable	Description
.IndexType[x]	SetScurveAccDec
.Data_1[x]	Total time for the axis to reach the Velocity Rate. The value is entered in ms.
.Data_2[x]	The percentage of time spent in constant jerk. A value of 80(%) percent means 40% of the acceleration time spent in constant jerk, 20% in constant acceleration and another 40% in constant jerk. This value must be set to 100 or less.
.Data_3[x]	Maximum velocity, the value is entered in LU/Min.
.Dwell[x]	The amount of time between the end of the move or index and the start of the index or step. The time is entered in milliseconds and has a range of 0 to 4,294,967,295 ms.
.Next Index[x]	The number of the next index or step to execute when Index Complete Mode is not set to Stop. The value can be between 0 and 299.
.Index Complete Mode[x]	Action taken when the index or step is complete. The following are valid values: 0. Stop. 1. Start next index when move is complete and after Dwell. 2. Start next index when Start Index Input is on. 3. Start next index before move is complete. 4. Optional Stop, stops at the end of the index if the Optional Stop input is on.

TURN OFF OUTPUT

Turns off one of the 8 user defined outputs. The IO.Output BOOL (0..8) variables are used to interface between the ladder the indexing program. These variables can be used in the ladder to control physical outputs or used in the ladder for user defined functions.

Data Variable	Description
.IndexType[x]	TurnOffOutput
.Data_1[x]	Output Number, 1-8.
.Data_2[x]	Not Used.
.Data_3[x]	Not Used.
.Dwell[x]	The amount of time between the end of the move or index and the start of the index or step. The time is entered in milliseconds and has a range of 0 to 4,294,967,295 ms.
.Next Index[x]	The number of the next index or step to execute when Index Complete Mode is not set to Stop. The value can be between 0 and 299.
.Index Complete Mode[x]	Action taken when the index or step is complete. The following are valid values: 0. Stop. 1. Start next index when move is complete and after Dwell. 2. Start next index when Start Index Input is on. 3. Start next index before move is complete. 4. Optional Stop, stops at the end of the index if the Optional Stop input is on.

TURN ON OUTPUT

Turns on one of the 8 user defined outputs. The IO.Output BOOL (0..8) variables are used to interface between the ladder the indexing program. These variables can be used in the ladder to control physical outputs or used in the ladder for user defined functions.

Data Variable	Description
.IndexType[x]	TurnOnOutput
.Data_1[x]	Output Number, 1-8.
.Data_2[x]	Not Used.
.Data_3[x]	Not Used.
.Dwell[x]	The amount of time between the end of the move or index and the start of the index or step. The time is entered in milliseconds and has a range of 0 to 4,294,967,295 ms.
.Next Index[x]	The number of the next index or step to execute when Index Complete Mode is not set to Stop. The value can be between 0 and 299.
.Index Complete Mode[x]	Action taken when the index or step is complete. The following are valid values: 0. Stop. 1. Start next index when move is complete and after Dwell. 2. Start next index when Start Index Input is on. 3. Start next index before move is complete. 4. Optional Stop, stops at the end of the index if the Optional Stop input is on.

USER COMMAND 1

Used to pass data from the indexing program to the main ladder. This data can then be used to perform special user defined functions in the main ladder. The data is only valid until the index is executed. Thus, it should be moved to different variables, if needed.

Data Variable	Description
.IndexType[x]	UserCommand1
.Data_1[x]	DINT data to be used in the main ladder.
.Data_2[x]	DINT data to be used in the main ladder
.Data_3[x]	UDINT data to be used in the main ladder
.Dwell[x]	The amount of time between the end of the move or index and the start of the index or step. The time is entered in milliseconds and has a range of 0 to 4,294,967,295 ms.
.Next Index[x]	The number of the next index or step to execute when Index Complete Mode is not set to Stop. The value can be between 0 and 299.
.Index Complete Mode[x]	Action taken when the index or step is complete. The following are valid values: 0. Stop. 1. Start next index when move is complete and after Dwell. 2. Start next index when Start Index Input is on. 3. Start next index before move is complete. 4. Optional Stop, stops at the end of the index if the Optional Stop input is on.

USER COMMAND 2

Used to pass data from the indexing program to the main ladder. This data can then be used to perform special user defined functions in the main ladder. The data is only valid until the index is executed. Thus, it should be moved to different variables, if needed.

Data Variable	Description
.IndexType[x]	UserCommand2
.Data_1[x]	DINT data to be used in the main ladder.
.Data_2[x]	DINT data to be used in the main ladder.
.Data_3[x]	UDINT data to be used in the main ladder.
.Dwell[x]	The amount of time between the end of the move or index and the start of the index or step. The time is entered in milliseconds and has a range of 0 to 4,294,967,295 ms.
.Next Index[x]	The number of the next index or step to execute when Index Complete Mode is not set to Stop. The value can be between 0 and 299.
.Index Complete Mode[x]	Action taken when the index or step is complete. The following are valid values: 0. Stop. 1. Start next index when move is complete and after Dwell. 2. Start next index when Start Index Input is on. 3. Start next index before move is complete. 4. Optional Stop, stops at the end of the index if the Optional Stop input is on.

VELOCITY BACKWARD

Start a velocity move in the backward or minus direction. If the software end limits are enabled, the axis stops when it reaches these positions. The NewRate command can be use to change the rate. The VelocityStop command is used to stop the move. This command uses the VEL_STRT function to set execute the move, reference the PiCPro Function/Function Block Reference Guide for details on the VEL_STRT function.

Data Variable	Description
.IndexType[x]	VelocityBackward
.Data_1[x]	Not Used.
.Data_2[x]	Not Used.
.Data_3[x]	Maximum velocity, The value is entered in LU/Min.
.Dwell[x]	The amount of time between the end of the move or index and the start of the index or step. The time is entered in milliseconds and has a range of 0 to 4,294,967,295 ms.
.Next Index[x]	The number of the next index or step to execute when Index Complete Mode is not set to Stop. The value ranges from 0 to 299.
.Index Complete Mode[x]	Action taken when the index or step is complete. The following are valid values: 0. Stop. 1. Start next index when move is complete and after Dwell. 2. Start next index when Start Index Input is on. 3. Start next index before move is complete. 4. Optional Stop, stops at the end of the index if the Optional Stop input is on.

VELOCITY FORWARD

Start a velocity move in the forward or plus direction. If the software end limits are enabled, the axis stops when it reaches these positions. The NewRate command can be use to change the rate. The VelocityStop command is used to stop the move. This command uses the VEL_STRT function to set execute the move, reference the PiCPro Function/Function Block Reference Guide for details on the VEL_STRT function.

Data Variable	Description
.IndexType[x]	VelocityForward
.Data_1[x]	Not Used.
.Data_2[x]	Not Used.
.Data_3[x]	Maximum velocity. The value is entered in LU/Min.
.Dwell[x]	The amount of time between the end of the move or index and the start of the index or step. The time is entered in milliseconds and has a range of 0 to 4,294,967,295 ms.
.Next Index[x]	The number of the next index or step to execute when Index Complete Mode is not set to Stop. The value can be between 0 and 299.
.Index Complete Mode[x]	Action taken when the index or step is complete. The following are valid values: 0. Stop. 1. Start next index when move is complete and after Dwell. 2. Start next index when Start Index Input is on. 3. Start next index before move is complete. 4. Optional Stop, stops at the end of the index if the Optional Stop input is on.

VELOCITY STOP

Stop the active VelocityForward or VelocityBackward move. This command uses the VEL_END function to set execute the move, reference the PiCPro Function/Function Block Reference Guide for details on the VEL_END function.

Data Variable	Description
.IndexType[x]	VelocityStop
.Data_1[x]	Not Used.
.Data_2[x]	Not Used.
.Data_3[x]	Not Used.
.Dwell[x]	The amount of time between the end of the move or index and the start of the index or step. The time is entered in milliseconds and has a range of 0 to 4,294,967,295 ms.
.Next Index[x]	The number of the next index or step to execute when Index Complete Mode is not set to Stop. The value ranges from 0 to 299.
.Index Complete Mode[x]	Action taken when the index or step is complete. The following are valid values: 0. Stop. 1. Start next index when move is complete and after Dwell. 2. Start next index when Start Index Input is on. 3. Start next index before move is complete. 4. Optional Stop, stops at the end of the index if the Optional Stop input is on.

WAIT FOR IN POSITION

Waits for the axis to be in position, the axis is in position when the axis is with-in the in-position bandwidth which is set in the servo setup or by the WriteServoData variable 40 command. This function is used when a move command was issued and the next index was started before the move was completed. It allows the user to program Wait For …, Turn On/Off Output, and New Rate commands while motion is active.

Data Variable	Description
.IndexType[x]	WaitForInPosn
.Data_1[x]	Not Used.
.Data_2[x]	Not Used.
.Data_3[x]	Not Used.
.Dwell[x]	The amount of time between the end of the move or index and the start of the index or step. The time is entered in milliseconds and has a range of 0 to 4,294,967,295 ms.
.Next Index[x]	The number of the next index or step to execute when Index Complete Mode is not set to Stop. The value can be between 0 – 127.
.Index Complete Mode[x]	Action taken when the index or step is complete. The following are valid values: 0. Stop. 1. Start next index when move is complete and after Dwell. 2. Start next index when Start Index Input is on. 3. Start next index before move is complete. 4. Optional Stop, stops at the end of the index if the Optional Stop input is on.

WAIT FOR INPUT OFF

Waits for one of the 8 user defined inputs to go off. The IO.Input BOOL (0..8) variables are used to interface between the ladder the indexing program. These variables can be connected to physical input or turned on/off in the ladder for user defined functions.

Data Variable	Description
.IndexType[x]	WaitForInputOff
.Data_1[x]	Input number 1-8.
.Data_2[x]	Not Used.
.Data_3[x]	Not Used.
.Dwell[x]	The amount of time between the end of the move or index and the start of the index or step. The time is entered in milliseconds and has a range of 0 to4,294,967,295 ms.
.Next Index[x]	The number of the next index or step to execute when Index Complete Mode is not set to Stop. The value can be between 0 and 299.
.Index Complete Mode[x]	Action taken when the index or step is complete. The following are valid values: 0. Stop. 1. Start next index when move is complete and after Dwell. 2. Start next index when Start Index Input is on. 3. Start next index before move is complete. 4. Optional Stop, stops at the end of the index if the Optional Stop input is on

WAIT FOR INPUT ON

Waits for one of the 8 user defined inputs to go on. The IO.Input BOOL (0..8) variables are used to interface between the ladder the indexing program. These variables can be connected to physical input or turned on/off in the ladder for user defined functions.

Data Variable	Description
.IndexType[x]	WaitForInputOn
.Data_1[x]	Input number 1-8.
.Data_2[x]	Not Used.
.Data_3[x]	Not Used.
.Dwell[x]	The amount of time between the end of the move or index and the start of the index or step. The time is entered in milliseconds and has a range of 0 to 4,294,967,295 ms.
.Next Index[x]	The number of the next index or step to execute when Index Complete Mode is not set to Stop. The value can be between 0 and 299.
.Index Complete Mode[x]	Action taken when the index or step is complete. The following are valid values: 0. Stop. 1. Start next index when move is complete and after Dwell. 2. Start next index when Start Index Input is on. 3. Start next index before move is complete. 4. Optional Stop, stops at the end of the index if the Optional Stop input is on.

WAIT FOR POSITION GREATER THAN OR EQUAL

Waits for the actual position to be “Greater Than or Equal To” a programmed value. This command is typically used along with one of the motion commands. A typically application would be to change the rate of a move after it reaches a certain position. The program would be as follows:

Start an Absolute position move

Start next index before move is complete

Wait for the position to be Greater Than or Equal To the speed change position

Change the velocity using the NewRate command

Wait for the move to stop using the WaitForInPosition command.

Data Variable	**Description**
.IndexType[x]	WaitForPosnGE
.Data_1[x]	Position, value is entered in LU.
.Data_2[x]	Not Used.
.Data_3[x]	Not Used.
.Dwell[x]	The amount of time between the end of the move or index and the start of the index or step. The time is entered in milliseconds and has a range of 0 to 4,294,967,295 ms.
.Next Index[x]	The number of the next index or step to execute when Index Complete Mode is not set to Stop. The value can be between 0 and 299.
.Index Complete Mode[x]	Action taken when the index or step is complete. The following are valid values: 0. Stop. 1. Start next index when move is complete and after Dwell. 2. Start next index when Start Index Input is on. 3. Start next index before move is complete. 4. Optional Stop, stops at the end of the index if the Optional Stop input is on.

Wait For Position Less Than Or Equal

Waits for the actual position to be “Less Than or Equal To” a programmed value. This command is typically used along with one of the motion commands. A typically application would be to change the rate of a move after it reaches a certain position. The program would be as follows:

Start an Absolute position move

Start next index before move is complete

Wait for the position to be Less Than or Equal To the speed change position

Change the velocity using the NewRate command

Wait for the move to stop using the WaitForInPosition command

Data Variable	Description
.IndexType[x]	WaitForPosnLE
.Data_1[x]	Position, value is entered in LU.
.Data_2[x]	Not Used.
.Data_3[x]	Not Used.
.Dwell[x]	The amount of time between the end of the move or index and the start of the index or step. The time is entered in milliseconds and has a range of 0 to 4,294,967,295 ms.
.Next Index[x]	The number of the next index or step to execute when Index Complete Mode is not set to Stop. The value can be between 0 and 299.
.Index Complete Mode[x]	Action taken when the index or step is complete. The following are valid values: 0. Stop. 1. Start next index when move is complete and after Dwell. 2. Start next index when Start Index Input is on. 3. Start next index before move is complete. 4. Optional Stop, stops at the end of the index if the Optional Stop input is on.

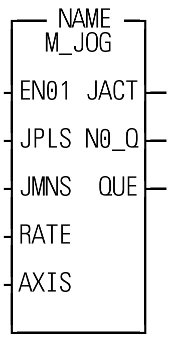
Programming Quick Reference Guide

Index Type	Data 1	Data 2	Data 3
Incremental	Incremental Distance LU	Not Used	Maximum velocity LU/Min
Registration	Maximum Incremental Distance LU	Distance to go past the registration mark LU	Maximum velocity LU/Min
Absolute	Absolute position LU	Not Used	Maximum velocity LU/Min
RotaryShortest	Absolute position LU	Not Used	Maximum velocity LU/Min
RotaryForward	Absolute position LU	Not Used	Maximum velocity LU/Min
RotaryBackward	Absolute position LU	Not Used	Maximum velocity LU/Min
VelocityForward	Not Used	Not Used	Maximum velocity LU/Min
VelocityBackward	Not Used	Not Used	Maximum velocity LU/Min
VelocityStop	Not Used	Not Used	Not Used
SetRollover	Roll Over Position LU	Not Used	Not Used
PartRef	Reference dimension LU	Not Used	Not Used
SetIgnoreDist	Ignore Distance LU	Not Used	Not Used
SetMaxMarkMissed	Maximum number of consecutive bad registration allowed	Not Used	Not Used
SetScurveAccDec	Acceleration-Deceleration Time mSec	Percent of time spent in constant Jerk, 0-100%	Maximum velocity LU/Min
SetLinearAccDec	Acceleration Time mSec	Deceleration Time mSec	Maximum velocity LU/Min
WaitForPosnGE	Position LU	Not Used	Not Used
WaitForPosnLE	Position LU	Not Used	Not Used
WaitForInPosn	Not Used	Not Used	Not Used

Index Type	Data 1	Data 2	Data 3
NewRate	Not Used	Not Used	Maximum velocity LU/Min
NOP	Not Used	Not Used	Not Used
TurnOnOutput	User Output Number 1-8	Not Used	Not Used
TurnOffOutput	User Output Number 1-8	Not Used	Not Used
WaitForInputOn	User Input Number 1-8	Not Used	Not Used
WaitForInputOff	User Input Number 1-8	Not Used	Not Used
IncCycleCount	Number of Times to repeat cycle, 0 – 65,535	Not Used	Not Used
SetLoopCount	Loop Counter Number, 1-2	Value to set the loop counter to, 0 – 65,535	Not Used
IncLoopCount	Loop Counter Number, 1-2	Number of times to loop, 0 – 65,535	Number of index to loop to
UserCommand1	User defined, -2,147,483,648 to +2,147,483,647	User defined, -2,147,483,648 to +2,147,483,647	User defined, 0 to 4,294,967,295
UserCommand2	User defined, -2,147,483,648 to +2,147,483,647	User defined, -2,147,483,648 to +2,147,483,647	User defined, 0 to 4,294,967,295
NOP	Not Used	Not Used	Not Used
EndOfProgram	Not Used	Not Used	Not Used

Index Setup Spread Sheet

There is an Excel Spreadsheet called “C:\G&L Motion Control Data\Applications V16.0.1\Tools\MIndexEx.xls” that can be used to generate a CSV file that can be loaded onto RAMDISK of MMCD controller. The CSV file can then be used by the M_INDEX ASFB to read in the programmed Indexer Data. The M_INDEX ASFB can also write the CSV which allows the user to back up the programs. Open up the file with Excel and save it under another name and create your program. Copy it to the RAMDISK using PiCPro. You can then use the File Structure variables to read the file and convert the text to the Data Structure.



Inputs: EN01 (BOOL) - enables execution

JPLS (BOOL) - enables a jog in the plus direction

JMNS (BOOL) - enables a jog in the minus direction

RATE (UDINT) - feedrate at which motion occurs (entered in LU/MIN)

AXIS (USINT) - identifies axis

Outputs: JACT (BOOL) - indicates jogging is active when set; indicates no motion is occurring when not set

NO_Q (BOOL) - active queue for the specified axis was not available

QUE (USINT) - number of move for queue

<<INSTANCE NAME>>:M_JOG(EN01 := <<BOOL>>, JPLS := <<BOOL>>, JMNS := <<BOOL>>, RATE := <<UDINT>>, AXIS := <<USINT>>, JACT => <<BOOL>>), NO_Q => <<BOOL>>, QUE => <<USINT>>);

This function block is designed to simplify the task of doing a manual jog (velocity) move on a closed loop axis. The manual jog is defined as a move that would be triggered by the operator physically pressing a switch or a button to move an axis on the machine to a different location, without actually running a cycle.

Before this function block can be used, the axis must be initialized and placed in servo lock. If the enable input is active, triggering the jog plus (JPLS) or jog minus (JMNS) input will cause the specified axis to move at the indicated rate in the corresponding direction. When the input is deactivated motion will stop.

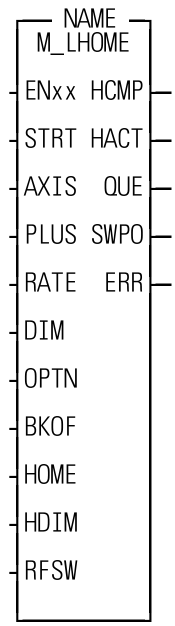
This function block is used to jog an axis that has been initialized and placed in servo lock with the close loop function. It checks the queue of the selected axis to be certain that no other moves are being executed. This function block should be used to allow the operator to manually move an axis on the machine. It is not designed for any other purpose.

IMPORTANT
If the enable is disabled while a move is under way, the move will end.

The JPLS input enables a move in the positive direction for the selected axis. The JMNS input enables a move in the negative direction for the selected axis. If both the JPLS and JMNS inputs are set, motion will stop until one of them is dropped. At that time motion will resume in the direction still selected.

M_LHOME

Performs a Home Cycle using a Ladder Reference **USER/M_REF**



Inputs: ENxx (BOOL) - enables execution, (xx indicates revision #)

STRT (BOOL) - enables the home cycle

AXIS (USINT) - identifies axis

PLUS (BOOL) - indicates direction of home cycle

RATE (UDINT) - feedrate at which motion occurs (entered in LU/MIN)

DIM (DINT) - reference dimension for the nearest resolver null or the next encoder index mark when the reference switch is set (entered in LUs)

OPTN (WORD) - provides referencing options 0=No option 1=Ignore index or null 2=Back to index 4=Use index only for reference

BKOF (BOOL) - selects backoff of reference switch option

HOME (BOOL) - selects homing after referencing option

HDIM (DINT) - home location to move to after reference is complete

RFSW (BOOL) - reference switch on axis

Outputs: HCMP (BOOL) - home cycle is complete

HACT (BOOL) - home cycle is being executed

QUE (USINT) - number of move for queue

SWPO (DINT) - distance in feedback units (FUs) from the reference switch to the index mark of an encoder or the null of a resolver.

ERR (BYTE) - report an error 1-4 if input data is invalid

<<INSTANCE NAME>>:M_LHOME(ENxx := <<BOOL>>, STRT := <<BOOL>>, AXIS := <<USINT>>, PLUS := <<BOOL>>, RATE := <<UDINT>>, DIM := <<DINT>>, OPTN := <<WORD>>, BKOF := <<BOOL>>, HOME := <<BOOL>> HDIM := <<DINT>>, RFSW := <<BOOL>>, HCMP => <<BOOL>>, HACT => <<BOOL>>, QUE => <<USINT>>, SWPO <<DINT>>, ERR => <<BYTE>>);

This function block performs a ladder reference cycle on an axis, followed by a homing (position) move to a designated location.

Before this function block can be used, the axis must be initialized and the position loop closed.

The reference cycle will cause the selected axis to move in the designated direction until the reference switch is sensed. In a ladder reference this reference switch is wired to an input module in the PiC900 and updated each scan of the ladder. When the reference switch is sensed the axis will reference (assign a value) to the next index mark of an encoder or the nearest null of a resolver. After the value is assigned the axis will decelerate to a stop and set the reference done flag.

The OPTN input controls how the reference switch and index are utilized.

> If OPTN is set to one, the index is ignored.
>
> If OPTN is set to two, the axis will travel to the reference switch, stop, and then travel back to the previous index mark.
>
> If OPTN is set to four, the reference switch is ignored during the reference cycle. The axis will reference only to the index mark.

If the HOME input is on when the reference done has been sensed the home move will automatically be triggered to position the axis at a desired location.

If the BKOF input is on when the reference is requested and if the axis is on the reference switch the axis will move in the opposite direction until the reference switch opens, and will then move back onto the reference switch. If the BKOF input is not on the axis will move in the specified direction until it sees an off to on transition of the limit switch.

This function block is used to perform a ladder reference, immediately followed by a position move to a selected home position. It should be executed every scan unless a home cycle will only be performed when the machine is started. In that case a normally closed contact of the output of HCMP may be used.

The inputs to this function block are similar to those of the FAST_REF function. There are four additional inputs listed below.

The BKOF input selects the backoff reference switch option.

The HOME input selects the homing after referencing option.

The HDIM input assigns the home dimension to move to.

The RFSW input is the reference switch.

If the axis is sitting on the limit switch when the home cycle is requested, and the BKOF input is on, the axis will move in the opposite direction of that indicated by the PLUS input until the switch opens and then will complete the home cycle in the normal manner.

The SWPO output is used to determine if the reference switch location will allow for repeatable referencing. If the reference switch is not properly located in relationship to the index marker of an encoder or the null of a resolver it could possibly reference a revolution off. To prevent this, the value reported by this output should be as follows:

- For an encoder system the value of this output should be greater than 25% and less than 75% of the total counts (FUs) per revolution. Example: For 8000 FUs/Rev, the value should be >2000 and <6000.
- For a resolver system the value of this output should be less than 25% or greater than 75% of the total counts (FUs) per revolution. Example: For 4000 FUs/Rev, the value should be <1000 or >3000.

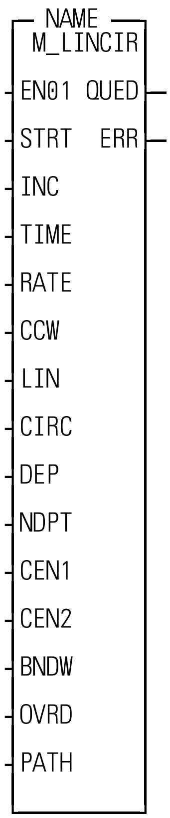
If the value is out of range either the reference switch will have to be moved or the transducer coupling shifted.

The ERR output indicates that invalid data was entered on one of the inputs. See the table below for a description of ERR output values.

ERR	Description
0	No error
1	The queue was not empty when the reference was requested
2	An error occurred in backing off of the reference switch
3	An error occurred in referencing
4	An error occurred in homing

M_LINCIR

Performs Linear and Circular Moves **USER/M_MOVE**



Inputs:

EN01 (BOOL) - enables execution

STRT (BOOL) - enables the coordinated move

INC (WORD) - defines incremental or absolute mode (0=absolute, 1=incremental)

TIME (BOOL) - defines if move is feedrate or time of move (0=feedrate, 1=time of move)

RATE (DINT) - feedrate or time of move

CCW (BOOL) - defines direction of circular move (0=clockwise, 1=counter-clockwise)

LIN (WORD) - defines which axes to move in a linear mode

CIRC (WORD) - defines which axes to move in a circular mode

DEP (WORD) - defines which axes to move in a simultaneous endpoint arrival mode

NDPT (DINT(0..16)) - endpoints or distances to move

CEN1 (DINT) - circle center for lowest numbered circular axis

CEN2 (DINT) - circle center for highest numbered circular axis

BNDW (DINT) - circular endpoint bandwidth

OVRD (USINT) - feedrate override percentage

PATH (USINT) - path number

Outputs:

QUED (BOOL) - move was queued without error

ERR (INT) - error number describing error that occurred when the move was queued

<<INSTANCE NAME>>:M_LINCIR(EN01 := <<BOOL>>, STRT := <<BOOL>>, INC := <<WORD>>, TIME := <<BOOL>>, RATE := <<DINT>>, CCW := <<BOOL>>, LIN := <<WORD>>, CIRC := <<WORD>>, DEP := <<WORD>>, NDPT := <<DINT>>, CEN1 := <<DINT>>, CEN2 := <<DINT>>, BNDW := <<DINT>>, OVRD := <<USINT>>, PATH := <<USINT>>, QUED => <<BOOL>>, ERR => <<INT>>);

This function block performs linear, circular, or third axis departure (simultaneous endpoint arrival) moves on a set of axes.

Before this function can be used, the axes must be initialized, the position loop must be closed, and a queue must be available on all axes to be used in the move.

This function block provides the interface from the application .LDO to the RATIO_RL and CORD2RL functions in order to perform linear coordinated, circular, and third axis departure (simultaneous endpoint arrival) motions.

Up to four separate paths of coordinated motion can be controlled. Each path of motion requires a separate instantiation of the M_LINCIR function block. Each path must control a unique set of axes. Only one M_LINCIR function block per path can be used within the application .LDO.

This function block can control up to 16 axes.

The EN01 input of this function block must be set every scan.

The STRT input must be one-shot. When it is one-shot, the function block will start the coordinated move, or enter it in the queue for the axes. It is the user's responsibility to ensure that there is a queue available on all of the axes involved in the move before pulsing this input.

The INC input defines whether each axis should move in the absolute or incremental mode. One bit of this WORD is reserved for each of the sixteen possible axes. Bit 0 is set if axis 1 is incremental, or reset if axis 1 is absolute, bit 1 is set if axis 2 is incremental, reset if axis 2 is absolute, etc.

The TIME input defines whether the move should be executed as a path feedrate move or a time of move. This input should be reset for path feedrate, or set for time of move.

If the TIME input is reset, then the RATE input is the path feedrate for the move in ladder units/minute. If the TIME input is set, then the RATE input is the time for the move in milliseconds.

The CCW input is only used for circular moves. If it is reset, then the move is clockwise, if it is set, then the move is counter-clockwise.

The LIN input defines which axes in the move are to be moved in a linear mode. One bit of the WORD is reserved for each of the sixteen axes. The bit must be set for the axis to do a linear move. Axes who have their bits set will be included in the calculations for the path feedrate.

The CIRC input defines which axes in the move are to be moved in a circular mode. One bit of the WORD is reserved for each of the sixteen axes. The bit must be set for the axis to do a circular move. Axes who have their bits set will be included in the calculations for the path feedrate.

The DEP input defines which axes in the move are to be moved in a simultaneous endpoint arrival mode. One bit of the WORD is reserved for each of the sixteen axes. The bit must be set for the axis to move. Axes who have their bits set will not be included in the calculations for the path feedrate, but they will arrive at their endpoints simultaneously with the axes that are.

The LIN, CIRC, and DEP words may never have the same bits set in them at a time. You must always set a bit for every axis ever used in the path, even if the axis is not to move in this particular move. In this case, you would set either the LIN or DEP bit for the axis, set the INC bit for the axis, and program an endpoint of zero for the axis.

The NDPT array holds the endpoints for the axes used in the move. The 0th element is not used. If the INC bit is set for the axis, this is the distance to move, if the INC bit is reset for the axis, then this is the position to move to. The endpoints are entered in ladder units.

The CEN1 and CEN2 inputs define the circle centers if a circular move is being performed. The CEN1 input is the center for the lowest numbered circular axis, and the CEN2 input is the center for the highest numbered circular axis. The centers are always programmed as an incremental distance from the starting point of the circle, even if the INC bit for the axes is not set. The centers are entered in ladder units. For example, if a circle were being done with axes 4 and 6, then CEN1 would be the center for axis 4, and CEN2 would be the center for axis 6.

The BNDW input defines a bandwidth for circular moves. When a circular move is requested, the distance from the start point to the center point and the distance from the endpoint to the center point are compared for both axes. If these distances differ by more than the bandwidth entered here, then the move will not execute and error 14 will be returned on the ERR output. This bandwidth is entered in ladder units.

The OVRD input defines the feedrate override value. This can be changed at any time, even if the STRT input is not energized. This adjusts the actual feedrate or time to be from 0 to 255 percent of the programmed feedrate or time.

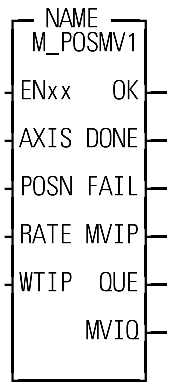
The PATH input defines the number of the path. Up to four totally independent paths of coordinated motion can be defined. This must be a number from 1 to 4. This should not be changed once it is set. Each path uses a time axis. Path 1 uses time axis 25. Path 2 uses time axis 26. Path 3 uses time axis 27. Path 4 uses time axis 28.

The QUED output will be set for one scan when STRT is pulsed and the move has been successfully queued on all axes defined. If an error occurred in queueing the move, this output will be reset when STRT is pulsed, and an error code will be stored in the ERR output.

The ERR output will be non-zero if an error occurs in queueing a move. A list of error codes is shown on the following table.

Note: WRITE_SV variable 25 Fast Queuing is enabled for the selected axes when STRT is set. Fast queuing will remain on for those axes until turned off by you.

ERR	Description
0	No error
1	No bits were set in the LIN, CIRC, or DEP WORDs
2	The same bit was set in the LIN and CIRC WORDs
3	The same bit was set in the DEP and CIRC WORDs
4	The same bit was set in the LIN and DEP WORDs
5	The number of bits set in the CIRC WORD was not 0 or 2
6	Not used
7	Not used
8	The time of move or feedrate was negative
9	The time of move or feedrate was zero
10	The feedrate was too high or the time was too low to calculate
11	The feedrate was too low or the time was too high to calculate
12	An axis that was selected was not initialized by the servo setup function
13	The STRTSRV function has not been called
14	Endpoint not on circle
1XX	When the distance to move was converted to feedback units, it was too positive to fit into 32 bits. XX = Axis number
2XX	When the distance to move was converted to feedback units, it was too negative to fit into 32 bits. XX = Axis number
3XX	The path feedrate or time entered causes an axis to exceed its velocity limit from servo setup. XX = Axis number
32766	The time axis could not be started. NOTE: For PiCPro 13.0 and higher you need to add the time axes to your servo setup.
32767	One of the OKs on the RATIO_RL functions did not get set. NOTE: For PiCPro 13.0 and higher you need to add the time axes to your servo setup.



Inputs:
ENxx (BOOL) - Enables execution (one shot)
AXIS (USINT) - Axis number
POSN (DINT) - End point in LU
RATE (UDINT) - Feed rate in LU / minute
WTIP (BOOL) - Wait for In Position signal before completing

Outputs:
OK (BOOL) - Command was accepted
DONE (BOOL) - Position move is complete
FAIL (BOOL) - No queue available or move is E-stopped or C-stopped
MVIP (BOOL) - Position move is in progress
QUE (USINT) - Queue number for move
MVIQ (BOOL) - Queued waiting for active move to end

```
<<INSTANCE NAME>>:M_POSMV1(ENxx := <<BOOL>>, AXIS :=
  <<USINT>>, POSN := <<DINT>>, RATE := <<UDINT>>, WTIP :=
  <<BOOL>>, OK => <<BOOL>>, DONE => <<BOOL>>, FAIL =>
  <<BOOL>>, MVIP => <<BOOL>>, QUE => <<USINT>>, MVIQ =>
  <<BOOL>>);
```

This block will initiate and track progress on a Position move for the specified axis. It is mainly intended for use in sequential move applications where each move is followed by another non motion or different axis action. In these situations, the move will typically become active immediately.

However, the block will queue a move as long as a queue is available and it will start when the active move completes.

E-stops and C-stops will cause the motion being monitored by the block to end. In these cases, the OK output will go off and the FAIL output will come on. The DONE output will stay off.

Aborts will also cause the move to terminate. The block cannot tell that an Abort has been done, and the OK and DONE outputs will come on in these cases. There is also one specific case where the MVIQ output will stay on incorrectly. This will occur if a move triggered by this block is in the queue and it is aborted and another Position move is started before this block runs.

INPUTS:

ENxx - (BOOL) one shot to trigger move

AXIS - (USINT) axis number to position

POSN - (DINT) end point for position move

RATE - (UDINT) feed rate for move in LU / Minute

WTIP - (BOOL) wait for in position. If this is on, the DONE output will not turn on until the axis is within the in position band. If off, DONE will turn on when the command position stops iterating. If a move is in the queue behind this move, then this setting has no effect and the DONE will be on when the move stops iterating.

OUTPUTS:

OK - (BOOL) turns on if move is accepted and goes off if move does not complete normally

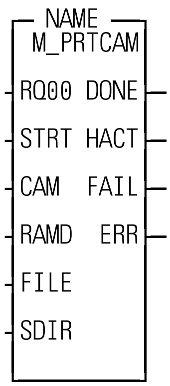
DONE - (BOOL) turns on when move completes, turns off when new move started

FAIL - (BOOL) on if move stopped due to E or C stop or no queue is available when EN turns on

MVIP - (BOOL) position move is in progress

QUE - (USINT) queue number for position move

MVIQ - (BOOL) move is in queue waiting for active move to end



Inputs: RQ00 (BOOL) - request file generation and write

CAM (STRUC 0..) - CAM structure input of the RATIOCAM function

RAMD (BOOL) - If enabled, allows file to be written to the RAMDISK. If disabled, file is written to the PC running PiCPro.

FILE (STRING) - name of the file

SDIR (STRING) - identifies the subdirectory where the file will be written to.

Outputs: DONE (BOOL) - set when the file is generated and written, reset when RQ00 goes on.

FAIL (BOOL) - set if an error occurs and reset when RQ00 goes on.

ERR (INT) - number of error that occurred. These errors are defined in Appendix B.

<<INSTANCE NAME>>:M_PRTCAM(RQ00 := <<BOOL>>, CAM := <<MEMORY AREA>>, RAMD := <<BOOL>>, FILE := <<STRING>> SDIR := <<STRING), DONE => <<BOOL>>, FAIL => <<BOOL>> ERR => <<INT>>);

This function block creates a text file for a RATIOCAM CAM structure. The file can be created on either the RAMDISK in the PiC or on the PC running PiCPro. A positive transition of RQ00 requests that the data specified by the CAM input be converted to ASCII code, concatenated, and written to the RAMDISK or to the PiCPro port. The CAM input is an array of structures and must have the following members:

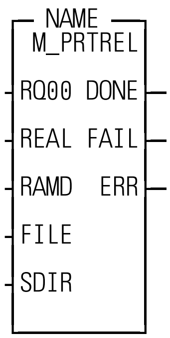
Name	Data Type	Definition
CAM	STRUCT (0..998)	The structure of the RATIOCAM profile
.M	INT	Master segment size
.S	INT	Slave segment size

The FILE input requires a string data type variable with the filename as an initial value. The format is “FILENAME.EXT”.

The SDIR input requires a string data type. A subdirectory is not required if you are writing the file to the RAMDISK. If you are writing the file to a PC running PiCPro, then the SDIR is required. It must contain the drive and subdirectory path. The following are examples showing the drive and subdirectory path:

C: indicates that the file will be written to C:Filename.ext.

C:\PRT_CAM indicates the file will be written to the directory C:\PRT_CAM\Filename.ext.



Inputs: RQ00 (BOOL) - request file generation and write

REAL (STRUC 0..) - REAL structure input of the RATIO_RL function

RAMD (BOOL) - If enabled, allows file to be written to the RAMDISK. If disabled, file is written to the PC running PiCPro.

FILE (STRING) - name of the file

SDIR (STRING) - identifies the subdirectory where the file will be written to.

Outputs: DONE (BOOL) - set when the file is generated and written, reset when RQ00 goes on.

FAIL (BOOL) - set if an error occurs and reset when RQ00 goes on.

ERR (INT) - number of error that occurred. These errors are defined in Appendix B.

<<INSTANCE NAME>>:M_PRTREL(RQ00 := <<BOOL>>, REAL := <<MEMORY AREA>>, RAMD := <<BOOL>>, FILE := <<(STRING>>, SDIR := <<STRING>>, DONE => <<BOOL>>, FAIL => <<BOOL>>, ERR => <<INT>>);

This function block creates a text file for a RATIO_RL structure. The file can be created on the RAMDISK in the PiC or on the PC running PiCPro. A positive transition of RQ00 requests that the data specified by the REAL input be converted to ASCII code, concatenated, and written to the RAMDISK or to the PiCPro port. The REAL input is an array of structures and must have these members:

Name	Data Type	Definition
REAL	STRUCT (0..998)	The structure of the RATIO_RL profile
.M	DINT	Master segment size
.S	DINT	Slave segment size
.LEN	LREAL	Length or K1
.AMPL	LREAL	Amplitude or K2
.STANGL	LREAL	Start angle or K3
.SPARE	LREAL	Spare for future use
.FLAGS	DWORD	Flags

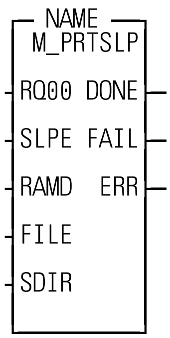
The FILE input requires a string data type variable with the filename as an initial value. The format is "FILENAME.EXT". The SDIR input requires a string data type. A subdirectory is not required if you are writing the file to the RAMDISK. If you are writing the file to a PC running PiCPro, then the SDIR is required. It must contain the drive and subdirectory path. The following are examples showing the drive and subdirectory path:

C: indicates that the file will be written to C:Filename.ext.

C:\PRT_CAM indicates the file will be written to the directory C:\PRT_CAM\Filename.ext.

M_PRTSLP

Creates a RATIOSLP text file **USER/M_DATA**



Inputs: RQ00 (BOOL) - request file generation and write

SLPE (STRUC 0..) - SLPE structure input of the RATIOSLP function

RAMD (BOOL) - If enabled, allows file to be written to the RAMDISK. If disabled, file is written to the PC running PiCPro.

FILE (STRING) - name of the file

SDIR (STRING) - identifies the subdirectory where the file will be written to.

Outputs: DONE (BOOL) - set when the file is generated and written, reset when RQ00 goes on.

FAIL (BOOL) - set if an error occurs and reset when RQ00 goes on.

ERR (INT) - number of error that occurred. These errors are defined in Appendix B.

<<INSTANCE NAME>>:M_PRTSLP(RQ00 := <<BOOL>>, SLPE := <<MEMORY AREA>> RAMD := <<BOOL>>, FILE := <<STRING>> SDIR := <<STRING>>, DONE => <<BOOL>>, FAIL => <<BOOL>>, ERR => <<INT>>);

This function block creates a text file for a RATIOSLP structure. The file can be created on either the RAMDISK in the PiC or on the PC running PiCPro.

A positive transition of RQ00 requests that the data specified by the SLPE input be converted to ASCII code, concatenated, and written to the RAMDISK or to the PiCPro port.

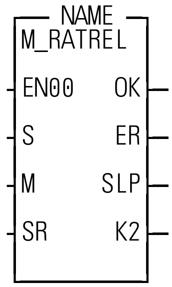
The REAL input is an array of structures and must have the following members:

Name	Data Type	Definition
SLPE	STRUCT (0..998)	The structure of the RATIOSLP profile
.M	INT	Master segment size
.S	INT	Slave segment size
.SLP	DINT	Slope of segment
.SR	DINT	Start ratio
.FLAGS	DWORD	Default flags

The FILE input requires a string data type variable with the filename as an initial value. The format is “FILENAME.EXT”. The SDIR input requires a string data type. A subdirectory is not required if you are writing the file to the RAMDISK. If you are writing the file to a PC running PiCPro, then the SDIR is required. It must contain the drive and subdirectory path. The following are examples showing the drive and subdirectory path:

C: indicates that the file will be written to C:Filename.ext.

C:\PRT_CAM indicates the file will be written to the directory C:\PRT_CAM\Filename.ext.



Inputs: EN01 (BOOL) - enables execution
S (DINT) - slave distance
M (DINT) - master distance
SR (LREAL) - starting ratio

Outputs: OK (BOOL) - execution complete
ER (LREAL) - ending ratio
SLP (LREAL) - slope
K2 (LREAL) - slope output divided by 2

<<INSTANCE NAME>>:M_RATREL(EN01 := <<BOOL>>, S := <<DINT>>, M := <<DINT>>, SR := <<LREAL>>, OK => <<BOOL>>, ER => <<LREAL>>, SLP => <<LREAL>>, K2 => <<LREAL>>);

This function block calculates the ending ratio, slope, and K2 (slope/2) used in the ratio real structure from the master distance, slave distance, and starting ratio.

This function block calculates the ending ratio and slope to be used with the RATIO_RL structure as one segment of the RATIO_RL profile. Refer to the documentation in the PiCPro Online Help regarding RATIO_RL for more information.

The slave and master segments (S and M) are entered in feedback units.

The starting ratio for the first segment of a RATIO_RL profile is normally zero. The starting ratio is called LEN or K1 in the ratio real documentation.

The formulas used by this function for calculation are as follows:

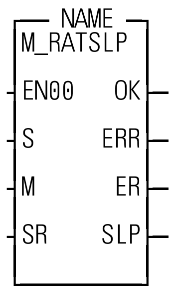
ER = (2S / M) - SR

SLP = (ER - SR) / M

K2 = SLP / 2

where ER is the ending ratio, SR is the starting ratio, S is the slave distance, M is the master distance, SLP is the slope, and K2 is the slope divided by 2. K2 is the AMPL structure member of the RATIO_RL REAL structure for a linear move.

The ending ratio is not an input to the RATIO_RL structure. However the ending ratio of one segment is normally used as the starting ratio of the next segment.



Inputs: EN01 (BOOL) - enables execution

S (INT) - slave distance

M (INT) - master distance

SR (DINT) - starting ratio

Outputs: OK (BOOL) - execution complete without error

ERR (INT) - error number

ER (DINT) - ending ratio

SLP (DINT) - slope

<<INSTANCE NAME>>:M_RATREL(EN01 := <<BOOL>>, S := <<INT>>, M := <<INT>>, SR := <<DINT>>, OK => <<BOOL>>, ERR => <<INT>>, ER => <<DINT>>, SLP => <<DINT>>);

This function block calculates the ending ratio and slope used in the ratio slope structure from the master distance, slave distance, and starting ratio.

This function block calculates the ending ratio and slope to be used with the RATIOSLP structure as one segment of the RATIOSLP profile. Refer to the documentation in the PiCPro Online Help regarding RATIOSLP for more information. The slave and master segments (S and M) are entered in feedback units.

The starting ratio for the first segment of a slope profile is normally zero. Non zero starting ratios must already be multiplied by the scaling factor of 16777216 before being used as an input to this function.

The formulas used by this function for calculation are as follows:

$$ER = (2S / M) - SR$$

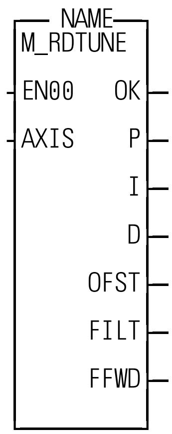
$$SLP = (ER - SR) / M$$

where ER is the ending ratio, SR is the starting ratio, S is the slave distance, M is the master distance, and SLP is the slope.

The ending ratio and slope that are outputs of this function have been multiplied by the scaling factor of 16777216. The ending ratio is not an input to the RATIOSLP structure. However, the ending ratio of one segment is normally used as the starting ratio of the next segment.

ERR	Description
1	The calculation for ER failed when S was between -64 and +63 (inclusive)
2	The calculation for ER failed when S was less than -64 or greater than +63
3	The calculation for SLP failed

Note:An M value of zero results in an error due to an attempt to divide by 0. No master distance can have a value of zero in a RATIOSLP profile.



Inputs: EN00 (BOOL) - enables execution
AXIS (USINT) - identifies axis

Outputs: OK (BOOL) - execution complete
P (DINT) - proportional gain
I (DINT) - integral gain
D (DINT) - derivative gain
OFST (DINT) - analog output offset
FILT (DINT) - slow speed filter value
FFWD (DINT) - feedforward percentage

<<INSTANCE NAME>>:M_RDTUNE(EN00 := <<BOOL>>, AXIS := <<USINT>>, OK => <<BOOL>>, P => <<DINT>>, I => <<DINT>>, D => <<DINT>>, DFST => <<DINT>>, FILT => <<DINT>>, FFWD => <<DINT>>);

This function block allows you to read all six tuning parameters from the TUNEREAD function in a single function.

This function block requires the numeric processor or a 486 DX processor.

The proportional gain for AXIS will be returned in P. P is in ladder units per minute per ladder unit of following error (LU / MIN / LUFE).

The integral gain for AXIS will be returned in I. I is in ladder units per minute per ladder units of following error times minutes (LU / MIN / LUFE * MIN).

The derivative gain for AXIS will be returned in D. D is in ladder units per minute per ladder unit of following error per minute (LU / MIN / LUFE / MIN).

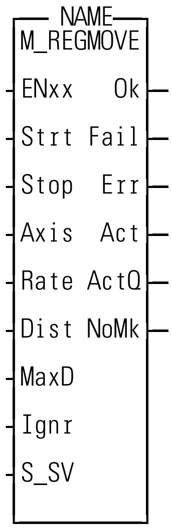
The analog output offset voltage for AXIS will be returned in OFST. OFST is in millivolts.

The slow speed filter value for AXIS will be returned in FILT. FILT is in milliseconds.

The feedforward percentage for AXIS will be returned in FFWD. FFWD will be from 0 to 100.

M_REGMOV

Performs Registration Move **USER/M_MOVE**



Inputs: ENxx (BOOL) - Enable FB - on at all times (xx indicates revision #)

Strt (BOOL) - Starts Registration Move - one shot on to start

Stop (BOOL) - Stops Registration Move - turn on to stop

Axis (USINT) - Axis number

Rate (UDINT) - Velocity rate in LU per min

Dist (DINT) - Distance you want to go past the registration mark in LU

MaxD (DINT) - Maximum distance to travel in LU

Ignr (DINT) - Distance to ignore registration marks in LU

S_SV (WORD) - Output of STATUSSV function

Outputs: Ok (BOOL) - Registration move was queued up

Fail (BOOL) - Registration move failed to execute

Err (INT) - Error code, see table below

Act (BOOL) - Registration move is active, axis is moving

ActQ (USINT) - Queue number of active registration move

NoMk (BOOL) - No Mark, no registration mark occurred with the maximum distance

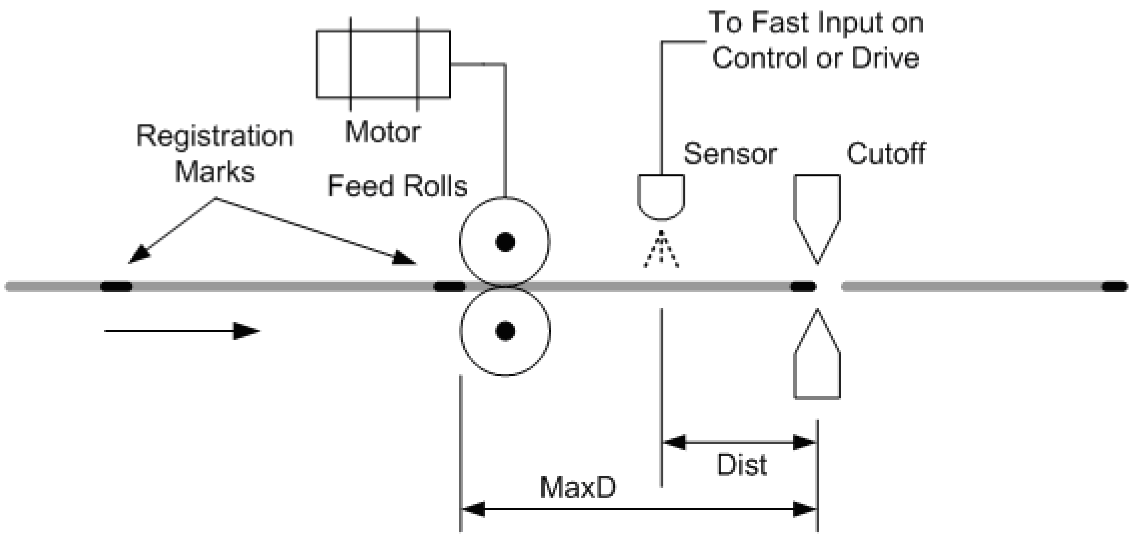
<<INSTANCE NAME>>:M_REGMOV(ENxx := <<BOOL>>, Strt := <<BOOL>>, Stop := <<BOOL>>, Axis := <<USINT>>, Rate := <<UDINT>>, Dist <<DINT>>, MaxD := <<DINT>>, Ignr := <<DINT>>, S_SV := <<WORD>>, Ok => <<BOOL>>, Fail => <<BOOL>>, Err => <<INT>>, Act => <<BOOL>>, ActQ => <<USINT>>, NoMk => <<BOOL>>);

This function block is used to perform an incremental distance / registration move. It allows you to program the axis to travel an incremental distance past a registration mark. A sensor must be wired to the fast input for the axis to detect the registration mark.

NOTE: The fast input that the registration sensor is wired to must not be declared as an I/O point in the software declarations. The M_REGMOV uses the FAST_QUE function which has control over the fast input. If you declare it as an I/O, then you may miss registration marks.

Figure 1 below shows a typical setup for a cut to length application.

Figure 1. Example Cut to Length



The feed rolls are used to index or move (MaxD) the material to cut. If the Sensor detects the registration mark on the material, the move distance is adjusted so that the registration mark is under the cutoff (Dist).

Inputs

The ENxx input must be enabled every scan.

The Strt input must be a one shot. When it goes on, the function block will start the registration move sequence as follows:

1. An incremental DISTANCE move of MaxD is queued up.
2. When the DISTANCE move is active and traveled past the Ignr distance, and the fast input / sensor is off, the FAST_QUE function is called and a second DISTANCE move of Dist is queued up.
3. If the fast input / sensor is triggered before the MaxD move is complete, the first distance move is aborted and the second distance move becomes active. The axis will stop the Dist past the registration mark sensor.
4. If no registration mark is detected before the MaxD. The axis will stop at the MaxD distance and abort the FAST_QUE and second DISTANCE move. The NoMK output will be turned on.

The Stop input will abort the move sequence and the axis will decelerate to a stop.

The Axis input is the axis number of the axis to move. The registration sensor must be wired to the fast input of this axis.

The Rate input is the maximum velocity or feedrate in LU/Min that the axis will move.

The Dist input is the distance in LU the axis will travel past the registration sensor.

The MaxD input is the maximum distance in LU the axis will travel.

The Ignr input is the distance in LU from the start of the move that the function will ignore any fast inputs.

The S_SV input is the STAT output of the STATUSSV function. This function block uses the status bits to determine if the fast input is on and when the distance move is started. You can not have more than one instance per axis of the STATUSSV function in your ladder.

NOTE: Reference the "PiCPro Function/Function Block Reference Guide" for details on the DISTANCE, FAST_QUE, and STATUSSV functions.

Outputs

The OK output is on when the function block is enabled and no faults are active.

The Fail output is on when a fault is active and the registration move sequence failed to execute.

The Err output is an error number that describes the error or fault that occurred. See the error listings below.

The Act output is on when the registration move sequence is active, i.e. the axis is in motion.

The ActQ output is the queue number of the active DISTANCE move.

The NoMk output is turned on when the MaxD distance was traveled without seeing a fast input.

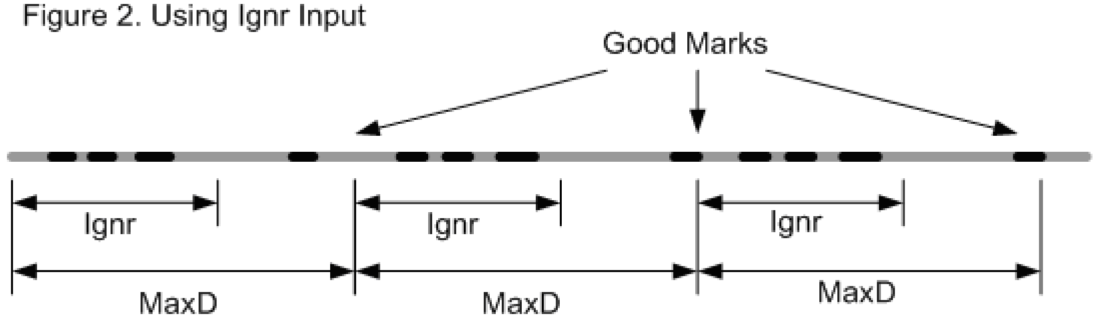
Error Codes

1	Another move was active when the Strt Input went on. No moves can be active when the function is called.
2	The first DISTANCE move failed, check MaxD and Rate inputs or the WriteSV to enable Fast Queuing failed.
3	A move was queued up outside of the M_REGMOV ASFB. No motion commands should be executed in the ladder when the registration move is active or started.
4	FAST_QUE function or second DISTANCE failed. Check Dist and Rate inputs.

5	A move was queued up outside of the M_REGMOV ASFB when the first DISTANCE move was active and before the FAST_QUE and second DISTANCE moves were called. No motion commands should be executed in the ladder when the registration move is active or started.
6	The fast input for the Registration Sensor never went off.

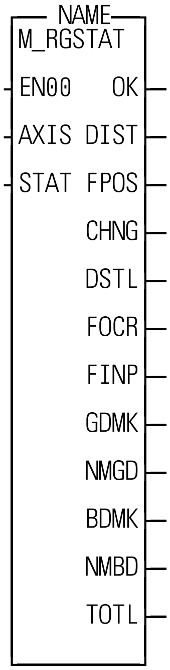
If you have multiple marks on your material you can use the Ignr input to ignore marks as show below in figure 2.

Figure 2. Using Ignr Input



M_RGSTAT

Returns Registration Data **USER/M_DATA**



Inputs: EN00 (BOOL) - enables execution

AXIS (USINT) - axis number

STAT (WORD) - status word from STATUSSV function

Outputs: OK (BOOL) - execution complete

DIST (DINT) - fast input distance

FPOS (DINT) - fast input position

CHNG (DINT) - registration/referencing position change

DSTL (BOOL) - indicates distance plus tolerance has been exceeded

FOCR (BOOL) - fast input occurred

FINP (BOOL) - fast input on

GDMK (BOOL) - good mark detected

NMGD (DINT) - number of good registration marks

BDMK (BOOL) - bad mark detected

NMBD (DINT) - number of bad registration marks

TOTL (DINT) - total number of fast inputs that have occurred

<<INSTANCE NAME>>:M_RGSTAT(EN00 := <<BOOL>>, AXIS := <<USINT>>, STAT := <<WORD>>, OK => <<BOOL>>, DIST => <<DINT>>, FPOS => <<DINT>>, CHNG => <<BOOL>>, DSTL => <<BOOL>>, FOCR => <<BOOL>>, FINP => <<BOOL>>, GDMK => <<BOOL>>, NMGD => <<DINT>>, BDMK => <<BOOL>>, NMBD => <<DINT>>, TOTL => <<DINT>>);

This function block obtains information about registration. The information gathered is distance between fast inputs, fast input position, registration reference change, number of good marks, number of bad marks, total number of marks, and the state of STATUSSV flags.

This function block should be enabled every scan.

The input at AXIS determines which axis the output information is for. AXIS can be a closed loop or digitizing axis.

The STAT input is the status word read from the STATUSSV function. STATUSSV can only be called once per scan, so its output is used as an input to this function.

The OK output will not be set if the axis has not been initialized.

The DIST output is the distance between the most recent fast input and the previous fast input in ladder units.

The FPOS output is the actual position of the axis at the point where the most recent fast input occurred in ladder units.

The CHNG output is the amount the position of the axis has changed in ladder units due to registration or the last machine reference.

The DSTL output will be set if the distance from the last mark exceeds the value of DIST + TOLR whether or not a mark has occurred. It will be reset when any mark occurs.

The FOCR output will be set if a fast input has occurred since the last time the STATUSSV function was called.

The FINP output is set if the fast input is on, and reset if the fast input is off.

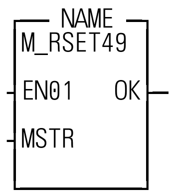
The GDMK output will be set if a good mark has been detected since the last time the STATUSSV function was called.

The NMGD output holds the total number of good registration marks that have been detected.

The BDMK output will be set if a bad mark has been detected since the last time the STATUSSV function was called.

The BAD output holds the number of bad registration marks that have been detected.

The TOTL output holds the total number of fast input transitions that have occurred.



Inputs: EN01 (BOOL) - enables execution

MSTR (BOOL) - machine start input

Outputs: OK (BOOL) - execution complete

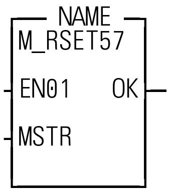
<<INSTANCE NAME>>:M_RSET49(EN01 := <<BOOL>>, MSTR := <<BOOL>>, OK => <<BOOL>>);

This function block is used to reset the E-stop errors on digitizing axes 49 through 56 when the machine start input is pulsed.

This function block should be enabled every scan.

The machine start input must go through a positive transition (off to on) to reset the errors.

On a positive transition of MSTR, this function will reset all E-stop errors on axes 49 through 56.



Inputs: EN01 (BOOL) - enables execution

MSTR (BOOL) - machine start input

Outputs: OK (BOOL) - execution complete

<<INSTANCE NAME>>:M_RSET57(EN01 := <<BOOL>>, MSTR := <<BOOL>>, OK => <<BOOL>>);

This function block is used to reset the E-stop errors on digitizing axes 57 through 64 when the machine start input is pulsed.

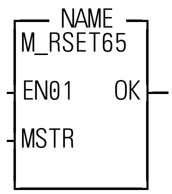
This function block should be enabled every scan.

The machine start input must go through a positive transition (off to on) to reset the errors.

On a positive transition of MSTR, this function will reset all E-stop errors on axes 57 through 64.

M_RSET65

Reset Errors on Digitizing Axes 65 to 72 **USER/M_INIT**



Inputs: EN01 (BOOL) - enables execution

MSTR (BOOL) - machine start input

Outputs: OK (BOOL) - execution complete

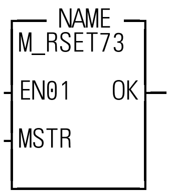
<<INSTANCE NAME>>:M_RSET65(EN01 := <<BOOL>>, MSTR := <<BOOL>>, OK => <<BOOL>>);

This function block is used to reset the E-stop errors on digitizing axes 65 through 72 when the machine start input is pulsed.

This function block should be enabled every scan.

The machine start input must go through a positive transition (off to on) to reset the errors.

On a positive transition of MSTR, this function will reset all E-stop errors on axes 65 through 72.



Inputs: EN01 (BOOL) - enables execution

MSTR (BOOL) - machine start input

Outputs: OK (BOOL) - execution complete

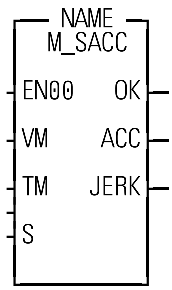
<<INSTANCE NAME>>:M_RSET73(EN01 := <<BOOL>>, MSTR := <<BOOL>>, OK => <<BOOL>>);

This function block is used to reset the E-stop errors on digitizing axes 73 through 80 when the machine start input is pulsed.

This function block should be enabled every scan.

The machine start input must go through a positive transition (off to on) to reset the errors.

On a positive transition of MSTR, this function will reset all E-stop errors on axes 73 through 80.



Inputs: EN00 (BOOL) - enables execution (one-shot)

VM (UDINT) - maximum velocity of move in ladder units/min

TM (REAL) - total time to reach velocity VM if axis starts from rest

S (USINT) - percentage of time spent in constant jerk

Outputs: OK (BOOL) - execution completed without error

ACC (LREAL) - maximum acceleration rate in ladder units or counts/min/sec

JERK (LREAL) - constant jerk in ladder units or counts/min/sec^2

```
<<INSTANCE NAME>>:M_SACC(EN00 := <<BOOL>>, VM :=
  <<UDINT>>, TM :=<<REAL>>, S := <<USINT>>, OK => <<BOOL>>),
  ACC => <<LREAL>>),  JERK => <<LREAL>>);
```

This function block is used to calculate the ACC and JERK values to be used with the ACC_JERK function.

Note: This function block is not intended to be used directly with the SCURVE or M_SCRVLC because the units for those functions are different (e.g ACC is Counts/min/min).

Inputs:

The EN00 input of this function block would normally be one-shot.

The VM input is set to the maximum velocity for the servo or time axis move to be executed.

The TM input sets the total time to reach velocity VM if the axis starts from rest. Typical values might be 0.1 seconds or 10 seconds. This value must be positive or the OK will not be set.

The S input sets the percentage of time spent in constant jerk. A value of 80(%) percent means 40% of the acceleration time spent in constant jerk, 20% in constant acceleration and another 40% in constant jerk. This value must be set to 100 or less or the OK will not be set.

Outputs:

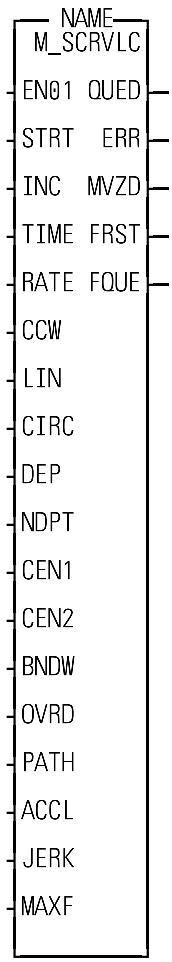
The OK is set if the input values are within range and the output values were calculated.

The ACC output is the maximum acceleration rate for the axis expressed in ladder units/min/sec for a servo axis and counts/min/sec for a time axis.

The JERK output is the constant jerk in ladder units/min/sec for a servo axis and counts/min/sec^2 for a time axis.

M_SCRVLC

Performs Linear and Circular Moves with S-Curve **USER/M_MOVE**



Inputs:

EN01 (BOOL) - enables execution

STRT (BOOL) - enables the coordinated move

INC (WORD) - defines incremental or absolute mode for up to 16 axes (0=absolute, 1=incremental)

TIME (BOOL) - defines if move is feedrate or time of move (0=feedrate, 1=time of move)

RATE (DINT) - feedrate or time of move

CCW (BOOL) - defines direction of circular move (0=clockwise, 1=counter-clockwise)

LIN (WORD) - defines which axes to move in a linear mode

CIRC (WORD) - defines which axes to move in a circular mode

DEP (WORD) - defines which axes to move in a simultaneous endpoint arrival mode

NDPT (DINT(0..16)) - endpoints or distances to move

CEN1 (DINT) - circle center for lowest numbered circular axis

CEN2 (DINT) - circle center for highest numbered circular axis

BNDW (DINT) - circular endpoint bandwidth

OVRD (USINT) - feedrate override percentage

PATH (USINT) - path number

ACCL (LREAL) - path acceleration in ladder units/min^2

JERK (LREAL) - path jerk in ladder units/min^3

MAXF (DINT) - maximum path feedrate in ladder units/minute

Outputs: QUED (BOOL) - move was queued without error

ERR (INT) - error number describing error that occurred when the move was queued

MVZD (BOOL) - linear move was queued with all distances being zero

FRST (USINT) - axis number of the first axis in the path

FQUE (USINT) - queue number for first axis in the path

```
<<INSTANCE NAME>>:M_SCRVLC(EN01 := <<BOOL>>, STRT :=
  <<BOOL>>, INC := <<WORD>>, TIME := <<BOOL>>, RATE :=
  <<DINT>>, CCW := <<BOOL>>, LIN  := <<WORD>>, CIRC :=
  <<WORD>>, DEP := <<WORD>>, NDPT := <<DINT (0..16)>>, CEN1 :=
  <<DINT>>, CEN2 := <<DINT>>, BNDW := <<DINT>>, OVRD :=
  <<USINT>>, PATH := <<USINT<<, ACCL := <<LREAL>>, JERK :=
  <<LREAL>>, MAXF := <<DINT>>, QUED => <<BOOL>>, ERR =>
  <<INT>>, MVZD => <<BOOL>>, FRST => <<USINT>>, FQUE =>
  <<USINT>>);
```

The M_SCRVLC function block provides the interface from the application .LDO to the RATIO_RL function in order to perform linear coordinated, circular, or third axis departure (simultaneous endpoint arrival) moves with S-curve acceleration and deceleration. Before this function can be used, the axes must be initialized, the position loop must be closed, and a queue must be available on all axes to be used in the move.

Up to four separate paths of coordinated motion can be controlled. Each path of motion requires a separate instantiation of the M_SCRVLC function block. Each path must control a unique set of axes. Only one M_SCRVLC function block per path can be used with the application .LDO. Each path uses a time axis. Path 1 uses time axis 25. Path 2 uses time axis 26. Path 3 uses time axis 27. Path 4 uses time axis 28.

This function block can control up to 16 axes.

Note:This function block requires a numeric processor or a 486 DX processor in the PiC900 and version 6.2 or higher of PiCPro.

Inputs

The EN01 input of this function block must be set every scan.

The STRT input must be one-shot. When it is one-shot, the function block will start the coordinated move, or enter it in the queue for the axes. It is the user's responsibility to ensure that there is a queue available on all of the axes involved in the move before pulsing this input.

The INC input defines whether each axis should move in the absolute or incremental mode. One bit of this WORD is reserved for each of the sixteen possible axes. Bit 0 is set if axis 1 is incremental, or reset if axis 1 is absolute, bit 1 is set if axis 2 is incremental, reset if axis 2 is absolute, etc.

The TIME input defines whether the move should be executed as a path feedrate move or a time of move. This input should be reset for path feedrate, or set for time of move.

If the TIME input is reset, then the RATE input is the path feedrate for the move in ladder units/minute. It the TIME input is set, then the RATE input is the time for the move in milliseconds.

The RATE is the path feedrate or the time for the move to execute depending on the TIME input.

The CCW input is only used for circular moves. If it is reset, then the move is clockwise, if it is set, then the move is counter-clockwise.

The LIN input defines which axes in the move are to be moved in a linear mode. One bit of the WORD is reserved for each of the sixteen axes. The bit must be set for the axis to do a linear move. Axes who have their bits set will be included in the calculations for the path feedrate.

The CIRC input defines which axes in the move are to be moved in a circular mode. One bit of the WORD is reserved for each of the sixteen axes. The bit must be set for the axis to do a circular move. Axes who have their bits set will be included in the calculations for the path feedrate.

The DEP input defines which axes in the move are to be moved in a simultaneous endpoint arrival mode. One bit of the WORD is reserved for each of the sixteen axes. The bit must be set for the axis to move. Axes who have their bits set will not be included in the calculations for the path feedrate, but they will arrive at their endpoints simultaneously with the axes that are.

Note:The LIN, CIRC, and DEP words may never have the same bits set in them at a time. You must always set a bit for every axis ever used in the path, even if the axis is not to move in this particular move. In this case, you would set either the LIN or DEP bit for the axis, set the INC bit for the axis, and program an endpoint of zero for the axis.

The NDPT array holds the endpoints for the axes used in the move. The 0th element is not used. If the INC bit is set for the axis, this is the distance to move, if the INC bit is reset for the axis, then this is the position to move to. The endpoints are entered in ladder units.

The CEN1 and CEN2 inputs define the circle centers if a circular move is being performed. The CEN1 input is the center for the lowest numbered circular axis, and the CEN2 input is the center for the highest numbered circular axis. The centers are always programmed as an incremental distance from the starting point of the circle, even if the INC bit for the axes is not set. The centers are entered in ladder units. For example, if a circle were being done with axes 4 and 6, then CEN1 would be the center for axis 4, and CEN2 would be the center for axis 6.

The BNDW input defines a bandwidth for circular moves. When a circular move is requested, the distance from the start point to the center point and the distance from the endpoint to the center point are compared for both axes. If these distances differ by more than the bandwidth entered here, then the move will not execute and error 14 will be returned on the ERR output. This bandwidth is entered in ladder units.

The OVRD input defines the feedrate override value. This can be changed at any time, even if the STRT input is not energized. This adjusts the actual feedrate or time to be from 0 to 255 percent of the programmed feedrate or time.

The PATH input defines the number of the path. Up to four totally independent paths of coordinated motion can be defined. This must be a number from 1 to 4. This should not be changed once it is set.

The ACCL is the path acceleration in ladder units/min^2. The JERK is the path jerk in ladder units/min^3. The MAXF is the maximum path feedrate in ladder units/min. This should not be changed once it is set.

Outputs

The QUED output will be set for one scan when STRT is pulsed and the move has been successfully queued on all axes defined. If an error occurred in queueing the move, this output will be reset when STRT is pulsed, and an error code will be stored in the ERR output.

The ERR output will be non-zero if an error occurs in queueing a move. A list of error codes is shown below:

ERR	Description
0	No error
1	No bits were set in the LIN, CIRC, or DEP WORDs
2	The same bit was set in the LIN and CIRC WORDs
3	The same bit was set in the DEP and CIRC WORDs
4	The same bit was set in the LIN and DEP WORDs
5	The number of bits set in the CIRC WORD was not 0 or 2
6	Not used
7	Not used
8	The time of move or feedrate was negative
9	The time of move or feedrate was zero
10	The feedrate was too high or the time was too low to calculate

11	The feedrate was too low or the time was too high to calculate
12	An axis that was selected was not initialized by the servo setup function
13	The STRTSRV function has not been called
14	Endpoint not on circle
1XX	When the distance to move was converted to feedback units, it was too positive to fit into 32 bits. XX = Axis number
2XX	When the distance to move was converted to feedback units, it was too negative to fit into 32 bits. XX = Axis number
3XX	The path feedrate or time entered causes an axis to exceed its velocity limit from servo setup. XX = Axis number
32766	The time axis could not be started. NOTE: For PiCPro 13.0 and higher you need to add the time axis to your servosetup.
32767	One of the OKs on the RATIO_RL functions did not get set or the OK on the time axis distance move did not get set. NOTE: For PiCPro 13.0 and higher you need to add the time axis to your servosetup.

Calculating ACCL and JERK

This section explains how to calculate the ACCL and JERK inputs for the function block.

The drawing below illustrates an S-curve acceleration.

V_m = Maximum path velocity

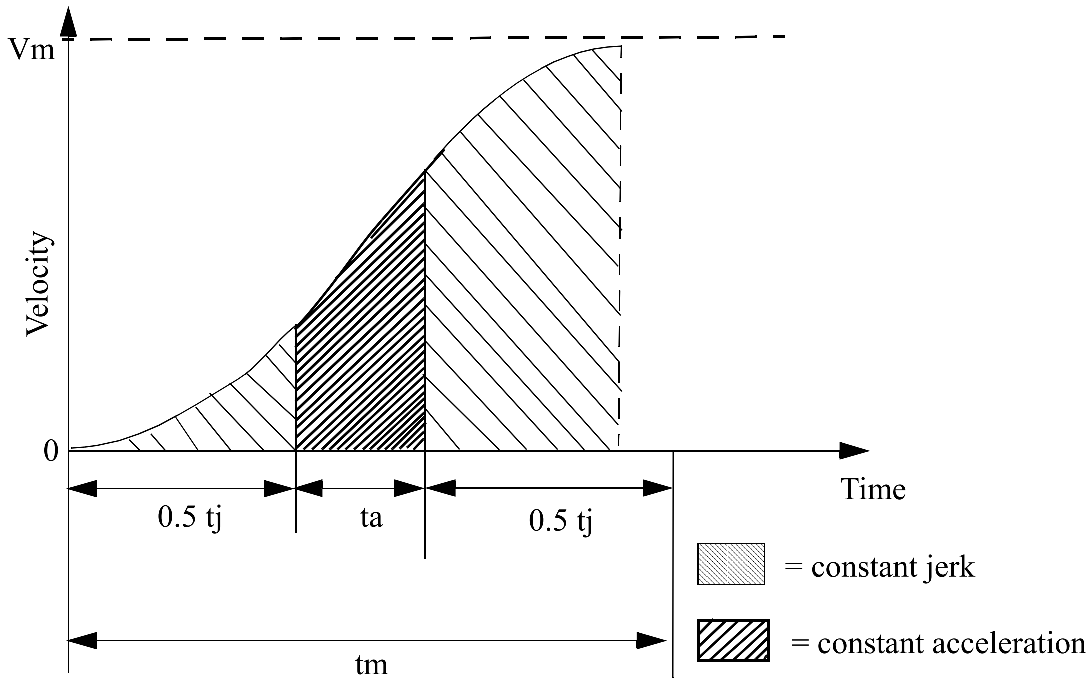
t_m = The total time it takes to get to velocity V_m if the axis starts at 0.

s = The percentage of time (t_m) spent in constant jerk.

From 0 to t_1, the axis will be in constant jerk
From t_1 to t_2, the axis will be in constant acceleration.
From t_2 to t_m, the axis will again be in constant jerk.

The formulas below show the relationship between t_m, t_1, t_2, and s.

$$t_1 = t_m - t_2 = \left(\frac{1}{2} \times s \times t_m\right)$$



$$t_2 = t_m - \left(\frac{1}{2} \times s \times t_m\right)$$

For a 10% S-curve, 10% of the time (t_m) is spent in constant jerk.
This means that s = 0.1.
For a 20% S-curve, 20% of the time (tm) is spent in constant jerk.
This means that s = 0.2, etc.

If you know V_m, t_m, and s, then you can calculate jerk and acceleration using the following formulas.

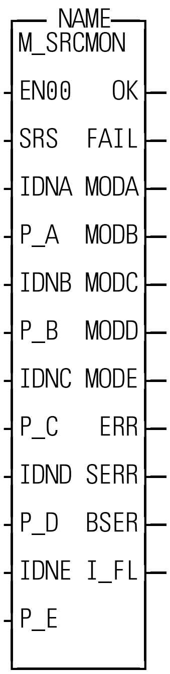
$$JERK = \frac{2 \times Vm}{s \times t_m{}^2(1 - 0.5 \times s)}$$

$$ACCL = \frac{Vm}{t_m \times (1 - 0.5 \times s)}$$

The units for JERK are ladder units per minute^3; therefore, V_m is in ladder units per minute and t_m is in minutes. The units for ACCL are ladder units per minute^2.

M_SRCMON

Monitors up to five SERCOS IDNs **USER/M_SERCOS**



Inputs:

EN00 (BOOL) - enables execution

SRS (STRUCT) - slot, ring, and slave to monitor

IDNA (UINT) - number of first IDN to monitor

P_A (BOOL) - set for Product IDN, reset for System IDN

IDNB (UINT) - number of second IDN to monitor

P_B (BOOL) - set for Product IDN, reset for System IDN

IDNC (UINT) - number of third IDN to monitor

P_C (BOOL) - set for Product IDN, reset for System IDN

IDND (UINT) - number of fourth IDN to monitor

P_D (BOOL) - set for Product IDN, reset for System IDN

IDNE (UINT) - number of fifth IDN to monitor

P_E (BOOL) - set for Product IDN, reset for System IDN

Outputs:

OK (BOOL) - execution complete

FAIL (BOOL) - execution failure

MODA (REAL) - value of first IDN

MODB (REAL) - value of second IDN

MODC (REAL) - value of third IDN

MODD (REAL) - value of fourth IDN

MODE (REAL) - value of fifth IDN

ERR (INT) - SERCOS error*

SERR (UINT) - SERCOS slave error*

BSER (INT) - SERCOS block specific error*

I_FL (UINT) - indicates the IDN that failed (1 through 5 corresponding to A through E) if an error occurs during a read

*See error tables at end of the M_SRCWTL function block section.

```
<<INSTANCE NAME>>:M_SRCMON(EN00 := <<BOOL>>, SRS :=
  <<MEMORY AREA>>, IDNA := <<UINT>>, P_A := <<BOOL>>, IDNB
  := <<UINT>>, P_B := <<BOOL>>, IDNC  := <<UINT>>, P_C :=
  <<BOOL>>, IDND := <<UINT>>, P_D := <<BOOL>>, IDNE :=
  <<UINT>>, P_E := <<BOOL>>, OK => <<BOOL>>, FAIL =>
  <<BOOL>>, MODA => <<REAL>>, MODB => <<REAL>>, MODC =>
  <<REAL>>, MODD => <<REAL>>, MODE => <<REAL>>, ERR =>
  <<INT>>, SERR => <<UINT>>, BSER => <<INT>>, I_FL =>
  <<UINT>>);
```

The M_SRCMON function block monitors up to five SERCOS IDNs for a single SERCOS slave. The operation data for each IDN is continuously read as long as the EN00 input is energized.

The IDNA through IDNE inputs can be used or left blank. When the EN00 input transitions from off to on, the attributes of each IDN are read and saved in the function block. These attributes are used to scale the data being monitored into engineering units for the output. If the IDNA through IDNE inputs are changed while monitoring, the EN00 input must be dropped and then re-energized so that the attributes for each IDN are read again.

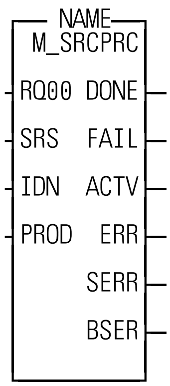
The SRS input is used to indicate which SERCOS slave to monitor. Slot, ring, and slave are used instead of an axis number so that this function block can be used in phase 2 initialization if desired. The SRS structure must be declared as follows:

Name	**Data Type**	**Definition**
SRS	STRUCT	
.SLOT	UINT	Slot number of the SERCOS module
.RING	UINT	Ring number on the module
.SLAVE	UINT	Slave number on the ring
	END_STRUCT	

If FAIL is set, ERR or BSER will be non-zero indicating the type of error. If ERR = 128 indicating Slave Error, SERR will be non-zero indicating the type of slave error.

M_SRCPRC

Executes SERCOS procedure command function **USER/M_SERCOS**



Inputs: RQ00 (BOOL) - requests execution of a procedure command function (one-shot)

SRS (STRUCT) - slot, ring, and slave number of the SERCOS slave to execute the procedure command function

IDN (UINT) - IDN number of procedure command function

PROD (BOOL) - set for Product IDN, reset for System IDN

Outputs: DONE (BOOL) - procedure command function complete

FAIL (BOOL) - procedure command function failure

ACTV (BOOL) - set while the procedure command function is active

ERR (INT) - SERCOS error*

SERR (UINT) - SERCOS slave error*

BSER (INT) - SERCOS block specific error*

*See error tables at end of the M_SRCWTL function block section.

<<INSTANCE NAME>>:M_SRCPRC(RQ00 := <<BOOL>>, SRS := <<MEMORY AREA>>, IDN := <<UINT>>, PROD := <<BOOL>>, DONE => <<BOOL>>, ACTV => <<BOOL>>, ERR => <<INT>>, SERR => <<UINT>>, BSER => <<INT>>);

The M_SRCPRC function block executes a SERCOS procedure command function for a single SERCOS slave. The RQ00 input of this function block should be one-shot to initiate the procedure command function. While the procedure command function is executing within the SERCOS slave, the ACTV output will be set. If the procedure command function completes without error, the DONE output will be set and the ACTV output will be reset. If the procedure command function fails, the FAIL output will be set and the ACTV output will be reset. The DONE or FAIL output will remain set until the RQ00 input is one-shot again.

The SRS input is used to indicate which SERCOS slave is to execute the procedure command function. Slot, ring, and slave are used instead of an axis number so that this function can be used in phase 2 initialization if desired. The SRS structure must be declared as shown in the following table:

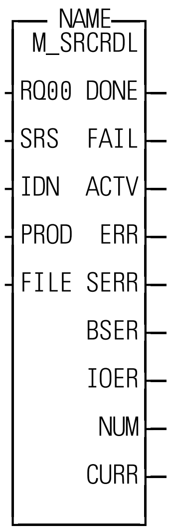
Name	Data Type	Definition
SRS	STRUCT	
.SLOT	UINT	Slot number of the SERCOS module
.RING	UINT	Ring number on the module
.SLAVE	UINT	Slave number on the ring
	END_STRUCT	

If FAIL is set, ERR or BSER will be non-zero indicating the type of error.

If ERR = 128 indicating slave error, SERR will be non-zero indicating the type of slave error.

M_SRCRDL

Reads SERCOS IDNs **USER/M_SERCOS**



Inputs: RQ00 (BOOL) - requests execution (one-shot)

SRS (STRUCT) - slot, ring, and slave number

IDN (UINT) - IDN number that will return a list of IDNs

PROD (BOOL) - set for Product IDN, reset for System IDN

FILE (STRING [80]) - filename of the file to save to

Outputs: DONE (BOOL) - execution complete

FAIL (BOOL) - execution failed

ACTV (BOOL) - set while executing

ERR (INT) - SERCOS error*

SERR (UINT) - SERCOS slave error*

BSER (INT) - SERCOS block specific error*

IOER (INT) - I/O function block error (See Appendix B in the online help.)

NUM (UINT) - number of IDNs in the list

CURR (UINT) - current member being read

*See error tables at end of the M_SRCWTL function block section.

<<INSTANCE NAME>>:M_SRCRDL(RQ00 := <<BOOL>>, SRS := <<MEMORY AREA>>, IDN := <<UINT>>, PROD := <<BOOL>>, FILE := <<STRING>>, DONE => <<BOOL>>, FAIL => <<BOOL>>, ACTV => <<BOOL>>, ERR => <<INT>>, SERR => <<UINT>> BSER => <<INT>>, IOER => <<INT>>, NUM => <<UINT>>, CURR => <<UINT>>);

The M_SRCRDL function block reads a list of up to 400 IDNs and saves the list to the PiC RAMDISK or workstation as an ASCII file along with the name, units, and operation data limits for each IDN in the list. Each IDN appears in a single line in the file. The data for each IDN is separated by tabs. This function block can be used in conjunction with M_SRCWTL to read and write lists of IDNs to and from a SERCOS slave.

The IDN number specified with the IDN and PROD inputs must return a list of IDNs in order to use this function block.

The RQ00 input must be one-shot. While the function block is reading the list of IDNs, the ACTV output will be set. If the read completes without error, the DONE output will be set and the ACTV output will be reset. If an error occurs during reading, the FAIL output will be set and the ACTV output will be reset. The DONE or FAIL output will remain set until the RQ00 input is one-shot again.

The NUM output indicates the total number of IDNs that exist in the list being read. The CURR output indicates the current member of the list being read and will range from 0 to NUM.

The SRS input is used to indicate from which SERCOS slave the list of IDNs will be read. Slot, ring, and slave are used instead of an axis number so that this function block can be used in phase 2 initialization if desired. The SRS structure must be declared as follows:

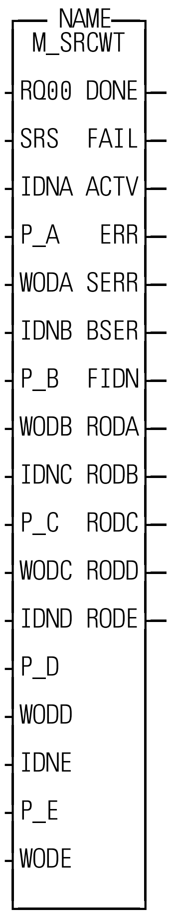
Name	**Data Type**	**Definition**
SRS	STRUCT	
.SLOT	UINT	Slot number of the SERCOS module
.RING	UINT	Ring number on the module
.SLAVE	UINT	Slave number on the ring
	END_STRUCT	

FILE is a string containing the full file specification of the file in which the list of IDNs is saved. This string must be terminated by the null character $00, (i.e. RAMDISK:\IDNFILE.DAT$00).

If FAIL is set ERR, BSER, or IOER will be non-zero indicating the type of error. If ERR = 128 indicating Slave Error, SERR will be non-zero indicating the type of slave error.

M_SRCWT

Writes and reads SERCOS IDNs **USER/M_SERCOS**



Inputs:

RQ00 (BOOL) - requests execution (one-shot)

SRS (STRUCT) - slot, ring, and slave number

IDNA (UINT) - number of first IDN to write

P_A (BOOL) - set for Product IDN, reset for System IDN

WODA (REAL) - value of operation datum for IDNA

IDNB (UINT) - number of second IDN to write

P_B (BOOL) - set for Product IDN, reset for System IDN

WODB (REAL) - value of operation datum for IDNB

IDNC (UINT) - number of third IDN to write

P_C (BOOL) - set for Product IDN, reset for System IDN

WODC (REAL) - value of operation datum for IDNC

IDND (UINT) - number of fourth IDN to write

P_D (BOOL) - set for Product IDN, reset for System IDN

WODD (REAL) - value of operation datum for IDND

IDNE (UINT) - number of fifth IDN to write

P_E (BOOL) - set for Product IDN, reset for System IDN

WODE (REAL) - value of operation datum for IDNE

Outputs:

DONE (BOOL) - set when the writes and reads are complete

FAIL (BOOL) - set if write or read fails

ACTV (BOOL) - set when operation is in process

ERR (INT) - SERCOS error*

SERR (UINT) - SERCOS slave error*

BSER (INT) - SERCOS block specific error*

FIDN (UINT) - the IDN the operation failed on

RODA (REAL) - value of operation datum read back from IDNA

RODB (REAL) - value of operation datum read back from IDNB

RODC (REAL) - value of operation datum read back from IDNC

RODD (REAL) - value of operation datum read back from IDND

RODE (REAL) - value of operation datum read back from IDNE

*See error tables at end of M_SRCWTL function block section.

```
<<INSTANCE NAME>>:M_SRCWT(RQ00 := <<BOOL>>, SRS :=
  <<MEMORY AREA>>, IDNA := <<UINT>>, P_A := <<BOOL>>, WODA
  := <<REAL>>, IDNB := <<UINT>>, P_B := <<BOOL>>, WODB :=
  <<REAL>>, IDNC := <<UINT>>, P_C := <<BOOL>> WODC :=
  <<REAL>>, IDND := <<UINT>>, P_D := <<BOOL>>, WODD :=
  <<REAL>>, IDNE := <<UINT>>, P_E := <<BOOL>>, WODE :=
  <<REAL>>, DONE => <<BOOL>>, FAIL => <<BOOL>>, ACTV =>
  <<BOOL>>, ERR => <<INT>>, SERR => <<UINT>> BSER => <<INT>>,
  FIDN => <<UINT>>, RODA => <<REAL>>, RODB => <<REAL>>,
  RODC => <<REAL>>, RODD => <<REAL>>, RODE => <<REAL>>);
```

The M_SRCWT function block writes and reads up to five SERCOS IDNs.

The M_SRCWT function block will write and read back operation data to a maximum of five IDNs on a SERCOS slave. The operation data for each IDN is written and read once when the RQ00 input is energized.

The IDNA through IDNE inputs can be used or left blank. When the RQ00 input transitions from off to on, the attributes of each IDN are read and saved in the function block. These attributes are used to scale the data at the input to the correct units for the SERCOS slave. After the attributes are read the operation data is written and read back again to verify that the write was successful. While this process is happening, the ACTV output will remain set. If the process completes without error, the DONE output will be set and the ACTV output will be reset. If an error occurs, the FAIL output will be set and the ACTV output will be reset.

The RQ00 input must be one-shot each time you wish to write data to the SERCOS slave. A second request cannot be made while the first one is still active. If this happens, the second request will be ignored.

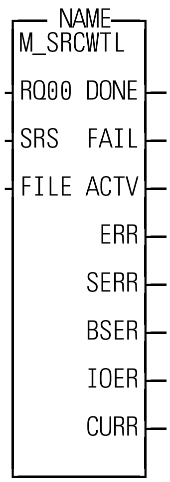
The SRS input is used to indicate which SERCOS slave to write to. Slot, ring, and slave are used instead of an axis number so that this function can be used in phase 2 initialization if desired. The SRS structure must be declared as follows:

Name	**Data Type**	**Definition**
SRS	STRUCT	
.SLOT	UINT	Slot number of the SERCOS module
.RING	UINT	Ring number on the module
.SLAVE	UINT	Slave number on the ring
	END_STRUCT	

If FAIL is set, ERR or BSER will be non-zero indicating the type of error. If ERR = 128 indicating Slave Error, SERR will be non-zero indicating the type of slave error.

M_SRCWTL

Writes SERCOS IDNs **USER/M_SERCOS**



Inputs: RQ00 (BOOL) - requests execution (one-shot)

SRS (STRUCT) - slot, ring, and slave number

FILE (STRING [80]) - filename

Outputs: DONE (BOOL) - execution complete

FAIL (BOOL) - execution failed

ACTV (BOOL) - execution active

ERR (INT) - SERCOS error*

SERR (UINT) - SERCOS slave error*

BSER (INT) - SERCOS block specific error*

IOER (INT) - I/O function block error (See Appendix B in the online help.)

CURR (UINT) - current IDN being written

*See error tables at end of the this function block section.

<<INSTANCE NAME>>:M_SRCWTL(RQ00 := <<BOOL>>, SRS := <<MEMORY AREA>>, FILE := <<STRING>>, DONE => <<BOOL>>, FAIL => <<BOOL>>, ACTV => <<BOOL>>, ERR => <<INT>>, SERR => <<UINT>> BSER => <<INT>>, IOER => <<INT>>, CURR => <<UINT>>);

The M_SRCWTL function block writes a list of SERCOS IDNs.

The M_SRCWTL function block reads a list of IDNs from an ASCII file on the PiC RAMDISK or workstation and writes the operation data from the list to a SERCOS slave. The ASCII file must be of the same format used for the M_SRCRDL function block. M_SRCWTL can be used in conjunction with M_SRCRDL to read and write lists of IDNs to and from a SERCOS slave.

The RQ00 input to this function must be one-shot. While the function block is writing the list of IDNs, the ACTV output will be set. If the write completes without error, the DONE output will be set and the ACTV output will be reset. If an error occurs during the write, the FAIL output will be set and the ACTV output will be reset. The DONE or FAIL output will remain set until the RQ00 input is one-shot again.

The CURR output indicates the current IDN being written to the SERCOS slave. This will continually update while the function block is active.

The SRS input is used to indicate which SERCOS slave the list of IDNs will be written to. Slot, ring, and slave are used instead of an axis number so that this function can be used in phase 2 initialization if desired. The SRS structure must be declared as shown in the following table:

Name	Data Type	Definition
SRS	STRUCT	
.SLOT	UINT	Slot number of the SERCOS module
.RING	UINT	Ring number on the module
.SLAVE	UINT	Slave number on the ring
	END_STRUCT	

FILE is a string containing the full file specification of the file in which the list of IDNs is saved. This string must be terminated by the null character $00 (i.e. RAMDISK:\IDNFILE.DAT$00).

If FAIL is set ERR, BSER, or IOER will be non-zero indicating the type of error. If ERR = 128 indicating Slave Error, SERR will be non-zero indicating the type of slave error.

M_SERCOS Function Block Errors

There are three types of error outputs that can appear on the M_*SERCOS* function blocks. They are described in the three tables that follow.

ERR Output

Table 1 contains the list of SERCOS errors that can appear at the ERR output of the M_*SERCOS* function blocks.

Table 1 - List of ERR Codes

Err #	Description
0	No error
1	IDN queue was busy when called.
2	Quantity specified in the .AVAIL structure member is not large enough for received data.
3	Axis is not initialized, is not a SERCOS axis, or the slot/ring/slave specification is incorrect.
4	Invalid data in DATA input structure
5	Error reset function could not be completed.
6	SERCOS ring 1 busy*
7	SERCOS ring 2 busy*
8	SERCOS ring 1 configuration size error**
9	SERCOS ring 2 configuration size error**
10	Function block enabled while already in process
11	Bit 3 or bit 8 set in the procedure command acknowledgment (data status) Either operation data invalid or procedure command error
12	Not enough pool memory available
13	Change bit in status word was zero after reference complete.
14	The IDN queue was cleared during an IDN transfer, typically caused by calling the SC_INIT function while an IDN is being read or written.
15	SERCOS module is unavailable for IDN transfer because the phase-to-phase transition in progress is between phase 2 and phase 4.
16	Slave response timed out
17	The SERCOS module did not receive an expected AT response. SERCOS cable may be disconnected.
18	Number of SERCOS slots equals zero.
19	The SERCOS module did not receive an expected MDT response. SERCOS cable may be disconnected.
20	Phase 0 detected that the ring is not complete. The optic cable could be open or drive turned off.
21	The SERCOS module firmware is outdated for the features requested from a newer version of the motion library.
22	The SERCOS module firmware is a newer version and the motion library is outdated and unable to interface.
23	The data (user function) is outdated for the features requested from the library or the SERCOS module firmware.
24	The data is a newer version and the library is unable to interface.
25	A two-ring SERCOS module was specified in SERCOS setup but the module is a one-ring SERCOS module.
30	The drive status word (bit 13=1) indicates an error.

31	An E-stop condition exists for this axis in the PiC900.
32	Incorrect phase number, contact Danaher Motion
33	Incorrect address error, contact Danaher Motion
34	Incorrect AT number error, contact Danaher Motion
35	Variable 48 is set to 1 and you attempt to close the loop
36	OPTN input is invalid.
48	Service channel not ready when attempt to send/receive non-cyclic data
49	No data to send or receive
50	The value of the .SIZE member of the TASK input structure does not match the byte count in the SERCOS module.
51	The value of the .SIZE member of the MAIN input structure does not match the byte count in the SERCOS module.
65	Error occurred calculating when MDT should occur.
66	Error occurred calculating when drive data valid.
67	Error occurred calculating when feedback data valid.
68	Error occurred calculating total time required for communication cycle.
69	Error occurred calculating cyclic data memory for SERCON processor.
70	Error occurred calculating cyclic data memory for internal memory map.
71	Error occurred calculating service channel memory map.
72	Incorrect ring error, contact Danaher Motion
73	Incorrect AT count error, contact Danaher Motion
74	CPU on SERCOS module has too many tasks during update.
128	Slave error occurred. Read SERR output to identify error. The SLV output indicates the slave number.
136	Slave will not respond in phase 1. The SLV output indicates the slave number.
144	Procedure command error - The slave number can be viewed at the SLV output and the IDN number at the IDN output.

*This busy error may occur if the SC_INIT function is not one-shotted and a second store operation is attempted before the first one is done.

**This size error will occur if too many IDNs are defined in the SERCOS setup data.

SERR Output

Table 2 contains the list of slave errors that can appear at the SERR output of M_*SERCOS* function blocks.

Table 2 - List of SERR Error Codes

SERR #	Description
0	No error
4097	This IDN does not exist.
4105	The data for this IDN may not be accessed.
8193	The name does not exist
8194	The name transmission is too short
8195	The name transmission is too long
8196	The name may not be changed
8197	The name is write-protected
12290	The attribute transmission is too short
12291	The attribute transmission is too long
12292	The attribute may not be changed
12293	The attribute is write-protected at this time
16385	The units do not exist
16386	The units transmission is too short
16387	The units transmission is too long
16388	The units may not be changed
16389	The units are write-protected at this time
20481	The minimum value does not exist
20482	The minimum value transmission is too short
20483	The minimum value transmission is too long
20484	The minimum value may not be changed
20485	The minimum value is write-protected
24577	The maximum value does not exist
24578	The maximum value transmission is too short
24579	The maximum value transmission is too long
24580	The maximum value may not be changed
24581	The maximum value is write-protected
28674	The data is too short.
28675	The data is too long
28676	The data may not be changed.
28677	The data is write-protected at this time.
28678	The data is smaller than the minimum value.
28679	The data is larger than the maximum value.
28680	The bit pattern for this IDN is invalid.

BSER Output

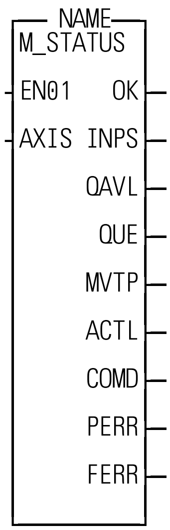
Table 3 contains the list of block specific errors that can appear at the BSER output of M_*SERCOS* function blocks.

Table 3 - Block Specific Error Codes

BSER #	Description
0	No error
1	Request to execute but not in phase 2 or 4
2	IDN is a procedure command
3	Data is variable length
4	Data is reserved
5	IDN is not a procedure command

M_STATUS

Return Axis Data **USER/M_DATA**



Inputs: EN01 (BOOL) - enables execution

AXIS (USINT) - axis number

Outputs: OK (BOOL) - execution completed without error

INPS (BOOL) - set when axis is in position

QAVL (BOOL) - set when next queue is empty

QUE (USINT) - queue number of move in active queue

MVTP (DINT) - type of move in active queue

ACTL (DINT) - actual position of axis in ladder units

COMD (DINT) - commanded position of axis in ladder units

PERR (DINT) - position error of axis in ladder units

FERR (DINT) - filter error of axis in ladder units

<<INSTANCE NAME>>:M_STATUS(EN01 := <<BOOL>>, AXIS := <<USINT>>, OK => <<BOOL>>, INPS => <<BOOL>>, QAVL => <<BOOL>>, QUE => <<USINT>>, MVTP => <<DINT>> ACTL => <<DINT>>, COMD => <<DINT>>, PERR => <<DINT>>, FERR => <<DINT>>);

This function block obtains information for a digitizing, time, or closed loop axis. It returns the in position flag, the queue available flag, the active queue number, the active move type, the actual position, the commanded position, the position error, and the filter error for the axis.This function block should be enabled every scan.

The input at AXIS determines which axis the output information is for.

The INPS output is set whenever the following error of the axis is within the in position limit entered in servo setup. It will be reset while the axis is in motion.

The QAVL output is set whenever the next queue or both the next and active queues are empty. When set it means another move can be put in the axis queue.

The QUE output holds the queue number of the move in the active queue. The queue number is assigned to each move when the move function is enabled. If no moves are active, the QUE number will be 0.

The MVTP output holds the type of the move in the active queue. If no move is active, this will be 0. The moves types are defined below:

MVTP	Description
11	POSITION
12	DISTANCE
14	VEL_STRT
16	FAST_REF or LAD_REF
18	RATIOPRO
20	RATIOSYN or RATIO_GR
22	RATIOCAM
23	RATIOSLP
24	RATIO_RL

The ACTL output holds the actual position of the axis in ladder units.

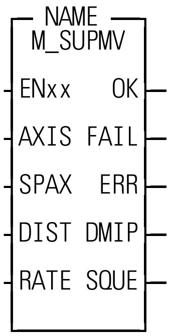
The COMD output holds the commanded position of the axis in ladder units.

The PERR output holds the position error of the axis in ladder units.

The FERR output holds the filter error of the axis in ladder units.

This function block can be used for a digitizing axis, a time axis, or a closed loop axis. If used for a digitizing axis only the ACTL and COMD outputs are used and there is no need to enter variables for the INPS, QAVL, QUE, MVTP, PERR, or FERR outputs. If used for a time axis, only the ACTL output is used and there is no need to enter variables for the INPS, QAVL, QUE, MVTP, COMD, PERR, and FERR outputs.

The OK output will not be set if the axis has not been initialized.



Inputs: ENxx (BOOL) - One Shot to start the motion

AXIS (USINT) - Servo axis on which move will be performed.

SPAX (USINT) - Virtual axis used to effect superimposed move.

DIST (DINT) - Distance of superimposed move in LU

RATE (UDINT) - Rate of superimposed move in LU / minute

Outputs: OK (BOOL) - Function block operation OK

FAIL (BOOL) - Command failed - see ERR for reasons

ERR (USINT) - Error number showing reason for FAIL

DMIP (BOOL) - Superimposed move in progress

SQUE (USINT) - Superimposed move queue number

```
<<INSTANCE NAME>>:M_SUPMV(ENxx := <<BOOL>>, AXIS :=
  <<USINT>>, SPAX := <<USINT>>, DIST := <<DINT>>, RATE :=
  <<UDINT>>, OK => <<BOOL>>, FAIL => <<BOOL>>, ERR =>
  <<USINT>>, DMIP => <<BOOL>>, SQUE => <<USINT>>);
```

This block will perform an incremental (Distance) move on a servo axis that already has an active move in progress. It can be used to cause a phase shift between master and slave. The shift may be determined by a registration eye or other means. The move can cause motion reversal. It is up to the programmer to determine that the effect of the move is acceptable in terms of speed and direction.

The active move must be a Ratio_xxxx move type or a Velocity move type. See the Help for READ_SV (WRITE_SV) variable 66 for more information. The block requires the use of a Virtual axis and internally commands this Virtual axis to move. The use of VAR66 causes the Virtual axis move to be added on top of the servo axis motion. Make sure that the Virtual axis has the same servo setup parameters as the servo axis, including interrupt rate. The acceleration and deceleration used will be those of the virtual axis. Also, make sure to close the loop on the virtual axis, Estop it, reset faults etc exactly the same as the servo axis.

INPUTS:

ENxx - One shot to start motion

AXIS - Servo (Ratio slave) axis number

SPAX - Virtual axis used to effect superimposed move

DIST - Distance and sign of move in LU

RATE - Velocity of move in LU / minute

OUTPUTS:

OK - block executed ok

FAIL - an error occurred - see ERR for causes

ERR - error code indicating reason for FAIL - see below for detail

DMIP - superimposed move is in progress

SQUE - queue number of superimposed move

ERRORS:

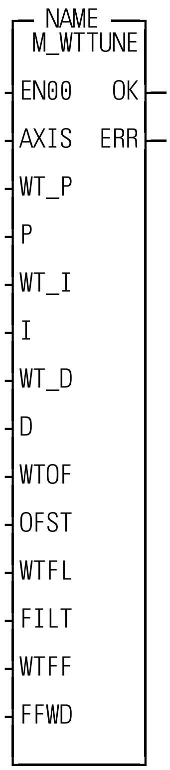
1 - The virtual axis loop is not closed.

2 - The virtual axis was already in motion when ENxx was pulsed.

3 - The servo axis would not accept the VAR66 write. Servo and virtual interrupt rates may be different.

4 - The servo axis does not have a valid move type active - RATIO_xxxx or Velocity.

10 - The axis motion was interrupted by an Estop or other event.



Inputs: EN00 (BOOL) - enables execution

AXIS (USINT) - identifies axis

WT_P (BOOL) - enables write of proportional gain

P (DINT) - proportional gain

WT_I (BOOL) - enables write of integral gain

I (DINT) - integral gain

WT_D (BOOL) - enables write of derivative gain

D (DINT) - derivative gain

WTOF (BOOL) - enables write of analog output offset

OFST (DINT) - analog output offset

WTFL (BOOL) - enables write of slow speed filter value

FILT (DINT) - slow speed filter value

WTFF (BOOL) - enables write of feedforward percentage

FFWD (DINT) - feedforward percentage

Outputs: OK (BOOL) - execution complete

ERR (INT) - error number

<<INSTANCE NAME>>:M_WTTUNE(EN00 := <<BOOL>>, AXIS := <<USINT>>, WT_P := <<BOOL>>, P := <<DINT>>, WT_1 := <<BOOL>>, I := <<DINT>>, WT_D := <<BOOL>> D := <<DINT>>, WTOF := <<BOOL>>, OFST := <<DINT>>, WTFL := <<BOOL>>, FILT := <<DINT>>, WTFF := <<BOOL>>, FFWD := <<DINT>>, OK => <<BOOL>>, ERR => <<INT>>);

This function block allows you to write all six tuning parameters from the TUNE-WRIT function in a single function.

This function block requires the numeric processor or a 486 DX processor.

The EN00 input of this function should be set every scan.

The AXIS input identifies which axis to write data to. It must be between 1 and 16 or between 101 and 116, inclusive.

Note: LU = ladder units, MIN = minutes, LUFE = ladder units of following error.

When the WT_P input is set, the proportional gain of AXIS will be changed to the value entered at P. P is in LU / MIN / LUFE and must be between 0 and 20000.

When the WT_I input is set, the integral gain of AXIS will be changed to the value entered at I. I is in LU / MIN / LUFE * MIN. I must be from 0 to 32000.

When the WT_D input is set, the derivative gain of AXIS will be changed to the value entered at D. D is in LU / MIN / LUFE / MIN.

When the WTOF input is set, the analog output offset voltage of AXIS will be changed to the value entered at OFST. OFST must be from -10000 to +10000 millivolts.

When the WTFL input is set, the slow speed filter value of AXIS will be changed to the value entered at FILT. FILT must be from 0 to 10000 milliseconds.

When the WTFF input is set, the feedforward percentage of AXIS will be changed to the value entered at FFWD. FFWD must be from 0 to 100%.

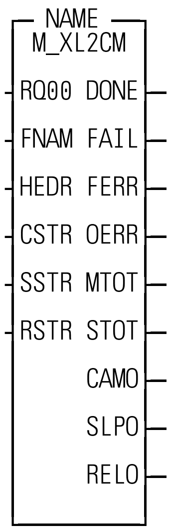
The WT_P, WT_I, WT_D, WTOF, WTFL and WTFF inputs can be one-shot. The parameters will remain changed until the axis is re-initialized or until this function block or the TUNEWRIT function is called again for AXIS.

The OK output will be set if the function executes without error. If an error occurs, OK will not be set and ERR will hold a number describing the error that occurred. A listing of errors is shown below:

ERR	Description
0	No error
1	Tried to change P for AXIS number that was not initialized or is out of range
3	Data for P is out of range or can not be calculated
101	Tried to change I for AXIS number that was not initialized or is out of range
103	Data for I is out of range or can not be calculated
201	Tried to change D for AXIS number that was not initialized or is out of range
203	Data for D is out of range or can not be calculated
301	Tried to change OFST for AXIS number that was not initialized or is out of range
303	Data for OFST is out of range or can not be calculated
401	Tried to change FILT for AXIS number that was not initialized or is out of range
403	Data for FILT is out of range or can not be calculated
501	Tried to change FFWD for AXIS number that was not initialized or is out of range
503	Data for FFWD is out of range or can not be calculated

M_XL2CM

Excel to Cam Profile **USER/M_XL2CM**



Inputs: RQ00 (BOOL) - one-shot to start conversion

FNAM (STRING) - file name of csv file that is to be converted

HEDR (STRUCT) - structure containing information about the
profile such as master axis, scaling

CSTR (STRUCT) - array of structures used to hold RATIOCAM
profile data

SSTR (STRUCT) - array of structures used to hold RATIOSLP
profile data

RSTR (STRUCT) - array of structures used to hold RATIO_RL
profile data

Outputs: DONE (BOOL) - file conversion completed without error

FAIL (BOOL) - file conversion failed

FERR (INT) - error number from OPEN, READ, or CLOSE
functions

OERR (INT) - error number from operation of function

MTOT (DINT) - Total master distance moved in the profile
(feedback units)

STOT (DINT) - Total slave distance moved in the profile
(feedback units)

CAMO (BOOL) - set if the profile type is RATIOCAM

SLPO (BOOL) - set if the profile type is RATIOSLP

RELO (BOOL) - set if the profile type is RATIO_RL

<<INSTANCE NAME>>:M_XL2CM(RQ00 := <<BOOL>>, FNAM := <<STRING>>, HEDR := <<STRUCT>>, CSTR := <<STRUCT>>, SSTR :=

<<STRUCT>>, RSTR := <<STRUCT>>, DONE => <<BOOL>>, FAIL => <<BOOL>>, FERR => <<INT>>, OERR => <<INT>>, MTOT => <<DINT>>, STOT => <<DINT>>, CAMO => <<BOOL>>, SLPO => <<BOOL>>, RELO => <<BOOL>>);

This Excel to Cam application specific function block is used to convert an ASCII CSV file in the correct format to a data structure that can be called directly by RATIOCAM, RATIOSLP or RATIO_RL.

The CSV files must be created using the Excel spreadsheet that is included with this function (the default spreadsheet is called Cam Profile Design Tool.XLS). Using Excel, enter profile information such as master axis, slave axis, and scaling. You then select the profile type (RATIOCAM, RATIOSLP or RATIO_RL) and enter the point pairs. The spreadsheet allows you to see a graph of the profile. Once you have entered the data, you export the appropriate data to a CSV file. Instructions on how to do this are included with the spreadsheet. The resulting CSV file is copied to the control's RAMDISK so this function can read it.

The string specified at the FNAM input must include the entire file path (e.g. RAMDISK:File.CSV$00).

The HEDR structure contains the following members, which are all read from the csv file:

.VERSION STRING[8] - the version number
.COMMENT STRING[40] - the comment
.SLAVE_AXIS USINT - axis number of the slave
.SLAVE_LABEL STRING[32] - slave description
.MASTER_AXIS USINT - axis number of the master .MASTER_LABEL STRING[32] - master description
.SLAVE_FU DINT - slave scaling info
.SLAVE_LU DINT - slave scaling info
.MASTER_FU DINT - master scaling info
.MASTER_LU DINT - master scaling info
.PROFILE_TYPE STRING[32] - cam, slope or real .NUMBER_OF_SEGS INT - number of segments in the profile

The CSTR, SSTR and RSTR inputs are all arrays of structures. The structure format is the same as that specified by the associated function: RATIOCAM, RATIOSLP and RATIO_RL respectively. In the example program each data structure is declared to be the maximum size so no data overruns occur. You can make the structures smaller depending on your application.

Either the DONE or the FAIL output will be set when the conversion is complete. If the FAIL output is set, either the FERR or the OERR output will indicate an error number. The FERR output is used to indicate an error from the OPEN, READ or CLOSE function and the error numbers can be found in Appendix B of the online help. The OERR output indicates an error inside the function and the codes are as follows:

The MTOT and STOT outputs indicate the total master and slave distance moved over the entire profile.

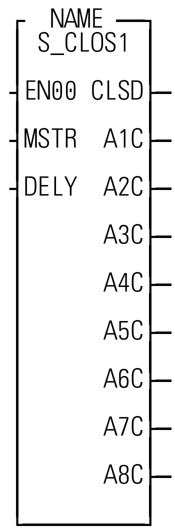
The CAMO, SLPO and RELO outputs are used to indicate the profile type. Only one of them will be set when the DONE is set. CAMO indicates RATIOCAM, SLPO indicates RATIOSLP and RELO indicates RATIO_RL. These outputs are used in the example program to decide which function to call after the profile data is read.

OERR DESCRIPTION:

ERR	Description
7001	Invalid version number in csv file
7002	Invalid comment line
7003	Invalid slave axis number and label
7004	Invalid master axis number and label
7005	Invalid slave scaling
7006	Invalid master scaling
7007	Invalid profile type found in file
7008	Invalid number of segments
7009	Invalid column labels line in csv file

S_CLOS1

Close Loop on SERCOS Servo Axes 1 to 8 **USER/S_ASFB**



Inputs: EN00 (BOOL) - enables execution

MSTR (BOOL) - machine start input

DELY (TIME) - amount of time that will elapse after a positive
transition of MSTR until the loops will be closed

Outputs: CLSD (BOOL) - one or more of axes 1 to 8 have their position loops closed

A1C (BOOL) - set when the loop is closed on axis 1

A2C (BOOL) - set when the loop is closed on axis 2

A3C (BOOL) - set when the loop is closed on axis 3

A4C (BOOL) - set when the loop is closed on axis 4

A5C (BOOL) - set when the loop is closed on axis 5

A6C (BOOL) - set when the loop is closed on axis 6

A7C (BOOL) - set when the loop is closed on axis 7

A8C (BOOL) - set when the loop is closed on axis 8

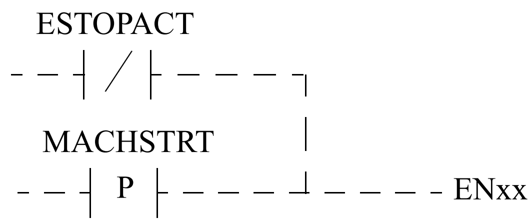
<<INSTANCE NAME>>:S_CLOS1(EN00 := <<BOOL>>, MSTR := <<BOOL>>, DELAY := <<TIME>>, CLSD => <<BOOL>>, A1C => <<INT>>, A2C => <<BOOL>>, A3C => <<BOOL>>, A4C => <<BOOL>>, A5C => <<BOOL>>, A6C => <<BOOL>>, A7C => <<BOOL>>, A8C => <<BOOL>>);

This function block is a replacement for M_CLOS1 for SERCOS axes. It is not for analog controlled axes.

This function block is used to reset the E-stop, C-stop, and programming errors on SERCOS servo axes 1 through 8 when the machine start input is pulsed. It also sends a class one diagnostics fault reset to the SERCOS drive. It closes the loop on SERCOS servo axes 1 through 8 after the machine start input is pulsed and a programmable time delay has elapsed. If there are no E-stop faults and the drive is enabled the loop will be closed and the closed output will be energized.

This function block can be enabled every scan. If the enable input changes from ON to OFF during the time delay after machine start, the function block will abort the time delay and not close the position loops.

If there are conditions that should abort the sequence to close the position loops (such as an electrical E-stop condition during the time delay), then the enable should include both the positive transition of the machine start input and the current state of the electrical E-stop status as shown below.



The reason for these two input conditions is to provide the enable at the start of the time delay (with the P contact of the machine start signal) and to maintain the enable during the time delay as needed (with the NC contact for the electrical E-stop condition).

The MMC example applications located on the Applications CD (in the examples sub-directory) illustrate the recommended ladder logic for the E-stop handling of the S_CLOS1 application. Please refer to MMC4_SOI.LDO for an example of S_CLOS1.

The machine start input must go through a positive transition (off to on) to reset the errors and close the loop.

The time DELY is normally in the range from 500 ms to 2 seconds.

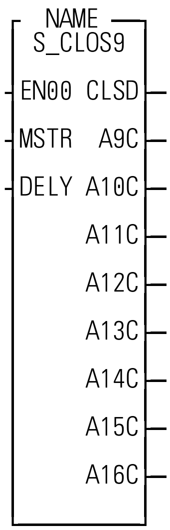
On a positive transition of MSTR, this function will send a procedure command to the SERCOS drive to reset class one diagnostic errors on axes 1 through 8.

The positive transition of MSTR enables a timer with a preset time of DELY. After DELY has elapsed, all E-stop, C-stop, and programming errors are reset on axes 1 through 8. If the E-stops are reset and the SERCOS drive is enabled, the loops will be closed on axes 1 to 8. CLSD will be energized if one or more of axes 1 to 8 have their position loops closed.

If an E-stop fault occurs on an axis 1 to 8, its loop closed output (A1 to A8) will be dropped. CLSD is true as long as one or more of axes 1 to 8 have their position loops closed.

S_CLOS9

Close Loop on SERCOS Servo Axes 9 to 16 **USER/S_ASFB**



Inputs: EN00 (BOOL) - enables execution

MSTR (BOOL) - machine start input

DELY (TIME) - amount of time that will elapse after a positive transition of MSTR until the loops will be closed

Outputs: CLSD (BOOL) - one or more of axes 9 to 16 have their position loops closed

A9C (BOOL) - set when the loop is closed on axis 9

A10C (BOOL) - set when the loop is closed on axis 10

A11C (BOOL) - set when the loop is closed on axis 11

A12C (BOOL) - set when the loop is closed on axis 12

A13C (BOOL) - set when the loop is closed on axis 13

A14C (BOOL) - set when the loop is closed on axis 14

A15C (BOOL) - set when the loop is closed on axis 15

A16C (BOOL) - set when the loop is closed on axis 16

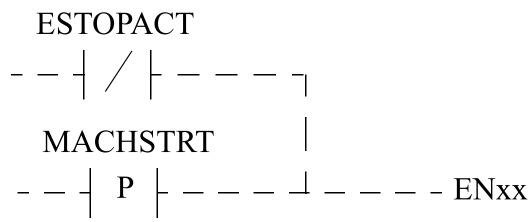
<<INSTANCE NAME>>:S_CLOS9(EN00 := <<BOOL>>, MSTR := <<BOOL>>, DELAY := <<TIME>>, CLSD => <<BOOL>>, A9C => <<INT>>, A10C => <<BOOL>>, A11C => <<BOOL>>, A12C => <<BOOL>>, A13C => <<BOOL>>, A14C => <<BOOL>>, A15C => <<BOOL>>, A16C => <<BOOL>>);

This function block is a replacement for M_CLOS9 for SERCOS axes. It is not for analog controlled axes.

This function block is used to reset the E-stop, C-stop, and programming errors on SERCOS servo axes 9 through 16 when the machine start input is pulsed. It also sends a class one diagnostics fault reset to the SERCOS drive. It closes the loop on SERCOS servo axes 9 through 16 after the machine start input is pulsed and a programmable time delay has elapsed. If there are no E-stop faults and the drive is enabled the loop will be closed and the closed output will be energized.

This function block can be enabled every scan. If the enable input changes from ON to OFF during the time delay after machine start, the function block will abort the time delay and not close the position loops.

If there are conditions that should abort the sequence to close the position loops (such as an electrical E-stop condition during the time delay), then the enable should include both the positive transition of the machine start input and the current state of the electrical E-stop status as shown below.



The reason for these two input conditions is to provide the enable at the start of the time delay (with the P contact of the machine start signal) and to maintain the enable during the time delay as needed (with the NC contact for the electrical E-stop condition).

The MMC example applications located on the Applications CD (in the examples sub-directory) illustrate the recommended ladder logic for the E-stop handling of the S_CLOSx application. Please refer to MMC4_SOI.LDO for an example of S_CLOSx.

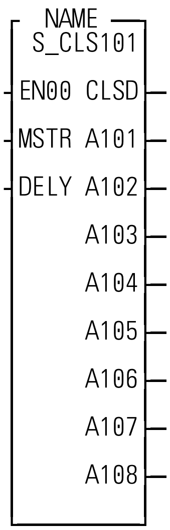
The machine start input must go through a positive transition (off to on) to reset the errors and close the loop.

The time DELY is normally in the range from 500 ms to 2 seconds.

On a positive transition of MSTR, this function will send a procedure command to the SERCOS drive to reset class one diagnostic errors on axes 9 through 16.

The positive transition of MSTR enables a timer with a preset time of DELY. After DELY has elapsed, all E-stop, C-stop, and programming errors are reset on axes 9 through 16. If the E-stops are reset and the SERCOS drive is enabled, the loops will be closed on axes 9 to 16. CLSD will be energized if one or more of axes 9 to 16 have their position loops closed.

If an E-stop fault occurs on an axis 9 to 16, its loop closed output (A9 to A16) will be dropped. CLSD is true as long as one or more of axes 9 to 16 have their position loops closed.



Inputs: EN00 (BOOL) - enables execution

MSTR (BOOL) - machine start input

DELY (TIME) - amount of time that will elapse after a positive
transition of MSTR until the loops will be closed

Outputs: CLSD (BOOL) - one or more of axes 101 to 108 have their position loops closed

A101 (BOOL) - set when the loop is closed on axis 101

A102 (BOOL) - set when the loop is closed on axis 102

A103 (BOOL) - set when the loop is closed on axis 103

A104 (BOOL) - set when the loop is closed on axis 104

A105 (BOOL) - set when the loop is closed on axis 105

A106 (BOOL) - set when the loop is closed on axis 106

A107 (BOOL) - set when the loop is closed on axis 107

A108 (BOOL) - set when the loop is closed on axis 108

<<INSTANCE NAME>>:S_CLOS101(EN00 := <<BOOL>>, MSTR := <<BOOL>>, DELAY := <<TIME>>, CLSD => <<BOOL>>, A101 => <<INT>>, A102 => <<BOOL>>, A103 => <<BOOL>>, A104 => <<BOOL>>, A105 => <<BOOL>>, A106 => <<BOOL>>, A107 => <<BOOL>>, A108 => <<BOOL>>);

This function block is a replacement for M_CLS101 for SERCOS axes. It is not for analog controlled axes.

This function block is used to reset the E-stop, C-stop, and programming errors on SERCOS servo axes 101 through 108 when the machine start input is pulsed. It also sends a class one diagnostics fault reset to the SERCOS drive. It closes the loop on SERCOS servo axes 101 through 108 after the machine start input is pulsed and a programmable time delay has elapsed. If there are no E-stop faults and the drive is enabled the loop will be closed and the closed output will be energized.

This function block can be enabled every scan. If the enable input changes from ON to OFF during the time delay after machine start, the function block will abort the time delay and not close the position loops.

If there are conditions that should abort the sequence to close the position loops (such as an electrical E-stop condition during the time delay), then the enable should include both the positive transition of the machine start input and the current state of the electrical E-stop status as shown below.

```
      ESTOPACT
  - - ┤ / ├ - - - - ┐
                    │
      MACHSTRT      │
  - - ┤ P ├ - - - - ┴ - - - - ENxx
```

The reason for these two input conditions is to provide the enable at the start of the time delay (with the P contact of the machine start signal) and to maintain the enable during the time delay as needed (with the NC contact for the electrical E-stop condition).

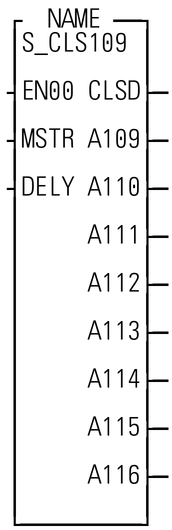
The machine start input must go through a positive transition (off to on) to reset the errors and close the loop.

The time DELY is normally in the range from 500 ms to 2 seconds.

On a positive transition of MSTR, this function will send a procedure command to the SERCOS drive to reset class one diagnostic errors on axes 101 through 108.

The positive transition of MSTR enables a timer with a preset time of DELY. After DELY has elapsed, all E-stop, C-stop, and programming errors are reset on axes 101 through 108. If the E-stops are reset and the SERCOS drive is enabled, the loops will be closed on axes 101 to 108. CLSD will be energized if one or more of axes 101 to 108 have their position loops closed.

If an E-stop fault occurs on an axis 101 to 108, its loop closed output (A101 to A108) will be dropped. CLSD is true as long as one or more of axes 101 to 108 have their position loops closed.



Inputs: EN00 (BOOL) - enables execution

MSTR (BOOL) - machine start input

DELY (TIME) - amount of time that will elapse after a positive
transition of MSTR until the loops will be closed

Outputs: CLSD (BOOL) - one or more of axes 109 to 116 have their position loops closed

A109 (BOOL) - set when the loop is closed on axis 109

A110 (BOOL) - set when the loop is closed on axis 110

A111 (BOOL) - set when the loop is closed on axis 111

A112 (BOOL) - set when the loop is closed on axis 112

A113 (BOOL) - set when the loop is closed on axis 113

A114 (BOOL) - set when the loop is closed on axis 114

A115 (BOOL) - set when the loop is closed on axis 115

A116 (BOOL) - set when the loop is closed on axis 116

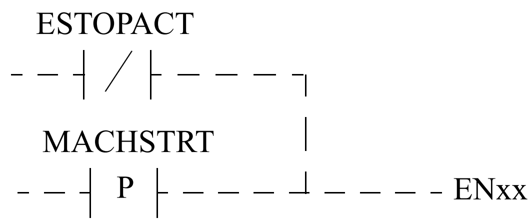
<<INSTANCE NAME>>:S_CLOS109(EN00 := <<BOOL>>, MSTR := <<BOOL>>, DELAY := <<TIME>>, CLSD => <<BOOL>>, A109 => <<INT>>, A110 => <<BOOL>>, A111 => <<BOOL>>, A112 => <<BOOL>>, A113 => <<BOOL>>, A114 => <<BOOL>>, A115 => <<BOOL>>, A116 => <<BOOL>>);

This function block is a replacement for M_CLS109 for SERCOS axes. It is not for analog controlled axes.

This function block is used to reset the E-stop, C-stop, and programming errors on SERCOS servo axes 109 through 116 when the machine start input is pulsed. It also sends a class one diagnostics fault reset to the SERCOS drive. It closes the loop on SERCOS servo axes 109 through 116 after the machine start input is pulsed and a programmable time delay has elapsed. If there are no E-stop faults and the drive is enabled the loop will be closed and the closed output will be energized.

This function block can be enabled every scan. If the enable input changes from ON to OFF during the time delay after machine start, the function block will abort the time delay and not close the position loops.

If there are conditions that should abort the sequence to close the position loops (such as an electrical E-stop condition during the time delay), then the enable should include both the positive transition of the machine start input and the current state of the electrical E-stop status as shown below.



The reason for these two input conditions is to provide the enable at the start of the time delay (with the P contact of the machine start signal) and to maintain the enable during the time delay as needed (with the NC contact for the electrical E-stop condition).

The machine start input must go through a positive transition (off to on) to reset the errors and close the loop.

The time DELY is normally in the range from 500 ms to 2 seconds.

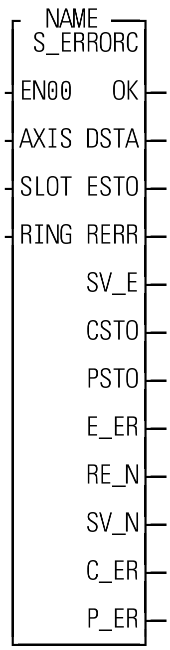
On a positive transition of MSTR, this function will send a procedure command to the SERCOS drive to reset class one diagnostic errors on axes 109 through 116.

The positive transition of MSTR enables a timer with a preset time of DELY. After DELY has elapsed, all E-stop, C-stop, and programming errors are reset on axes 109 through 116. If the E-stops are reset and the SERCOS drive is enabled, the loops will be closed on axes 109 to 116. CLSD will be energized if one or more of axes 109 to 116 have their position loops closed.

If an E-stop fault occurs on an axis 109 to 116, its loop closed output (A109 to A116) will be dropped. CLSD is true as long as one or more of axes 109 to 116 have their position loops closed.

S_ERRORC

Axis Error Checking Centurion SERCOS Drives **USER/S_ASFB**



Inputs: EN00 (BOOL) - enables execution

AXIS (USINT) - identifies SERCOS axis

SLOT (USINT) - slot number for the SERCOS module

RING (USINT) - ring the axis is connected to

Outputs: OK (BOOL) - execution complete

DSTA (USINT) - indicates the drive status

ESTO (BOOL) - indicates an E-stop is active when set

RERR (BOOL) - indicates a ring error

SV_E (BOOL) - indicates a slave (drive) error

CSTO (BOOL) - indicates a C-stop is active when set

PSTO (BOOL) - indicates a programming error has occurred

E_ER(WORD) - identifies E-stop errors

RE_N (INT) - identifies ring error number

SV_N (UINT) - identifies slave (drive) error number

C_ER (WORD) - identifies C-stop errors

P_ER (WORD) - identifies programming errors

<<INSTANCE NAME>>:S_ERRORC(EN00 := <<BOOL>>, AXIS := <<USINT>>, SLOT := <<USINT>>, RING := <<USINT>>, OK => <<BOOL>>, DSTA => <<USINT>>, ESTO => <<BOOL>>, RERR => <<BOOL>>, SV_E => <<BOOL>>, CSTO => <<BOOL>>, PSTO => <<BOOL>>, E_ER => <<WORD>>, RE_N => <<INT>>, SV_N =><<UINT>>, C_ER => <<WORD>>, P_ER => <<WORD>>);

This function block is a replacement for M_ERROR for a SERCOS axis with a Centurion drive. It is not for an analog controlled axis.

This function block is used to report errors that occur on a SERCOS servo axis. The types of errors include ring errors, drive errors, E-stop, C-stop and programming errors. These conditions may be caused by the SERCOS hardware, SERCOS drive, servo software or the ladder programming. If defined by the programmer, they will be triggered using the E-STOP or C_STOP functions. All of these errors for the defined axis are reported by this one function block.

The enable input of this function should be directly connected to the rail with a wire, causing this function block to be executed each scan.

The boolean outputs can be used as flags in the ladder to report error conditions.

The E_ER, C_ER and P_ER word outputs can be converted to HEX display by using the Module Monitor Edit View List command and inserting the variables. Alternately, they can be given an initial value of 16#0 for a hex value during animation. After monitoring them in HEX, refer to the tables in the manual of functions E_ERRORS, C_ERRORS and P_ERRORS to help identify the exact problem. The RE_N value (ring error number) value can be identified by refering to the SCR_ERR function in the Function/Function Block Reference Guide. Refer to the SERCOS drive manual for the description of errors occuring on the SV_N value (drive error number).

The DSTA, drive status output reflects the state the drive is currently at in decimal format. It is defined by the upper two bits of the SERCOS drive status word. The four possible states are:

0- drive not ready for power-up, internal checks not yet concluded successfully

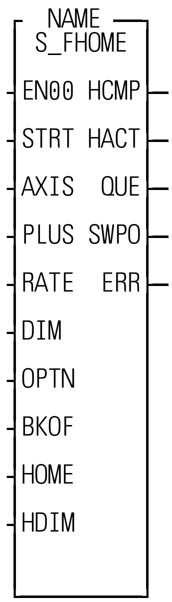
1- drive logic OK and ready for main power

2 - drive ready and main power applied but not enabled, no torque to motor

3 - drive is enabled with torque applied to motor

S_FHOME

Performs a SERCOS Home Cycle using a Fast Reference **USER/S_ASFB**



Inputs: EN00 (BOOL) - enables execution

STRT (BOOL) - enables the home cycle

AXIS (USINT) - identifies SERCOS axis

PLUS (BOOL) - indicates direction of home cycle

RATE **(**UDINT) - feedrate at which motion occurs (entered in LU/MIN)

DIM (DINT) - reference dimension for the nearest resolver null or the next encoder index mark when the reference switch is set (entered in LUs)

OPTN (WORD) - provides referencing options (0 or 1) 0 = no option, 1 = Ignore index or null

BKOF (BOOL) - selects backoff of reference switch option

HOME (BOOL) - selects homing after referencing option

HDIM (DINT) - home location to move after reference is
complete

Outputs: HCMP (BOOL) - home cycle is complete

HACT (BOOL) - home cycle is being executed

QUE (USINT) - number of moves for queue

SWPO (DINT) - distance in feedback units (FUs) from the
reference switch to the index mark of an encoder or the null of a
resolver

ERR (BYTE) - report an error 1-4 if input data is invalid

<<INSTANCE NAME>>:S_FHOME(EN00 := <<BOOL>>, STRT := <<BOOL>>, AXIS := <<USINT>>, PLUS := <<BOOL>>, RATE := <<UDINT>>, DIM := <<DINT>>, OPTN := <<WORD>>, BKOF := <<BOOL>>, HOME := <<BOOL>>, HDIM := <<DINT>>, HCMP => <<BOOL>>, HACT => <<BOOL>>, QUE => <<USINT>>, SWPO => <<DINT>>, ERR => <<BYTE>>);

This function block is a replacement for M_FHOME for a SERCOS axis. It is not for an analog controlled axis.

This function block performs a fast reference cycle on a SERCOS axis, followed by a homing (position) move to a designated location.

Before this function can be used, the SERCOS axis must be initialized and the position loop must be closed.

The reference cycle will cause the selected SERCOS axis to move in the designated direction until the reference switch is sensed.

In the Centurion SERCOS drive the reference switch is wired to the input number two of the selected axis on the Centurion drive. This function block uses SCA_RFIT to initialize the SERCOS drive's fast input for the reference cycle and to direct the SERCOS drive to latch the position upon that input.

When the fast input occurs, the position of the axis is latched by the hardware in the drive independent of the ladder scan.

When the reference switch is sensed, the axis will reference (assign a value) to the next index mark of an encoder or the nearest null of a resolver. After the value is assigned, the axis will decelerate to a stop and set the reference done flag.

If the HOME input is on when the reference done has been sensed, the home move will automatically be triggered to position the axis at a desired location.

If the BKOF input is on when the reference is requested, and the axis is on the reference switch, the axis will move in the opposite direction of that indicated by the PLUS input until the switch opens and then will complete the home cycle in the normal manner. If the BKOF input is not on the axis will move in the specified direction until it sees an off to on transition of the limit switch.

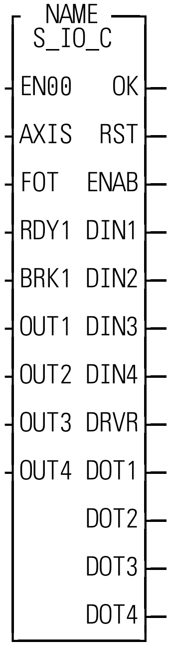
This function block is used to perform a fast reference, immediately followed by a position move to a selected home position. It should be executed every scan unless a home cycle will only be performed when the machine is started. In that case a normally closed contact of the output of HCMP may be used.

The SWPO output is used to determine if the reference switch location will allow for repeatable referencing. If the reference switch is not properly located in relationship to the index marker of an encoder or the null of a resolver it could possibly reference a revolution off. To prevent this, the value reported by this output should be as follows:

- For an encoder system the value of this output should be greater than 25% and less than 75% of the total counts (FUs) per revolution. Example: For 8000 FUs/ Rev, the value should be >2000 and <6000.
- For a resolver system the value of this output should be less than 25% or greater than 75% of the total counts (FUs) per revolution. Example: For 4000 FUs/ Rev, the value should be <1000 or >3000.

If the value is out of range either the reference switch will have to be moved or the transducer coupling shifted.The ERR output indicates that invalid data was entered on one of the inputs. The possible errors are listed in the following table:

ERR	Description
0	No error
1	The queue was not empty when the reference was requested
2	An error occurred in backing off of the reference switch
3	An error occurred in referencing
4	An error occurred in homing
5	An error occurred within the SERCOS drive, either during the initialization of the SERCOS drive (its probe input) or during the monitoring of the SERCOS drive while it is referencing. The SERCOS ring and slave error values can be obtained by animating this function block after the error.



Inputs: EN00 (BOOL) - enables execution

AXIS (USINT) - identifies SERCOS axis

FOT (BOOL) - force the desired output states to drive outputs

RDY1 **(**BOOL) - desired state of the drive ready output

BRK1 (BOOL) - desired state of the drive brake output

OUT1 (BOOL) - desired state of the drive output one

OUT2 (BOOL) - desired state of the drive output two

OUT3 (BOOL) - desired state of the drive output three

OUT4 (BOOL) - desired state of the drive output four

Outputs: OK (BOOL) - execution complete

RST (BOOL) - state of the drive reset input

ENAB (BOOL) - state of the drive enable input

DIN1 (BOOL) - state of the drive input one

DIN2 (BOOL) - state of the drive input two

DIN3 (BOOL) - state of the drive input three

DIN4 (BOOL) - state of the drive input four

DRVR (BOOL) - state of the drive ready output

DRVB (BOOL) - state of the drive brake output

DOT1 (BOOL) - state of the drive output one

DOT2 (BOOL) - state of the drive output two

DOT3 (BOOL) - state of the drive output three

DOT4 (BOOL) - state of the drive output four

<<INSTANCE NAME>>:S_IO_C(EN00 := <<BOOL>>, AXIS := <<USINT>>, FOT := <<BOOL>>, RDY1 := <<BOOL>>, BRK1 := <<BOOL>>, OUT1 := <<BOOL>>, OUT2 := <<BOOL>>, OUT3 := <<BOOL>>, OUT4 := <<BOOL>>, OK => <<BOOL>>, RST => <<BOOL>>, ENAB => <<BOOL>>, DIN1 => <<BOOL>>, DIN2 => <<BOOL>>, DIN3 => <<BOOL>>, DIN4 => <<BOOL>>, DRVR => <<BOOL>>, DRVB =>

<<BOOL>>, DOT1 => <<BOOL>>, DOT2 => <<BOOL>>, DOT3 =>
<<BOOL>>, DOT4 => <<BOOL>>);

The SERCOS drive I/O function block provides ladder access to the digital inputs and outputs of a Centurion DSM SERCOS servo drive through the SERCOS service channel. It is not intended for analog or non-Centurion DSM SERCOS drives.

The Digital Output Override IDN P0036 must be set in the drive to use this feature. Refer to the Centurion SERCOS Drives IDN Manual.

The EN enable input should always be enabled allowing the function block to be executed every ladder scan. When the input is enabled the function block will wait until the SERCOS and Servo axis setup files have been initiated before any service channel activity begins. Once initialized, the drive ID is read to verify it is a Centurion SERCOS drive, and if so, the drive inputs and outputs are then continuously read.

When the FOT is enabled the state of the function block RDY1, BRK1, OUT1, OUT2, OUT3 and OUT4 inputs will be written to the drive. After a successful write, the function block outputs will reflect the desired outputs. While the FOT is enabled, the function block inputs are evaluated and any state change will be sent to the drive. The new drive outputs will again be reflected at the function block outputs.

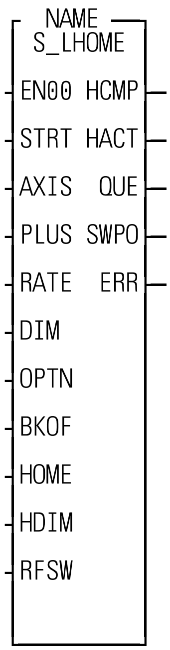
NOTE
If the FOT input is disabled, all drive outputs are set to zero.

The OK output will be energized only if the axis is initialized, the drive is of type Centurion DSM SERCOS, and the drive I/O reads and writes are successful. If any failures occur, the OK output will not be energized. If a failure occurs while reading or writing I/O, the related function block outputs are set to zero.

NOTE
If an I/O read or write failure occurs, the related function block outputs may not reflect the actual state of the drive inputs and outputs

S_LHOME

Perform a SERCOS Home Cycle using a Ladder Reference **USER/S_ASFB**



Inputs: EN00 (BOOL) - enables execution

STRT (DINT) - move from reference switch and move back to home position after referencing.

AXIS (USINT) - identifies SERCOS axis

PLUS **(**BOOL) - indicates direction of home cycle

RATE (UDINT) - feedrate at which motion occurs (entered in LU/MIN)

DIM (DINT) - reference dimension for the nearest resolver null or the next encoder index mark when the reference switch is set
(entered in LUs)

OPTN (WORD) - provides referencing options (0 or 1)
0 = No option, 1 = Ignore index or null

BKOF (BOOL) - selects backoff of reference switch option

HOME (BOOL) - selects homing after referencing option

HDIM (DINT) - home location to move to after reference is complete

RFSW (BOOL) - references switch on axis

Outputs: HCMP (BOOL) - home cycle is complete

HACT (BOOL) - home cycle is being executed

QUE (USINT) - number of move for queue

SWP0 (DINT) - distance in feedback unit (FUs) from the
reference switch to the index mark of an encoder or the null of a resolver

ERR (BYTE) - report an error 1-4 if input data is invalid

```
<<INSTANCE NAME>>:S_LHOME(EN00 := <<BOOL>>,
  STRT := <<DINT >> AXIS := <<USINT>>, PLUS := <<BOOL>>, RATE :=
  <<UDINT>>, DIM := <<DINT>>, OPTN := <<WORD>>, BKOF :=
  <<BOOL>>, HOME := <<BOOL>>, HDIM := <<DINT>>, RFSW :=
  <<BOOL>>, HCMP => <<BOOL>>, HACT => <<BOOL>>, QUE =>
  <<USINTL>>, SWPO => <<DINT>>, ERR => <<BYTE>>);
```

This function block is a replacement for M_LHOME for a SERCOS axis. It is not for an analog controlled axis.

This function block performs a ladder reference cycle on a SERCOS axis, followed by a homing (position) move to a designated location.

Before this function block can be used, the SERCOS axis must be initialized and the position loop closed.

The reference cycle will cause the selected SERCOS axis to move in the designated direction until the reference switch is sensed. This function block uses SCA_RFIT to direct the drive to ignore the fast input for the reference and to monitor the position while the ladder checks for the reference switch. In a ladder reference, this reference switch is wired to an input in the MMC or in an input module within the PiC rack and updated each scan of the ladder. When the reference switch is sensed the SERCOS axis will reference (assign a value) to the next index mark of an encoder or the nearest null of a resolver. After the value is assigned the axis will decelerate to a stop and set the reference done flag.

If the HOME input is on when the reference done has been sensed the home move will automatically be triggered to position the SERCOS axis at a desired location.

If the BKOF input is on when the reference is requested and if the axis is on the reference switch, the axis will move in the opposite direction of that indicated by the PLUS input until the reference switch opens, and then will complete the home cycle in the normal manner. If the BKOF input is not on, the axis will move in the specified direction until it sees an off to on transition of the limit switch.

This function block is used to perform a ladder reference, immediately followed by a position move to a selected home position. It should be executed every scan unless a home cycle will only be performed when the machine is started. In that case a normally closed contact of the output of HCMP may be used.

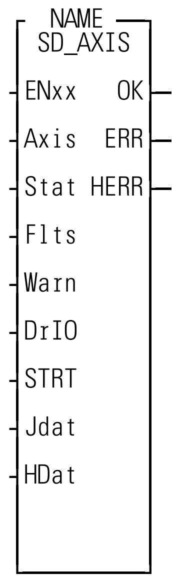
The SWPO output is used to determine if the reference switch location will allow for repeatable referencing. If the reference switch is not properly located in relationship to the index marker of an encoder or the null of a resolver it could possibly reference a revolution off. To prevent this, the value reported by this output should be as follows:

- For an encoder system the value of this output should be greater than 25% and less than 75% of the total counts (FUs) per revolution. Example: For 8000 FUs/ Rev, the value should be >2000 and <6000.
- For a resolver system the value of this output should be less than 25% or greater than 75% of the total counts (FUs) per revolution. Example: For 4000 FUs/ Rev, the value should be <1000 or >3000.

If the value is out of range either the reference switch will have to be moved or the transducer coupling shifted.

The ERR output indicates that invalid data was entered on one of the inputs. The possible errors are listed below:

ERR	Description
0	No error
1	The queue was not empty when the reference was requested
2	An error occurred in backing off of the reference switch
3	An error occurred in referencing
4	An error occurred in homing
5	An error occurred within the SERCOS drive, either during the initialization of the SERCOS drive (its probe input) or during the monitoring of the SERCOS drive while it is referencing. The SERCOS ring and slave error values can be obtained by animating this function block after the error.



Inputs:

ENxx (BOOL) - enables the block and should be on at all times once drive communications are established. (xx indicates revision #)

Axis (USINT) - specifies the axis number

Stat (STRUCT) - axis and drive status information

Flts (STRUCT) - axis and drive fault information

Warn (STRUCT) - axis and drive warning information

DrIO (STRUCT) - status of drive inputs and settings for outputs

STRT (STRUCT) - signals to control the position loop

Jdat (STRUCT) - Jog data, signals and status for the jog functionality

HDat (STRUCT) - Home data, signals and status for the home functionality

Outputs:

OK (BOOL) - function OK

ERR (INT) - error number from block

HERR (INT) - error number from home cycle

<<INSTANCE NAME>>:SD_AXIS(ENxx := 1, Axis := <<USINT>>, Stat := <<MEMORY AREA>>, Flts := <<MEMORY AREA>>, Warn := <<MEMORY AREA>>, DrIO := <<MEMORY AREA>>, STRT := <<MEMORY AREA>>, Jdat := <<MEMORY AREA>>, HDat := <<MEMORY AREA>>, OK => <<BOOL>>, ERR => <<INT>>, HERR => <<INT>>);

This function block includes the SD_STAT1 function to provide axis and drive faults and warnings and the SD_IO function to read and write drive I/O. It also contains the position loop closing and opening logic as well as homing and jogging functions.

INPUTS:

NAME	DATA TYPE	DESCRIPTION OR FUNCTION USED FOR DATA
ENxx	BOOL	Enables Execution
Axis	USINT	Axis Number
Stat	STRUCT	Value written to drive output 1, READ_SV variable 72
.InPosition	BOOL	IN_POS?
.QueAvailable	BOOL	Q_AVAIL?
.ActiveQue	USINT	Q_NUMBER
.MoveType	DINT	READ_SV variable 2
.Actual Position	DINT	READ_SV variable 1
.CommandedPosition	DINT	READ_SV variable 3
.PositionError	DINT	READ_SV variable 4
.FilterError	DINT	READ_SV variable 5
.MotorCurrent	DINT	READ_SV variable 73
.MotorAvgCurrent	DINT	READ_SV variable 74
.MotorTemp	DINT	READ_SV variable 77
.StartupCommutation Complete	BOOL	READ_SV variable 67 Bit 0
.AtZeroSpeed	BOOL	READ_SV variable 67 Bit 1
.InSpeedWindow	BOOL	READ_SV variable 67 Bit 2
.UpToSpeed	BOOL	READ_SV variable 67 Bit 3
.AtPlusCurrentLimit	BOOL	READ_SV variable 67 Bit 4
.AtMinusCurrentLimit	BOOL	READ_SV variable 67 Bit 5
.DriveBusCharged	BOOL	READ_SV variable 67 Bit 6
.DriveEnabled	BOOL	READ_SV variable 67 Bit 7
.DriveReady	BOOL	READ_SV variable 67 Bit 8
.BrakeReleased	BOOL	READ_SV variable 67 Bit 9
.DriveFault	BOOL	READ_SV variable 67 Bit 10
.DriveWarning	BOOL	READ_SV variable 67 Bit 11
.ShuntOn	BOOL	READ_SV variable 67 Bit 12
.DriveReadyAndBusCharged	BOOL	READ_SV variable 67 Bit 13
.Reserved	BOOL (0..5)	READ_SV variable 67 Bits 14-23
.HardwareEnableLine	BOOL	READ_SV variable 67 Bit 24
.AuxiliaryFeedbackLossOfFeedback	BOOL	READ_SV variable 67 Bit 25
.Reserved_26_31	BOOL (0..9)	READ_SV variable 67 Bits 26-31
.StructureSize	UINT	Set to the Size of STAT Structure in Bytes

END_STUCT

NAME	DATA TYPE	DESCRIPTION OR FUNCTION USED FOR DATA
Flts	STRUCT	Axis and Drive Faults
.EstopActive	BOOL	E_STOP?
.EstopErr	WORD	E_ERRORS
.LossOfFeedbackEstop	BOOL	E_ERRORS Bit 0
.ExcessPositionErrorEstop	BOOL	E_ERRORS Bit 1
.SlaveOverflowDeltaEstop	BOOL	E_ERRORS Bit 2
.UserLadderEstop	BOOL	E_ERRORS Bit 3
.SERCOSDriveEstop	BOOL	E_ERRORS Bit 4
.SERCOSCycDataEstop	BOOL	E_ERRORS Bit 5
.ASIU_Timeout	BOOL	E_ERRORS Bit 6
.DriveFaultEstop	BOOL	E_ERRORS Bit 7
.DriveCommFaultEstop	BOOL	E_ERRORS Bit 8
.ReservedEstopbits	BOOL (0..5)	E_ERRORS Bits 9-14
.AnyEstopBitSet	BOOL	E_ERRORS Bit 15
.CstopActive	BOOL	C_STOP?
.CstopErr	WORD	C_ERRORS
.SWPositiveEndLimitCstop	BOOL	C_ERRORS Bit 0
.SWNegative EndLimitCstop	BOOL	C_ERRORS Bit 1
.UserLadderCstop	BOOL	C_ERRORS Bit 2
.MachRefDimCstop	BOOL	C_ERRORS Bit 3
.FeedrateCstop	BOOL	C_ERRORS Bit 4
.DistPosnDimCstop	BOOL	C_ERRORS Bit 5
.PartRefDimCstop	BOOL	C_ERRORS Bit 6
.PartRefCstop	BOOL	C_ERRORS Bit 7
.ReservedCstopbits	BOOL (0..6)	C_ERRORS Bits 8-14
.AnyCstopBitSet	BOOL	C_ERRORS Bit 15
.PstopActive	BOOL	P_STOP?
.PstopErr	WORD	P_ERRORS
.MasterStartPosnPstop	BOOL	P_ERRORS Bit 0
.ReservedBit1Pstop	BOOL	P_ERRORS Bit 1
.ReservedBit2Pstop	BOOL	P_ERRORS Bit 2
.ReservedBit3Pstop	BOOL	P_ERRORS Bit 3
.ReservedBit4Pstop	BOOL	P_ERRORS Bit 4
.MasterNotAvailPstop	BOOL	P_ERRORS Bit 5
.InvalidProfilePstop	BOOL	P_ERRORS Bit 6
.FastQuePstop	BOOL	P_ERRORS Bit 7
.SlaveDistPstop	BOOL	P_ERRORS Bit 8

.MasterDistPstop	BOOL	P_ERRORS Bit 9
.SlaveBeyondStrtPointPstop	BOOL	P_ERRORS Bit 10
.MasterBeyondStrtPointPstop	BOOL	P_ERRORS Bit 11
.ReservedBit13Pstop	BOOL	P_ERRORS Bit 12
.ReservedBit14Pstop	BOOL	P_ERRORS Bit 13
.ReservedBit15Pstop	BOOL	P_ERRORS Bit 14
.AnyPstopBitSet	BOOL	P_ERRORS Bit 15
.DriveFaultDW	DWORD	READ_SV variable 68 converted to a DWORD
.DFLT	UINT	Identifies Drive Fault Code, see the MMC Smart Drive and Digital MMC Control Hardware Manual for detailed description
.NoDriveFaults	BOOL	Set when no drive fault is active
.DriveMemoryFault_11	BOOL	READ_SV variable 68 Bit 0
.DriveBusOverVoltageFault_12	BOOL	READ_SV variable 68 Bit 1
.DrivePM1OverCurrentFault_13	BOOL	READ_SV variable 68 Bit 2
.DriveBusUnderVoltageFault_14	BOOL	READ_SV variable 68 Bit 3
.MotorTempFault_15	BOOL	READ_SV variable 68 Bit 4
.ContinuousCurrentFaul_16	BOOL	READ_SV variable 68 Bit 5
.DriveHeatsinkTempFault_17	BOOL	READ_SV variable 68 Bit 6
.DriveF2FeedbackFault_21	BOOL	READ_SV variable 68 Bit 7
.DriveF1FeedbackFault_22	BOOL	READ_SV variable 68 Bit 8
.DriveAmbientTempFault_23	BOOL	READ_SV variable 68 Bit 9
.MotorCalculatedTempFault_24	BOOL	READ_SV variable 68 Bit 10
.DriveTimingFault_25	BOOL	READ_SV variable 68 Bit 11
.DriveInterfaceFault_26	BOOL	READ_SV variable 68 Bit 12
.UserSetFault_27	BOOL	READ_SV variable 68 Bit 13
.DriveF1CommunicationFault_31	BOOL	READ_SV variable 68 Bit 14
.OverSpeedFault_32	BOOL	READ_SV variable 68 Bit 15
.OverCurrentFault_33	BOOL	READ_SV variable 68 Bit 16
.ControlPanelDisconnectFault_34	BOOL	READ_SV variable 68 Bit 17
.DrivePowerModuleFault_35	BOOL	READ_SV variable 68 Bit 18
.FeedbackTypeMismatchFault_36	BOOL	READ_SV variable 68 Bit 19
.EndatFault_37	BOOL	READ_SV variable 68 Bit 20
.DriveRelayFault_41	BOOL	READ_SV variable 68 Bit 21
.DrivePM2OverCurrentFault_42	BOOL	READ_SV variable 68 Bit 22
.DrivePMTempFault_43	BOOL	READ_SV variable 68 Bit 23
.MotorGroundFault_44	BOOL	READ_SV variable 68 Bit 24
.DriveACInputOverVoltageFault_45	BOOL	READ_SV variable 68 Bit 25
.OvertravelPlusFault_46	BOOL	READ_SV variable 68 Bit 26

.OvertravelMinusFault_47	BOOL	READ_SV variable 68 Bit 27
.DigitalLinkCommunicationError_51	BOOL	READ_SV variable 68 Bit 28
.InvalidSwitchSettingFault_52	BOOL	READ_SV variable 68 Bit 29
.HardwareFailureFault_53	BOOL	READ_SV variable 68 Bit 30
.Reserved4	BOOL	READ_SV variable 68 Bit 31
.StuctureSize	UINT	Set to the Size of Flts Structure in Bytes
	END_STUCT	

NAME	DATA TYPE	DESCRIPTION OR FUNCTION USED FOR DATA
Warn	STRUCT	Drive Warnings, see MMC Smart Drive and Digital MMC Control Hardware Manual for detailed descriptions
.DriveWarnDW	DWORD	READ_SV variable 69 converted to a DWORD
.DWRN	UINT	Same as DWRN output. Identifies Drive Warning Code, see the MMC Smart Drive and Digital
.Nowarnings	BOOL	Set when no drive warning is active
.DriveHeatsinkTempWarning_01	BOOL	READ_SV variable 69 Bit 0
.DriveAmbientTempWarning_02	BOOL	READ_SV variable 69 Bit 1
.MotorTempWarning_03	BOOL	READ_SV variable 69 Bit 2
.MotorCalculatedTempWarning_04	BOOL	READ_SV variable 69 Bit 3
.OvertravelPlusWarning_05	BOOL	READ_SV variable 69 Bit 4
.OvertravelMinusWarning_06	BOOL	READ_SV variable 69 Bit 5
.Reserved	BOOL (0..25)	READ_SV variable 69 Bits 6-31
.StructureSize	UINT	Set to the Size of Warn Structure in Bytes
		END STUCT

NAME	DATA TYPE	DESCRIPTION OR FUNCTION USED FOR DATA
DrIO	STRUCT	Status of drive inputs and settings for outputs.
.A_In	DINT	Value of drive analog input, READ_SV variable 71
.W_01	BOOL	Value written to drive output 1, READ_SV variable 72
.W_02	BOOL	Value written to drive output 2, READ_SV variable 72
.W_03	BOOL	Value written to drive output 3, READ_SV variable 72
.W_04	BOOL	Value written to drive output 4, READ_SV variable 72
.WBRK	USINT	Value written to drive brake output, READ_SV variable 72
.R_01	BOOL	Current state of drive output 1, READ_SV variable 72
.R_02	BOOL	Current state of drive output 2, READ_SV variable 72
.R_03	BOOL	Current state of drive output 3, READ_SV variable 72
.R_04	BOOL	Current state of drive output 4, READ_SV variable 72
.RBRK	BOOL	Current state of drive brake output, READ_SV variable 72
.In_1	BOOL	Current state of drive input 1, READ_SV variable 71
.In_2	BOOL	Current state of drive input 2, READ_SV variable 71
.In_3	BOOL	Current state of drive input 3, READ_SV variable 71
.In_4	BOOL	Current state of drive input 4, READ_SV variable 71
.In_5	BOOL	Current state of drive input 5, READ_SV variable 71
.In_6	BOOL	Current state of drive input 6, READ_SV variable 71
.In_7	BOOL	Current state of drive input 7, READ_SV variable 71
.In_8	BOOL	Current state of drive input 8, READ_SV variable 71

.StructureSize	UINT	used to test the external and internal structures to ensure they are the same size
	END_STRUCT	

NAME	DATA TYPE	DESCRIPTION OR FUNCTION USED FOR DATA
STRT	STRUCT	Signals to control the position loop
.MSTR	BOOL	input - clears axis and drive errors and closes the position loop
.DELY	TIME	input - delay time after MSTR input before position loop is closed
.ESTP	BOOL	input - activates a USER ESTOP if no other E_ERROR active and opens position loop
.CSTP	BOOL	input - activates a USER CSTOP if no other C_ERROR active
.CLSD	BOOL	output - indicates state of position loop
.NO_DC_BUS	BOOL	output - indicates an attempt was made to close the loop with no dc bus power available. Also puts a 999 fault in the drive faults code in ladder
.StructureSize	UINT	used to test the external and internal structures to ensure they are the same size
	END_STRUCT	

The STRT structure input members are used as coils in the ladder diagram to trigger the actions described above. The output members are used as contacts in the ladder to carry out any interlocking or actions needed. See the example ladders provided with PiCPro Applications for more detail

NAME	DATA TYPE	DESCRIPTION OR FUNCTION USED FOR DATA
Jdat	STRUCT	Controls jog
.JPLS	BOOL	input - jogs in a + direction at JRat speed or moves incremental distance
.JMNS	BOOL	input -jogs in a - direction at JRat speed or moves incremental distance
.JRate	UDINT	input - axis jog rate in LU / MIN
.JogActive	BOOL	output - velocity jog is active
.JogQue	USINT	output - velocity jog queue number #
.IncDistance	DINT	input - incremental jog distance
.IncMove	BOOL	input - causes incremental distance move with JPLS or JMNS inputs
.IncQue	USINT	output -incremental distance move queue #
.IncAct	BOOL	output - incremental move is active
.OKtoJog	BOOL	input - must be on to allow jog to take place with JPLS and JMNS
.Hdwhl_jog	BOOL	input - when on, locks axis to hand wheel (digitizing axis)
.Hdwhl_axis	USINT	input - axis number for hand wheel
.Axis_SDST	DINT	input - axis LU moved when hand wheel moves its distance
.Hdwhl_MDST	DINT	input - hand wheel LU for axis to move previous entry
.Hdwhl_que	USINT	output - queue number of hand wheel move
.HdwhlAct	BOOL	output - on when hand wheel jog active
.StructureSize	UINT	used to test external and internal structures for the same size
	END_STRUCT	

The JDAT structure input members are used as coils in the ladder diagram to trigger the actions described above. Output members are used as contacts in the ladder as needed.

Three types of jog are possible - OktoJog must be on for all to work

- Continuous Jog - axis will move as long as the inputs are made

- Incremental Jog - axis will move the specified distance each time an input is made
- Hand wheel Jog - axis is electronically geared to an encoder hand wheel

It is the responsibility of the main ladder to turn on ONLY one at a time.

See the example ladders provided with PiCPro Applications for more detail.

NAME	DATA TYPE	DESCRIPTION OR FUNCTION USED FOR DATA
HDat	STRUCT	configures homing cycle options
.Strt_Home	BOOL	input - starts axis home cycle
.RFSW	BOOL	input - triggers end of home cycle for ladder input applications only
.AutoStart	BOOL	input - starts home cycle on loop closure if not already homed
.Mode	INT	input - chooses mode of homing - see later list
.PlusDir	BOOL	input - chooses initial direction of home motion
.Rate	UDINT	input - sets home rate in LU/Min
.DIM	DINT	input - position axis is set to once home cycle reaches its final target
.BackOff	BOOL	input - causes axis to reverse off switch if on it when cycle is started
.HDim	DINT	input - position axis will move to after reference if move option cho-sen
.MaxDist	DINT	input - Limit cycle motion will stop if limit not seen before this point
.Target	DINT	input - value of drive current / fol-lowing error used as hard stop detect
.FErrBW_Pct	DINT	input - following error limit is set to this during Limit cycle
.Current_Limit_Pct	DINT	input - current limit is set to this dur-ing Limit cycle
.HDne	BOOL	output - set when home cycle com-plete
.HAct	BOOL	output - set when home cycle is active
.HQue	USINT	output - queue number of home cycle
.SPos	DINT	
.Herr	INT	output - error number - see below
.StructureSize	UINT	used to test external and internal structures for the same size
	END_STRUCT	output - distance from home switch to marker pulse

Inputs must be provided from the ladder diagram. Many of these are fixed and can be set in Initial Values in Software Declarations or through a MOVE block or a ST Network, triggered by a one-shot on the first ladder scan. Others must be triggered from the ladder diagram using coils or MOVE functions. BOOL outputs may be used in the ladder diagram as contacts.

Notes on Choosing Type of Homing

Proximity Switch

Where axis travel is more than one revolution of the motor, use of either a proximity or limit switch is the most common arrangement. This can be connected to the drive I/O fast or regular input, or to a General or Block input.

Fast home #1 will provide the most repeatable accuracy using only a target switch. Its accuracy will be independent of speed and ladder scan time. Ladder home #11 allows the use of a General input point for the home switch but will vary in repeatability due to ladder scan time. In cases where no other drive I/O is used, it may save a breakout connection arrangement from the drive I/O connector.

The repeatability of both cycles can be improved by choosing to use the motor marker pulse after the switch. However, if the motor or any of the drive train components are replaced then the home offset value may need changing.

Where the switch used is an end of travel switch, use the reverse to marker option #15 or 16, or the move to HDIM options to back off the switch at the end of the home cycle.

Marker Only

Where the axis travel is only one rev of the motor, a marker only reference is possible.

Limit Setting

The axis may be moved against a mechanical stop to detect home. Doing this causes the drive current to rise and the position loop following error will also increase. Either one of these 2 quantities can be used to detect the stop mechanism. Once detected, the command position will be ahead of the target position by the amount of the following error. The equipment should not be left in this condition as it may cause damage. Always make sure that the axis is backed off the stop using either the reverse to marker or move to HDIM option.

Operator Home

The operator can jog the axis to a known position and assign a value using #50.

Homing Cycles using Fast Input on Drive

1 - Home to switch, preset DIM and stop

2 - Home to switch, continue to marker, preset DIM and stop

3 - Home to switch, preset DIM then move to HDIM

4 - Home to switch, continue to marker, preset DIM then move to HDIM

Homing Cycles using non Fast Input (Ladder Home)

11 - Home to switch, preset DIM and stop

12 - Home to switch, continue to marker, preset DIM and stop

13 - Home to switch, preset DIM then move to HDIM

14 - Home to switch, continue to marker, preset DIM then move to HDIM

15 - Home to switch, back to 1st marker after leaving switch, preset DIM and stop

16 - Home to switch, back to 1st marker after leaving switch, preset DIM, move to HDIM

Homing Cycles using Motor Feedback Marker Only

21 - Home to marker, preset DIM and stop

22 - Home to marker, preset DIM and move to HDIM

Homing Cycles to a Hard Mechanical Stop using Current

31 - Home to drive current target, preset DIM and stop

32 - Home to drive current target, reverse to marker, preset DIM and stop

33 - Home to drive current target, preset DIM, move to HDIM and stop

34 - Home to drive current target, rev to marker, preset DIM, move to HDIM and stop

Homing Cycles to a Hard Mechanical Stop using Following Error

41 - Home to following error target, preset DIM and stop

42 - Home to following error target, reverse to marker, preset DIM and stop

43 - Home to following error target, preset DIM, move to HDIM and stop

44 - Home to following error target, rev to marker, preset DIM, move to HDIM and stop

Operator Established Home

50 - Part Reference to DIM on Home start - home complete immediately

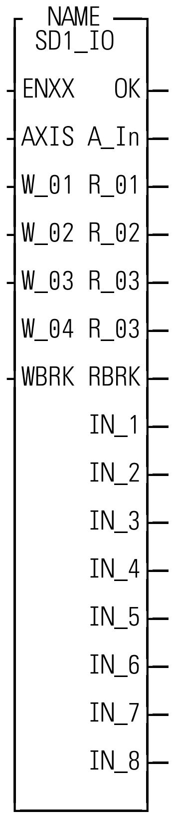
ERR Errors

These are ladder programming errors caused by mismatched structure sizes. The table below describes the different errors that may occur.

ERR	Description
1	Mismatch size in Stat structure between SD_STAT1 and SD_AXIS
2	Mismatch size in Flts structure between SD_STAT1 and SD_AXIS
3	Mismatch size in Warn structure between SD_STAT1 and SD_AXIS
4	Miscellaneous FB not OK conditions - SD_AXIS OK will not be ON
10	Mismatch size in Stat structure between SD_AXIS and main ladder
11	Mismatch size in Flts structure between SD_AXIS and main ladder
12	Mismatch size in Warn structure between SD_AXIS and main ladder
13	Mismatch size in DrIO structure between SD_AXIS and main ladder
14	Mismatch size in Strt structure between SD_AXIS and main ladder
15	Mismatch size in HDat structure between SD_AXIS and main ladder
16	Mismatch size in JDat structure between SD_AXIS and main ladder

HERR Errors

HERR	Description
1	Final home move point is beyond hard stop
2	Invalid option selected
3	Active queue not available to start home move
4	Failed to read / write rollover or expanded following error limit
5	Distance move did not start
6	Hard stop reached - no move in progress
7	Hard stop not reached - distance move ended first
8	Distance move abort failed
9	Part Reference failed
10	Move to index mark did not start
11	Reference End function not OK
12	Move to final position did not start
13	Home cycle stopped from Stop input
14	Axis C stop or P stop occurred during home cycle
15	Restore of original rollover or following error limit failed
16	Operator Part Reference Failed



Inputs:

ENxx (BOOL) - enables execution

AXIS (USINT) - identifies axis

W_01 (BOOL) - value written to drive output 1

W_02 (BOOL) - value written to drive output 2

W_03 (BOOL) - value written to drive output 3

W_04 (BOOL) - value written to drive output 4

WBRK (BOOL) - value written to drive brake output

Outputs:

OK (BOOL) - execution complete

A_IN (DINT) - value of drive analog input

R-01 (BOOL) - current state of drive output 1

R-02 (BOOL) - current state of drive output 2

R-03 (BOOL) - current state of drive output 3

R-04 (BOOL) - current state of drive output 4

R-BRK (BOOL) - current state of drive output brake

IN_1 (BOOL) - current state of drive input 1

IN_2 (BOOL) - current state of drive input 2

IN_3 (BOOL) - current state of drive input 3

IN_4 (BOOL) - current state of drive input 4

IN_5 (BOOL) - current state of drive input 5

IN_6 (BOOL) - current state of drive input 6

IN_7 (BOOL) - current state of drive input 7

IN_8 (BOOL) - current state of drive input 8

<<INSTANCE NAME>>:SD_10(ENXX=1,Axis:=<<W_01:=<<BOOOL>>, W02:=<<BOOL>>,W_03:=<<BOOL>>,W_04:=<<BOOL>>,W_BRK:=<<BOOL>>,OK=><<BOOL>>,A_In=><<DINT>>,R_01=><<BOOL>>,R_02=><<BOOL>>,R_03=><<BOOL>>,R_04=><<BOOL>>,R_BRK=><<BOOL>>,In_1=><<BOOL>>,In_2=><<BOOL>>,In_3=><<BOOL>>,In_4=><<BOOL>>,In_5=><<BOOL>>,In_6=><<BOOL>>,In_7=><<BOOL>>,In_8=><<BOOL>>

This function block reads the current values of the analog and DC inputs and reads/writes the outputs for the MMC Digital Smart Drive.

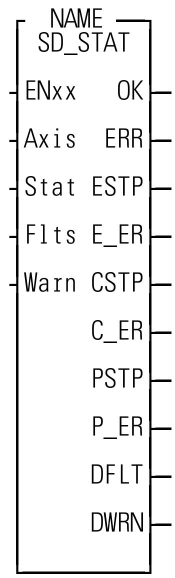
NOTE: The MMC Application Input/Output assignment must be checked in the drive setup for the inputs and outputs to be read/written by this function. See READ_SV variables 70, 71, and 72 for more details.

INPUTS

Name	**Data Type**	**Description or Function Used for Data**
ENxx	BOOL	Enables Execution
Axis	USINT	Axis Number
W_01	BOOL	Value written to drive output 1, READ_SV variable 72
W_02	BOOL	Value written to drive output 2, READ_SV variable 72
W_03	BOOL	Value written to drive output 3, READ_SV variable 72
W_04	BOOL	Value written to drive output 4, READ_SV variable 72
WBRK	USINT	Value written to drive brake output, READ_SV variable 72

OUTPUTS:

Name	**Data Type**	**Description or Function Used for Data**
OK	BOOL	Execution complete without error
A_IN	DINT	Value of drive analog input, READ_SV variable 71
R_01	BOOL	Current state of drive output 1, READ_SV variable 72
R_02	BOOL	Current state of drive output 2, READ_SV variable 72
R_03	BOOL	Current state of drive output 3, READ_SV variable 72
R_04	BOOL	Current state of drive output 4, READ_SV variable 72
RBRK	BOOL	Current state of drive brake output, READ_SV variable 72
IN_1	BOOL	Current state of drive input 1, READ_SV variable 71
IN_2	BOOL	Current state of drive input 2, READ_SV variable 71
IN_3	BOOL	Current state of drive input 3, READ_SV variable 71
IN_4	BOOL	Current state of drive input 4, READ_SV variable 71
IN_5	BOOL	Current state of drive input 5, READ_SV variable 71
IN_6	BOOL	Current state of drive input 6, READ_SV variable 71
IN_7	BOOL	Current state of drive input 7, READ_SV variable 71
IN_8	BOOL	Current state of drive input 8, READ_SV variable 71



Inputs:

ENxx (BOOL) - enables execution

Axis (USINT) - identifies axis

Stat (STRUCT) - provides axis and drive status information

Flts (STRUCT) - provides axis and drive fault information

Warn (STRUCT) - provides axis and drive warning information

Outputs:

OK (BOOL) - execution complete

ERR (INT) - reports an error number, see error listing below

ESTP (BOOL) - indicates an E-Stop is active

E_ER (WORD) - identifies E-Stop errors

CSTP (BOOL) - indicates a C-Stop is active

C_ER (WORD) - identifies C-Stop errors

PSTP (BOOL) - indicates a P-Stop is active

P_ER (WORD) - identifies P-Stop errors

DFLT (UINT) - identifies Drive Fault Code

DWRN (UINT) - identifies Drive Warning Code

<<INSTANCE NAME>>:SD_STAT(ENXX:=1, Axis := <<USINT>>, Stat := <<MEMORY AREA>>, Flts := <<MEMORY AREA>>, Warn := <<MEMORY AREA>>, OK => <<BOOL>, ERR => <<INT>>, ESTP => <<BOOL>>, E_ER => <<WORD>>, CSTP => <<BOOL>>, C_ER => <<WORD>>, PSTP => <<BOOL>>, P_ER => <<WORD>>, DFLT => <<UINT>>, DWRN => <<UINT>>);

This function block provides status and fault information using the READ_SV variables and other common Motion functions for an MMC Digital Servo Axis. The structure inputs for this ASFB are quite large and need to match the descriptions below to function correctly. Instead of re-entering these declarations we suggest that you either start with one of the Digital Smart Drive Examples included with the PiCPro 15.0 or higher Applications CD or copy and paste the appropriate networks from the example into your ladder.

NOTE: The Stat.StructureSize, Flts.StructureSize, and Warn.StructureSize variables must set to the size of the structures in bytes. Use the SIZEOF function to determine the size of the data structure as shown in the example ladders. The ASFB compares the sizes of the structures with the internal structures to make sure that they match. If they don't match an error will be reported and the function will not execute.

INPUTS:

NAME	DATA TYPE	DESCRIPTION OR FUNCTION USED FOR DATA
ENxx	BOOL	Enables Execution
Axis	USINT	Axis Number
Stat	STRUCT	Value written to drive output 1, READ_SV variable 72
.InPosition	BOOL	IN_POS?
.QueAvailable	BOOL	Q_AVAIL?
.ActiveQue	USINT	Q_NUMBER
.MoveType	DINT	READ_SV variable 2
.Actual Position	DINT	READ_SV variable 1
.CommandedPosition	DINT	READ_SV variable 3
.PositionError	DINT	READ_SV variable 4
.FilterError	DINT	READ_SV variable 5
.MotorCurrent	DINT	READ_SV variable 73
.MotorAvgCurrent	DINT	READ_SV variable 74
.MotorTemp	DINT	READ_SV variable 77
.StartupCommutationComplete	BOOL	READ_SV variable 67 Bit 0
.AtZeroSpeed	BOOL	READ_SV variable 67 Bit 1
.InSpeedWindow	BOOL	READ_SV variable 67 Bit 2
.UpToSpeed	BOOL	READ_SV variable 67 Bit 3
.AtPlusCurrentLimit	BOOL	READ_SV variable 67 Bit 4
.AtMinusCurrentLimit	BOOL	READ_SV variable 67 Bit 5
.BusCharged	BOOL	READ_SV variable 67 Bit 6
.DriveEnabled	BOOL	READ_SV variable 67 Bit 7
.DriveReady	BOOL	READ_SV variable 67 Bit 8
.BrakeReleased	BOOL	READ_SV variable 67 Bit 9
.DriveFault	BOOL	READ_SV variable 67 Bit 10
.DriveWarning	BOOL	READ_SV variable 67 Bit 11
.ShuntOn	BOOL	READ_SV variable 67 Bit 12
.DriveReadyAndBusCharged	BOOL	READ_SV variable 67 Bit 13
.Reserved	BOOL (0..5)	READ_SV variable 67 Bits 14-23
.HardwareEnableLine	BOOL	READ_SV variable 67 Bit 24
.AuxiliaryFeedbackLossOfFeedback	BOOL	READ_SV variable 67 Bit 25
.Reserved_26_31	BOOL (0..9)	READ_SV variable 67 Bits 26-31

StructureSize	UINT	Set to the Size of STAT Structure in Bytes END_STUCT

NAME	DATA TYPE	DESCRIPTION OR FUNCTION USED FOR DATA
Flts	STRUCT	Axis and Drive Faults
.EstopActive	BOOL	E_STOP?
.EstopErr	WORD	E_ERRORS
.LossOfFeedbackEstop	BOOL	E_ERRORS Bit 0
.ExcessPositionErrorEstop	BOOL	E_ERRORS Bit 1
.SlaveOverflowDeltaEstop	BOOL	E_ERRORS Bit 2
.UserLadderEstop	BOOL	E_ERRORS Bit 3
.SERCOSDriveEstop	BOOL	E_ERRORS Bit 4
.SERCOSCycDataEstop	BOOL	E_ERRORS Bit 5
.ASIU_Timeout	BOOL	E_ERRORS Bit 6
.DriveFaultEstop	BOOL	E_ERRORS Bit 7
.DriveCommFaultEstop	BOOL	E_ERRORS Bit 8
.ReservedEstopbits	BOOL (0..5)	E_ERRORS Bits 9-14
.AnyEstopBitSet	BOOL	E_ERRORS Bit 15
.CstopActive	BOOL	C_STOP?
.CstopErr	WORD	C_ERRORS
.SWPositiveEndlimitCstop	BOOL	C_ERRORS Bit 0
.SWNegativeEndlimitCstop	BOOL	C_ERRORS Bit 1
.UserLadderCstop	BOOL	C_ERRORS Bit 2
.MachRefDimCstop	BOOL	C_ERRORS Bit 3
.FeedrateCstop	BOOL	C_ERRORS Bit 4
.DistPosnDimCstop	BOOL	C_ERRORS Bit 5
.PartRefDimCstop	BOOL	C_ERRORS Bit 6
.PartRefCstop	BOOL	C_ERRORS Bit 7
.ReservedCstopbits	BOOL (0..6)	C_ERRORS Bits 8-14
.AnyCstopBitSet	BOOL	C_ERRORS Bit 15
.PstopActive	BOOL	P_STOP?
.PstopErr	WORD	P_ERRORS
.MasterStartPosnPstop	BOOL	P_ERRORS Bit 0
.ReservedBit1Pstop	BOOL	P_ERRORS Bit 1
.ReservedBit2Pstop	BOOL	P_ERRORS Bit 2
.ReservedBit3Pstop	BOOL	P_ERRORS Bit 3
.ReservedBit4Pstop	BOOL	P_ERRORS Bit 4
.MasterNotAvailPstop	BOOL	P_ERRORS Bit 5

.InvalidProfilePstop	BOOL	P_ERRORS Bit 6
.FastQuePstop	BOOL	P_ERRORS Bit 7
.SlaveDistPstop	BOOL	P_ERRORS Bit 8
.MasterDistPstop	BOOL	P_ERRORS Bit 9
.SlaveBeyondStrtPointPstop	BOOL	P_ERRORS Bit 10
.MasterBeyondStrtPointPstop	BOOL	P_ERRORS Bit 11
.ReservedBit13Pstop	BOOL	P_ERRORS Bit 12
.Reserved Bit14Pstop	BOOL	P_ERRORS Bit 13
.Reserved Bit15Pstop	BOOL	P_ERRORS Bit 14
.AnyPstopBitSet	BOOL	P_ERRORS Bit 15
.NoDriveFaults	BOOL	Set when no drive fault is active
.DriveMemoryFault_11	BOOL	READ_SV variable 68 Bit 0
.DriveBusOverVoltageFault_12	BOOL	READ_SV variable 68 Bit 1
.DrivePM1OverCurrentFault_13	BOOL	READ_SV variable 68 Bit 2
.DriveOverPowerFault_14	BOOL	READ_SV variable 68 Bit 3
.MotorTempFault_15	BOOL	READ_SV variable 68 Bit 4
.ContinuousCurrentFault_16	BOOL	READ_SV variable 68 Bit 5
.DriveHeatsinkTempFault_17	BOOL	READ_SV variable 68 Bit 6
.Reserved1	BOOL	READ_SV variable 68 Bit 7
.DriveF1FeedbackFault_22	BOOL	READ_SV variable 68 Bit 8
.DriveAmbientTempFault_23	BOOL	READ_SV variable 68 Bit 9
.MotorCalculatedTempFault_24	BOOL	READ_SV variable 68 Bit 10
.DriveTimingFault_25	BOOL	READ_SV variable 68 Bit 11
.DriveInterfaceFault_26	BOOL	READ_SV variable 68 Bit 12
.UserSetFault_27	BOOL	READ_SV variable 68 Bit 13
.DriveF1CommunicationFault_31	BOOL	READ_SV variable 68 Bit 14
.OverSpeedFault_32	BOOL	READ_SV variable 68 Bit 15
.OverCurrentFault_33	BOOL	READ_SV variable 68 Bit 16
.ControlPanelDisconnectFault_34	BOOL	READ_SV variable 68 Bit 17
.DrivePowerModuleFault_35	BOOL	READ_SV variable 68 Bit 18
.DriveSetupDataFault_36	BOOL	READ_SV variable 68 Bit 19
.Reserved2	BOOL	READ_SV variable 68 Bit 20
.DriveRelayFault_41	BOOL	READ_SV variable 68 Bit 21
.DrivePM2OverCurrentFault_42	BOOL	READ_SV variable 68 Bit 22
.DrivePMTempFault_43	BOOL	READ_SV variable 68 Bit 23
.MotorGroundFault_44	BOOL	READ_SV variable 68 Bit 24
.DriveACInputOverVoltageFault_45	BOOL	READ_SV variable 68 Bit 25
.OvertravelPlusFault_46	BOOL	READ_SV variable 68 Bit 26
.OvertravelMinusFault_47	BOOL	READ_SV variable 68 Bit 27
.DigitalLinkCommunicationError_51	BOOL	READ_SV variable 68 Bit 28

.InvalidSwitchSettingFault_52	BOOL	READ_SV variable 68 Bit 29
.Reserved3	BOOL	READ_SV variable 68 Bit 30
.Reserved4	BOOL	READ_SV variable 68 Bit 31
.StuctureSize	UINT	Set to the Size of Flts Structure in Bytes
	END_STUCT	

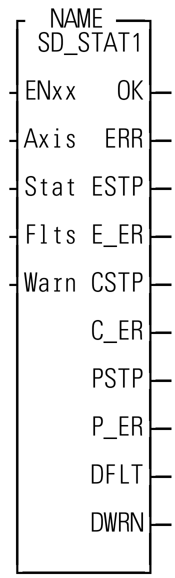
NAME	DATA TYPE	DESCRIPTION OR FUNCTION USED FOR DATA
Warn	STRUCT	Drive Warnings, see MMC Smart Drive and Digital MMC Control Hardware Manual for detailed descriptions
.Nowarnings	BOOL	Set when no drive warning is active
.DriveHeatsinkTempWarning_01	BOOL	READ_SV variable 69 Bit 0
.DriveAmbientTempWarning_02	BOOL	READ_SV variable 69 Bit 1
.MotorTempWarning_03	BOOL	READ_SV variable 69 Bit 2
.MotorCalculatedTempWarning_04	BOOL	READ_SV variable 69 Bit 3
.LossofFeedbackonAuxWarning_05	BOOL	READ_SV variable 69 Bit 4
.Reserved	BOOL (0..26)	READ_SV variable 69 Bits 5-31
.StructureSize	UINT	Set to the Size of Warn Structure in Bytes
	END STUCT	

OUTPUTS:

NAME	DATA TYPE	DESCRIPTION OR FUNCTION USED FOR DATA
OK	BOOL	Execution complete without error
ERR	INT	Reports error number, see error listing below
ESTP	BOOL	Indicates an E-Stop is active
E_ER	WORD	Identifies E-Stop errors, see E_ERRORS
CSTP	BOOL	Indicates a C-Stop is active
C_ER	WORD	Identifies C-Stop errors; see C_ERRORS
PSTP	BOOL	Indicates a P-Stop is active
P_ER	WORD	Identifies P-Stop errors, see P_ERRORS
DFLT	UINT	Identifies Drive Fault Code, see the MMC Smart Drive and Digital MMC Control Hardware Manual for detailed descriptions
DWRN	UINT	Identifies Drive Warning Code, see the MMC Smart Drive and Digital MMC Control Hardware Manual for detailed description

ERROR CODES FOR ERR:

ERR	Error Description
1	Invalid value for Stat.StructureSize, the Stat structure has the wrong number of elements and/or the wrong data types
2	Invalid value for Flts.StructureSize, the Flts structure has the wrong number of elements and/or the wrong data types
3	Invalid value for Warn.StructureSize, the Warn structure has the wrong number of elements and/or the wrong data types
4	One or more of the internal motion functions failed, check to make sure that the axis has been initialized



Inputs:

ENxx (BOOL) - enables execution (xx indicates revision #)

Axis (USINT) - identifies axis

Stat (STRUCT) - provides axis and drive status information

Flts (STRUCT) - provides axis and drive fault information

Warn (STRUCT) - provides axis and drive warning information

Outputs:

OK (BOOL) - execution complete

ERR (INT) - reports an error number, see error listing below

ESTP (BOOL) - indicates an E-Stop is active

E_ER (WORD) - identifies E-Stop errors

CSTP (BOOL) - indicates a C-Stop is active

C_ER (WORD) - identifies C-Stop errors

PSTP (BOOL) - indicates a P-Stop is active

P_ER (WORD) - identifies P-Stop errors

DFLT (UINT) - identifies Drive Fault Code

DWRN (UINT) - identifies Drive Warning Code

<<INSTANCE NAME>>:SD_STAT1(ENxx :=1, Axis := <<USINT>>, Stat := <<MEMORY AREA>>, Flts := <<MEMORY AREA>>, Warn := <<MEMORY AREA>>, OK => <<BOOL>, ERR => <<INT>>, ESTP => <<BOOL>>, E_ER => <<WORD>>, CSTP => <<BOOL>>, C_ER => <<WORD>>, PSTP => <<BOOL>>, P_ER => <<WORD>>, DFLT => <<UINT>>, DWRN => <<UINT>>);

This function block provides status and fault information using the READ_SV variables and other common Motion functions for an MMC Digital Servo Axis. The structure inputs for this ASFB are quite large and need to match the descriptions below to function correctly. Instead of re-entering these declarations we suggest that you either start with one of the Digital Smart Drive Examples included with the PiCPro 16.0 or higher Applications CD or copy and paste the appropriate networks from the example into your ladder.

NOTE: The Stat.StructureSize, Flts.StructureSize, and Warn.StructureSize variables must set to the size of the structures in bytes. Use the SIZEOF function to determine the size of the data structure as shown in the example ladders. The ASFB compares the sizes of the structures with the internal structures to make sure that they match. If they don't match an error will be reported and the function will not execute.

INPUTS:

NAME	DATA TYPE	DESCRIPTION OR FUNCTION USED FOR DATA
ENxx	BOOL	Enables Execution
Axis	USINT	Axis Number
Stat	STRUCT	Value written to drive output 1, READ_SV variable 72
.InPosition	BOOL	IN_POS?
.QueAvailable	BOOL	Q_AVAIL?
.ActiveQue	USINT	Q_NUMBER
.MoveType	DINT	READ_SV variable 2
.Actual Position	DINT	READ_SV variable 1
.CommandedPosition	DINT	READ_SV variable 3
.PositionError	DINT	READ_SV variable 4
.FilterError	DINT	READ_SV variable 5
.MotorCurrent	DINT	READ_SV variable 73
.MotorAvgCurrent	DINT	READ_SV variable 74
.MotorTemp	DINT	READ_SV variable 77
.StartupCommutationComplete	BOOL	READ_SV variable 67 Bit 0
.AtZeroSpeed	BOOL	READ_SV variable 67 Bit 1
.InSpeedWindow	BOOL	READ_SV variable 67 Bit 2
.UpToSpeed	BOOL	READ_SV variable 67 Bit 3
.AtPlusCurrentLimit	BOOL	READ_SV variable 67 Bit 4
.AtMinusCurrentLimit	BOOL	READ_SV variable 67 Bit 5
.BusCharged	BOOL	READ_SV variable 67 Bit 6
.DriveEnabled	BOOL	READ_SV variable 67 Bit 7
.DriveReady	BOOL	READ_SV variable 67 Bit 8
.BrakeReleased	BOOL	READ_SV variable 67 Bit 9
.DriveFault	BOOL	READ_SV variable 67 Bit 10
.DriveWarning	BOOL	READ_SV variable 67 Bit 11

.ShuntOn	BOOL	READ_SV variable 67 Bit 12
.DriveReadyAndBusCharged	BOOL	READ_SV variable 67 Bit 13
.Reserved	BOOL (0..5)	READ_SV variable 67 Bits 14-23
.HardwareEnableLine	BOOL	READ_SV variable 67 Bit 24
AuxiliaryFeedbackLossOfFeedback	BOOL	READ_SV variable 67 Bit 25
.Reserved_26_31	BOOL (0..9)	READ_SV variable 67 Bits 26-31
StructureSize	UINT	Set to the Size of STAT Structure in Bytes
	END_STRUCT	

NAME	DATA TYPE	DESCRIPTION OR FUNCTION USED FOR DATA
Flts	STRUCT	Axis and Drive Faults
.EstopActive	BOOL	E_STOP?
.EstopErr	WORD	E_ERRORS
.LossOfFeedbackEstop	BOOL	E_ERRORS Bit 0
.ExcessPositionErrorEstop	BOOL	E_ERRORS Bit 1
.SlaveOverflowDeltaEstop	BOOL	E_ERRORS Bit 2
.UserLadderEstop	BOOL	E_ERRORS Bit 3
.SERCOSDriveEstop	BOOL	E_ERRORS Bit 4
.SERCOSCycDataEstop	BOOL	E_ERRORS Bit 5
.ASIU_Timeout	BOOL	E_ERRORS Bit 6
.DriveFaultEstop	BOOL	E_ERRORS Bit 7
.DriveCommFaultEstop	BOOL	E_ERRORS Bit 8
.ReservedEstopbits	BOOL (0..5)	E_ERRORS Bits 9-14
.AnyEstopBitSet	BOOL	E_ERRORS Bit 15
.CstopActive	BOOL	C_STOP?
.CstopErr	WORD	C_ERRORS
.SWPositiveEndlimitCstop	BOOL	C_ERRORS Bit 0
.SWNegative EndlimitCstop	BOOL	C_ERRORS Bit 1
.UserLadderCstop	BOOL	C_ERRORS Bit 2
.MachRefDimCstop	BOOL	C_ERRORS Bit 3
.FeedrateCstop	BOOL	C_ERRORS Bit 4
.DistPosnDimCstop	BOOL	C_ERRORS Bit 5
.PartRefDimCstop	BOOL	C_ERRORS Bit 6
.PartRefCstop	BOOL	C_ERRORS Bit 7
.ReservedCstopbits	BOOL (0..6)	C_ERRORS Bits 8-14
.AnyCstopBitSet	BOOL	C_ERRORS Bit 15
.PstopActive	BOOL	P_STOP?
.PstopErr	WORD	P_ERRORS

.MasterStartPosnPstop	BOOL	P_ERRORS Bit 0
.ReservedBit1Pstop	BOOL	P_ERRORS Bit 1
.ReservedBit2Pstop	BOOL	P_ERRORS Bit 2
.ReservedBit3Pstop	BOOL	P_ERRORS Bit 3
.ReservedBit4Pstop	BOOL	P_ERRORS Bit 4
.MasterNotAvailPstop	BOOL	P_ERRORS Bit 5
.InvalidProfilePstop	BOOL	P_ERRORS Bit 6
.FastQuePstop	BOOL	P_ERRORS Bit 7
.SlaveDistPstop	BOOL	P_ERRORS Bit 8
.MasterDistPstop	BOOL	P_ERRORS Bit 9
.SlaveBeyondStrtPointPstop	BOOL	P_ERRORS Bit 10
.MasterBeyondStrtPointPstop	BOOL	P_ERRORS Bit 11
.ReservedBit13Pstop	BOOL	P_ERRORS Bit 12
.ReservedBit14Pstop	BOOL	P_ERRORS Bit 13
.ReservedBit15Pstop	BOOL	P_ERRORS Bit 14
.AnyPstopBitSet	BOOL	P_ERRORS Bit 15
.DriveFaultDW	DWORD	READ_SV variable 68 converted to a DWORD
.DFLT	UINT	Identifies Drive Fault Code, see the MMC Smart Drive and Digital MMC Control Hardware Manual for detailed description
.NoDriveFaults	BOOL	Set when no drive fault is active
.DriveMemoryFault_11	BOOL	READ_SV variable 68 Bit 0
.DriveBusOverVoltageFault_12	BOOL	READ_SV variable 68 Bit 1
.DrivePM1OverCurrentFault_13	BOOL	READ_SV variable 68 Bit 2
.DriveBusUnderVoltageFault_14	BOOL	READ_SV variable 68 Bit 3
.MotorTempFault_15	BOOL	READ_SV variable 68 Bit 4
.ContinuousCurrentFault_16	BOOL	READ_SV variable 68 Bit 5
.DriveHeatsinkTempFault_17	BOOL	READ_SV variable 68 Bit 6
.DriveF2FeedbackFault_21	BOOL	READ_SV variable 68 Bit 7
.DriveF1FeedbackFault_22	BOOL	READ_SV variable 68 Bit 8
.DriveAmbientTempFault_23	BOOL	READ_SV variable 68 Bit 9
.MotorCalculatedTempFault_24	BOOL	READ_SV variable 68 Bit 10
.DriveTimingFault_25	BOOL	READ_SV variable 68 Bit 11
.DriveInterfaceFault_26	BOOL	READ_SV variable 68 Bit 12
.UserSetFault_27	BOOL	READ_SV variable 68 Bit 13
.DriveF1CommunicationFault_31	BOOL	READ_SV variable 68 Bit 14
.OverSpeedFault_32	BOOL	READ_SV variable 68 Bit 15
.OverCurrentFault_33	BOOL	READ_SV variable 68 Bit 16
.ControlPanelDisconnectFault_34	BOOL	READ_SV variable 68 Bit 17

.DrivePowerModuleFault_35	BOOL	READ_SV variable 68 Bit 18
.FeedbackTypeMismatchFault_36	BOOL	READ_SV variable 68 Bit 19
.EndatFault_37	BOOL	READ_SV variable 68 Bit 20
.DriveRelayFault_41	BOOL	READ_SV variable 68 Bit 21
.DrivePM2OverCurrentFault_42	BOOL	READ_SV variable 68 Bit 22
.DrivePMTempFault_43	BOOL	READ_SV variable 68 Bit 23
.MotorGroundFault_44	BOOL	READ_SV variable 68 Bit 24
.DriveACInputOverVoltageFault_45	BOOL	READ_SV variable 68 Bit 25
.OvertravelPlusFault_46	BOOL	READ_SV variable 68 Bit 26
.OvertravelMinusFault_47	BOOL	READ_SV variable 68 Bit 27
.DigitalLinkCommunicationError_51	BOOL	READ_SV variable 68 Bit 28
.InvalidSwitchSettingFault_52	BOOL	READ_SV variable 68 Bit 29
.HardwareFailureFault_53	BOOL	READ_SV variable 68 Bit 30
.Reserved4	BOOL	READ_SV variable 68 Bit 31
.StuctureSize	UINT	Set to the Size of Flts Structure in Bytes
	END_STRUCT	

NAME	DATA TYPE	DESCRIPTION OR FUNCTION USED FOR DATA
Warn	STRUCT	Drive Warnings, see MMC Smart Drive and Digital MMC Control Hardware Manual for detailed descriptions
.DriveWarnDW	DWORD	READ_SV variable 69 converted to a DWORD
.DWRN	UINT	Same as DWRN output. Identifies Drive Warning Code, see the MMC Smart Drive and Digital
.Nowarnings	BOOL	Set when no drive warning is active
.DriveHeatsinkTempWarning_01	BOOL	READ_SV variable 69 Bit 0
.DriveAmbientTempWarning_02	BOOL	READ_SV variable 69 Bit 1
.MotorTempWarning_03	BOOL	READ_SV variable 69 Bit 2
.MotorCalculatedTempWarning_04	BOOL	READ_SV variable 69 Bit 3
.OvertravelPlusWarning_05	BOOL	READ_SV variable 69 Bit 4
.OvertravelMinusWarning_06	BOOL	READ_SV variable 69 Bit 5
.Reserved	BOOL (0..25)	READ_SV variable 69 Bits 6-31
.StructureSize	UINT	Set to the Size of Warn Structure in Bytes
	END_STRUCT	

OUTPUTS:

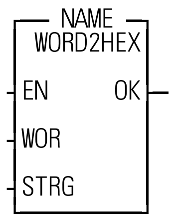
NAME	DATA TYPE	DESCRIPTION OR FUNCTION USED FOR DATA
OK	BOOL	Execution complete without error
ERR	INT	Reports error number, see error listing below
ESTP	BOOL	Indicates an E-Stop is active
E_ER	WORD	Identifies E-Stop errors, see E_ERRORS
CSTP	BOOL	Indicates a C-Stop is active
C_ER	WORD	Identifies C-Stop errors; see C_ERRORS
PSTP	BOOL	Indicates a P-Stop is active
P_ER	WORD	Identifies P-Stop errors, see P_ERRORS
DFLT	UINT	Identifies Drive Fault Code, see the MMC Smart Drive and Digital MMC Control Hardware Manual for detailed descriptions
DWRN	UINT	Identifies Drive Warning Code, see the MMC Smart Drive and Digital MMC Control Hardware Manual for detailed description

ERROR CODES FOR ERR:

ERR	Error Description
1	Invalid value for Stat.StructureSize, the Stat structure has the wrong number of elements and/or the wrong data types
2	Invalid value for Flts.StructureSize, the Flts structure has the wrong number of elements and/or the wrong data types
3	Invalid value for Warn.StructureSize, the Warn structure has the wrong number of elements and/or the wrong data types
4	One or more of the internal motion functions failed, check to make sure that the axis has been initialized

WORD2HEX

Converts a word to a hex value **USER/M_COMMON**



Inputs: EN (BOOL) - enables execution

WORD (WORD) - value to convert

STRG (STRING) - Converted value

Outputs: OK (BOOL) - execution complete

<<INSTANCE NAME>>:WORD2HEX(EN00 := <<BOOL>>, WORD := << WORD>> STRG := <<STRING>>, OK => <<BOOL>>);

This function block places the hexadecimal notation of the value at WORD into the string at STRG.

Example: If 26,854 is entered at the WORD input, 68E6 will be reported at STRG.

NOTES

APPENDIX A M_DSMCOM Commands

This appendix contains the commands that can be entered at the CMD input of the M_DSMCOM function block. These commands allow you to communicate with the DSM100 drive over the communications port. The tables that follow contain detailed descriptions of the commands, applicable values, responses, and exceptions.

Exception Responses

If a command is received by the drive without a communication error, but cannot be processed normally, an exception response is generated. The table below lists the possible exception responses.

Response Data	Exception Type	Description	Applicable Commands
01	Invalid Data	The command data parameter was unacceptable, and the parameter was not changed in the drive.	Non-Range Variable Commands, Low Level Commands
02	Command Not Enabled	The command is disabled and is dependent on another command for enabling.	Manufacturing, Firmware Upgrades
03	EEPROM Write Error	The command required a write to EEPROM, and the data was not able to be written.	All
04	Data Accepted After Limiting to Minimum	The command data was out of range, but was modified to the minimum value.	Range Variable Commands
05	Data Accepted After Limiting to Maximum	The command data was out of range, but was modified to the maximum value.	Range Variable Commands
06	Command Disabled When Drive is Enabled	The command cannot be complied with, because the drive is enabled.	All
07	Flash Programming Error	The command required the flash memory to be altered and an error occurred.	Flash Memory Altering Commands
08	Invalid Function Code	The master function code was not recognized by the drive.	All
09	Command Disabled When Drive is Disabled	The command cannot be complied with, because the drive is disabled.	All

Host Command Set

The tables below use the following symbols to specify data widths.

Note: Every byte in the specified data field is sent in the command encoded as two ASCII-hex characters):

Data	Signed	Unsigned
8-bit data	[c1]..[cn]	[b1]..[bn]
16-bit data	[s1]..[sn]	[w1]..[wn]
32-bit data	[L1]..[Ln]	[d1]..[dn]

On numeric parameters, the Range of Data Values field contains the range of values in user units and the resolution [denoted by (ε: xxxxxx)]. The field also contains the range of command hexadecimal values expected for the parameter.

In addition, the Units field contains the multiplier for converting the user units to the command hexadecimal values. These multipliers are presented in hexadecimal also.

Common Product Line Commands

These commands will remain consistent across product lines. The bit definitions on the Powerup Status command may change between products, but the command must return zero (00) on a successful powerup.

Parameter	Range of Data Values	Units	Command		Command Data	Response Data	Exception Responses
Product Type Identifies the type of product. Currently only the BCM-03 is known. but is provided for future expansion.	[b1] - Type 0- BCM-03	-	Read	000	-	[b1]	
			Write	-	-	-	
Powerup Status The status of the drive during power up testing. The bit definitions of the READ Powerup Status command may change between products, but the command will always return zero (00) on a successful powerup.	[b1] - Status 00 - Successful Power-Up 51 - Boot Block Checksum Error 52 - Non-Boot Block Checksum Error 53 - Uninitialized Personality EEPROM Error 54 - Personality EEPROM Read Error 55 - Personality EEPROM Data Corruption Error 56 - Main Processor Watchdog Error 57 - Sub Processor Watchdog Error 58 - Main Processor RAM Error 59 - Sub Processor RAM Error 60 - Uninitialized Service EEPROM Error 61 - Service EEPROM Read Error 62 - Service EEPROM Data Corruption Error 63 - Main Processor A/D Converter Error 64 - Sub Processor A/D Converter Error	-	Read	001	-	[b1]	
	65 - Analog1 Output Error 66 - Gate Array Error 67 - Analog2 Output Error 68 - Inter-Processor Communication Error 69 - Sub Processor Initialization Error 70 - Sub Processor SRAM Error 71 - Sub Processor Code Loading Error 72 - Sub Processor Startup Error 73 - Sub Processor Checksum Error 74 - Personality EEPROM Write Error 75 - Service EEPROM Write Error 76 - Software Clock Error 77 - Sub Processor Communication Checksum Error 78 - Sub Processor Sine Table Generation Error 79 - Personality Data Out Of Range 80 - Service Data Out Of Range 81 - Motor Block Checksum Error		Write	-	-	-	
Main Firmware Version The version number of the drive's main firmware.	[b1] - Major Version 0..255 (00..ff)	-	Read	002	-	[b1] [b2]	
	[b2] - Minor Revision 0..255 (00..ff)		Write	-	-	-	
Boot Firmware Version The version number of the drive's boot firmware.	[b1] - Major Version 0..255 (00..ff)	-	Read	003	-	[b1] [b2]	
	[b2] - Minor Revision 0..255 (00..ff)		Write	-	-	-	

General Commands

Parameter	Range of Data Values	Units	Command		Command Data	Response Data	Exception Responses
Reset Personality EEPROM Resets the personality EEPROM to its factory settings.	[No Data]	-	Read	-	-	-	
			Write	010	-	-	03, 06
Drive Name Identifies the drive in a multidrop system.	[b1]..[b32] - Name Name is a 32-character string.	-	Read	011	-	[b1]..[b32]	
			Write	012	[b1]..[b32]	-	03
Position Scale Value The position scale used by the host computer for scaling position variables. This information is not necessary for drive operation. The scale is a 32 bit IEEE floating point value.	[d1] - Value -3.4e+38..3.4e+38 (ε: 1.19e-7)	units / count	Read	013	-	[d1]	
	(80000000..7fffffff)		Write	014	[d1]	-	03
Position Scale Text The position scale text used by the host computer to identify the units of the position scale.	[b1]..[b8] - Name Name is an 8-character string.	-	Read	015	-	[b1]..[b8]	
			Write	016	[b1]..[b8]	-	03
Velocity Scale Value The velocity scale used by the host computer for scaling velocity variables. This information is not necessary for drive operation. The scale is a 32 bit IEEE floating point value.	[d1] - Value -3.4e+38..3.4e+38 (ε: 1.19e-7)	units / RPM	Read	017	-	[d1]	
	(80000000..7fffffff)		Write	018	[d1]	-	03
Velocity Scale Text The velocity scale text used by the host computer to identify the units of the velocity scale.	[b1]..[b8] - Name Name is an 8-character string.	-	Read	019	-	[b1]..[b8]	
			Write	01a	[b1]..[b8]	-	03
Acceleration Scale Value The acceleration scale used by the host computer for scaling acceleration variables. This information is not necessary for drive operation. The scale is a 32 bit IEEE floating point value.	[d1] - Value -3.4e+38..3.4e+38 (ε: 1.19e-7)	units /RPM /second	Read	01b	-	[d1]	
	(80000000..7fffffff)		Write	01c	[d1]	-	03
Acceleration Scale Text The acceleration scale text used by the host computer to identify the units of the acceleration scale.	[b1]..[b8] - Name Name is an 8-character string.	-	Read	01d	-	[b1]..[b8]	
			Write	01e	[b1]..[b8]	-	03
Torque Scale Value The torque scale used by the host computer for scaling torque variables. This information is not necessary for drive operation.	[d1] - Value -3.4e+38..3.4e+38 (ε: 1.19e-7)	units / Amp	Read	01f	-	[d1]	
The scale is a 32 bit IEEE floating point value.	(80000000..7fffffff)		Write	020	[d1]	-	03
Torque Scale Text The torque scale text used by the host computer to identify the units of the torque scale.	[b1]..[b8] - Name Name is an 8-character string.	-	Read	021	-	[b1]..[b8]	
			Write	022	[b1]..[b8]	-	03

Position Loop Commands

Parameter	Range of Data Values	Units	Command		Command Data	Response Data	Exception Responses
Position Loop Proportional Gain	[w1] - Gain 0.0..31.98 (ε: 7.8e-3)	in/min/mil	Read	030	-	[w1]	
	(0000..0fff)	(× 0080)	Write	031	[w1]	-	03, 05
Position Loop Integral Gain	[w1] - Gain 0..31.98 (ε: 7.8e-3)	-	Read	032	-	[w1]	
	(0000..0fff)	(× 0080)	Write	033	[w1]	-	03, 05
Position Loop Derivative Gain	[w1] - Gain 0..31.98 (ε: 7.8e-3)	-	Read	034	-	[w1]	
	(0000..0fff)	(× 0080)	Write	035	[w1]	-	03, 05
Position Loop Feedforward Gain	[w1] - Gain 0..200 (ε: 1)	-	Read	036	-	[w1]	
	(0000..00c8)	(× 0001)	Write	037	[w1]	-	03, 05
Integrator Zone Maximum position error which the integrator is still active. If the position error is greater than the I Zone, the integrator is reset	[w1] - Zone 0..32767 (ε: 1)	counts	Read	038	-	[w1]	
	(0000..7fff)	(× 0001)	Write	039	[w1]	-	03, 05
Position Window Size Maximum position error which allows the In Position flag to remain set.	[w1] - Size 0..32767 (ε: 1)	counts	Read	03a	-	[w1]	
	(0000..7fff)	(× 0001)	Write	03b	[w1]	-	03, 05
Position Window Time The minimum time which the position error must be less than the Position Window Size to set the In Position flag.	[b1] - Time 0..255 (ε: 1)	millisec-onds	Read	03c	-	[b1]	
	(00..ff)	(× 01)	Write	03d	[b1]	-	03
Position Error Limit Minimum position error which allows the excess Position Error flag to remain clear	[d1] - Limit 1..2147483647 (ε: 1)	counts	Read	03e	-	[d1]	
	(00000001..7fffffff)	(× 00000001)	Write	03f	[d1]	-	03, 04, 05
Position Error Time The minimum time which the position error must be greater than the Position Error Limit to cause an Excess Position Error fault.	[w1] - Time 0..65535 (ε: 1)	millisec-onds	Read	040	-	[w1]	
	(00..ffff)	(× 0001)	Write	041	[w1]	-	03
Gear Ratio The ratio between the motor and master counts for following.	[s1] - Motor -32767..32767 (ε: 1) (8001..7fff)	counts (× 0001)	Read	042	-	[s1] [s2]	
	[s2] - Master 1..32767 (ε: 1) (0001..7fff)	master counts (× 0001)	Write	043	[s1] [s2]	-	03, 04
Master Rotation The rotation direction of the master encoder in follower mode, and the polarity of the direction input in the step/direction mode.	[b1] - Direction 0 - forward direction (TP: Normal)	-	Read	044	-	[b1]	
	1 - reverse direction (TP: Reverse)		Write	045	[b1]	-	01, 03, 06
Slew Rate The acceleration limit for the motor when used in a follower mode.	[d1] - Rate 0..2147483647 (ε: 1)	RPM/sec	Read	046	-	[d1]	
	(00000000..7fffffff)	(× 00000001)	Write	047	[d1]	-	03, 05
Slew Enable Determines if the slew rate is used in follower mode	[b1] - Flag 0 - Disabled (TP: Off) 1 - Enabled (TP: On)	-	Read	048	-	[b1]	
			Write	049	[b1]	-	01, 03

Velocity Loop Commands

Parameter	Range of Data Values	Units	Command		Command Data	Response Data	Exception Responses
Velocity Loop Proportional Gain	[w1] - Gain 0..1000 (ε: 1)	-	Read	04a	-	[w1]	
	(0000..03e8)	(× 0001)	Write	04b	[w1]	-	03, 05
Velocity Loop Integral Gain	[w1] - Gain 0..1000 (ε: 1)	-	Read	04c	-	[w1]	
	(0000..03e8)	(× 0001)	Write	04d	[w1]	-	03, 05
Velocity Loop Derivative Gain	[s1] - Gain -1000..1000 (ε: 1)	-	Read	04e	-	[s1]	
	(fc18..03e8)	(× 0001)	Write	04f	[s1]	-	03, 04, 05
Zero Speed Limit Maximum motor velocity which allows the ZeroSpeed flag to remain set.	[d1] - Limit 0..32767.99998 (ε: 1.53e-5)	RPM	Read	050	-	[d1]	
	(00000000..7fffffff)	(× 00010000)	Write	051	[d1]	-	03, 05
Speed Window Size Maximum motor velocity error which allows the Speed Window flag to remain set.	[d1] - Limit 0..32767.99998 (ε: 1.53e-5)	RPM	Read	052	-	[d1]	
	(00000000..7fffffff)	(× 00010000)	Write	053	[d1]	-	03, 05
Over Speed Limit Minimum motor velocity which causes the Overspeed fault to occur.	[d1] - Limit 0..32767.99998 (ε: 1.53e-5)	RPM	Read	054	-	[d1]	
	(00000000..7fffffff)	(× 00010000)	Write	055	[d1]	-	03, 05
At Speed Limit Minimum motor velocity which causes the At Speed flag to be set	[d1] - Limit 0..32767.99998 (ε: 1.53e-5)	RPM	Read	056	-	[d1]	
	(00000000..7fffffff)	(× 00010000)	Write	057	[d1]	-	03, 05
Velocity Loop Update Period Velocity control loop execution period.	[b1] - Period 0 - 200 µsecond 1 - 400 µsecond 2 - 600 µsecond 3 - 800 µsecond	-	Read	058	-	[b1]	
	4 - 1000 µsecond 5 - 1200 µsecond 6 - 1400 µsecond 7 - 1600 µsecond		Write	059	[b1]	-	01, 03, 06
Velocity Error Limit Sets or returns the minimum velocity error which allows the Excess Velocity Error flag to remain clear.	[d1] - limit 0..32767.99998 (ε: 1.53e-5)	RPM	Read	05A	-	[d1]	03, 05
	(00000000..7fffffff)	(× 00010000)	Write	05B	[d1]	-	
Velocity Error Time Sets or returns the minimum time which the velocity error must be greater than the Velocity Error Limit to cause an Excess Velocity Error fault	[w1] - time 0..65535 (ε: 1.53e-5)	millisecond	Read	05C	-	[w1]	03
	(00.. ffff)	(x0001)	Write	05D	[w1]	-	

Torque Current Conditioning Commands

Parameter	Range of Data Values	Units	Command		Command Data	Response Data	Exception Responses
Low Pass Filter Bandwidth Cutoff frequency of the low pass filter.	[w1] - Bandwidth 1..992 (ε: 1)	Hz	Read	070	-	[w1]	
	(0001..03e0)	(× 0001)	Write	071	[w1]	-	03, 04, 05
Low Pass Filter Enable Determines if the low pass filter is used in the control loop.	[b1] - Flag 0 - Disabled (TP: Off)	-	Read	076	-	[b1]	
	1 - Enabled (TP: On)		Write	077	[b1]	-	01, 03
Software Positive Current Limit User specified positive current limit for the drive.	[w1] - Limit 0..255.992 (ε: 7.8e-3)	Amps	Read	07a	-	[w1]	
The minimum of this value, the peak rating of the drive, the peak rating of the motor, and the +ILIMIT analog input is used as the limiting value.	(0000..7fff)	(× 0080)	Write	07b	[w1]	-	03, 05
Software Negative Current Limit User specified negative current limit for the drive.	[w1] - Limit 0..255.992 (ε: 7.8e-3)	Amps	Read	07c	-	[w1]	
The minimum of this value, the peak rating of the drive, the peak rating of the motor, and the -ILIMIT analog input is used as the limiting value.	(0000..7fff)	(× 0080)	Write	07d	[w1]	-	03, 05
Continuous Current Limit User specified current faulting value.	[w1] - Limit 0..255.992 (ε: 7.8e-3)	Amps	Read	07e	-	[w1]	
This parameter is provided to allow a faulting current value which is less than the capacity of the drive and motor.	(0000..7fff)	(× 0080)	Write	07f	[w1]	-	03, 05
PWM Frequency Switching Disable Sets or returns the flag which indicates if the PWM frequency changes with the speed and current demands of the motor.	[b1] - Flag 00 - Enabled	-	Read	1A8	-	[b1]	
	01 - Disabled	-	Write	1A9	[b1]	-	01, 03

Motor Commands

Note: All Motor Commands other than *Motor ID* are disabled and return the exception response 02 unless Motor ID is set to 65535 (ffff).

Parameter (Motor Commands)	Range of Data Values	Units	Command		Command Data	Response Data	Exception Responses
Motor ID Identifies the motor in the drive's motor parameter table currently being used. The word is separated into various groups of bit fields to specify the encoder resolution, motor, type, and table ID.	[w1] - Number 0..65535 (0000..ffff) BITS USAGE	-	Read	090	-	[w1]	
The setting 0 (0000) indicates that no motor has been selected, and the setting 65535 (ffff) indicates motor parameters were set Individually and not read from the drive's motor parameter table.	15..12 Table ID 11..8 Encoder Resolution 7 Type (0 = synch., 1 = induct.) 6..0 Motor Number		Write	091	[w1]	-	03, 06
Encoder Lines The number of lines on the motor encoder.	[w1] - Lines 100..15000 (ε: 1)	lines/rev	Read	092	-	[w1]	
	(0064..3a98)	(× 0001)	Write	093	[w1]	-	01, 02, 03, 06
Maximum Motor Speed The minimum speed of the motor which causes an Overspeed fault.	[d1] - Speed 0..32767.99998 (ε: 1.53e-5)	RPM	Read	094	-	[d1]	
	(00000000..7fffffff)	(× 00010000)	Write	095	[d1]	-	01, 02, 03, 06
Motor Peak Current The peak current which the motor can handle.	[w1] - Current 0..255.992 (ε: 7.8e-3)	Amps	Read	096	-	[w1]	
	(0000..7fff)	(× 0080)	Write	097	[w1]	-	01, 02, 03, 06
Motor Continuous Current The continuous current which the motor can handle.	[w1] - Current 0..255.992 (ε: 7.8e-3)	Amps	Read	098	-	[w1]	
	(0000..7fff)	(× 0080)	Write	099	[w1]	-	01, 02, 03, 06
Torque Constant The sine wave torque constant of the motor.	[w1] - K_t 0.00024..15.9998 (ε: 2.44e-4)	N-m/Amp	Read	09a	-	[w1]	
	(0001..ffff)	(× 1000)	Write	09b	[w1]	-	01, 02, 03, 06
Rotor Inertia J_m	[w1] - J_m 0.0156..1023.98 (ε: 1.56e-2)	kg-cm^2	Read	09c	-	[w1]	
	(0001..ffff)	(× 0040)	Write	09d	[w1]	-	01, 02, 03, 06
Back EMF Constant K_e	[w1] - K_e 0.0039..255.996 (ε: 3.91e-3)	Volts / 1000 RPM	Read	09e	-	[w1]	
	(0001..ffff)	(× 0100)	Write	09f	[w1]	-	01, 02, 03, 06
Winding Resistance The phase to phase resistance of the motor windings at 25° C.	[w1] - Resistance 0.0039..255.996 (ε: 3.91e-3)	Ohms	Read	0a0	-	[w1]	
	(0001..ffff)	(× 0100)	Write	0a1	[w1]	-	01, 02, 03, 06
Winding Inductance The phase to phase inductance of the motor windings.	[w1] - Inductance 0.0039..255.996 (ε: 3.91e-3)	mH	Read	0a2	-	[w1]	
	(0001..ffff)	(× 0100)	Write	0a3	[w1]	-	01, 02, 03, 06
Thermostat Flag Indicates if the motor contains a thermostat.	[b1] - Flag 0 - no thermostat present	-	Read	0a4	-	[b1]	
	1 - thermostat is present		Write	0a5	[b1]	-	01, 02, 03, 06

Motor Commands (Continued)

Parameter (Motor Commands)	Range of Data Values	Units	Command		Command Data	Response Data	Exception Responses
Commutation Type	[b1] - Type 0 - induction motor 1 - 6-step ABS/Index	-	Read	0a6	-	[b1]	
	2 - 8-step ABS/Index 3 - Hall/Index 4 - Hall/Hall		Write	0a7	[b1]	-	01, 02, 03, 06
Current Feedforward	[s1] - Value -127.996..127.996 (ε: 3.91e-3)	degrees/ kRPM	Read	0a8	-	[s1]	
	(8001..7fff)	(× 0100)	Write	0a9	[s1]	-	02, 03, 04, 06
Thermal Time Constant The thermal time constant for protecting the motor.	[w1] - Time 0..65535 (ε: 1)	seconds	Read	0aa	-	[w1]	
	(0000..ffff)	(× 0001)	Write	0ab	[w1]	-	02, 03, 06
Pole Count Number of poles.	[b1] - Poles 0 - 2 Poles 1 - 4 Poles	-	Read	0ac	-	[b1]	
	2 - 6 Poles 3 - 8 Poles		Write	0ad	[b1]	-	01, 02, 03, 06
Hall Offset The offset of the Hall-effect sensor relative to the rotor.	[w1] - Offset 0..359 (ε: 1)	electrical degrees	Read	0ae	-	[w1]	
	(0000..0167)	(× 0001)	Write	0af	[w1]	-	01, 02, 03, 06
Index Offset The offset of the motor encoder relative to the rotor.	[w1] - Offset 0..359 (ε: 1)	electrical degrees	Read	0b0	-	[w1]	
	(0000..0167)	(× 0001)	Write	0b1	[w1]	-	01, 02, 03, 06
Motor Table Information Information about the Motor Table in the drive. The information returned includes the number of synchronous motors and induction motors in the table, and the table ID number.	[b1] - Sync. Records 0..255 (00..ff) [b2] - Induction Records 0..255		Read	0b2	-	[b1] ..[b3]	
	(00..ff) [b3] - Table ID 0..31 (00..1f)		Write	-	-	-	
Motor Table Record Size Information about the Motor Table records in the drive. The information returned includes the synchronous motor and induction motor record sizes.	[w1] - Sync. Record Size 0..65535 (0000..ffff)		Read	0b3	-	[w1] ..[w2]	
	[w2] - Induction Record Size 0..65535 (0000..ffff)		Write	-	-	-	
Motor Table Version Version of the Motor Table in the drive.	[b1] - Major Version 0..255 (00..ff)		Read	0b4	-	[b1]..[b2]	
	[b2] - Minor Revision 0..255 (00..ff)		Write	-	-	-	
Thermal Time Constant Enable Sets or returns the flag which indicates if the Thermal Time Constant is used for protecting the motor.	[b1] - Flag 00 - Disabled 01 - Enabled	-	Read	1A6	-	[b1]	
			Write	1A7	[b1]	-	02, 03, 06

Motor Commands (Continued)

Parameter	Range of Data Values	Units	Command		Command Data	Response Data	Exception Responses
Motor Forward Direction Flag Sets or returns the motor's forward direction when viewed from the shaft end.	[b1] - Flag 00 - clockwise 01 - counterclockwise		Read	1AA	-	[b1]	
			Write	1AB	[b1]	-	01, 02, 03, 06

Digital I/O Commands

Parameter	Range of Data Values	Units	Command		Command Data	Response Data	Exception Responses
Digital Input Configuration Register Determines which flag is (or flags are) controlled by the specified digital input. If no bits are set for an input, it is unassigned. The Preset Select lines can be used together or separately to select the desired preset. Unassigned select lines are set to 0. The select codes are as follows:	[b1] - Input Number 0 - Input1 1 - Input2 2 - Input3 3 - Input4 [w2] - Flag Number Bit 0 - Torque Override (TP: TrqMode) Bit 1 - Integrator Inhibit (TP: IntInh) Bit 2 - Follower Enable (TP: FolEnab)		Read	0c0	[b1]	[w2]	
Preset C B A 0 0 0 0 1 0 0 1 2 0 1 0 3 0 1 1 4 1 0 0 5 1 0 1 6 1 1 0 7 1 1 1	Bit 3 - Forward Enable (TP: FClamp) Bit 4 - Reverse Enable (TP: RClamp) Bit 5 - Analog Override (TP: Overide) Bit 6 - Preset Select Line A (TP: PreSelA) Bit 7 - Preset Select Line B (TP: PreSelB) Bit 8 - Preset Select Line C (TP: PreSelC)		Write	0c1	[b1] [w2]		01, 03, 06
Digital Output Configuration Register Determines which flag is (or flags are) monitored on the specified digital output. I f no bits are set for an input, it is unassigned.	[b1] - Output Number 0 - Output1 1 - Output2 2 - Output3 3 - Output4 [w2] - Flag Number Bit 0 - In-Position (TP: InPos) Bit 1 - Within Position Window (TP: PosWin) Bit 2 - Zero Speed (TP: 0 Speed)		Read	0c2	[b1]	[w2]	
	Bit 3 - Within Speed Window (TP: SpdWin) Bit 4 - Positive ILimit (TP: +ILimit) Bit 5 - Negative ILimit (TP: -ILimit) Bit 6 - At Speed (TP: AtSpeed) Bit 7 - Drive Enabled (TP: DrvEnab) Bit 8 - DC Bus Charged (TP: BusChg) Bit 9 - Disabling Fault		Write	0c3	[b1] [w2]		01, 03, 06
Override Digital Output Overrides the digital output control to allow the user to write the output bits directly.	[b1] - State 0 - Normal	-	Read	0c4	-	[b1]	
	1 - Override		Write	0c5	[b1]	-	01
Digital Output Write Mask Contains the bit pattern to write to the digital outputs when in override control.	[w1] - States Bit 0 - READY Output State Bit 1 - BRAKE Output State Bit 2 - OUTPUT1 Output State	-	Read	0c6	-	[w1]	
	Bit 3 - OUTPUT2 Output State Bit 4 - OUTPUT3 Output State Bit 5 - OUTPUT4 Output State		Write	0c7	[w1]	-	01
BRAKE Active Delay The time delay between disabling the drive, and activating the BRAKE output. Negative values indicate the time that the BRAKE is active before disabling the drive.	[s1] - Delay -32767..32767 (ε: 1)	millisec-onds	Read	0c8	-	[s1]	
	(8001..7fff)	(× 0001)	Write	0c9	[s1]	-	03, 04
BRAKE Inactive Delay The time delay between enabling the drive and deactivating the BRAKE output. Negative values indicate the time that the BRAKE is inactive before enabling the drive.	[s1] - Delay -32767..32767 (ε: 1)	millisec-onds	Read	0ca	-	[s1]	
	(8001..7fff)	(× 0001)	Write	0cb	[s1]	-	03, 04

Analog I/O Commands

Parameter (Analog I/O Commands)	Range of Data Values	Units	Command		Command Data	Response Data	Exception Responses
COMMAND Velocity Offset The offset applied to the COMMAND analog input when being used for velocity command.	[s1] - Offset -10000..10000 (ε: 1)	millivolts	Read	0cc	-	[s1]	
	(d8f0..2710)	(× 0001)	Write	0cd	[s1]	-	03, 04
COMMAND Velocity Scale The scale applied to the COMMAND analog input when being used for velocity command.	[s1] - Scale -32767..32767 (ε: 1)	RPM / Volt	Read	0ce	-	[s1]	
	(8001..7fff)	(× 0001)	Write	0cf	[s1]	-	03, 04
COMMAND Torque Offset The offset applied to the COMMAND analog input when being used for torque command.	[s1] - Offset -10000..10000 (ε: 1)	millivolts	Read	0d0	-	[s1]	
	(d8f0..2710)	(× 0001)	Write	0d1	[s1]	-	03, 04
COMMAND Torque Scale The scale applied to the COMMAND analog input when being used for torque command.	[s1] - Scale -127.996..127.996 (ε: 3.91e-3)	Amps / Volt	Read	0d2	-	[s1]	
	(8001..7fff)	(× 0100)	Write	0d3	[s1]	-	03, 04
Analog Output Configuration Register Determines which signal is monitored on the specified analog output.	[b1] - Output Number 0 - Output1 1 - Output2 [b2] - Signal Number 0 - Current Command (TP: I Cmd) 1 - Current Average Command (TP: Avg I) 2 - Current Positive Peak (TP: +IPeak) 3 - Current Negative Peak (TP: -IPeak) 4 - Positive ILimit (TP: +ILimit) 5 - Negative ILimit (TP: -ILimit) 6 - Motor Velocity (TP: MtrVel) 7 - Velocity Command (TP: VelCmd) 8 - Velocity Error (TP: VelErr) 9 - Motor Position (TP: MtrPos) 10 - Position Command Slewed (TP: PosCmd)		Read	0d4	[b1]	[b2]	
	11 - Position Error (TP: PosErr) 12 - Position Peak Positive Error (TP: +PosEPk) 13 - Position Peak Negative Error (TP: -PosEPk) 20 - Master Position (TP: MstrPos) 21 - Position Loop Output (TP: [not avail]) 22 - Velocity Loop Output (TP: [not avail]) 23 - Filter Output (TP: [not avail]) 24 - Notch Output (TP: [not avail]) 25 - R Phase Current (TP: [not avail]) 26 - T Phase Current (TP: [not avail]) 27 - Torque Current (TP: [not avail]) 28 - Field Current (TP: [not avail]) 29 - Torque Voltage (TP: [not avail]) 30 - Field Voltage (TP: [not avail]) 31 - Scaled A/D Command Value (TP: [not avail]) 32 - Bus Voltage (TP: [not avail])		Write	0d5	[b1] [b2]		01, 03
Analog Offset The offset applied to the specified Analog output.	[b1] - Output Number 0 - Output1 1 - Output2		Read	0d6	[b1]	[s2]	
	[s2] - Offset -32767..32767 (ε: 1) (8001..7fff)	millivolts (× 0001)	Write	0d7	[b1] [s2]	-	01, 03, 04

Analog I/O Commands (Continued)

Parameter (Analog I/O Commands)	Range of Data Values	Units	Command		Command Data	Response Data	Exception Responses
Analog Scale The scale applied to the specified Analog output.	[b1] - Output Number 0 - Output1 1 - Output2	Drive Internal Units	Read	0d8	[b1]	[s2]	
	[s2] - Scale -32767..32767 (ε: 1) (8001..7fff)	(Dependent on selected signal)	Write	0d9	[b1] [s2]	-	01, 03, 04
Override Analog Outputs Overrides the analog output control to write the outputs directly.	[b1] - State 0 - Normal 1 - Override	-	Read	0da	-	[b1]	
			Write	0db	[b1]	-	01
Analog Output Write Value Contains the value to write to the analog outputs when in override cotrol	[b1] - Output Number 0 - Output1 1 - Output2	-	Read	0dc	[b1]	[s2]	
	[s2] - Value -10000..10000 (ε: 1) (d8f00.2710)	millivolts (x 0001)	Write	0dd	[b1] [s2]	-	01

Serial Port Commands

Parameter	Range of Data Values	Units	Command		Command Data	Response Data	Exception Responses
Serial Port Baud Rate The drive's serial port baud rate. If the baud rate is changed, it will not take effect until the drive is reset.	[b1] - Rate 0 - 1200 (TP: 1200) 1 - 2400 (TP: 2400)	-	Read	0de	-	[b1]	
	2 - 4800 (TP: 4800) 3 - 9600 (TP: 9600) 4 - 19200 (TP: 19200)		Write	0df	[b1]	-	01, 03
Serial Port Frame Format The drive's serial port baud rate. If the baud rate is changed, it will not take effect until the drive is reset.	[b1] - Frame 0 - 7 data bits, even parity, 1 stop bit (TP: 7D1SEP) 1 - 7 data bits, odd parity, 1 stop bit (TP: 7D1SOP)	-	Read	0e0	-	[b1]	
	2 - 8 data bits, no parity, 1 stop bit (TP: 8D1SNP) 3 - 8 data bits, even parity, 1 stop bit (TP: 8D1SEP) 4 - 8 data bits, odd parity, 1 stop bit (TP: 8D1SOP)		Write	0e1	[b1]	-	01, 03
Software Drive ID The ID used for drive addressing when the rotary DIP switch is set to position "F".	[b1] - ID 0..255	-	Read	0e2	-	[b1]	
	(00..ff)		Write	0e3	[b1]	-	03

Operating Mode Commands

Parameter (Operating Mode Commands)	Range of Data Values	Units	Command		Command Data	Response Data	Exception Responses
Encoder Output Configuration Register The divisor for the motor encoder quadrature output.	[b1] - Divisor 0 - Divide by 1 (TP: ÷ by 1) 1 - Divide by 2 (TP: ÷ by 2)	-	Read	0f0	-	[b1]	
	2 - Divide by 4 (TP: ÷ by 4) 3 - Divide by 8 (TP: ÷ by 8)		Write	0f1	[b1]	-	01, 03
Command Source The signal used for the drive's command source.	[b1] - Source 0 - Analog COMMAND Input (TP: Analog) 1 - Presets (TP: Presets)		Read	0f2	-	[b1]	
	2 - Master Encoder (TP: AuxEnc) 3 - Step/Direction (TP: StepDir) 4 - Step+/Step- (TP: Step+/-)		Write	0f3	[b1]		01, 03, 06
Drive Mode The flag which determines if the velocity control loop is active.	[b1] - Mode 0 - Velocity (TP: Velocity)		Read	0f4	-	[b1]	
	1 - Torque (TP: Torque)		Write	0f5	[b1]		01, 03, 06
Velocity Preset The command velocity levels used when the drive is configured with Presets as the Command Source, and Velocity as the drive mode.	[b1] - Preset 0..7 (00..07) [L2] - Velocity	- RPM	Read	0f6	[b1]	[L2]	
	-32767.99998..32767.99998 (ε: 1.53e-5) (80000001..7fffffff)	(× 00010000)	Write	0f7	[b1] [L2]	-	01, 03, 04
Torque Preset The command torque levels used when the drive is configured with Presets as the Command Source, and Torque as the drive mode.	[b1] - Preset 0..7 (00..07) [s2] - Torque	- Amps	Read	0f8	[b1]	[s2]	
	-255.992..255.992 (ε: 7.81e-3) (8001..7fff)	(× 0080)	Write	0f9	[b1] [s2]	-	01, 03, 04
Analog Input Acceleration Limit The acceleration value used when the analog command input changes while the drive is in velocity mode.	[d1] - Rate 0..2147483647 (ε: 1)	RPM/sec	Read	0fa	-	[w1]	
	(00000000..7fffffff)	(× 00000001)	Write	0fb	[w1]	-	03, 05
Analog Input Deceleration Limit The deceleration value used when the analog command input changes while the drive is in velocity mode.	[d1] - Rate 0..2147483647 (ε: 1)	RPM/sec	Read	0fc	-	[d1]	
	(00000000..7fffffff)	(× 00000001)	Write	0fd	[d1]	-	03, 05
Preset Input Acceleration Limit The acceleration value used when changing between velocity presets. This limit is only used while the drive is in velocity mode and the Command Source is set to Preset input.	[d1] - Rate 0..2147483647 (ε: 1)	RPM/sec	Read	0fe	-	[d1]	
	(00000000..7fffffff)	(× 00000001)	Write	0ff	[d1]	-	03, 05
Preset Input Deceleration Limit The deceleration value used when changing between velocity presets. This limit is only used while the drive is in velocity mode and the Command Source is set to Preset input.	[d1] - Rate 0..2147483647 (ε: 1)	RPM/sec	Read	100	-	[d1]	
	(00000000..7fffffff)	(× 00000001)	Write	101	[d1]	-	03, 05
Tuning Direction Flag Sets or returns the flag which indicates the direction the motor rotates during tuning.	[b1] - Flag 00 - Bi-directional		Read	1A0	-	[b1]	
	01 - Forward 02 - Reverse		Write	1A1	[b1]	-	01, 03, 06

Operating Mode Commands (Continued)

Parameter (Operating Mode Commands)	Range of Data Values	Units	Command		Command Data	Response Data	Exception Responses
Analog Input Acceleration Limits Enable Sets or returns the flag which indicates that acceleration limits are enabled. This flag is only used while the drive is in velocity mode and the Command Source is set to analog COMMAND input.	[b1] - Flag 00 - Disabled 01 - Enabled		Read	1A2	-	[b1]	
			Write	1A3	[b1]	-	01,03
Preset Acceleration Limits Enable Sets or returns the flag which indicates that acceleration limits are enabled. This flag is only used while the drive is in velocity mode and the Command Source is set to Preset input.	[b1] - Flag 00 - Disabled 01 - Enabled		Read	1A4	-	[b1]	
			Write	1A5	[b1]	-	01, 03
Change Direction Flag Sets or returns the flag which indicates if the normal direction has been changed (reversed).	[b1] - Flag 00 - Normal 01 - Reversed		Read	1AC	-	[b1]	
			Write	1AD	[b1]	-	01, 03, 06

Alternative Operating Mode Commands

Parameter (Operating Mode Commands)	Range of Data Values	Units	Command		Command Data	Response Data	Excep-tion Responses
Operating Mode The operating mode for the drive. Usually, the drive is in Normal mode. The mode can be changed for tuning, encoder alignment, and encoder resolution detection.	[b1] - Mode 0 - Normal (TP: Normal) 1 - AutoTuning (TP: Auto)		Read	102	-	[b1]	
	2 - Manual Tuning (Velocity Step) (TP: Man Vel) 3 - Manual Tuning (Position Step) (TP: Man Pos) 4 - Encoder Alignment (TP: Align) 5 - Encoder Resolution Detection (TP: [not avail])		Write	103	[b1]		01, 06
Operating Mode Status Contains status bits for above alternative operating modes.	[w1] - Status Bit 0 - AutoTuning Complete Bit 1 - Encoder Alignment Complete Bit 2 - Motor Index Detected		Read	104	-	[w1]	
	Bit 3 - Master Index Detected Bit 4 - Motor Encoder Resolution Determined Bit 5 - Master Encoder Resolution Determined Bit 6 - AutoTune Failed		Write	-	-	-	
Autotune Maximum Current The maximum current used in the autotuning algorithm.	[w1] - Current 0.0078..255.992 (ε: 7.81e-3)	Amps	Read	105	-	[w1]	
	(0001..7fff)	(× 0080)	Write	106	[w1]	-	03, 04, 05
Autotune Maximum Distance The maximum distance the motor can travel in the autotuning algorithm.	[d1] - Distance 1..2147483647 (ε: 1)	Counts	Read	107	-	[d1]	
	(00000001..7fffffff)	(× 0001)	Write	108	[d1]	-	03, 04, 05
Manual Tune Position Period The period of the square wave used in the position step manual tuning mode.	[w1] - Period 1..32767 (ε: 1)	millisec-onds	Read	109	-	[w1]	
	(0001..7fff)	(× 0001)	Write	10a	[w1]	-	03, 04, 05
Manual Tune Position Step The amplitude of the square wave used in the position step manual tuning mode.	[w1] - Amplitude 1..32767 (ε: 1)	Counts	Read	10b	-	[w1]	
	(0001..7fff)	(× 0001)	Write	10c	[w1]	-	03, 04, 05
Manual Tune Velocity Period The period of the square wave used in the velocity step manual tuning mode.	[w1] - Period 1..32767 (ε: 1)	millisec-onds	Read	10d	-	[w1]	
	(0001..7fff)	(× 0001)	Write	10e	[w1]	-	03, 04, 05
Manual Tune Velocity Step The amplitude of the square wave used in the velocity step manual tuning mode.	[d1] - Amplitude 0.000015..32767.99998 (ε: 1.53e-5)	RPM	Read	10f	-	[d1]	
	(00000001..7fffffff)	(× 00010000)	Write	110	[d1]	-	03, 04, 05
Encoder Alignment Offset The offset of the motor encoder index pulse relative to the rotor phase location. This value is determined automatically in the Encoder Alignment Operating Mode and continually updates while in that mode. It can also be set when not in the Encoder	[s1] - Offset -180..179 (ε: 1) (ff4c..00b3)	electrical degrees (× 0001)	Read	111	-	[s1]	
Alignment Operating Mode by using the write command. (Note: The value in this parameter does not affect the commutation. It has to be set by using the Remove Alignment Offset command)			Write	112	[s1]	-	01

Alternative Operating Mode Commands (Continued)

Parameter (Alternative Operating Mode Com)	Range of Data Values	Units	Command		Command Data	Response Data	Exception Responses
Save Alignment Offset Corrects the encoder alignment by copying the Encoder Alignment Offset value into an encoder alignment compensation parameter. The alignment compensation parameter value is used in correcting the motor encoder input for commutation.	[No Data] (TP: ↑ to Rmv)		Read	-	-	-	
			Write	113	-	-	03
Motor Encoder Resolution The measured motor encoder counts between index pulses when in the	[w1] - Resolution 0..32767 (ε: 1)	counts	Read	114	-	[w1]	
Encoder Resolution Detection Operating Mode.	(0000..7fff)	(× 0001)	Write	-	-	-	
Master Encoder Resolution The measured master encoder counts between index pulses when in the Encoder Resolution Detection Operating Mode.	[w1] - Resolution 0..32767 (ε: 1)	counts	Read	115	-	[w1]	
	(0000..7fff)	(× 0001)	Write	-	-	-	
Motor Index Position The last recorded position of the motor encoder index.	[w1] - Position 0..65535 (ε: 1)	counts	Read	116	-	[w1]	
	(0000..ffff)	(× 0001)	Write	-	-	-	
Master Index Position The last recorded position of the master encoder index.	[w1] - Position 0..65535 (ε: 1)	master counts	Read	117	-	[w1]	
	(0000..ffff)	(× 0001)	Write	-	-	-	

Runtime Command and Control Commands

Parameter	Range of Data Values	Units	Command		Command Data	Response Data	Exception Responses
Reset Drive Resets the drive hardware and reboots the drive's processors.	[No Data] (TP: ↑to Reset)	-	Read	-	-	-	
			Write	120	-	-	06
Software Drive Enable/Disable If set to Enable Drive and the ENABLE input is active, the drive is enabled. If set to Disable Drive or the ENABLE input is not active, the drive is disabled.	[b1] - State 0 - Disable Drive (TP: Disable)	-	Read	121	-	[b1]	
	1 - Enable Drive (TP: Enable)		Write	122	[b1]	-	01
Torque Setpoint The torque command value used when the Drive Mode is Torque, and the Setpoint Control is Enabled.	[s1] - Torque -255.992..255.992 (ε: 7.81e-3)	Amps	Read	123	-	[s1]	
	(8001..7fff)	(× 0080)	Write	124	[s1]	-	04
Velocity Setpoint The velocity command value used when the Drive Mode is Velocity, and the Setpoint Control is Enabled.	[L1] - Velocity -32767.99998..32767.99998 (ε: 1.53e-5)	RPM	Read	125	-	[L1]	
	(80000001..7fffffff)	(× 00010000)	Write	126	[L1]	-	04
Setpoint Acceleration The acceleration value used when the Velocity Setpoint changes, and the Setpoint Control is Enabled.	[d1] - Rate 0..2147483647 (ε: 1)	RPM/sec	Read	127	-	[d1]	
	(00000000..7fffffff)	(× 00000001)	Write	128	[d1]	-	03, 05
Setpoint Control Enables or disables the setpoint control.	[b1] - State 0 - Disable Setpoint Control (TP: Normal)	-	Read	129	-	[b1]	
	1 - Enable Setpoint Control (TP: CtlPanl)		Write	12a	[b1]	-	01
Reset Faults Resets the fault detection circuitry.	[No Data] (TP: ↑toReset)		Read	-	-	-	
			Write	12b	-	-	

Runtime Status Commands

Parameter (Runtime Status Commands)	Range of Data Values	Units	Command		Command Data	Response Data	Exception Responses
Packed Drive Status The status of various flags in the drive. This status is repeatedly updated.	[d1] - Status Bit 0 - In-Position Bit 1 - Within Position Window Bit 2 - Zero Speed Bit 3 - Within Speed Window Bit 4 - Positive ILimit Bit 5 - Negative ILimit Bit 6 - At Speed Bit 7 - Drive Enabled Bit 8 - DC Bus Charged Bit 9 - Fault Disable Bit 10 - Fault Decel/Disable Bit 11 - Latched Fault Warning Bit 12 - Unlatched Fault Warning	-	Read	134	-	[d1]	
	Bit 14 - Brake Active Bit 15 - Drive Ready Bit 16 - Torque Mode Bit 17- Integrator Inhibit Bit 18 - Follower Enable Bit 19 - Forward Clamp Bit 20 - Reverse Clamp Bit 21 - Analog Override Bit 22 - Preset Select Line A Bit 23 - Preset Select Line B Bit 24 - Preset Select Line C Bit 30 - Reset Faults Bit 31 - Enable Active		Write	-	-	-	
Fault Status Identifies the present state of the possible fault conditions. If a specific Fault Group Mask Is set to unlatched warning, the appropriate bit is not latched in this register and may clear when the condition is removed. If the specific Fault Group Mask is *not* set to unlatched warning, the appropriate bit is latched in this register and will remain set until the drive is reset.	[d1] - Status Bit 0 - +24VDC Fuse Blown Bit 1 - +5VDC Fuse Blown Bit 2 - Encoder Power Fuse Blown Bit 3 - Motor Overtemperature Bit 4 - IPM Fault (Overtemperature/Overcurrent/Short Circuit) Bit 5 - Channel IM Line Break Bit 6 - Channel BM Line Break Bit 7 - Channel AM Line Break Bit 8 - Bus Undervoltage Bit 9 - Bus Overvoltage Bit 10 - Illegal Hall State Bit 11 - Sub processor Unused Interrupt	-	Read	135	-	[d1]	
	Bit 12 - Main processor Unused Interrupt Bit 16 - Excessive Average Current Bit 17 - Overspeed Bit 18 - Excess Following Error Bit 19 - Motor Encoder State Error Bit 20 - Master Encoder State Error Bit 21 - Motor Thermal Protection Bit 22 - IPM Thermal Protection Bit 27 - Enabled with No Motor Selected Bit 28 - Motor Selection not in Table Bit 29 - Personality Write Error Bit 30 - Service Write Error Bit 31 - CPU Communications Error		Write	-	-	-	

Runtime Status Commands (Continued)

Parameter (Runtime Status Commands)	Range of Data Values	Units	Command		Command Data	Response Data	Exception Responses
Run State Identifies the present state of the drive and possible fault conditions. The reported faults are only ones with Fault Mask values set to Disable Drive or Decel, Then Disable Drive. This command is added to support Touch-Pad background status polling operation. This implies that other products which use the touchpad will need to adhere to the following format. The state values 1..127 are reserved for fault indications. These values will cause the fault to be shown on the touchpad.	[c1] - State -01 - Drive Enabled 00 - Drive Ready 01 - +24VDC Fuse Blown 02 - +5VDC Fuse Blown 03 - Encoder Power Fuse Blown 04 - Motor Overtemperature 05 - IPM Fault (Overtemperature/Overcurrent/ Short Circuit) 06 - Channel IM Line Break 07 - Channel BM Line Break 08 - Channel AM Line Break 09 - Bus Undervoltage 10 - Bus Overvoltage 11 - Illegal Hall State	-	Read	136	-	[c1]	
The values 0..-128 are reserved for non-fault state information which is to be indicated, but not shown as a fault by the touchpad.	12 - Sub processor Unused Interrupt 13 - Main processor Unused Interrupt 17 - Excessive Average Current 18 - Overspeed 19 - Excess Following Error 20 - Motor Encoder State Error 21 - Master Encoder State Error 22 - Motor Thermal Protection 23 - IPM Thermal Protection 28 - Enabled with No Motor Selected 29 - Motor Selection not in Table 30 - Personality Write Error 31 - Service Write Error 32 - CPU Communications Error		Write	-	-	-	
Digital Input States Identifies the present state of the digital inputs.	[w1] - States Bit 0 - RESET FAULTS Input State Bit 1 - ENABLE Input State Bit 2 - INPUT1 Input State	-	Read	137	-	[w1]	
	Bit 3 - INPUT2 Input State Bit 4 - INPUT3 Input State Bit 5 - INPUT4 Input State		Write	-	-	-	
Digital Output States Identifies the present state of the digital outputs.	[w1] - States Bit 0 - READY Output State Bit 1 - BRAKE Output State Bit 2 - OUTPUT1 Output State	-	Read	138	-	[w1]	
	Bit 3 - OUTPUT2 Output State Bit 4 - OUTPUT3 Output State Bit 5 - OUTPUT4 Output State		Write	-	-	-	-

Runtime Data Commands

Parameter (Runtime Data Commands)	Range of Data Values	Units	Command		Command Data	Response Data	Exception Responses
Reset Peaks Resets the peak detection firmware for positive position error peak, negative position error peak, positive torque current, and negative current.	[No Data] (TP: ↑toReset)	-	Read	-	-	-	
			Write	140	-	-	
COMMAND Input The command input value before scaling and offsetting.	[s1] - Value -10000.. 10000 (ε: 1)	millivolts	Read	141	-	[s1]	
	(d8f0..2710)	(× 0001)	Write	-	-	-	
Positive ILimit Input The +ILimit input value.	[s1] - Value -255.992..255.992 (ε: 7.81e-3)	Amps	Read	142	-	[s1]	
	(8001..7fff)	(× 0080)	Write	-	-	-	
Negative ILimit Input The -ILimit input value.	[s1] - Value -255.992..255.992 (ε: 7.81e-3)	Amps	Read	143	-	[s1]	
	(8001..7fff)	(× 0080)	Write	-	-	-	
Analog Output The analog output values.	[b1] - Number 0 - Output 1 1 - Output 2		Read	144	[b1]	[s2]	
	[s2] - Value -10000..10000 (ε: 1) (d8f0..2710)	millivolts (× 0001)	Write	-	-	-	
Motor Position The value of the motor encoder register.	[L1] - Value -2147483647..2147483647 (ε: 1)	Counts	Read	145	-	[L1]	
	(80000001..7fffffff)	(× 00000001)	Write	-	-	-	
Master Position The value of the master input register.	[L1] - Value -2147483647..2147483647 (ε: 1)	Master Counts	Read	146	-	[L1]	
	(80000001..7fffffff)	(× 00000001)	Write	-	-	-	
Position Command The position command input to the position loop, which is the master position, after gearing and slew rate limiting.	[L1] - Value -2147483647..2147483647 (ε: 1)	Counts	Read	147	-	[L1]	
	(80000001..7fffffff)	(× 00000001)	Write	-	-	-	
Position Error The difference between the Position Command and the Motor Position.	[L1] - Value -2147483647..2147483647 (ε: 1)	Counts	Read	148	-	[L1]	
	(80000001..7fffffff)	(× 00000001)	Write	-	-	-	
Position Positive Peak Error The maximum amount the Position Command lead the Motor Position.	[L1] - Value 0..2147483647 (ε: 1)	Counts	Read	149	-	[L1]	
	(00000000..7fffffff)	(× 00000001)	Write	-	-	-	
Position Negative Peak Error The maximum amount the Position Command lagged the Motor Position.	[L1] - Value -2147483647..0 (ε: 1)	Counts	Read	14a	-	[L1]	
	(80000001..00000000)	(× 00000001)	Write	-	-	-	
Velocity Command The command value to the velocity loop.	[L1] - Value -32767.99998..32767.99998 (ε: 1.53e-5)	RPM	Read	14b	-	[L1]	
	(80000001..7fffffff)	(× 00010000)	Write	-	-	-	
Motor Velocity The feedback value to the velocity loop.	[L1] - Value -32767.99998..32767.99998 (ε: 1.53e-5)	RPM	Read	14c	-	[L1]	
	(80000001..7fffffff)	(× 00010000)	Write	-	-	-	

Runtime Data Commands (Continued)

Parameter (Runtime Data Commands)	Range of Data Values	Units	Command		Command Data	Response Data	Exception Responses
Velocity Error The difference between Velocity Command and Motor Velocity.	[L1] - Value -32767.99998..32767.99998 (ε: 1.53e-5)	RPM	Read	14d	-	[L1]	
	(80000001..7fffffff)	(× 00010000)	Write		-	-	
Current Command The command value to the current loop.	[s1] - Value -255.992..255.992 (ε: 7.81e-3)	Amps	Read	14e	-	[s1]	
	(8001..7fff)	(× 0080)	Write	-	-	-	
Average Current The average value of the Current Command(?).	[s1] - Value -255.992..255.992 (ε: 7.81e-3)	Amps	Read	14f	-	[s1]	
	(8001..7fff)	(× 0080)	Write	-	-	-	
Current Positive Peak The largest positive value of the Current Command(?).	[s1] - Value 0.0..255.992 (ε: 7.81e-3)	Amps	Read	150	-	[s1]	
	(0000..7fff)	(× 0080)	Write	-	-	-	
Current Negative Peak The largest negative value of the Current Command(?).	[s1] - Value -255.992..0.0 (ε: 7.81e-3)	Amps	Read	151	-	[s1]	
	(8001..0000)	(× 0080)	Write	-	-	-	
Bus Voltage The measured voltage of the DC bus.	[w1] - Value 0..32767 (ε: 1)	Volts	Read	152	-	[w1]	
	(0000..7fff)	(× 0001)	Write	-	-	-	
Field Current The calculated field current for induction motors.	[s1] - Value -255.992..255.992 (ε: 7.81e-3)	Amps	Read	153	-	[s1]	
	(8001..7fff)	(× 0080)	Write	-	-	-	
Torque Current The calculated torque current.	[s1] - Value -255.992..255.992 (ε: 7.81e-3)	Amps	Read	154	-	[s1]	
	(8001..7fff)	(× 0080)	Write	-	-	-	
R-Phase Current The calculated R-Phase current.	[s1] - Value -255.992..255.992 (ε: 7.81e-3)	Amps	Read	155	-	[s1]	
	(8001..7fff)	(× 0080)	Write	-	-	-	
T-Phase Current The calculated T-Phase current.	[s1] - Value -255.992..255.992 (ε: 7.81e-3)	Amps	Read	156	-	[s1]	
	(8001..7fff)	(× 0080)	Write	-	-	-	
Field Voltage Command The field voltage command for induction motors.	[s1] - Value -255.992..255.992 (ε: 7.81e-3)	Volts	Read	157	-	[s1]	
	(8001..7fff)	(× 0080)	Write	-	-	-	
Torque Voltage Command The torque voltage command.	[s1] - Value -255.992..255.992 (ε: 7.81e-3)	Volts	Read	158	-	[s1]	
	(8001..7fff)	(× 0080)	Write	-	-	-	
Average Motor Current The average current seen by the motor.	[s1] - Value 0.0..255.992 (ε: 7.81e-3)	Amps	Read	159	-	[s1]	
	(0000..7fff)	(× 0080)	Write	-	-	-	

Runtime Data Collection Commands

Parameter (Runtime Data Collection Commands)	Range of Data Values	Units	Command		Command Data	Response Data	Exception Responses
Channel 1 Source The signal values returned in channel 1 of Collected Data.	[b1] - Signal Number 0 - Current Command 1 - Current Average Command 2 - Current Positive Peak 3 - Current Negative Peak 4 - Positive ILimit 5 - Negative ILimit 6 - Motor Velocity 7 - Velocity Command 8 - Velocity Error 9 - Motor Position 10 - Position Command Slewed 11 - Position Error 12 - Position Peak Positive Error 13 - Position Peak Negative Error		Read	160	-	[b1]	
	20 - Master Position 21 - Position Loop Output 22 - Velocity Loop Output 23 - Filter Output 24 - Notch Output 25 - R Phase Current 26 - T Phase Current 27 - Torque Current 28 - Field Current 29 - Torque Voltage 30 - Field Voltage 31 - Scaled A/D Command Value 32 - Bus Voltage		Write	161	[b1]		01
Channel 2 Source The signal values returned in channel 2 of Collected Data.	[b1] - Signal Number See *Channel 1 Source* for selections		Read	162	-	[b1]	
			Write	163	[b1]		01
Trigger Source The signal used to trigger the data collection depending on the Trigger Mode.	[b1] - Signal Number See *Channel 1 Source* for selections		Read	164	-	[b1]	
			Write	165	[b1]		01
Timebase The time between samples returned in Collected Data.	[w1] - Value 0.0..13107.0 (ε: 0.2)	millisec-onds	Read	166	-	[w1]	
	(0000..ffff)	(× 0005)	Write	167	[w1]	-	
Trigger Mode Determines how data is collected when Arm Triggering command is sent.	[b1] - Mode 0 - Trigger immediately		Read	168	-	[b1]	
	1 - Trigger on positive transition of trigger source 2 - Trigger on negative transition of trigger source.		Write	169	[b1]	-	01
Trigger Threshold The value which must be crossed on the Trigger Source when the Trigger Mode is set to trigger in a transition.	[s1] - Value	Drive Internal Units	Read	16a	-	[s1]	
	(8001..7fff)	(Dependent on selected trigger source)	Write	16b	[s1]	-	04
Arm Triggering Arms the data collection to begin collecting data when the next trigger event occurs.	[No Data]		Read	-	-	-	
			Write	16c	-	-	
Trigger Status The status of the data collection process.	[b1] - Status 0 - Waiting for Trigger to Occur		Read	16d	-	[b1]	
	1 - Triggered, Collecting Data 2 - Data Collection Complete		Write	-	-	-	

Runtime Data Collection Commands (Continued)

Collected Data The data collected from the last trigger event.	[b1] - Group 0 - Channel 1, Samples 1 through 16 1 - Channel 1, Samples 17 through 32 2 - Channel 1, Samples 33 through 48 3 - Channel 1, Samples 49 through 64 4 - Channel 1, Samples 65 through 80 5 - Channel 1, Samples 81 through 96 6 - Channel 1, Samples 97 through 112 7 - Channel 1, Samples 113 through 128		Read	16e	[b1]	[s2].. [s17]	
	8 - Channel 2, Samples 1 through 16 9 - Channel 2, Samples 17 through 32 a - Channel 2, Samples 33 through 48 b - Channel 2, Samples 49 through 64 c - Channel 2, Samples 65 through 80 d - Channel 2, Samples 81 through 96 e - Channel 2, Samples 97 through 112 f - Channel 2, Samples 113 through 128 [s2]..[s17] Requested data Error Code 4 - Data Accepted After Limiting to Minimum.		Write	-	-	-	04

NOTES

APPENDIX B Press Transfer ASFBs

Introduction

This set of Application Specific Function Blocks (ASFBs) is designed to generate a profile for a slave axis in a press application. As the master axis moves, the slave axis moves in, dwells, and moves out in one master rotation (i.e., 360 degrees). A variation that could be supported as well would be for the slave axis to move in and dwell for each master rotation and that motion is repeated several times before the slave moves out to its initial position.

This profile can have different shapes. It can be triangular (the slave accelerates and decelerates without achieving a constant velocity) or trapezoidal (the slave accelerates to a maximum velocity for a portion of its motion before it decelerates). The acceleration and deceleration can also be configured for an ‘scurve’ where the corners of the motion transitions are smoothed.

To obtain a slave axis profile for two slave moves for one master axis rotation, the M_PRF2MV function block is called from the main application ladder. This function block has a number of inputs to direct the profile generation:

- RR - an array of structures configured in the format required by the RATIO_RL function block. This set of functions blocks is designed for a RATIO_RL application. RATIO_RL usage is detailed in the PiCPro function block reference guide. This structure has the following format:
- MAST_DIS - the distance of the master motion in this segment (in FU).
- SLAV_DIS - the distance of the slave motion in this segment (in FU).
- K1 - the K1 coefficient in the polynomial equation for RATIO_RL.
- K2 - the K2 coefficient in the polynomial equation for RATIO_RL.
- K3 - the K3 coefficient in the polynomial equation for RATIO_RL.
- SPARE - reserved for future use.
- FLAGS - indicate the execution of the polynomial function of RATIO_RL.

  This RR input to the function block must be defined as an array of structures in the calling function (which is usually the main application ladder). The actual size required for this array will depend upon the type of profile required; an scurve profile will have more segments than a simpler constant acceleration profile. The number of segments within the profile will be as follows for each move: acceleration portion (1 for no scurve, 3 with scurve), constant velocity portion (0 or triangular, 1 for trapezoidal), deceleration portion (1 for no scurve, 3 with scurve), and dwell portion (1 if a dwell is required). For example, for the application of M_PRF2MV, the size of the RR array **must be at least 17** to encompass the various combinations because 16 segments will be required.

The sizing of this array is very important. If the array is sized too small, run-time errors within the application are likely to occur (because other variables in PiC memory will be written during the calculations since the internal function blocks will assume enough memory has been allocated by the main application ladder).

- MOV1 and MOV2 - an input structure describes each of the two slave moves required. This structure provides the following information for each move:
- STRT_ANG - the angle of the master axis at the start of the slave's move (in degrees).
- STOP_ANG - the angle of the master axis at the end of the slave's move (in degrees).
- SLV_MOVE - the distance of the slave move (specified in input units that can be scaled).
- MAX_V - the maximum velocity that can be allowed for this slave move (specified as a ratio of slave FU to master FU).
- PCT_J - the percent of maximum possible jerk to be used for this slave move (in a range of 0.0 to 100.0). A value of zero means no jerk, and therefore, no scurve.
- PCT_A - the percent of maximum possible acceleration to be used for this slave move (in a range of 0.0 to 100.0).
- TRI_ONLY - a boolean flag to indicate a triangular profile is desired.
- SCURVE - a boolean flag to indicate the smoothed scurve accel/decel is desired.
- MDST - the number of master feedback units in one cycle or rotation.
- MSCL - the number of master feedback units per input unit in the input MOVx structure (i.e., the start and stop angles).
- SSCL - the number of slave feedback units per input unit in the input MOVx structure (i.e., the slave distance moved between two master angles).
- VLIM - the maximum allowable velocity for this master/slave application (specified as a ratio of slave FU to master FU). This limit is one that would reflect the inherent machine limitations. The individual move structures specify the maximum velocity that is desired for that specific move; that velocity for a move cannot exceed this VLIM value. This VLIM value would be one that is entered once for the application; the velocities for the individual moves could be specified via the user interface.

The input move structure can indicate the intent of a triangular slave move (TRI_ONLY). However, if the other parameters result in a trapezoidal profile achieving the required slave motion, this function block will generate the appropriate trapezoidal profile and it will set a boolean output that indicates this change in behavior. If the main ladder must get a triangular profile then it can take the appropriate actions, such as providing the user interface with a signal that the move parameters must be specified again. If the main ladder will tolerate either triangular or trapezoidal profile but it prefers the triangular profile then this is supported.

If the combination of parameters prevents the generation of a profile then the function block returns an appropriate error indicator. The main ladder must make sure that no errors were detected before trying to apply the generated profile.

The input move structure can direct the profile shape by specifying the percent of maximum acceleration (PCT_A). 100% of maximum acceleration would approximate a step function - immediately get to the maximum velocity for the slave's move (in most cases an unacceptable response for the slave). 0% of maximum acceleration would obtain the minimum slope for the slave's acceleration and still achieve the required slave motion. Values within this range obtain an intermediate behavior.

There is no separate deceleration rate provided as an input, so the deceleration portion of the profile will use the same parameters as the acceleration portion. However, there is an internal function block to generate the deceleration portion of the profile so it could be possible (but not supported at this time) for the generated profile to contain different acceleration and deceleration configurations.

The input move structure can indicate the intent of smoothed scurve acceleration and deceleration portions of the slave profile (SCURVE). This also requires a percentage of jerk to be specified (PCT_J). Maximum jerk (100%) would obtain no scurve behavior because there would be only constant acceleration. Minimum jerk (0.1%) would obtain the smoothest acceleration portion of the profile with no constant acceleration but with the highest peak acceleration rate.

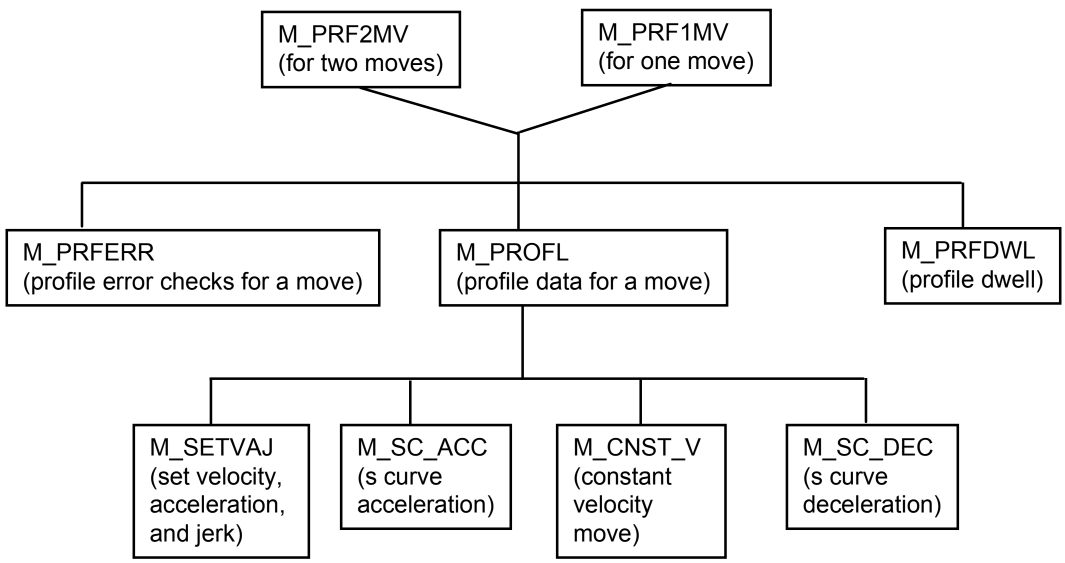
This is the set of function blocks whose purpose is to generate a slave profile for a press application.

- M_PRF2MV - this function block generates a slave axis profile for two slave moves for one master axis rotation. It will in turn call the M_PRFERR, M_PROFL and M_PRFDWL function blocks for each of the two slave axis moves in the profile.
- M_PRF1MV - this function block generates a slave axis profile for one slave move for one master axis rotation. It will in turn call the M_PRFERR, M_PROFL and M_PRFDWL function blocks to generate the profile. The M_PRF1MV function block has the same inputs as M_PRF2MV except that only one move is handled in the profile rather than two.
- M_PRFERR - this function block checks the validity of the input move's parameters.
- M_PROFL - this function block generates the portion of the profile when the slave is moving. It will in turn call the M_SETVAJ, M_SC_ACC, M_CNST_V and M_SC_DEC function blocks.
- M_PRFDWL - this function block generates the portion of the profile when the master axis is moving but the slave is not. This block is required because the RATIO_RL profile must account for all the master counts so that the profile can be repeated (i.e., for each master rotation, the slave performs the same profile). Therefore if the slave is moving only part of the time (which will occur in many press applications), then a portion of the profile contains the master's motion that has no corresponding slave motion. Also, because the real to integer calculations being performed during the generation of the profile might result in rounding, there could be a few counts of master or slave axis motion that could not be incorporated into the main part of the profile. Those remaining counts, if any, can be accounted for during this portion of the profile.
- M_SETVAJ - this function block calculates the acceleration, velocity and jerk to be used for this move. This function block also determines whether the move's parameters can support a triangular profile or whether it must be a trapezoidal profile.
- M_SC_ACC - this function block adds the acceleration portion of the profile into the main structure (for a RATIO_RL). Depending on the move's parameters, the acceleration will be constant acceleration or it will be an scurve (i.e., smoothed acceleration).
- M_CNST_V - this function block adds the constant velocity portion of the trapezoidal profile into the main structure (for a RATIO_RL). The constant velocity portion is the 'flat top' of the profile. A triangular profile does not have a constant velocity portion.

- M_SC_DEC - this function block adds the deceleration portion of the profile into the main structure (for a RATIO_RL). Depending on the move's parameters, the deceleration will be constant deceleration or it will be an scurve (i.e., smoothed deceleration).

If a specific application requires a different combination of slave moves for one (or more) master moves, these function blocks are the 'building blocks' for that application. The M_PRF1MV function block illustrates how to convert a defined move of a slave axis (i.e., the move structure) into a profile for RATIO_RL. Its contents can be merged into your application and then modified to concatenate other slave moves, each with its own definition specified in a move structure, into the single profile. Note that if a longer profile is to be generated you must make sure that the array of structures for the profile in the application is adequately sized.

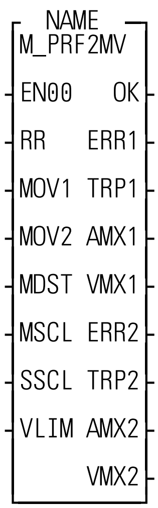
The following flow chart shows the relationship of the function blocks to each other.



Note: The M_PRF2MV function block contains two each of M_PRFERR, M_PROFL, and M_PRFDWL. There is one of these function blocks for each of the two moves.

M_PRF2MV

2 slave moves for master **USER/M_PROFL**



Inputs: EN00 (BOOL) - enables execution (Typically one-shot)

RR (STRUCTURE) - array of structures to be used for RATIO_RL profile.

MOV1 (STRUCTURE) - structure containing input data for first move

MOV2 (STRUCTURE) - structure containing input data for second move

MDST (DINT) - master feedback units/cycle where cycle is divided into 360 degrees

MSCL (REAL) - master feedback units/input unit, typically set to 1.0

SSCL (REAL) - Slave feedback units/input unit

VLIM (REAL) - Maximum allowable velocity

Outputs: OK (BOOL) - execution completed without error

ERR1 (BYTE) - error number for first move

TRP1 (BOOL) - move 1 changed to a trapezoid to achieve move

AMX1 (REAL) - maximum acceleration rate calculated for move 1

VMX1 (REAL) - maximum velocity rate calculated for move 1

ERR2 (BYTE) - error number for second move.

TRP2 (BOOL) - move 2 changed to a trapezoid to achieve move

AMX2 (REAL) - maximum acceleration rate calculated for move 2

VMX2 (REAL) - maximum velocity rate calculated for move 2

<<INSTANCE NAME>>:M_PRF2MV(EN00 := <<BOOL>>, RR := <<MEMORY AREA>> MOV1 := <<MEMORY AREA>>, MOV2 := <<MEMORY AREA>>, MDST := <<DINT>>, MSCL := <<REAL>>, SSCL := <<REAL>>, VLIM := <<REAL>>, OK => <<BOOL>>, ERR1 => <<BYTE>>, TRP1 => <<BOOL>>, AMX1 => <<REAL>>, VMX1 => <<REAL>>, ERR2 =>

<<BYTE>>, TRP2 => <<BOOL>>, AMX2 => <<REAL>>, VMX2 => <<REAL>>);

The M_PRF2MV and M_PRF1MV application specific function blocks are used for applications where there is a cyclic master moving in one direction and the slave is performing a distance move (or two distance moves) and an optional dwell during the master cycle. The function blocks take as inputs the distance the slave should move and over what portion of the master cycle it should move and converts that information into a RATIO_RL profile structure. M_PRF1MV is used if there is one move (with or without a dwell) during a master cycle. M_PRF2MV is used if there are two moves (with or without dwell) during a master cycle. The slave moves can be different directions or the same. The slave moves can be trapezoidal or triangular, and the functions allow a move to be smoothed with an "S-Curve" -like acceleration by input selection.

The information given to the function about the move such as slave distance and portion of the master cycle to move is converted to an array of structures that is used as an input to the RATIO_RL function. The structure must be formatted as shown in Table 2-1 and must have an array size of at least 17 (0..16) because the function clears out the structure data before it calculates the profile. This structure is filled in by the function block so it is not necessary for you to enter any data into it.

Table 2-1. Ratio Real Structure

Name	Data Type	Definition
RR	STRUCTURE	Array of Structures to be used for profile
.MAST_DIS	DINT	Master move distance in feedback units
.SLAV_DIS	DINT	Slave move distance relative to Master in feedback units
.K1	LREAL	VELOCITY co-efficient for polynomial
.K2	LREAL	ACCELERATION co-efficient for polynomial (A/2)
.K3	LREAL	JERK co-efficient for polynomial (J/6)
.SPARE	LREAL	Spare
.FLAGS	DWORD	Move type flags. Bits 2 & 3 = 0 for polynomial

The function assumes a rotary cyclic master moving in a single direction. The value at the MDST input indicates the number of master feedback units for the cycle. This is divided into 360 degrees for a cycle. The master information for the profile is entered in degrees. The MSCL input is typically set to 1.

The slave information for the profile is entered in slave input units. The SSCL input defines the number of slave feedback units per slave input unit. For example, if you want to enter slave data in revolutions and there are 8000 feedback units per revolution, you would enter a value of 8000.0 at the SSCL input and a value of 0.25 for slave distance in the move structure to move one quarter of a rev.

The shape of the slave profile is determined by the input parameters at MOV1 and MOV2. Each slave move can have a zero-speed dwell after it. The format for the MOV1 and MOV2 structures is shown in Table 2-2. MOV2 has the same structure format as MOV1.

The VLIM input to the function block is used as a limit on the master/slave ratio that is allowed. This value is compared with the individual MAX_V inputs in the move structures and an error is set if the individual MAX_V inputs exceed VLIM. The maximum ratio can be multiplied by the speed of the master to find the maximum velocity of the slave axis. Using master/slave ratio as opposed to velocity is a better method for applications where the master can run at variable speeds - in this case if the master is moving slower the slave will do so as well.

Table 2-2. Move Structure

Name	Data Type	Definition
MOV1	STRUCTURE	Structure containing move's input data
.STRT_ANG	REAL	Angle of master axis at start of slave move
.STOP_ANG	REAL	Angle of master axis at end of slave move
.SLV_MOVE	REAL	Distance of slave move
.MAX_V	REAL	Maximum desired velocity of the slave axis
.PCT_J	REAL	Percent of maximum possible jerk to be used
.PCT_A	REAL	Percent of maximum possible accel to be used
.TRI_ONLY	BOOL	Triangular profile desired
.SCURVE	BOOL	Smoothed scurve acc/dec desired

The profile will be calculated as a triangle if possible. If it is not possible to achieve a triangular profile because of the limit on ratio (MAX_V) then the profile will become trapezoidal in shape with MAX_V as the limit for master/slave ratio.

Regardless of the value of STRT_ANG, the slave move will be put in the first segment of the RATIO_RL profile. Therefore, if you want the slave to start its move at a specific master position, you should use the MSTR input of the RATIO_RL profile.

The acceleration parameter for the move is set to a value between 0 and 100%. 0% acceleration is the slowest amount and the move will be as close to triangular as possible. 100% acceleration is the fastest amount and the move will be almost rectangular in shape.

Any profile can be "smoothed" by adjusting the percent jerk. This will not change the basic shape of the profile, but will change the acceleration and deceleration portions of the move to resemble an “S-Curve”. The percent jerk parameter determines how much the acceleration (or deceleration) segment of the move will be smoothed. A 0% value is the most smoothing, and a 100% value will yield no smoothing. The maximum velocity of the move is not affected by “smoothing” nor is the average acceleration. It only affects how the acceleration (and deceleration) will be applied to obtain this velocity.

The ASFB checks the input data for errors. If an error is detected the profile will not be generated and the OK output will not be set. ERR1 and ERR2 are byte values. Each error that can occur is assigned a bit number in the byte. It is possible for more than one error to be set at a time. The bit assignments for ERR1 and ERR2 are shown in Table 2-3. The function BYT2BOOL is helpful in checking for specific errors.

Maximum velocity can be limited to allow an automatically adjusting profile which will be triangular until the maximum velocity is reached. It will then spread into a trapezoid using the minimum acceleration to achieve the move.

It is also possible to set up each move as a constant accel/decel triangular move or a trapezoidal move using operator inputs for the desired shape. Acceleration can be adjusted to change the shape from triangular to nearly rectangular by increasing the acceleration percent.

For any such profile, the acceleration can be "smoothed" by adjusting the jerk percent. This will not change the basic shape of the profile, but will change the acceleration and deceleration portions of the move to resemble an "S-Curve".

The percentage of jerk corresponds inversely to the portion of the Acceleration (or Deceleration) segment of the move which will be smoothed, until 100% equals no S-Curve. At minimum jerk, there is no constant accel portion. This will correspond to the highest acceleration rate. Maximum velocity during a move is not affected by “smoothing” or jerk, nor is the average acceleration. It only affects how the acceleration (and deceleration) will be applied to obtain this velocity.

The first portion of the ASFB checks the input data for any detectable errors. The bit assignments for ERR1 and ERR2 are shown in Table 2-3. The function BYT2BOOL is helpful in checking for specific errors.

Table 2-3. Error Definitions

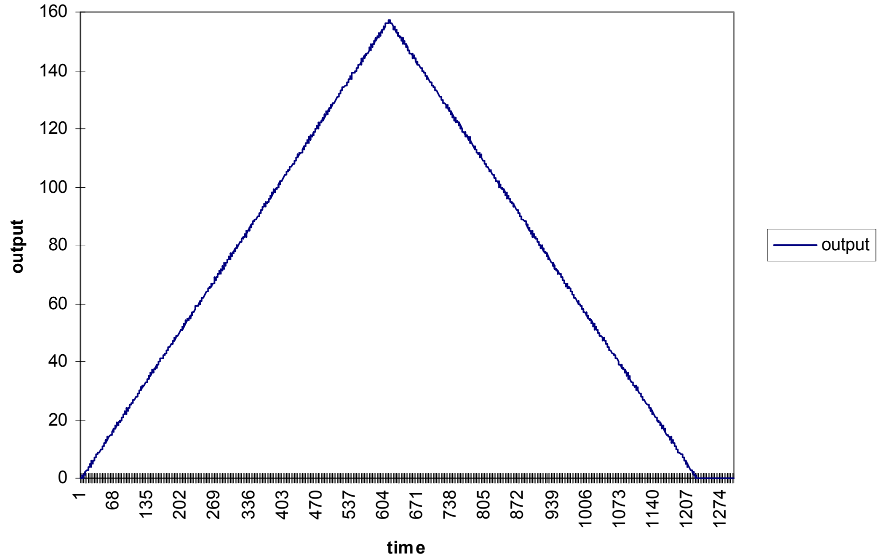
Fault Bit Number	Description
0	Starting angle is not within -180 to 360 degrees
1	Ending angle is not within -180 to 360 degrees
2	Acceleration percent value not within 0.0 to 100.0
3	Jerk percentage value not within 0.0 to 100.0
4	Desired velocity limit higher than allowed
5	Desired velocity limit is zero
6	Master Move 1 overlaps Master Move 2
7	Cannot set-up move with input parameters given

Example Profiles

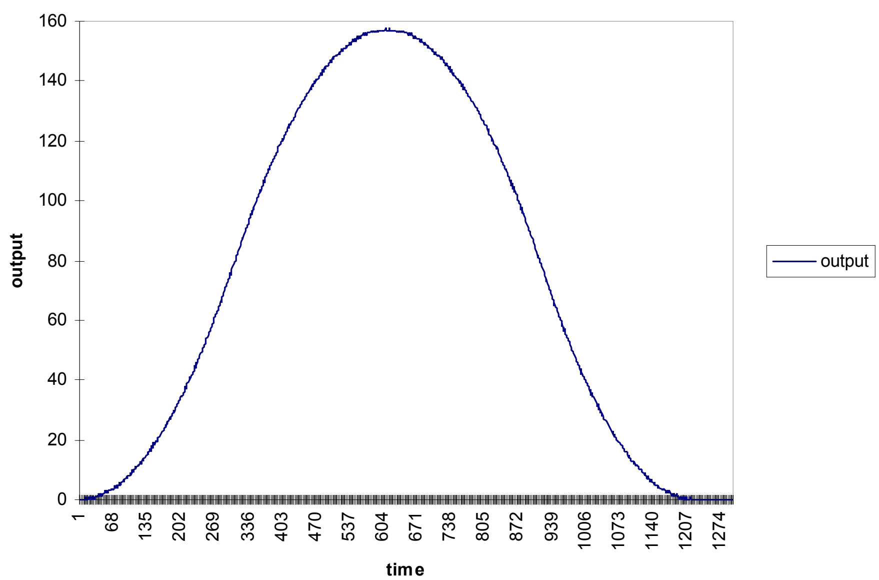
The following four examples illustrate the effects of jerk on a triangular profile. As can be seen, the lower the jerk percentage the smoother the profile.

1. Triangular profile with no scurve (and 50% acceleration).
2. Triangular profile with scurve and 5% jerk.
3. Triangular profile with scurve and 50% jerk.
4. Triangular profile with scurve and 95% jerk.

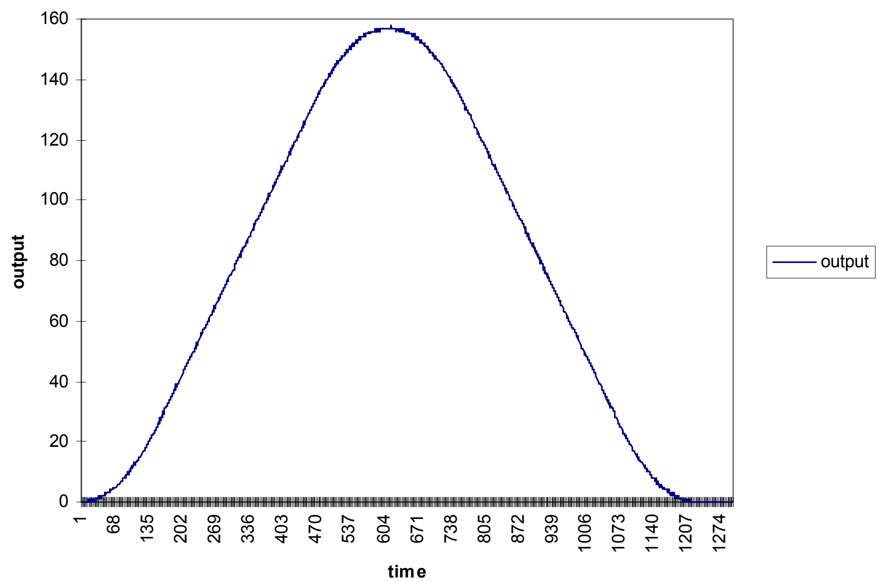
Figure 2-1. Triangular no scurve, 50% accel



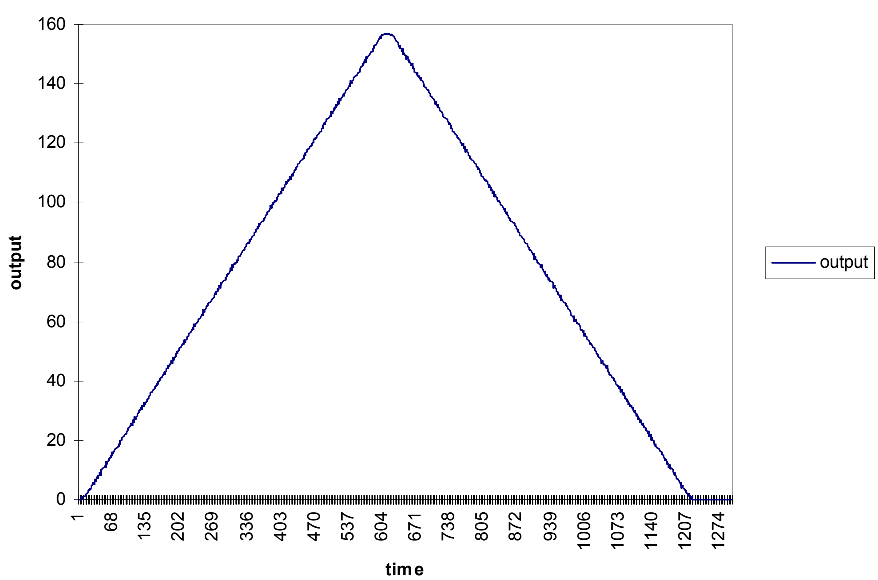
Triangular scurve, 50% accel, 5% jerk



Triangular scurve, 50% accel, 50% jerk

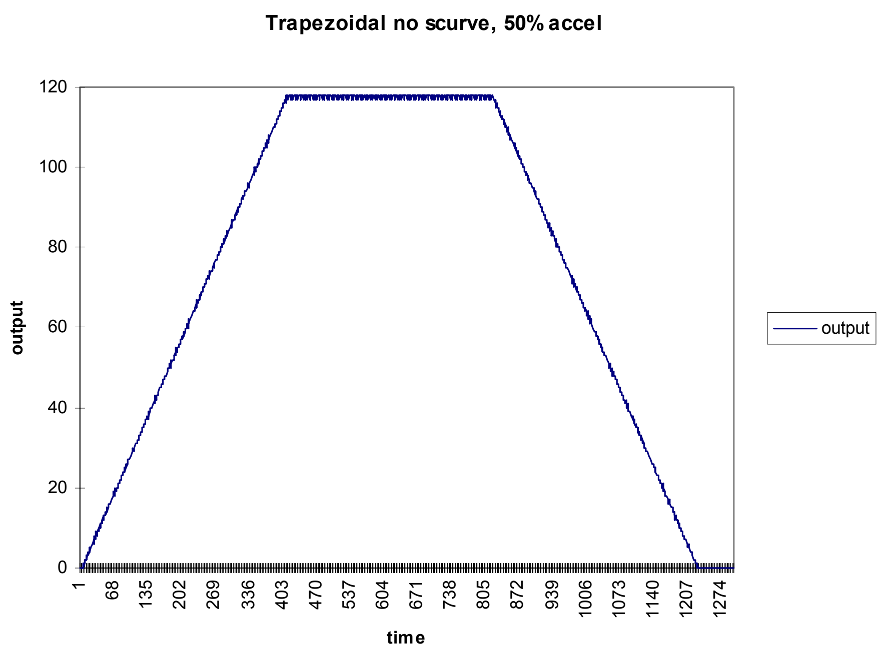


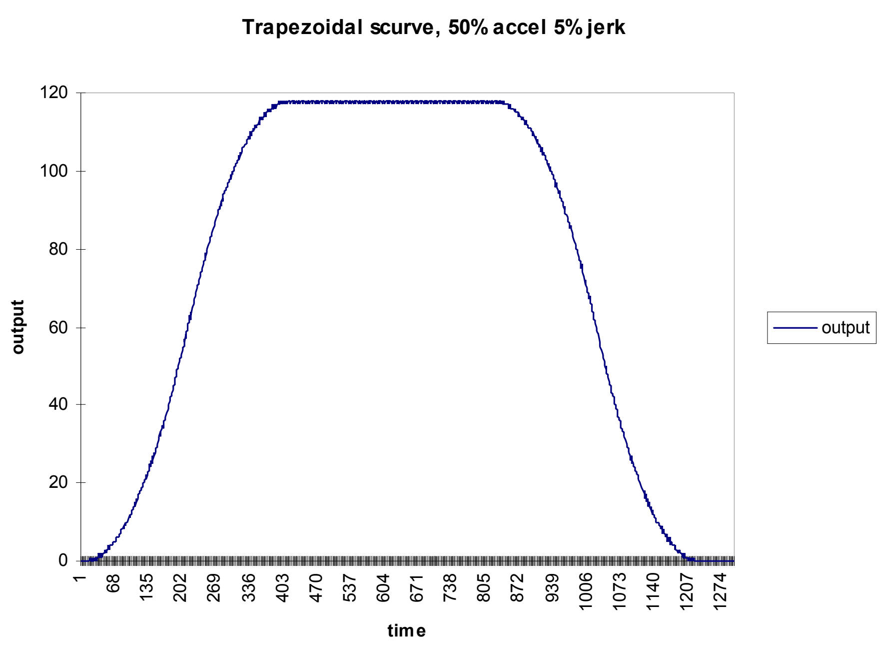
Triangular scurve, 50% accel, 95% jerk



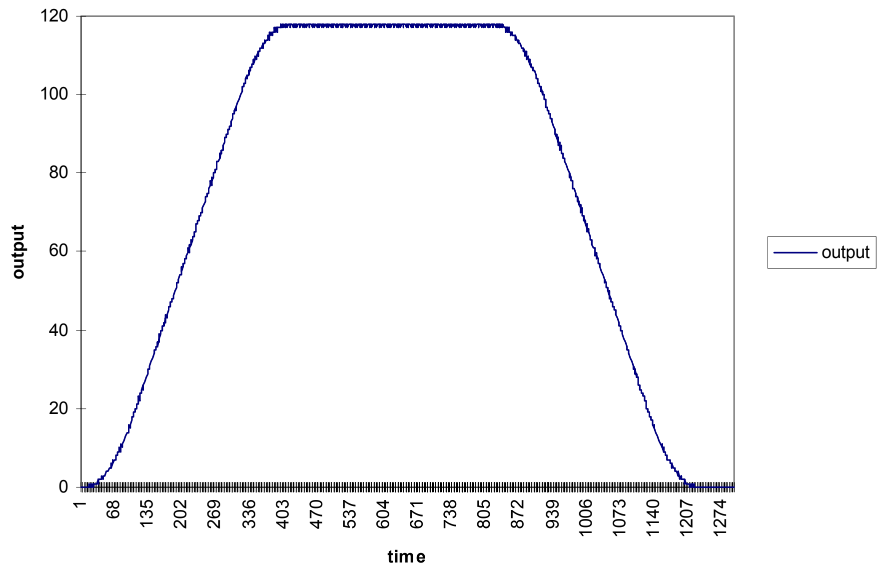
The following six examples illustrate the effects of acceleration and jerk on a trapezoidal profile. As can be seen, the higher the acceleration percentage the steeper the acceleration curve. Also, just as for the triangular profile, the lower the jerk percentage the smoother the profile.

1. Trapezoidal profile with no scurve and 50% acceleration.
2. Trapezoidal profile with scurve, 50% acceleration and 5% jerk.
3. Trapezoidal profile with scurve, 50% acceleration and 50% jerk.
4. Trapezoidal profile with scurve, 50% acceleration and 95% jerk.
5. Trapezoidal profile with scurve, 10% acceleration and 50% jerk.
6. Trapezoidal profile with scurve, 90% acceleration and 50% jerk.

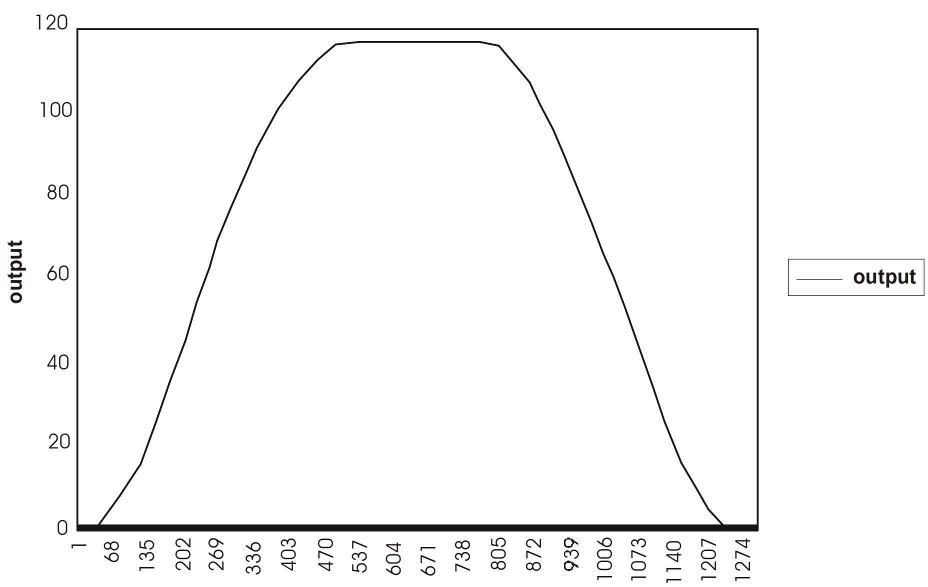

Trapezoidal no scurve, 50% accel
output
120
100
80
60
40
20
0
1
68
135
202
269
336
403
470
537
604
671
738
805
872
939
1006
1073
1140
1207
1274
time
output


Trapezoidal scurve, 50% accel 5% jerk
output
120
100
80
60
40
20
0
1
68
135
202
269
336
403
470
537
604
671
738
805
872
939
1006
1073
1140
1207
1274
time
output

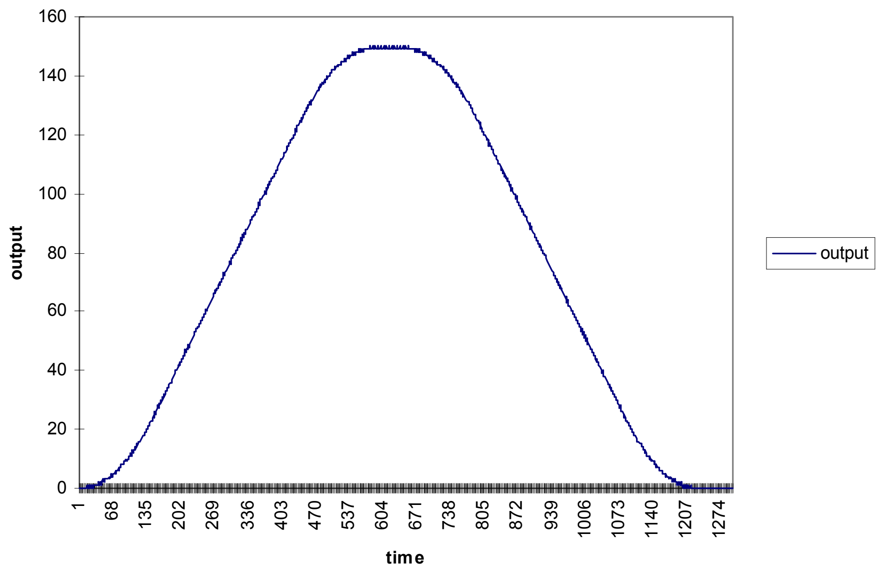
Trapezoidal scuve, 50% accel 50% jerk



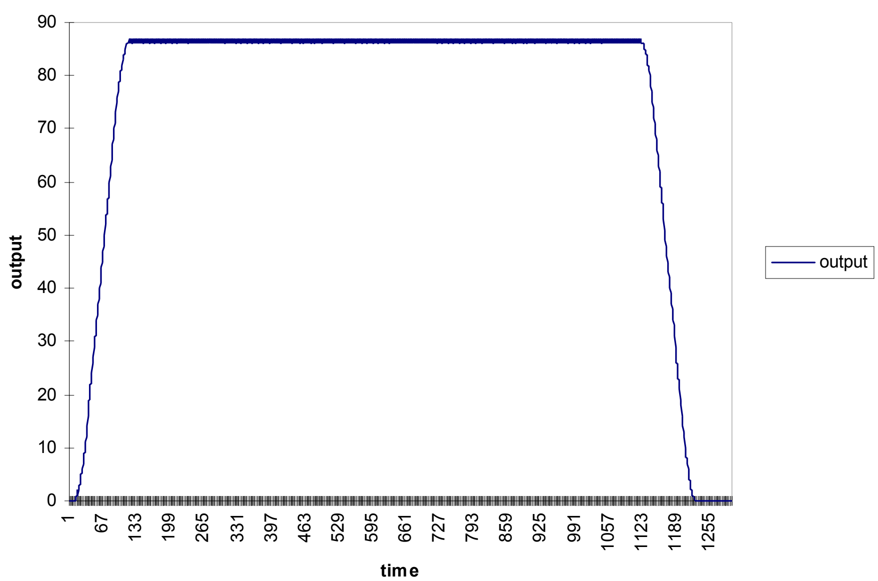
Trapezoidal scurve, 50% accel 95% jerk

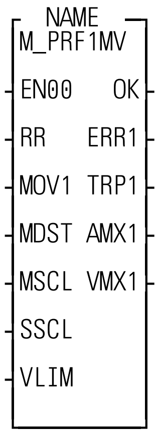


Trapezoidal scurve, 10% accel 50% jerk



Trapezoidal scurve, 90% accel, 50% jerk





Inputs: EN00 (BOOL) - enables execution (Typically one-shot)

RR (STRUCTURE) - Array of Structures to be used for profile.

MOV1 (STRUCTURE) - Structure containing 1st move's input data.

MDST (DINT) - Master feedback units/cycle.

MSCL (REAL) - Master feedback units/input unit.

SSCL (REAL) - Slave feedback units/input unit..

VLIM (REAL) - Maximum allowable velocity.

Outputs: OK (BOOL) - execution completed without error.

ERR1 (BYTE) - Error number for move.

TRP1 (BOOL) - Move changed to a trapezoid to achieve move.

AMX1 (REAL) - Maximum acceleration rate calculated for move.

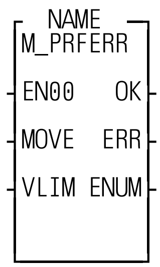
VMX1 (REAL) - Maximum velocity rate calculated for move.

<<INSTANCE NAME>>:M_PRF1MV(EN00 := <<BOOL>>, RR := <<MEMORY AREA>> MOV1 := <<MEMORY AREA>>, MDST := <<DINT>>, MCSL := <<REAL>>, SSCL := <<REAL>>, VLIM := <<REAL>>, OK => <<BOOL>>, ERR1 => <<BYTE>>, TRP1 => <<BOOL>>, AMX1 => <<REAL>>, VMX1 => <<REAL>>);

The M_PRF1MV function block sets up a single slave move in the master cycle.

The M_PRF1MV function block does the same processing as M_PRF2MV except that it processes only one slave move for the master cycle rather than two slave moves. Refer to the M_PRF2MV description for more information.

Note: Fault bit number 6 (Table 2-3: Error Definitions) is not used by M_PRF1MV.



Inputs: EN00 (BOOL) - enables execution (Typically one-shot)

MOVE (STRUCT) - Structure containing 1st move’s input data.

VLIM (REAL) - Maximum allowable velocity.

Outputs: OK (BOOL) - execution completed without error.

ERR (BOOL) - Error flag.

ENUM (BYTE) - Error number.

<<INSTANCE NAME>>:M_PRFERR(EN00 := <<BOOL>>, MOVE := <<MEMORY AREA>> VLIM := <<REAL>>, OK => <<BOOL>>, ERR => <<BOOL>>, ENUM => <<BYTE>>);

The M_PRFERR function block checks the validity of the move’s parameters that have been passed into a function block.

This function block checks the validity of the move’s parameters which have been passed into a function block. It is originally designed specifically for a rotary master and a linear slave axis. The format for the MOVE structure is shown in Table 2-4.

Table 2-4. MOVE Structure

Name	Data Type	Definition
MOVE	STRUCT	Structure of input data for move
.STRT_ANG	REAL	Angle of master axis at start of slave move
.STOP_ANG	REAL	Angle of master axis at end of slave move
.SLV_MOVE	REAL	Distance of slave move
.MAX_V	REAL	Maximum desired velocity of the slave axis
.PCT_J	REAL	Percent of maximum possible jerk to be used
.PCT_A	REAL	Percent of maximum possible accel to be used
.TRI_ONLY	BOOL	Triangular profile desired
.SCURVE	BOOL	Smoothed scurve acc/dec desired

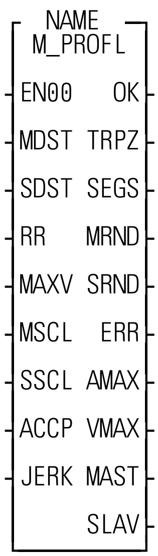
If any of the values are invalid, then a fault is flagged (ERR). All faults found are coded into a byte which is passed to the calling ladder. Use BYT2BOOL to decode faults. Only the first 6 bits are set in this function block. The upper 2 bits are used and set in the calling ladder. The Error Definitions are shown in Table 2-5.

Table 2-5. Error Definitions

Fault Bit Number	Description
0	Starting angle is not within -180 to 360 degrees
1	Ending angle is not within -180 to 360 degrees
2	Acceleration percent value not within 0.0 to 100.0
3	Jerk percentage value not within 0.0 to 100.0
4	Desired velocity limit higher than allowed
5	Desired velocity limit is zero

M_PROFL

Make profile for 1 move **USER/M_PROFL**



Inputs:

EN00 (BOOL) - enables execution (Typically one-shot)

MDST (REAL) - Master distance for this move.

SDST (REAL) - Slave distance for this move.

RR (STRUCT) - Array of structures to be used for profile.

MAXV (REAL) - Maximum desired velocity.

MSCL (REAL) - Master feedback units/input unit.

SSCL (REAL) - Slave feedback units/input unit.

ACCP (REAL) - Percent of maximum possible accel to be used.

JERK (REAL) - Percent of maximum possible jerk to be used

Outputs:

OK (BOOL) - execution completed without error.

TRPZ (BOOL) - Move changed to a trapezoid to achieve move.

SEGS (USINT) - Total number of structures used for this move.

MRND (DINT) - Master move rounding error detected in FUs.

SRND (DINT) - Slave move rounding error detected in FUs.

ERR (BOOL) - Cannot achieve the move with the given inputs.

AMAX (REAL) - Maximum acceleration rate calculated for move.

VMAX (REAL) - Maximum velocity rate calculated for move.

MAST (DINT) - Number of master feedback units used in move.

SLAV (DINT) - Number of Master feedback units used in move.

<<INSTANCE NAME>>:M_PROFL(EN00 := <<BOOL>>, MDST := <<REAL>>, SDST := <<REAL>> RR := <<MEMORY AREA>>, MAXV := <<REAL>>, MSCL := <<REAL>>, SSCL := <<REAL>>, ACCP := <<REAL>>, JERK := <<REAL>>, OK => <<BOOL>>, TRPZ => <<BOOL>>, SEGS => <<USINT>>, MRND => <<DINT>>, SRND => <<SRND>>, ERR => <<BOOL>>, AMAX => <<REAL>>, VMAX => <<REAL>>, MAST => <<DIST>>, SLAV => <<DINT>>);

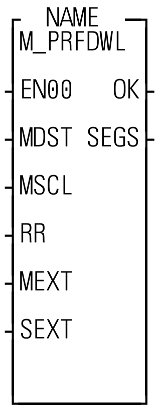
The M_PROFL function block sets up one move

Maximum velocity can be limited to allow an automatically adjusting profile which will be triangular until the maximum velocity is reached. It will then spread into a trapezoid using the minimum acceleration to achieve the move.

It is also possible to set up each move as a constant accel/decel triangular move or a trapezoidal move using operator inputs for the desired shape. Acceleration can be adjusted to change the shape from triangular to nearly rectangular by increasing the acceleration percent.

For any such profile, the acceleration can be “smoothed” by adjusting the jerk percent. This will not change the basic shape of the profile, but will change the acceleration and deceleration portions of the move to resemble an “S-Curve”. The percentage of jerk corresponds inversely to the portion of the Acceleration (or Deceleration) segment of the move which will be smoothed, until 100% equals no S-Curve. At minimum jerk, there is no constant accel portion. This will correspond to the highest acceleration rate. Maximum velocity during a move is not affected by “smoothing” or jerk, nor is the average acceleration. It only affects how the acceleration (and deceleration) will be applied to obtain this velocity.

The format for the RATIO_RL structure is shown in Table 2-1.



Inputs: EN00 (BOOL) - enables execution (Typically one-shot)

MDST (REAL) - Master distance for dwell.

MSCL (REAL) - Master scale factor.

RR (STRUCTURE) - Array of structures used for the profile.

MEXT (DINT) - Extra master feedback units.

SEXT (DINT) - Extra slave feedback units.

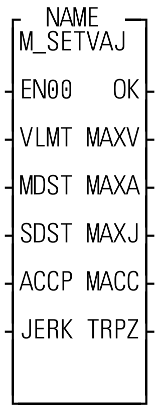
Outputs: OK (BOOL) - execution completed without error

SEGS (USINT) - Total number of structures used for move.

<<INSTANCE NAME>>:M_PRFDWL(EN00 := <<BOOL>>, MDST := <<REAL>>, MSCL := <<REAL>> RR := <<MEMORY AREA>>, MEXT := <<DINT>>, SEXT := <<DINT>>, OK => <<BOOL>>, SEGS => <<USINT>>);

This M_PRFDWL function block takes the array of structures pointer and fills the next structure with the necessary data for a Slave dwell in the profile. The Master distance will be adjusted by any rounding errors detected when the preceding move was calculated. The only slave motion will be any slave rounding errors detected. These will be applied at the end of the dwell.

There is no acceleration or jerk or initial velocity in a dwell move.



Inputs: EN00 (BOOL) - enables execution (Typically one-shot)

VLMT (REAL) - Maximum desired velocity.

MDST (REAL) - Master distance for this move.

SDST (REAL) - Slave distance for this move.

ACCP (REAL) - Percent of maximum possible accel to be used.

JERK (REAL) - Percent of maximum possible jerk to be used.

Outputs: OK (BOOL) - execution completed without error.

MAXV (USINT) - Maximum velocity calculated for move.

MAXA (LREAL) - Maximum acceleration rate calculated for move.

MAXJ (LREAL) - Maximum jerk calculated for move.

MACC (LREAL) - Master distance during acceleration (and deceleration).

TRPZ (BOOL) - Move changed to a trapezoid to achieve move.

<<INSTANCE NAME>>:M_SETVAJ(EN00 := <<BOOL>>, VLMT := <<REAL>>, MDST := <<REAL>> SDST := <<REAL>>, ACCP := <<REAL>>, JERK := <<REAL>>, OK => <<BOOL>>, MAXV => <<USINT>>, MAXJ => <<LREAL>>, MACC => <<LREAL>>, TRPZ => <<BOOL>>);

The M_SETVAJ function block calculates the acceleration, maximum velocity, and jerk to be used for this move.

The M_SETVAJ ASFB calculates the acceleration, maximum velocity, and jerk to be used for this move. Separate calls can be used if acceleration and deceleration are to be different. The distance that the master will move during the acceleration (and deceleration) is also calculated. If Accel is not the same as decel, care must be used to avoid invalid setup of the Ratio Real. Use MAXV for K1 of Decel. It is easy to have s-curve on either the acceleration or deceleration rather than both. The value of this is to allow the axis to accel or decel faster when inertia is lower, and allow more time for critical moves (i.e., a larger portion of a triangular move for the decel when concerned with a loaded part slipping out of the holding mechanism while the accel has no such concern).

MAXV is the maximum velocity that the profile will reach, and is the ending velocity for the accel portion of the move.

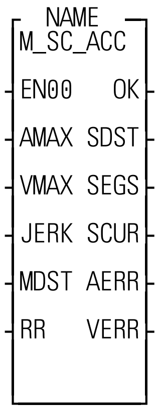
MACC is the distance that the Master axis will move during the accel.

For any such move profile, the acceleration can be "smoothed" by adjusting the jerk percent. This will change the acceleration (and deceleration) portions of the move to resemble an "S-Curve".

The percentage of jerk corresponds inversely to the portion of the Acceleration (or Deceleration) segment of the move which will be smoothed, until 100% equals no S-Curve. At minimum jerk, there is no constant accel portion. This will correspond to the highest acceleration rate. Maximum velocity during a move is not affected by “smoothing” or jerk, nor is the average acceleration. It only affects how the acceleration (and deceleration) will be applied to obtain this velocity.

M_SC_ACC

Acceleration segment **USER/M_PROFL**



Inputs: EN00 (BOOL) - enables execution (Typically one-shot)

AMAX (LREAL) - Maximum acceleration rate calculated for the move.

VMAX (LREAL) - Maximum velocity calculated for the move.

JERK (LREAL) - Maximum jerk calculated for this move.

MDST (LREAL) - Master distance during acceleration.

RR (STRUCTURE) - Array of structures to be used for profile.

Outputs: OK (BOOL) - Execution of function completed without error.

SDST (DINT) - Slave distance during acceleration.

SEGS (USINT) - Total number of segments used for acceleration.

SCUR (BOOL) - If set, acceleration uses an S curve move.

AERR (BOOL) - If set, acceleration equals 0.

VERR (BOOL) - If set, velocity equals 0..

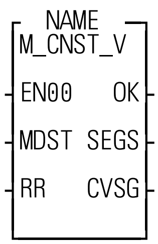
<<INSTANCE NAME>>:M__SC_ACC(EN00 := <<BOOL>>, AMAX := <<LREAL>>, VMAX := <<LREAL>> JERK := <<LREAL>>, MDST := <<LREAL>>, RR := <<MEMORY AREA>>, OK => <<BOOL>>, SDST => <<DINT>>, SEGS => <<USINT>>, SCUR => <<BOOL>>, AERR => <<BOOL>>, VERR => <<BOOL>>);

The M_SC_ACC function block adds the acceleration portion of the move to the profile.

The M_SC_ACC ASFB adds the necessary segments for the acceleration portion of the profile. If the JERK input is not equal to zero, there will be three segments for the acceleration. If the JERK input is zero, there will be one segment required.

M_CNST_V

Constant velocity segment **USER/M_PROFL**



Inputs: EN00 (BOOL) - enables execution (Typically one-shot)

MDST (REAL) - Master distance for this move.

RR (STRUCT) - Array of Structures to be used for profile.

Outputs: OK (BOOL) - Execution of function completed without error.

SEGS (USINT) - Total number of segments used for profile.

CVSG (BOOL) - A valid Constant Velocity structure was used flag.

<<INSTANCE NAME>>:M__CNST_V(EN00 := <<BOOL>>, MDST := <<REAL>>, RR := <<MEMORY AREA>> OK => <<BOOL>>, SEGS => <<USINT>>, CVSG => <<BOOL>>);

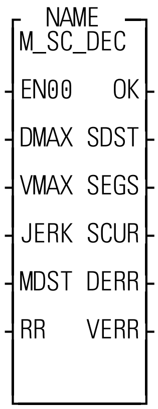
The M_CNST_V function block fills a Ratio Real structure with the necessary distances and polynomial co-efficients for a Constant Velocity move. The initial velocity is filled by an earlier function call.

A constant velocity move requires the initial velocity, K1 to be non-zero, and the initial and final values of both acceleration, K2/2, and Jerk, K3/6, to be zero.

The Master Move Distance cannot be zero for a RATIO_RL segment, therefore if this would be the case, then no constant velocity move is set up.

M_SC_DEC

Deceleration segment **USER/M_PROFL**



Inputs: EN00 (BOOL) - enables execution (Typically one-shot)

DMAX (LREAL) - Maximum acceleration rate calculated for the move.

VMAX (LREAL) - Maximum velocity calculated for the move.

JERK (LREAL) - Maximum jerk calculated for this move.

MDST (LREAL) - Master distance during acceleration.

RR (STRUCTURE) - Array of structures to be used for profile.

Outputs: OK (BOOL) - Execution of function completed without error.

SDST (DINT) - Slave distance during acceleration.

SEGS (USINT) - Total number of segments used for acceleration.

SCUR (BOOL) - If set, acceleration uses an S curve move.

DERR (BOOL) - If set, acceleration equals 0.

VERR (BOOL) - If set, velocity equals 0.

<<INSTANCE NAME>>:M__SC_DEC(EN00 := <<BOOL>>, DMAX := <<LREAL>>, VMAX := <<LREAL>> JERK := <<LREAL>>, MDST := <<LREAL>>, RR := <<MEMORY AREA>>, OK => <<BOOL>>, SDST => <<DINT>>, SEGS => <<USINT>>, SCUR => <<BOOL>>, DERR => <<BOOL>>, VERR => <<BOOL>>);

The M_SC_DEC function block adds the deceleration portion of the move to the profile.

The M_SC_DEC ASFB adds the necessary segments for the deceleration portion of the profile. If the JERK input is not equal to zero, there will be three segments for the deceleration. If the JERK input is zero, there will be one segment required.

NOTES

APPENDIX C Contents of the Applications CD

The Applications CD contains the Motion ASFBs, General ASFBs, Ethernet ASFBs and Example Applications for the Standalone MMC and the MMC for PC.

The following manuals can be found in the folder Manuals on your CD: Motion ASFB Manual, General Purpose ASFB Manual and Ethernet ASFB Manual. They are PDF files and can be opened, read, and printed using Adobe Acrobat Reader. Acrobat Reader can be downloaded free of charge from Adobe at http://www.adobe.com/supportservice/custsupport/download.html.

Note: You MUST use Acrobat Reader 4.0 or newer to view these manuals.

The PiCPro Applications can be used with either PiCPro Professional Edition or MMC-Limited Edition. The **Examples** folder has example projects for each edition of PiCPro.

The default folders when installing the Applications CD are (where Vxx.x.x represents the version of the applications software) :

- C:\G&L Motion Control Data\Applications Vxx.x.x\ASFB
- C:\G&L Motion Control Data\Applications Vxx.x.x\Examples
- C:\G&L Motion Control Data\Applications Vxx.x.x\Tools
- C:\G&L Motion Control Data\Applications Vxx.x.x\QuickStart

Note: You should not put any other files or applications into these folders.

The **ASFB** folder contains the Motion, General and Ethernet ASFBs.

The **Tools** folder contains the ServoSetupAssistant spreadsheet.

The **Examples** folder contains 8 Standalone MMC examples or starter ladders and several simple examples for the Motion, General and Ethernet ASFBs. There are also 3 MMC for PC examples or starter ladders included, although these cannot be downloaded using the MMC-Limited Edition.

The following is a description of how the examples are named.

- MMCxxxxx Standalone MMC Application Example.
- MPCxxxxx MMC for PC Application Example.
- SRCxxxxx SERCOS Setup Example.
- SRVxxxxx Standalone MMC Servo Setup Example.
- SRVPxxxx MMC for PC Servo Setup Example.
- ZSRCMMC SERCOS Setup Example Library.
- ZSRVMMC Servo Setup Example Library.
- xxxxxx.CPA Cimrex Operator Interface Application Example.
- xxxxxx.GLC G&L DeviceNet Scanner Configuration Example.
- M_xxxxxx Motion ASFB Example.
- G_xxxxxx General ASFB Example.
- E_xxxxxx Ethernet ASFB Example.

- U_xxxxxx MMC Application Example UDFB.

The following are brief descriptions of each Standalone MMC example (see Appendix D for more information).

- M_C2M_EX. The Cad2Motion ASFB, translates an M and G code format ASCII file into servo motion. Many applications require description of their motion path using CAD software. Third party packages (such as Gcode2000) will convert the CAD package DXF output to M and G code text files. M_C2M will translate the M and G code file to servo motion. Examples of applications include glue laying and textiles cutting. The Cad2Motion ASFB is not intended for applications that involve metal cutting machine tools such as lathes and mills and therefore does not support features required by CNC applications such as cutter radius compensation, tool offsets and constant surface speed.
- MMC4_EX. This example can be used as a starting point for a 4 axis application. It contains the logic required to initialize the servo axes, close the servo loops, jog an axis, and home an axis. It also contains basic E-Stop and C-Stop logic. This example does not have any operator interface support. All of the machine functions are performed with discrete I/O.
- MMC2_EX. This example can be used as a starting point for a 2 axis application. It contains the logic required to initialize the servo axes, close the servo loops, jog an axis, and home an axis. It also contains basic E-Stop and C-Stop logic. This example does not have any operator interface support. All of the machine functions are performed with discrete I/O.
- MMC4_OI. This example can be used as a starting point for a 4 axis application. It contains the logic required to initialize the servo axes, close the servo loops, jog an axis, and home an axis. It also contains basic E-Stop and C-Stop logic. This example uses a Cimrex 30 as an operator interface. The Cimrex application is called MMC4_C30.
- MMC2_OI. This example can be used as a starting point for a 2 axis application. It contains the logic required to initialize the servo axes, close the servo loops, jog an axis, and home an axis. It also contains basic E-Stop and C-Stop logic. This example uses a Cimrex 30 or Cimrex 20 as an operator interface. The Cimrex applications are called MMC2_C30 and MMC2_C20 (for the Cimrex 30 or Cimrex 20 units respectively).

- MMC4_SOI. This example can be used as a starting point for a SERCOS application having an operator interface device. It contains the logic required to initialize the SERCOS configuration, initialize the servo axes, close the servo loops, jog an axis, and home an axis. It also contains basic E-Stop and C-Stop logic. This example uses a Cimrex 30 as an operator interface. The Cimrex application is called MMC4_SC30.
- MMC4_PT. This example shows how to apply the press transfer application specific function blocks. It contains the logic required to initialize the servo axes, close the servo loops, jog an axis, and home an axis. It also contains basic E-Stop and C-Stop logic. It also contains logic to configure and run the specified slave profile(s). This example uses a Cimrex 30 as an operator interface. The Cimrex application is called M4PT_C30.
- MMC_STEP. This example shows how to apply the stepper block I/O module. It contains the logic required to initialize the stepper block and to control it. A single UDFB (U_STPRC) provides the control logic for a stepper axis. The features supported by the UDFB include home (zero reference), jog, move a distance, move to a position, control-stop, emergency-stop and c-stop/e-stop reset.

  MMC_DND. This example shows how to apply the Centurion DeviceNet Positioning MicroDSM drive. It contains the logic required to initialize and run the DeviceNet network. Two UDFBs provides the control logic for a DeviceNet axis to jog, move a distance or move to a position. A third UDFB provides the DeviceNet scanner status details. This example ladder and DeviceNet configuration are both referenced in the MicroDSM with DeviceNet installation manual.

The following is a brief description of each MMC for PC example (or starter ladder).

- MPC4_EX. This example can be used as a starting point for a 4 axis application. It contains the logic required to initialize the servo axes, close the servo loops, jog an axis, and home an axis. It also contains basic E-Stop and C-Stop logic. This example does not have any operator interface support. All of the machine functions are performed with discrete I/O. This example is almost identical to MMC4_EX.
- MPC4_OI. This example can be used as a starting point for a 4 axis application. It contains the logic required to initialize the servo axes, close the servo loops, jog an axis, and home an axis. It also contains basic E-Stop and C-Stop logic. This example uses a Cimrex C30 with an Ethernet module as an operator interface. The Cimrex application is called MPC4_C30. This example is almost identical to MMC4_OI.

- MPC4_SOI. This example can be used as a starting point for a SERCOS application having an operator interface device. It contains the logic required to initialize the SERCOS configuration, initialize the servo axes, close the servo loops, jog an axis, and home an axis. It also contains basic E-Stop and C-Stop logic. This example uses a Cimrex C30 with an Ethernet module as an operator interface. The Cimrex application is called MPC4_SC30. This example is almost identical to MMC4_SOI.

Note:For both of the MMC for PC with Cimrex examples, there will usually be a configuration change required before either works. The Cimrex configuration file calls out the Ethernet connection to the MMC for PC. However, the correct TCP/IP address of the PC hosting the MMC for PC is unknown. The actual host address must be verified in the Cimrex configuration before it is downloaded to the Cimrex unit.

APPENDIX D Standalone MMC Examples

Descriptions of the Standalone MMC Examples

The following is a description of the functionality provided with each Standalone MMC example.

The purpose of each example is to provide a foundation for a customer's own application. Each example provides all the basic features of most applications:

- Loading the servo configuration
- Closing the servo position loops
- Handling emergency stop and control stop conditions
- Jogging the axes
- Referencing (or homing) the axes

Typically, the ladder logic required for a specific application can be added to the end of an example.

Which example is the best starting point depends on the specific application (e.g. how many servo axes are required and will an operator interface device be used?).

Basic Application Examples

The following is a description of the two examples that lack an operator interface. They are MMC2_EX for a 2 axis MMC and MMC4_EX for a 4 axis MMC.

- The initial network with the STRTSERV function loads the servo setup configuration. Each example ladder includes a servo setup example (such as SRV2_EX for a 2 axis MMC or SRV4_EX for a 4 axis MMC). Each application must have its own servo setup configuration. Typically, this configuration is unique for every application as each could have different ratios of programming units (i.e., inches, millimeters, degrees, etc.) to the servo feedback units, different travel limits, different servo PID gains, etc.
- The M_CHK1 and M_CHK49 function blocks check which axes are configured within the loaded servo setup. These blocks make it easier to write sections of ladder code that support different numbers of axes; by checking the Booleans set here, the appropriate actions can be taken based on the axes that are configured.
- The M_CLOS1 function blocks will close the position loops for the configured servo axes. This action also includes resetting any internal e-stop or c-stop servo conditions (a servo e-stop will open the position loop).
- The M_STATUS and M_ERROR function blocks provide all the significant data for the servos in one place within the ladder. If the application is using the digitizing axis available with the MMC, there is a network that checks that axis for an e-stop condition (that can result from a 'loss of feedback' hardware error).

- The STATUSSV functions provide the ladder with information regarding the state of the respective servo fast inputs. Each axis has a DC input provided with a 'fast input' capability - so that (when enabled) the axis can latch the axis position at the instant the DC input occurs, based solely on the input condition rather than on the ladder scan time.
- The network with the ESTOPACT coil monitors for an emergency stop (or e-stop) condition. The purpose for this network is to ensure that if an e-stop condition is detected, all the servo axes will be e-stopped immediately.
- The network with the CSTOPACT coil monitors for a controlled stop (or c-stop) condition. The purpose for this network is to ensure that if a c-stop condition is detected, all the servo axes will be c-stopped immediately.
- The networks with the GEN_IO and AUX_IO labels provide a simple way to view all the MMC discrete input and output conditions using the ladder animation.
- The subsequent networks provide support for jogging and homing the axes. When jogging, only one axis can be selected and it can jog in one of three modes: at a constant velocity, to a specific incremental distance, or as a follower to a handwheel. Note that jogging will stop when a travel limit is reached because an e-stop will occur (in the ESTOPACT's coil network). However, if the Machine Start push-button is being pushed at the same time as the jog button, the operator can jog off a specific over-travel. For instance when in velocity mode, if the axis is on the plus over-travel switch, the axis cannot jog in the plus direction but it can jog in the minus direction.

The last networks in the basic examples provide support for the referencing (or homing) of the servo axes. The M_FHOME function block supports a homing sequence for the axis using the axis fast input as the reference switch. The M_LHOME function block supports a homing sequence for the axis using a specific discrete input as the reference switch. Both function blocks are provided for each servo axis in the ladder with M_FHOME actually connected to the rail. If the M_LHOME functionality is desired, then its code fragment (the function block and the surrounding elements) can be cut and pasted over the M_FHOME area.

Operator Interface Application Examples

The examples having operator interface support (MMC2_OI and MMC4_OI) have more ladder logic than the basic examples. Early in the ladder, the OI_SER function block enables the MMC variables (in the main ladder) that are given the G (global) attribute to be shared with the operator interface device (typically a Cimrex terminal) connected via an RS232 serial connection. Note that the global attribute is also used when the MMC is connected to an external PC via the Ethernet module and the MMC shares its data with the PC using the OPC Server technology.

There are a number of networks that update the state of the ALARM() Boolean array. This array is used to provide immediate indicators to the operator interface device regarding specific ladder conditions.

After the axis jogging and homing section of the ladder, there are several networks involved with the saving and retrieval of the axis setup data using a RAMDISK file. Rather than having the various motion rates or reference positions hard coded within the ladder logic, they can be set by the operator and saved in a RAMDISK file, AXSETUP.DAT.

The examples conclude with a few networks that are the basis for supporting other Cimrex features: trending or graphing and the control of the LEDs.

SERCOS Drive Interface Application Example

There is an example provided for the 8 axis SERCOS MMC unit. This example (MMC4_SOI) is an extension of the 4 axis MMC example with an operator interface. For the support of the SERCOS drives, several different ASFBs are used.

At the start of the ladder logic, the SC_INIT function initializes the SERCOS ring connected to the MMC. Each SERCOS application must have its own SERCOS setup configuration. The example uses SRC_MMC4 for its 4 SERCOS axis configuration. For an actual application, this SERCOS setup function must be replaced with the actual SERCOS setup configuration. The programmer's steps to do that are very similar to those for replacing the servo setup, except that steps 7 through 10 in the above procedure (when STARTING A NEW APPLICATION FROM AN EXAMPLE) are done with SERCOS setup, not servo setup. The example uses SRV_MMC4 for its servo setup configuration. For an actual application, this servo setup function must be replaced with the actual servo setup configuration, as previously described.

The S_CLOS1 function block will close the position loop for the configured SERCOS axes. This ASFB is different for SERCOS axes because the SERCOS drive itself might have error conditions to reset before the position loop can be closed.

The S_ERRORC function block provides all the significant error information for a specific SERCOS axis. This ASFB is different than M_ERROR because a SERCOS axis has additional error information.

The S_FHOME and S_LHOME function blocks provide support for the referencing of the SERCOS axes. They are different than M_FHOME and M_LHOME because the latching of the reference position is done within the SERCOS drive.

The S_IO_C function block is unique to a Centurion SERCOS drive application because the additional discrete input and output points at the SERCOS drive can be made available to the ladder.

Press Transfer Application Example

There is an example provided for the press transfer ASFB set within the motion ASFB package. This example (MMC4_PT) is an extension of the 4 axis MMC example with an operator interface. There is a detailed 'readme' file for this example's user interface and the accompanying ladder logic; that file is MMC4_PT.TXT.

Stepper Application Example

There is an example provided for the stepper block I/O module. This example (MMC_STEP) is a simple ladder to illustrate the control logic for a stepper module.

The UDFB U_STPRC handles the overhead required for a stepper control. This UDFB handles the stepper home or zero reference, jog, move a distance, move to a position, controlled-stop, emergency stop, and a reset of a c-stop or e-stop.

Centurion DeviceNet Positioning MicroDSM Drive Application Example

There is an example provided for the Centurion DeviceNet MicroDSM drive. This example (MMC_DND) is a simple example to illustrate the logic to control the DeviceNet scanner and to control the DeviceNet drive over DeviceNet.

The ASFB M_DNJOGC handles the overhead required for the jog motion. The ASFB M_DNPOSC handles the overhead required for the motion of either a move of an incremental distance or a move to an absolute position. Note that both of these ASFBs are for the Centurion DeviceNet MicroDSM drive. Other DeviceNet drives will have their own unique DeviceNet interface (i.e., the respective boolean, byte and long-word tag name assignments in the scanner's memory map).

The ASFB M_DNSTAT extracts the details for the DeviceNet module status into individual bytes and booleans.

This example ladder is referenced in the MicroDSM with DeviceNet installation manual. It serves as the programming example for the MicroDSM with DeviceNet drive. The MMC_DND.GLC DeviceNet scanner's configuration data file is also referenced in the same drive manual to illustrate how to define the tag names for the DeviceNet drive's input and output data.

Starting a New MMC Application from a Standalone MMC Example

The following is a description of how to start a new application from a Standalone MMC example.

To use one of the examples as a starting point for your application follow the steps below. The following procedure will use the MMC4_EX Professional Edition Project as the starting point. The target application will be called NEW_APPL and the servo setup library will be ZNEWAPPL; other appropriate names can replace these used in this procedure. The same steps can be followed with the PiCPro MMC Limited Edition.

Note: In the instructions below, xx.x and xx.x.x represent the current version of the software.

1. Start PiCPro Vxx.x Professional Edition.
2. File | Open, MMC4_EX Pro Edition.PRJ. This will open up the project.

3. Once the project is open, do a File | Save As and fill in the following fields:
   - Destination PRJ
     C:\G&L Motion Control Data\Applications Vxx.x.x\New_Appl\New_Appl.prj
     This will be the name and folder for the new project.
   - Destination G&L
     C:\G&L Motion Control Data\Applications Vxx.x.x\New_Appl\New_Appl.G&L.
     This is the file name of the compressed project. The compressed file can be stored on the MMC Flashdisk. This allows the end user the ability to open the project from the MMC without having the original source code.
4. Change the destination folders for Main LDO, Lib Path, UDFB Source, Servo Source, and Servo Lib's from:
   C:\G&L Motion Control Data\Applications Vxx.x.x\Examples
   To:
   C:\G&L Motion Control Data\Applications Vxx.x.x\New_Appl
5. Close the Save As Project by clicking OK.
6. Expand the Main .LDO and double click on MMC4_EX.LDO. This will open the main ladder and do a File | Save As with the new file name NEW_APPL.ldo.
7. In network 5 (the network with the STRTSERV function), right click on SRV4_EX and then View Servo Setup. Do a File | Save As to SRV4_NEW.
8. Make any necessary changes to the servo setup and then do a Compile | Make Function. When prompted for the library name for the servo function, change the name to ZNEWAPPL.
9. Close servo setup.
10. Replace the SRV4_EX function with the new servo setup function you just created by clicking on Ladder |Functions | ZNEWAPPL | SRV4_NEW.
11. Close the NEW_APPL ladder window and save the changes when prompted.
12. Go to back to the project window and delete MMC4_EX from the MAIN .LDO and add NEW_APPL.ldo.
13. Do a File | Update Project Tree to update the project.
14. Close the project. When prompted to compress the application select No. You do not need to compress the project until you are ready to store it on the flash disk.
15. Using Windows Explorer delete the MMC4_*.* files, the SRV4*.* files and ZSRVMMC.LIB file from the new project folder:
    C:\G&L Motion Control Data\Applications Vxx.x.x\New_Appl

Note: If the application is using SERCOS-controlled servos, the above procedure is followed starting with MMC4_SOI with additional steps to replace the example SERCOS setup file.

NOTES

APPENDIX E Digital Smart Drive Examples

Introduction

This document provides instructions on how to select and modify the best example ladder diagram for your application. Prior to doing so, you should have most of the following information worked out and available.

- Number of servo axes
- Number of digitizing axes
- Inputs and assignments used in application
- Outputs and assignments used in application
- Power On and Emergency Stop Methods
- Position Loop Closure Method
- Manual Features - jog
- Homing methods and sequence

The example ladders include the above areas. You will need to configure them according to your requirements.

You will also require information on the following to complete the application.

- Operator Interface
- Automatic Motions and Sequence of Operations

These are not part of the example ladders.

Purpose

Example ladders are provided as a starting base for customer applications. They include the basic functions needed for all applications in a recommended order and format and are the simplest way to get an application up and running.

The example ladders assign some input and output points to features only to illustrate techniques. You can stay with these assignments or change them as you wish. A specific configuration of features such as homing and jogging are set up. You will need to change these according to your requirements. This document contains information on how to do some of this. Other information can be found in the network comments and PiCPro Online Help or the Function Block manual. Other information may be found in the Application Notes on the Danaher Motion web site.

The following examples are provided in their own folders

- 1 Axis using the D1 Resident MMC Digital Smart Drive Module
- 2 Axis using the D2 Resident MMC Digital Smart Drive Module
- 4 Axis using the D4 Resident Digital Smart Drive Module
- 16 Axis using the D16 Resident Digital Smart Drive Module

- 32 Axis using the D32 Standalone MMC Digital Smart Drive Module
- 64 Axis using the D64 Standalone MMC Digital Smart Drive Module

In each folder are

- Ladder diagram (LDO) and network comments (REM) files
- Servo setup (SRV) and library (LIB) files
- Project (PRJ) file

The PRJ file can be used to open the example applications so they can be viewed. However, the examples should be left intact and not modified in place for an application.

Getting Started with PiCPro

The first step is to choose the example closest to the target application that uses the same MMC Digital Smart Drive Module. Since it is easier to remove axes than to add them, choose one with more axes if the exact number is not there. Copy ONLY the LDO, REM and SRV files to the working folder for the application. Rename the 3 files to what you want them called in the application using Windows Explorer.

Set up the PiCPro Libraries pointers to include Applications /ASFB and the working application folder.

Open the SRV file and by subtracting any unused axes. Then modify the remaining axes for the application and do a Compile / Make Function to create a library (LIB) file. See the notes on the SRV file later for more information on the settings in the example files.

Open the renamed LDO. It will give a message about a missing servo function. Put your own servo function in its place in the appropriate network.

If necessary, delete the 4 networks for any unused axes - see below.

At this point, do a Compile Only on the ladder to check for any errors. There should not be any. If there are, either correct them or start over.

Save the ladder and then do a File - New - Project. PiCPro will build a project tree showing the folders and files needed by the ladder diagram. Do a File Save when complete. You will be saving the newly created project (PRJ) file and from now on should open that file when starting a new PiCPro session.

Modifying the Ladders

A number of comment networks are provided at the beginning of the ladder. Some of these contain specific G&L information. You should read and understand the Warranty and Revision History network headers before proceeding. They contain important information.

You should adapt the contents to your specific situation. You are free to add more comment networks if you wish.

Network 1 - Danaher Motion Warranty Statement

Network 2 - comment network only for application information, revision history etc.

Network ….

Network Label SRV_INIT - Servo Setup Initialize reads the specific axis information from the servo setup file. Your own file is now in here.

Network Label AXES_CHK - Calculates information on axes that are initialized. See the MCHKALL ASFB Help for detail.

AXES_CNT - Axis and other data are stored in memory arrangements called Structures and are passed from the main ladder into function blocks that perform operations on them. It is critically important that the structures are the same in both places. This network calculates the sizes of the structures declared in the main ladder. They will be used later by the function blocks to compare with their own internally declared sizes. If the 2 differ, an error code will be output by the function block and it will not run.

Network Label GEN_IN - Shows contacts of all available inputs based on the hardware declarations. We recommend you connect coils of internal relays that you declare and name per the functionality and use contacts of the coil throughout the ladder. As a more conventional alternative, it is also possible to rename the point in Software Declarations and use the input point contacts in the ladder.

Network Label ESTPALL - The example ladders E stop and C stop all axes if any one has a stop of that type. This network makes sure that all axes see the signal by making sure it lasts for one full ladder scan. The network also contains logic to do the same thing for Drive Warnings.

Network Label DRVPWR - Energizes a coil to be used to energize a general output point for use with a starter that switches the main ac input to the drives. This is an important safety feature and should be included on all applications.

Network Label GEN_OUT - Takes contacts of coils energized in the ladder and turns on output points. You need to configure this network according to your application.

Network Label AX1INIT sets up initial data for axes features such as homing and jogging. This is only one way of doing this. It is also possible to place these values in the Initial Values column of Software Declarations. If the Software Declarations method is chosen, this network can be deleted. It is done here to highlight the options that need to be chosen. It is done in Structured Text format so that each line can be commented. It could also be done using a MOVE function block in ladder diagram format enabled by a positive transition contact of the First Scan coil. If a feature is not being used, then the data for it need not be set up. Be particularly careful with the last IF - END_IF statement. This contains code to write drive current limits on start up. The values there are for a specific drive type and need to be changed for the type being used. The statements are included to cover a possible but unlikely situation. The drive current limit can be reduced from the ladder while certain home operations are running. It will be restored at the end of the home cycle, but if the power was to fail and the ladder stopped scanning during that cycle, it would not be. Therefore some mechanism for restoring it without using PiCPro is needed. If you don't intend to use the current reduction feature, delete the lines completely.

Network Label - AX1DRIO provides drive input and output signals and control. These should be handled by the same methods described for general inputs and outputs. If drive I/O is not used for an axis, this network can be deleted for that axis.

Network Label - AX1_CTL provides the logic for closing the position loop as well as emergency and controlled stops. Also provides the logic for jogging and homing. You need to implement the logic for all the coils in this network per your application. If a feature such as jogging is not used, you can delete the coils and logic feeding them.

Network Label - AX1_FN contains the function block SD_AXIS that controls the axis position loop, errors and warning information, jogging and homing features.

Subsequent networks will repeat the functions of networks these 4 networks for the other included axes. They will take the same names as these networks modified by the axis number i.e. AX1INIT becomes AX2INIT and so on.

If you do not need 1 or more axes, then delete the 4 networks related to those axes.

APPENDIX F Servo Setup Assistant

The ServoSetupAssistant is an Excel spreadsheet tool that makes creating your servo setup information a little bit easier. You specify machine parameters such as time base and precision in the ServoSetupAssistant spreadsheet. Based on these parameters the servo setup information for an axis is automatically calculated and can be imported into PiCPro using the Import button that is provided under the 'Axis Data' tab. The Servo Setup Assistant spreadsheet (ServoSetup Assistant.xls) is part of the Applications Software CD installation. It can be found in the Tools folder (C:\G&L Motion Control Data\Applications Vxx.x.x\Tools, where xx.x.x represents the current version of the software). To use this spreadsheet:

1. Do a File Save As to save this spreadsheet as a different name to maintain the original. A different .XLS file should be made for each axis in your application.
2. Two sheets are provided for each type of axis that is allowed: Closed loop servo, Digitizing, and Closed loop SERCOS. The first sheet is used to input data, the second sheet holds the servo setup data that will be imported into PiCPro. You may delete the sheets for axes that are not being used. Double click on the tab for a sheet to change the name of the sheet to match the description of the axis.
3. Enter the data for the axis. The setup information is being automatically calculated on the next sheet as you enter. Enter only information with pull down boxes or data that appears in blue. When you are ready to save the information for an axis select the sheet called 'PiCPro 'X' Setup Data' (where X is Servo, Digitizing or SERCOS Servo). NOTE: Before you save the file in .CSV format, you must save the entire workbook by doing a File Save to keep your edits. If you do not do this, all formulas and changes will be lost when you save the file in .CSV format.
4. From the File menu select Save As and save the sheet under a different name, with file type .CSV (comma delimited variable). Excel will prompt you through the save process. All other sheets in the workbook and all formulas will be deleted in the .CSV file.
5. Close the files in Excel.
6. Start PiCPro. Import the data from the .CSV file by selecting the Import button under axis data.

NOTES

Index

L

M