Motion

Application Specific Function Block Manual

Part Number M.1300.7191
Version 12.0.1

Giddings & Lewis
Controls, Measurement and Sensing

NOTE

Progress is an on-going commitment at Giddings & Lewis. 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. Giddings & Lewis 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 Giddings & Lewis 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. Giddings & Lewis 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, Giddings & Lewis, 660 South Military Road, P.O. Box 1658, Fond du Lac, WI 54936-1658. Giddings & Lewis can be reached by telephone at (920) 921–7100.

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.

M.1300.7191

Release 0201

©1993, 94, 95, 96, 97, 98, 99, 2000, 2001Giddings & Lewis, LLC

IBM is a registered trademark of International Business Machines Corporation.
Windows 95, 98, NT, Microsoft, and MS-DOS are registered trademarks of Microsoft Corporation.
Pentium and PentiumPro are trademarks of Intel Corporation.
ARCNET is a registered trademark of Datapoint.
PiC900, PiCPro, MMC, PiCServoPro, PiCTune, PiCProfile, LDO Merge, PiCMicroTerm and PiC Progrmming Pendant are trademarks of Giddings & Lewis, LLC

Table of Contents: Motion ASFB Manual

CHAPTER 1-Application Specific Function Block Guidelines 1-1

Installation 1-1

Revisions 1-1

Network 1 1-1
Network 2 1-1
Network 3 1-2

ASFB Input/Output Descriptions 1-2

Network 4 1-2

Using ASFBs 1-2

CHAPTER 2-Motion ASFBs 2-1

ADDCKSUM 2-6
BYTE2HEX 2-7
CHKCKSUM 2-8
DWOR2HEX 2-9
HEX2BYTE 2-10
HEX2DWOR 2-11
HEX2WORD 2-12
M_CHK1 2-13
M_CHK101 2-14
M_CHK109 2-15
M_CHK49 2-16
M_CHK57 2-17
M_CHK65 2-18
M_CHK73 2-19
M_CHK9 2-20
M_CLOS1 2-21
M_CLOS9 2-23
M_CLS101 2-25
M_CLS109 2-27
M_CRSFIN 2-29
M_DATCAP 2-31
M_DATCPT 2-36
M_DNJOGC 2-40
M_DNPOSC 2-41
M_DNSTAT 2-43
M_DSMCOM 2-45
RS232 Connections 2-50
RS422/RS485 Connections 2-50
M_DW2BOO 2-51
M_ERROR 2-52

M_FHOME 2-53
M_INCPTR 2-56
M_JOG 2-57
M_LHOME 2-58
M_LINCIR 2-61
M_PRTCAM 2-65
M_PRTREL 2-66
M_PRTSLP 2-67
M_RATREL 2-68
M_RATSLP 2-69
M_RDTUNE 2-71
M_RGSTAT 2-72
M_RSET49 2-74
M_RSET57 2-75
M_RSET65 2-76
M_RSET73 2-77
M_SCRVLC 2-78
M_SRCMON 2-84
M_SRCPRC 2-86
M_SRCRDL 2-88
M_SRCWT 2-90
M_SRCWTL 2-92
ERR Output 2-93
SERR Output 2-95
BSER Output 2-97
M_STATUS 2-98
M_WTTUNE 2-100
S_CLOS1 2-102
S_CLOS9 2-104
S_ERRORC 2-106
S_FHOME 2-108
S_IO_C 2-110
S_LHOME 2-112
WORD2HEX 2-115

APPENDIX A-M_DSMCOM Commands A-1

Exception Responses A-1
Host Command Set A-2
Common Product Line Commands A-3
General Commands A-4
Position Loop Commands A-5
Velocity Loop Commands A-6
Torque Current Conditioning Commands A-7
Motor Commands A-8
Motor Commands (Continued) A-9
Motor Commands (Continued) A-10

Digital I/O Commands A-11
Analog I/O Commands A-12
Analog I/O Commands (Continued) A-13
Serial Port Commands A-14
Operating Mode Commands A-15
Operating Mode Commands (Continued) A-16
Alternative Operating Mode Commands A-17
Alternative Operating Mode Commands (Continued) A-18
Runtime Command and Control Commands A-19
Runtime Status Commands A-20
Runtime Status Commands (Continued) A-21
Runtime Data Commands A-22
Runtime Data Commands (Continued) A-23
Runtime Data Collection Commands A-24
Runtime Data Collection Commands (Continued) A-25

APPENDIX B-Press Transfer ASFBs B-1

M_PRF2MV B-6
M_PRF1MV B-17
M_PRFERR B-18
M_PROFL B-21
M_PRFDWL B-23
M_SETVAJ B-24
M_SC_ACC B-26
M_CNST_V B-27
M_SC_DEC B-28

Index Index-1

NOTES

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 Giddings & Lewis.

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:\Program Files\Giddings & Lewis\Applications **v*xx.x.r***\ASFB

and

C:\Program Files\Giddings & Lewis\Applications **v*xx.x.r***\Examples

where **v*xx.x*** *is the PiCPro for Windows 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 Giddings & Lewis 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 Giddings & Lewis personnel is 00 to 49. The range of numbers available for you is 50 to 99. See chart below.

Revision	Giddings & Lewis 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.

ASFB Input/Output Descriptions

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.

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_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_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_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_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_JOG	Jogs a closed loop axis.
	M_LINCIR	Performs linear, circular, and simultaneous endpoint arrival moves on closed loop axes.
	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_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_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.

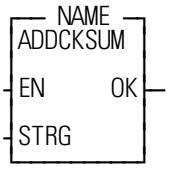
S_CLOS1	Closes the loop on SERCOS servo axes 1 to 8 (to replace M_CLOS1)
S_CLOS9	Closes the loop on SERCOS servo axes 9 to 16 (to replace M_CLOS9)
S_ERRORC	Returns e-stop, c-stop, and programming errors for a SERCOS servo axis or e-stop errors for a SERCOS digitizing axis; SERCOS ring and slave errors 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)

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 RATIOSLP 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

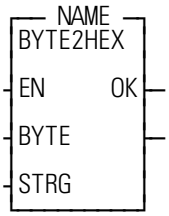
STRG(STRING) - input string

Outputs: OK (BOOL) - execution complete

This function 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

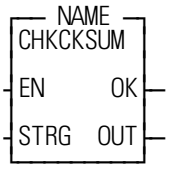
BYTE (BYTE) - value to convert

STRG(STRING) - converted value

Outputs: OK (BOOL) - execution complete

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.



Inputs:EN (BOOL) - enables execution

STRG(STRING) - input string to check

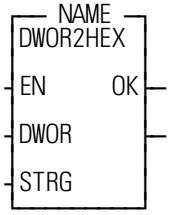
Outputs: OK (BOOL) - execution complete

OUT (BOOL) - checksum OK output

This function 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 (DWOR) - value to convert

STRG (STRING) - converted value

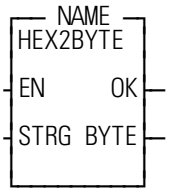
Outputs: OK (BOOL) - execution complete

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

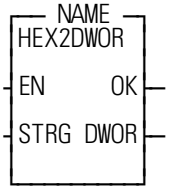
BYTE (BYTE) - converted value

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

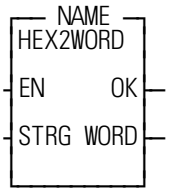
DWOR (DWOR) - converted value

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

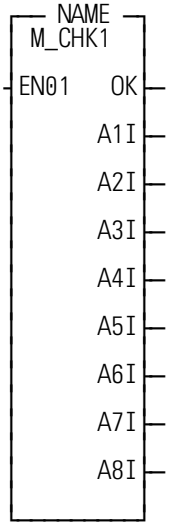
STRG (STRING) - hexadecimal value to convert

Outputs: OK (BOOL) - execution complete

WORD (WORD) - converted value

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: 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

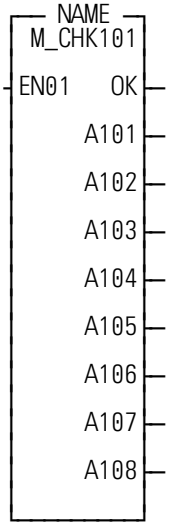
A8I (BOOL) - set if axis number 8 has been initialized

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.



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).

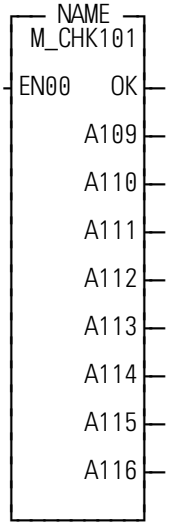
A108 (BOOL) - set if axis number 108 has been initialized (servo axis 24).

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.



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).

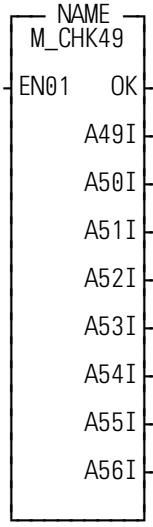
A116 (BOOL) - set if axis number 116 has been initialized (servo axis 32).

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.



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

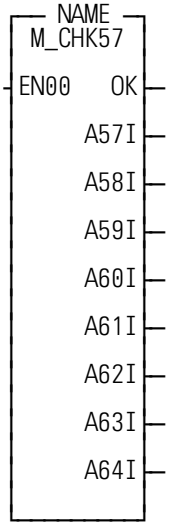
A56I (BOOL) - set if axis number 56 has been initialized

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

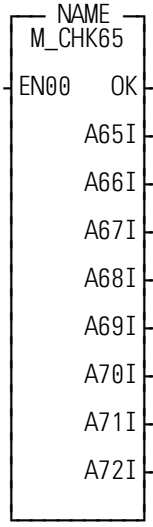
A64I (BOOL) - set if axis number 64 has been initialized

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.



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

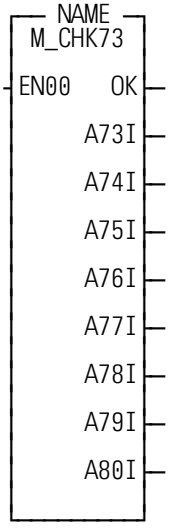
A72I (BOOL) - set if axis number 72 has been initialized

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.



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

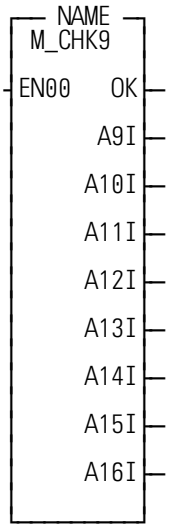
A80I (BOOL) - set if axis number 80 has been initialized

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: EN00 (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

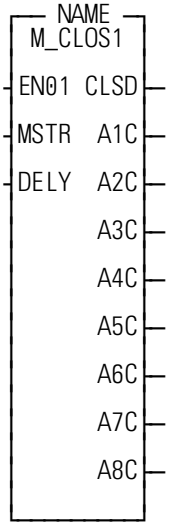
A16I (BOOL) - set if axis number 16 has been initialized

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: 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

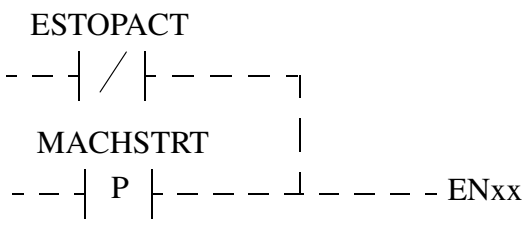
A7C (BOOL) - set when the loop is closed on axis 7

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

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.



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.

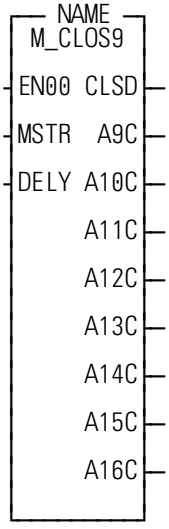
The machine start input must go through a positive transition (off to on) to reset the errors and close the loop.

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.



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

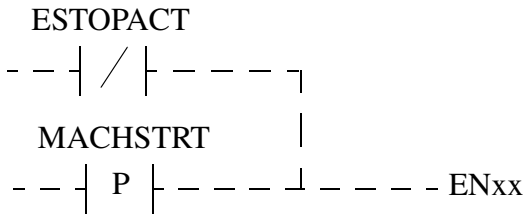
A15C (BOOL) - set when the loop is closed on axis 15

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

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.



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.

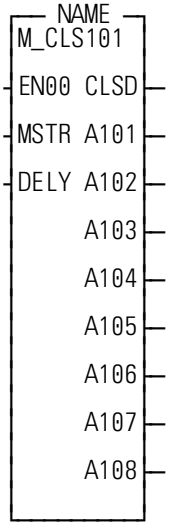
The machine start input must go through a positive transition (off to on) to reset the errors and close the loop.

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)

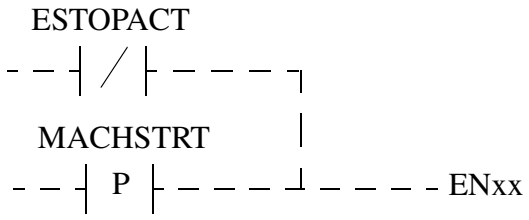
A107 (BOOL) - set when the loop is closed on axis called 107 (the 23rd defined axis)

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

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.

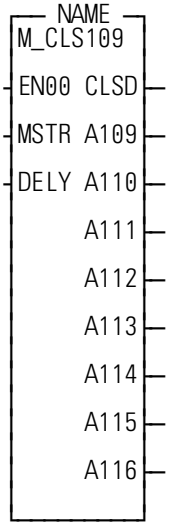
The machine start input must go through a positive transition (off to on) to reset the errors and close the loop.

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.



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)

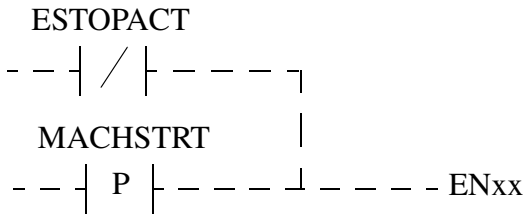
A115 (BOOL) - set when the loop is closed on axis called 115 (the 31st defined axis)

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

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.



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.

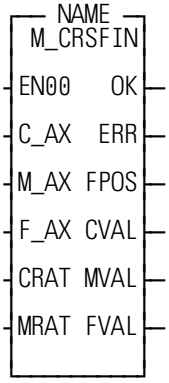
The machine start input must go through a positive transition (off to on) to reset the errors and close the loop.

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: 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

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:

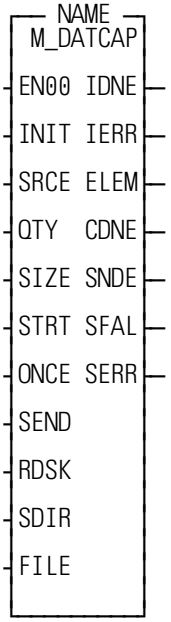
ERR	Description
0	No error
1	The OK from the READ_SV function for the fine axis was not set
2	The OK from the READ_SV function for the medium axis was not set
3	The OK from the READ_SV function for the coarse axis was not set
4	M_AX and C_AX inputs are both zero
5	M_AX was non-zero, but MRAT was zero
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.



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

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.

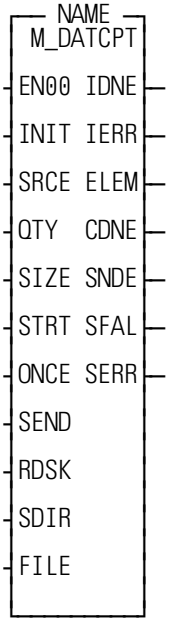
Once the data has been captured, it can be sent to a file on the RAMDISK or workstation. The data will be sent when the SEND input is pulsed. If the RDSK input is ON when the SEND input is pulsed, then the captured data will be sent to the PiC900 RAMDISK. If the RDSK input is OFF when the SEND input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



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

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.

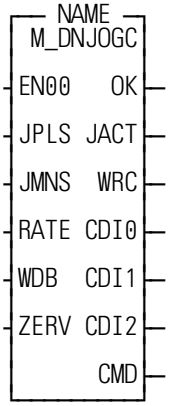
Once the data has been captured, it can be sent to a file on the RAMDISK or workstation. The data will be sent when the SEND input is pulsed. If the RDSK input is ON when the SEND input is pulsed, then the captured data will be sent to the PiC900 RAMDISK. If the RDSK input is OFF when the SEND input 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 Software Manual)
101 to 199	Error occurred on file write
201 to 299	Error occurred on file close



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

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.

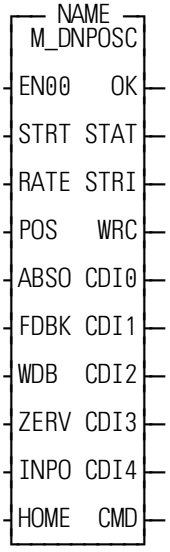
Important - If the enable is disabled while a move is underway the axis will continue to move until the jog switch is deactivated.

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

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:

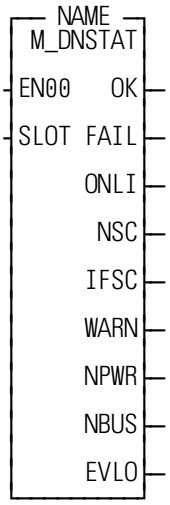
1 = Axis is Positioning

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

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.

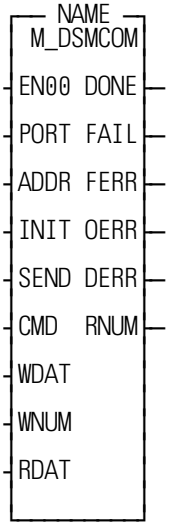
WARN is set when the communication warning threshold has been exceeded.

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

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.

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. These errors are defined in Appendix B. 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.

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

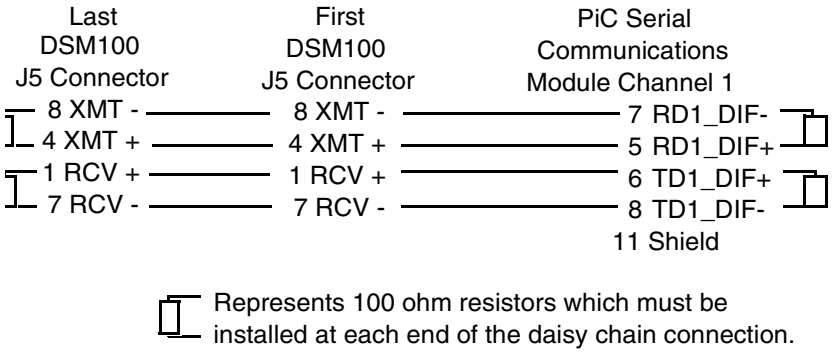
In single drive applications where the communications link is less than 50 feet, a three wire RS232 serial communication link may be used. The pinout is shown below.

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

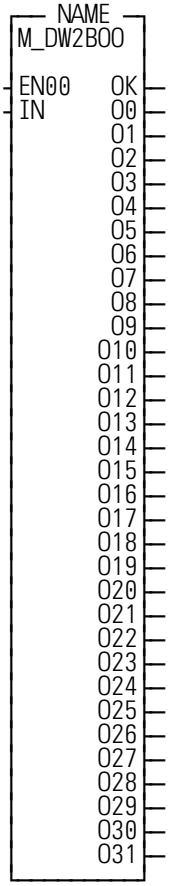
Typically, the M_DSMCOM function block will be used with RS422/RS485 serial communication. RS422/RS485 provides superior noise immunity, allows communication links greater than 50 feet, and allows multiple drive connections to one PiC.

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.

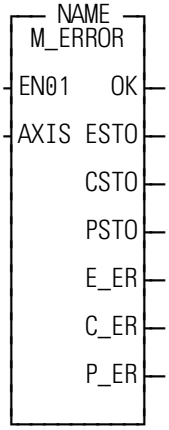


Inputs: EN01 (BOOL) - enables execution

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)

This function block converts a DWORD to 32 BOOLs.



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

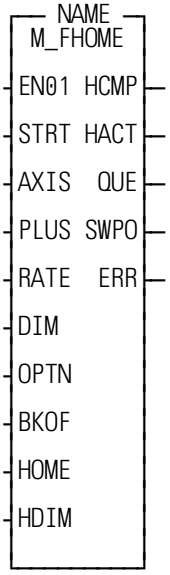
This function is used to report **servo** E-stop, C-stop and programming error conditions in the ladder. These conditions may be caused by the servo software or defined by the programmer. If defined by the programmer they will be triggered using the E-STOP or C_STOP functions. All of these errors for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

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.

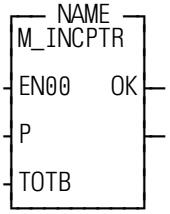
The SWPO output is used to determine if the reference switch location will allow for repeatable referencing. If the reference switch is not properly located in relationship to the index marker of an encoder or the null of a resolver it could possibly reference a revolution off. To prevent this, the value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



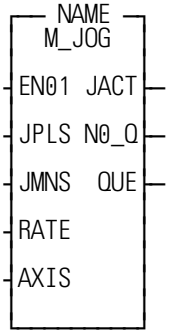
Inputs:EN00 (BOOL) - enable

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.



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

This function 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.

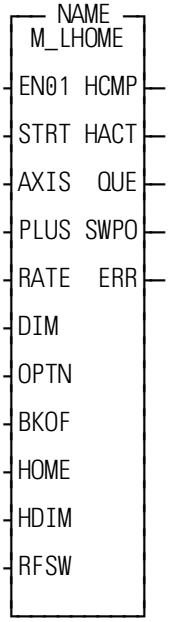
IMPORTANT

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.



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

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

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.

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.

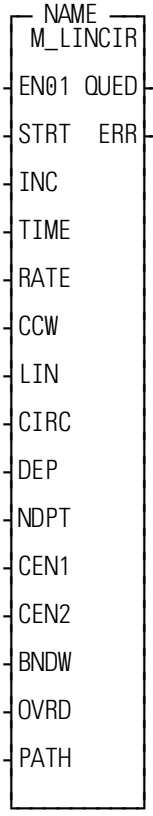
The SWPO output is used to determine if the reference switch location will allow for repeatable referencing. If the reference switch is not properly located in relationship to the index marker of an encoder or the null of a resolver it could possibly reference a revolution off. To prevent this, the value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



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

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.

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.

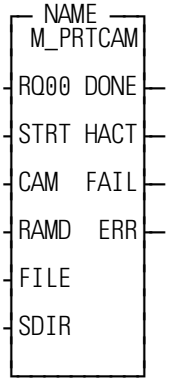
The ERR output will be non-zero if an error occurs in queueing a move. A list of error codes is shown on the following table.

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
32767	One of the OKs on the RATIO_RL functions did not get set

M_PRTCAM

Creates a RATIOCAM text file **USER/M_DATA**



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.

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.

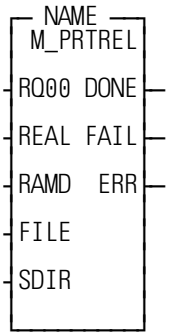
A positive transition of RQ00 requests that the data specified by the CAM input be converted to ASCII code, concatenated, and written to the RAMDISK or to the PiCPro port.

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. If a sub directory is desired, the initial value can have up to eight characters. A "\" is not required. If a subdirectory is not desired, the initial value should be left blank.



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.

This function block creates a text file for a RATIO_RL structure. The file can be created on either the RAMDISK in the PiC or on the PC running PiCPro.

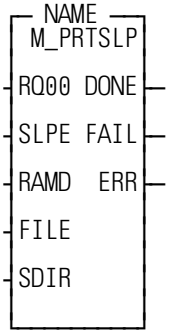
A positive transition of RQ00 requests that the data specified by the REAL input be converted to ASCII code, concatenated, and written to the RAMDISK or to the PiCPro port.

The REAL input is an array of structures and must have the following 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. If a sub directory is desired, the initial value can have up to eight characters. A "\" is not required. If a subdirectory is not desired, the initial value should be left blank.



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.

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.

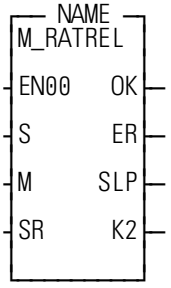
A positive transition of RQ00 requests that the data specified by the SLPE input be converted to ASCII code, concatenated, and written to the RAMDISK or to the PiCPro port.

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. If a sub directory is desired, the initial value can have up to eight characters. A "\" is not required. If a subdirectory is not desired, the initial value should be left blank.



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

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 PiC900 software manual 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:

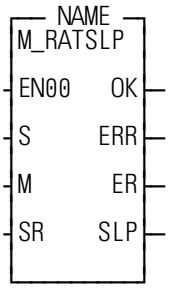
ER = (2S / M) - SR

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

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 PiC900 software manual 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:

$$ER = (2S / M) - SR$$

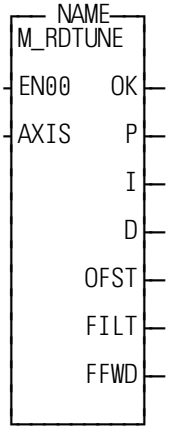
$$SLP = (ER - SR) / M$$

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

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).

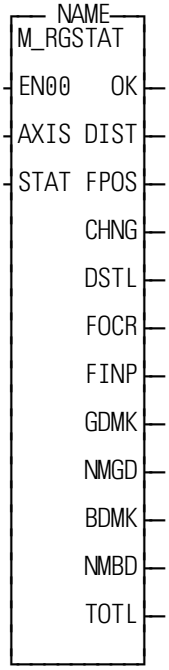
The integral gain for AXIS will be returned in I. I is in ladder units per minute per ladder units of following error times minutes (LU / MIN / LUFE * MIN).

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.



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

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.

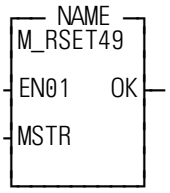
The GDMK output will be set if a good mark has been detected since the last time the STATUSSV function was called.

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

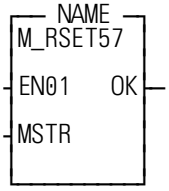
Outputs: OK (BOOL) - execution complete

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

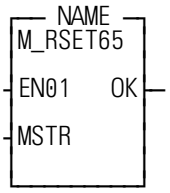
Outputs: OK (BOOL) - execution complete

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.



Inputs: EN01 (BOOL) - enables execution

MSTR (BOOL) - machine start input

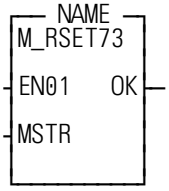
Outputs: OK (BOOL) - execution complete

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

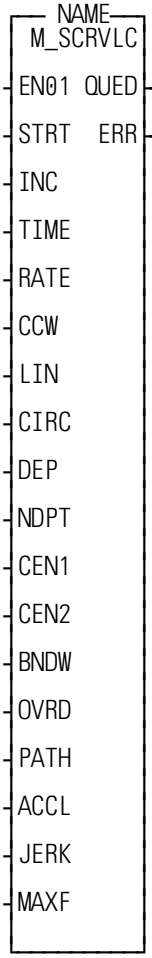
Outputs: OK (BOOL) - execution complete

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: 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

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.

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.

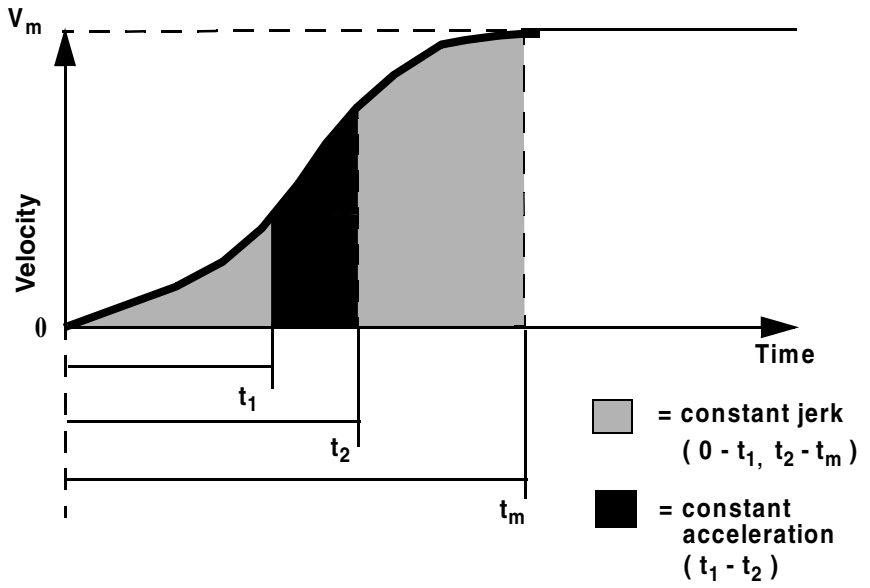
The QUED output will be set for one scan when STRT is pulsed and the move has been successfully queued on all axes defined. If an error occurred in queueing the move, this output will be reset when STRT is pulsed, and an error code will be stored in the ERR output.

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
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

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.

$$t_1 = t_m - t_2 = \left(\frac{1}{2} \times s \times t_m\right)$$

$$t_2 = t_m - \left(\frac{1}{2} \times s \times t_m\right)$$

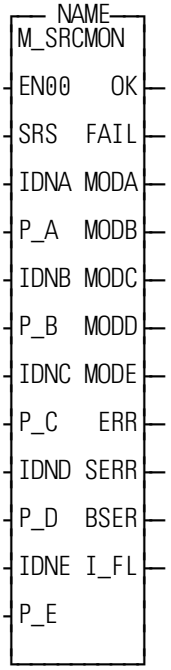
For a 10% S-curve, 10% of the time (t_m) is spent in constant jerk.
This means that s = 0.1.
For a 20% S-curve, 20% of the time (tm) is spent in constant jerk.
This means that s = 0.2, etc.

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.



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.

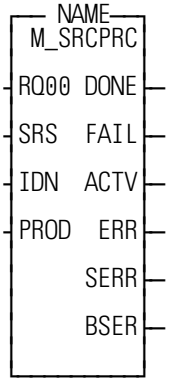
The M_SRCMON function block monitors up to five SERCOS IDNs for a single SERCOS slave. The operation data for each IDN is continuously read as long as the EN00 input is energized.

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.



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.

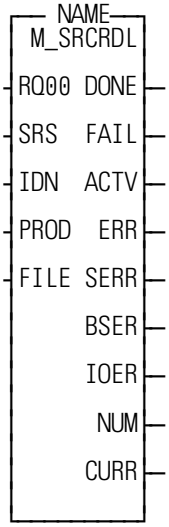
The M_SRCPRC function block executes a SERCOS procedure command function for a single SERCOS slave. The RQ00 input of this function block should be one-shot to initiate the procedure command function. While the procedure command function is executing within the SERCOS slave, the ACTV output will be set. 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.



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 Software Manual.)

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.

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.

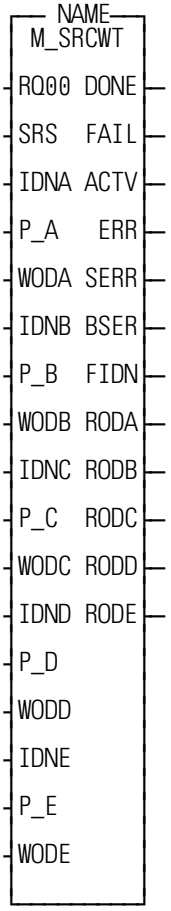
The NUM output indicates the total number of IDNs that exist in the list being read. The CURR output indicates the current member of the list being read and will range from 0 to NUM.

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.



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.

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.

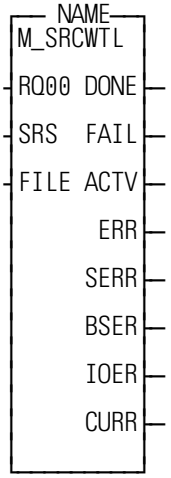
The IDNA through IDNE inputs can be used or left blank. When the RQ00 input transitions from off to on, the attributes of each IDN are read and saved in the function block. These attributes are used to scale the data at the input to the correct units for the SERCOS slave. After the attributes are read 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.



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 Software Manual.)

CURR (UINT) - current IDN being written

*See error tables at end of the this function block section.

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. RAM-DISK:\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 Giddings & Lewis.

33 Incorrect address error, contact Giddings & Lewis.

34 Incorrect AT number error, contact Giddings & Lewis.

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 Giddings & Lewis.
73 Incorrect AT count error, contact Giddings & Lewis.
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

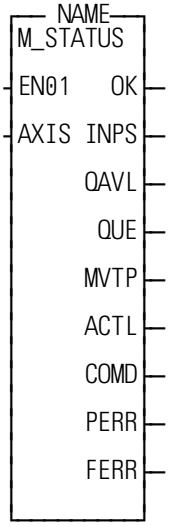
20483 The minimum value transmission is too long
20484 The minimum value may not be changed
20485 The minimum value is write-protected
24577 The maximum value does not exist
24578 The maximum value transmission is too short
24579 The maximum value transmission is too long
24580 The maximum value may 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



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

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.

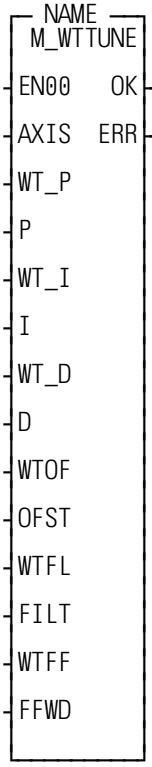
The COMD output holds the commanded position of the axis in ladder units.

The PERR output holds the proportional 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: 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

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.

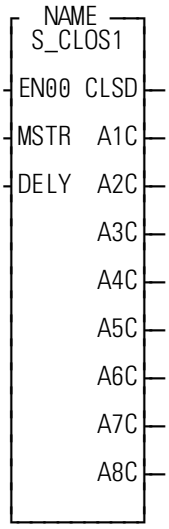
When the WTFL input is set, the slow speed filter value of AXIS will be changed to the value entered at FILT. FILT must be from 0 to 10000 milliseconds.

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



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

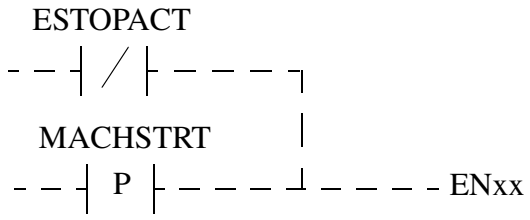
A8C (BOOL) - set when the loop is closed on axis 8

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.

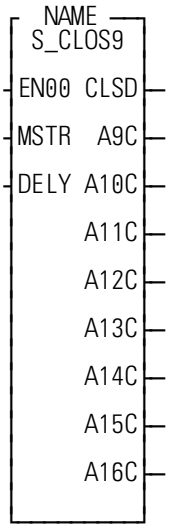
The machine start input must go through a positive transition (off to on) to reset the errors and close the loop.

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.



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

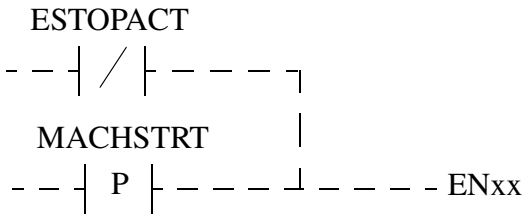
A16C (BOOL) - set when the loop is closed on axis 16

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.

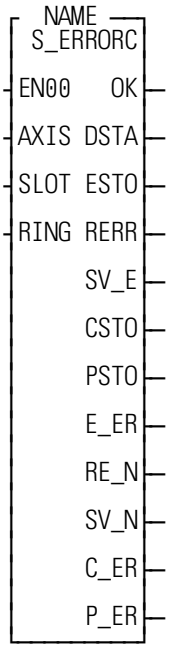
The machine start input must go through a positive transition (off to on) to reset the errors and close the loop.

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

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

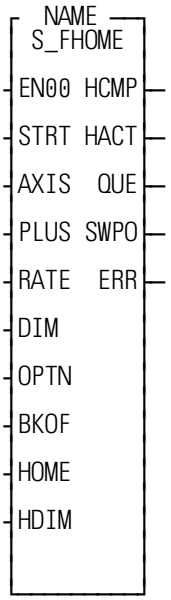
This function block is a replacement for M_ERROR for a SERCOS axis with a Centurion drive. It is not for an analog controlled axis.

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).



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

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.

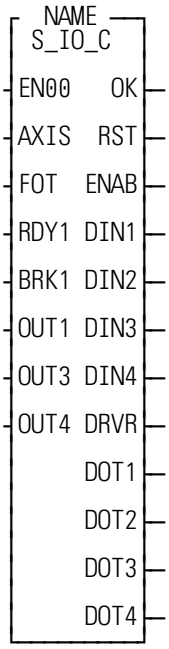
This function block is used to perform a fast reference, immediately followed by a position move to a selected home position. It should be executed every scan unless a home cycle will only be performed when the machine is started. In that case a normally closed contact of the output of HCMP may be used.

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 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: EN00 (BOOL) - enables execution

AXIS (USINT) - identifies SERCOS axis

FOT (BOOL) - force the outputs of a Centurion SERCOS drive

RDY1 (BOOL) - state to be sent to the drive ready output

BRK1 (BOOL) - state to be sent to the drive brake output

OUT1 (BOOL) - state to be sent to drive output one

OUT2 (BOOL) - state to be sent to drive output two

OUT3 (BOOL) - state to be sent to drive output three

OUT4 (BOOL) - state to be sent to drive output four

Outputs: OK (BOOL) - execution complete

RST (BOOL) - state of the drive reset input

ENAB (BOOL) - state of the drive reset 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 signal

DRVB (BOOL) - state of the drive brake signal

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

This function block provides the ladder access to the inputs and outputs of a Centurion SERCOS drive through the SERCOS service channel.

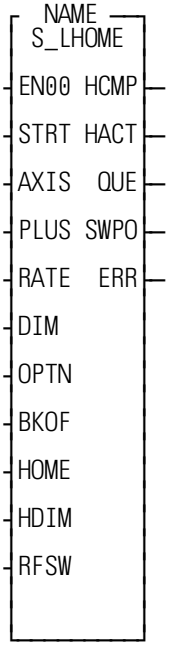
This function block provides the ladder access to the inputs and outputs of a Centurion SERCOS drive through the SERCOS service channel. It is not for a non-Centurion SERCOS drive and it is not for an analog controlled axis.

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 FOT when enabled will transfer the state of the next six function block inputs to the Centurion SERCOS drive outputs.

The digital output override IDN P0036 must be set in the drive to use this feature. Refer to the Centurion SERCOS Drive IDN Manual.

The function block outputs can be used as flags in the ladder to report the state of the SERCOS drive hardware.



Inputs: EN00 (BOOL) - enables execution

AXIS (USINT) - identifies SERCOS axis

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

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

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.

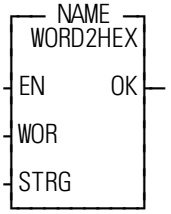
If the value is out of range either the reference switch will have to be moved or the transducer coupling shifted.

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.

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

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

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.

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	-	-	-	

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) (80000000..7fffffff)	units / count	Read	013	-	[d1]	
			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) (80000000..7fffffff)	units / RPM	Read	017	-	[d1]	
			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) (80000000..7fffffff)	units /RPM /second	Read	01b	-	[d1]	
			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. The scale is a 32 bit IEEE floating point value.	[d1] - Value -3.4e+38..3.4e+38 (ε: 1.19e-7) (80000000..7fffffff)	units / Amp	Read	01f	-	[d1]	
			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 isstill 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)	milliseconds	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)	milliseconds	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

Parameter	Range of Data Values	Units	Command		Com-mand Data	Response Data	Excep-tion 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

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. 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.	[w1] - Number 0..65535 (0000..ffff) BITS USAGE 15..12 Table ID 11..8 Encoder Resolution 7 Type (0 = synch., 1 = induct.) 6..0 Motor Number	-	Read	090	-	[w1]	
			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)	$kg\text{-}cm^2$	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

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

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

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: 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	[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]	
	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)	milliseconds	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)	milliseconds	Read	0ca	-	[s1]	
	(8001..7fff)	(× 0001)	Write	0cb	[s1]	-	03, 04

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

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

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) 2 - 4800 (TP: 4800) 3 - 9600 (TP: 9600) 4 - 19200 (TP: 19200)	-	Read	0de	-	[b1]	
			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) 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)	-	Read	0e0	-	[b1]	
			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 (00..ff)	-	Read	0e2	-	[b1]	
			Write	0e3	[b1]	-	03

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 thedrive 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

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

Parameter (Operating Mode Commands)	Range of Data Values	Units	Command		Command Data	Response Data	Exception Responses
Operating Mode The operating mode for the drive. Usually, the drive is in Normal mode. The mode can bechanged 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 algorithim.	[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 algorithim.	[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)	milliseconds	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)	milliseconds	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

Parameter (Alternative Operating Mode Com)	Range of Data Values	Units	Command		Com-mand Data	Response Data	Excep-tion Responses
Save Alignment Offset Corrects the encoder alignment by copying the Encoder Alignment Offset value into an encoder alignment compensation parame-ter. 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	-	-	-	

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	-	-	

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	-	-	-	

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 TouchPad 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	-	-	-	-

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	-	-	-	

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	-	-	-	

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)	milliseconds	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	-	-	-	

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.

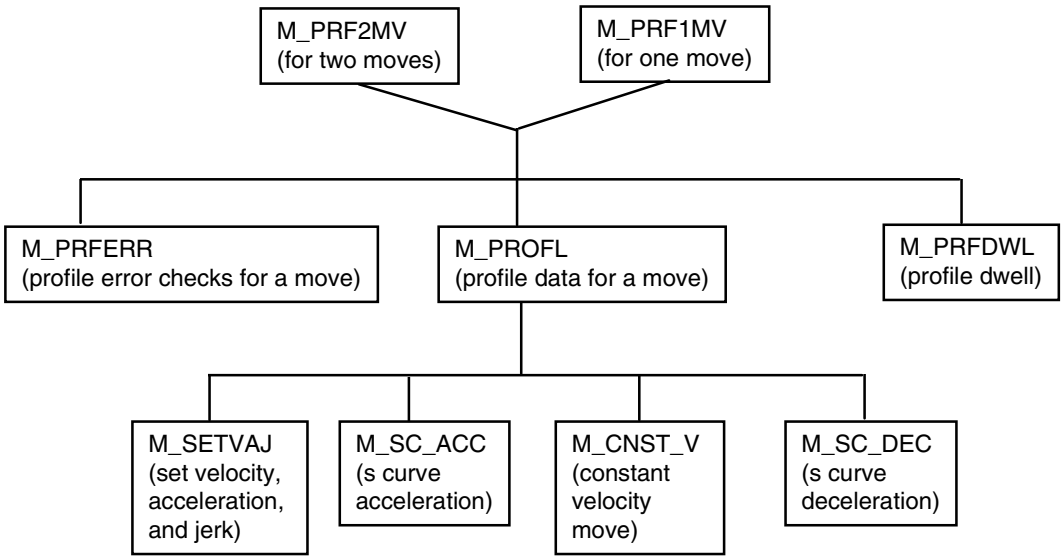
This is the set of function blocks whose purpose is to generate a slave profile for a press application.

- 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.

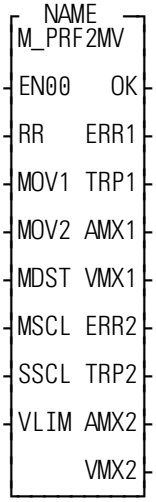
- M_SC_DEC - this function block adds the deceleration portion of the profile into the main structure (for a RATIO_RL). Depending on the move's parameters, the deceleration will be constant deceleration or it will be an scurve (i.e., smoothed deceleration).

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.



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.

MOV2 (STRUCTURE) - Structure containing 2nd 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 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.

The M_PRF2MV function block sets up 2 slave moves in the master cycle.

This function block is designed for a rotary master axis such as a press. It sets up a two move Ratio Real profile, with dwell segments after each. The moves can be different directions or the same, and different directions or the same. Either move can use smoothed "S-Curve" acceleration by input selection. Either move can be

trapezoidal or triangular, again by input selection (smoothing can be used in either case).

This function block controls the setup for the profile. The successful call of this function block results in a filled array of structures for a Ratio_Real profile which is ready to be started by a call of RATIO_RL using the array of structures set up herein. The format for the RATIO REAL structure is shown in Table 2-1.

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
.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. Reserved for possible future features.
.FLAGS	DWORD	Move type flags. Bits 2 & 3 = 0 for polynomial

The shape of the profile is determined by the input parameters. There are two separate moves in this profile, with zero-speed slave dwells after each. The format for the MOV1 and MOV2 structures is shown in Table 2-2. MOV2 has the same structure format as MOV1.

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

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

Assuming no errors, the data is separated and scaled for both the Master & Slave moves. This data is checked, then fed into the appropriate Move or Dwell function block.

Final error checking is done before returning OK.

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. If either a smaller or a slightly larger acceleration rate is given, the profile will look the same because the slave must still move the same distance for the same master motion. However if a large enough acceleration rate is given for the maximum velocity limits and the respective master and slave distances specified in the input move structure then the generated profile will become trapezoidal and the output boolean variable will be set to indicate that change in behavior.

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.

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.

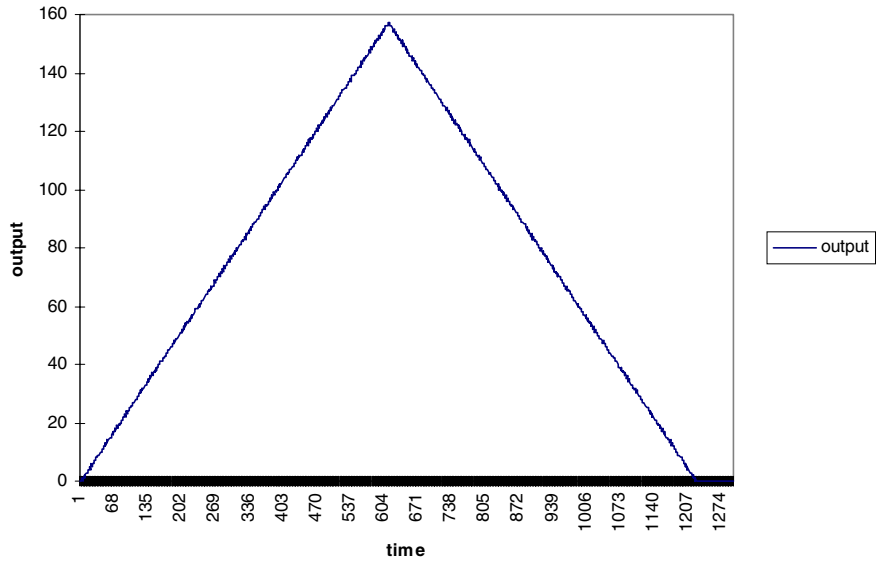
3. Trapezoidal profile with scurve, 50% acceleration and 50% jerk.

4. Trapezoidal profile with scurve, 50% acceleration and 95% jerk.

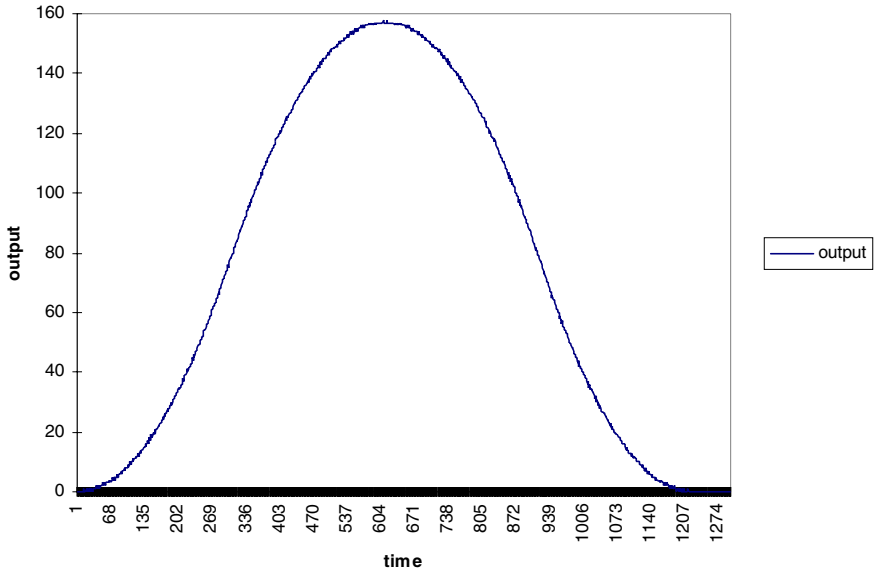
5. Trapezoidal profile with scurve, 10% acceleration and 50% jerk.

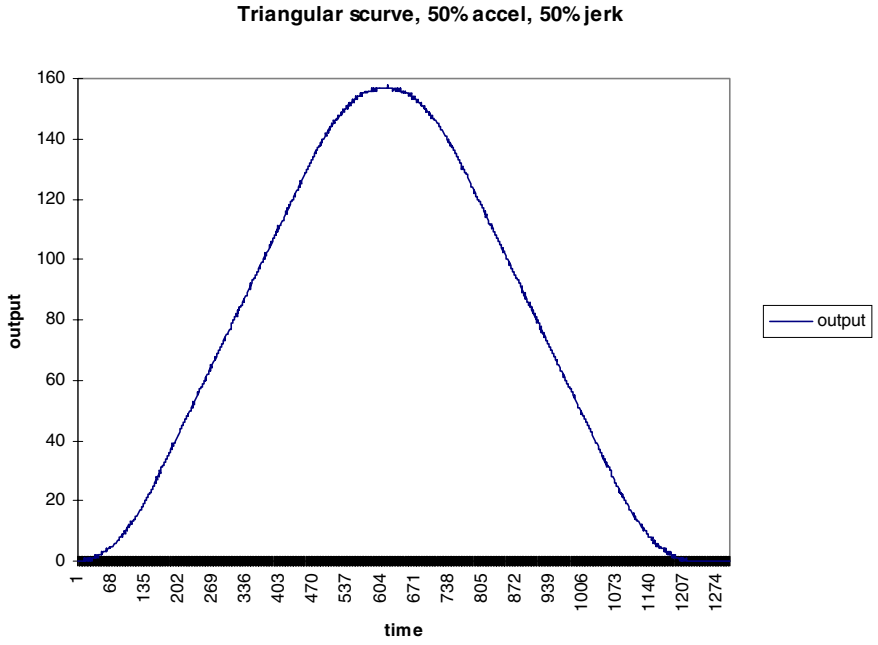
6. Trapezoidal profile with scurve, 90% acceleration and 50% jerk.

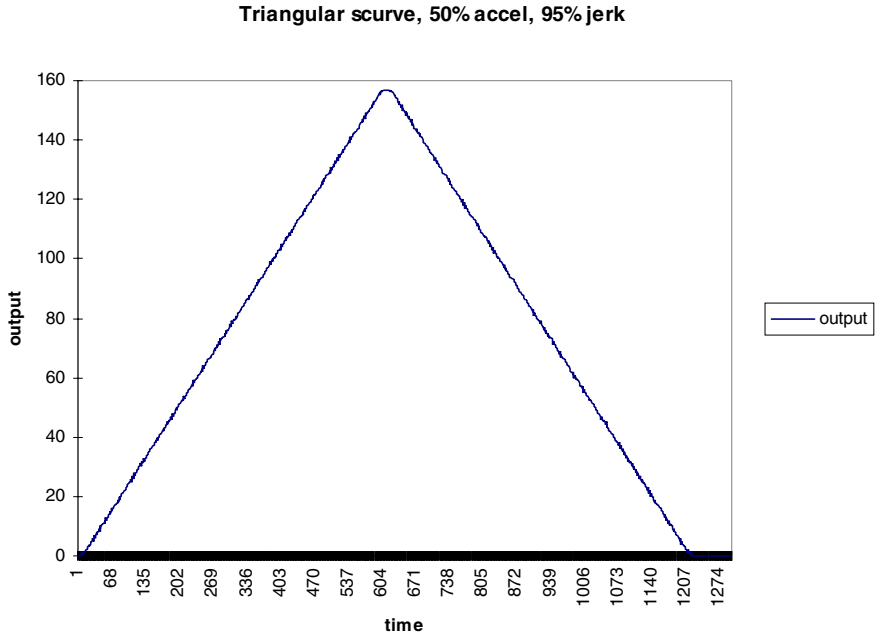
Figure 2-1. Triangular no scurve, 50% accel

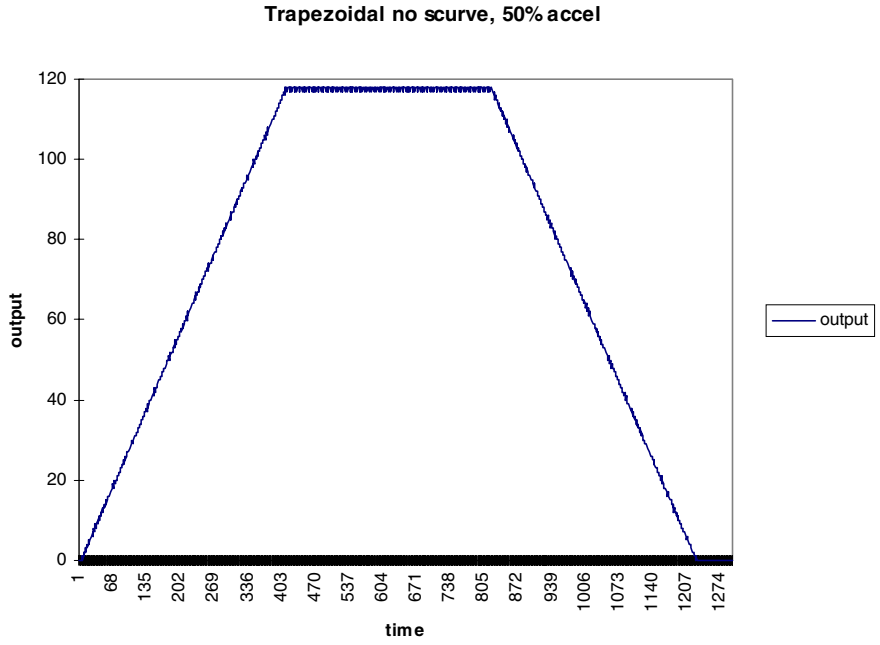


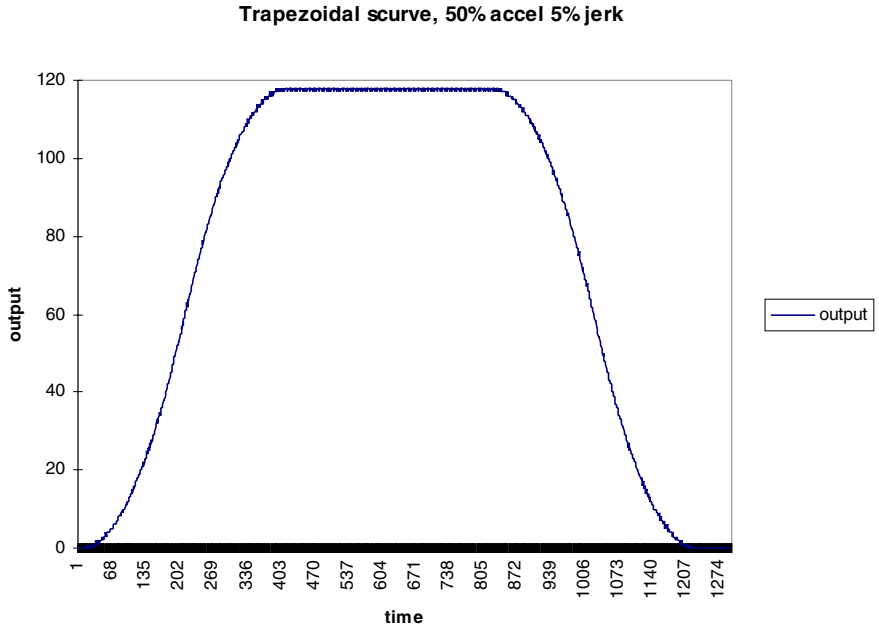
Triangular scurve, 50% accel, 5% jerk

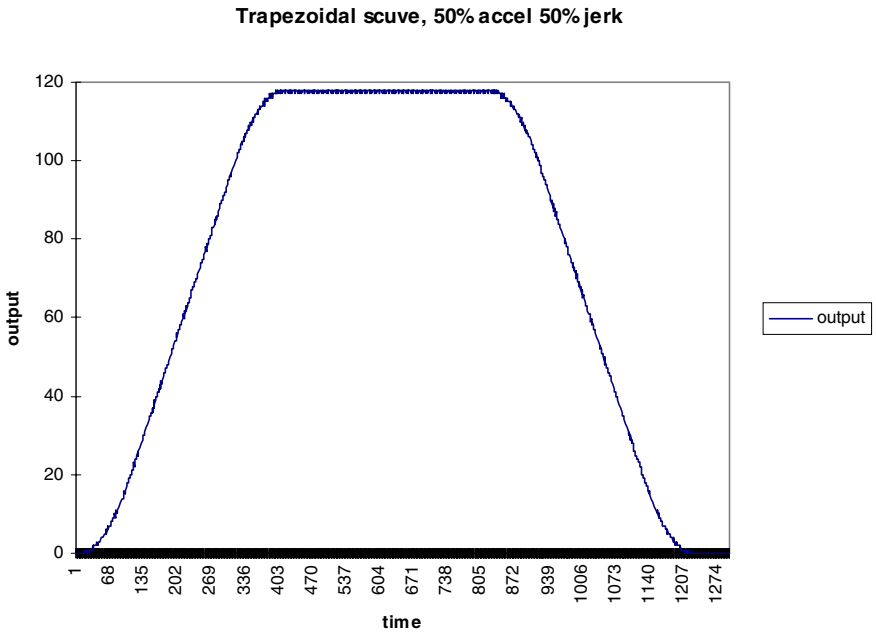


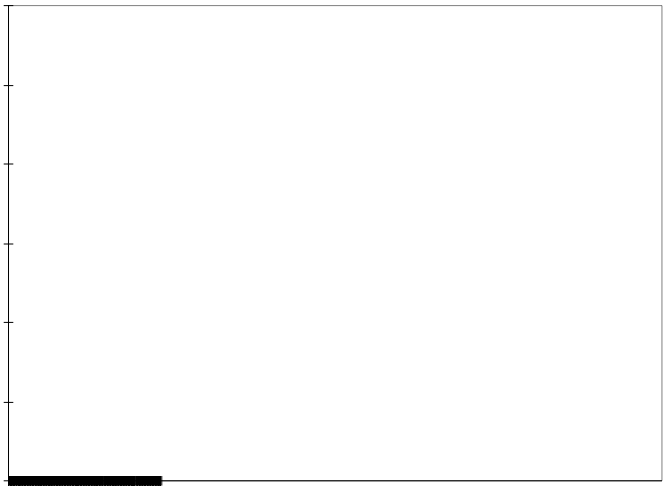

Triangular scurve, 50% accel, 50% jerk
output
time
output
0
20
40
60
80
100
120
140
160
1
68
135
202
269
336
403
470
537
604
671
738
805
872
939
1006
1073
1140
1207
1274

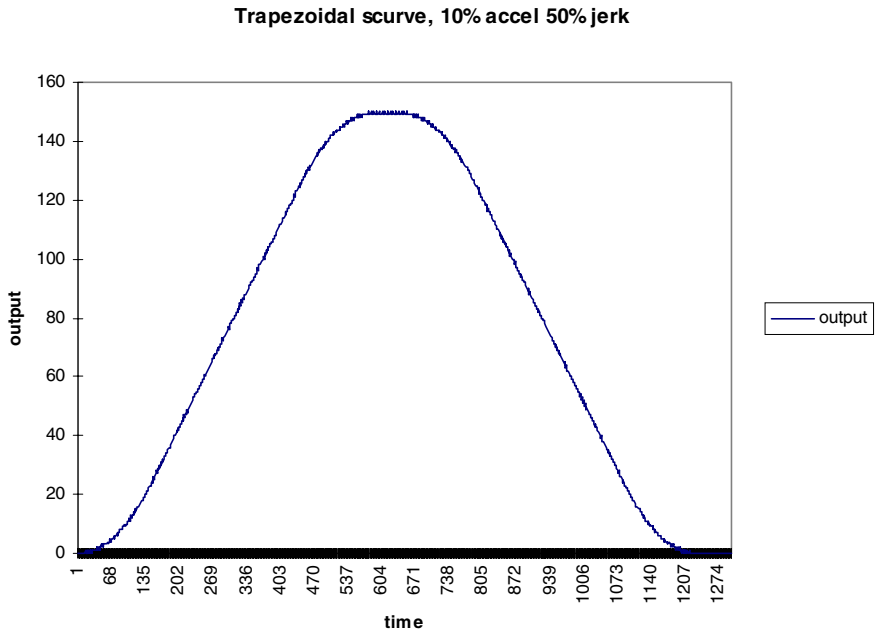

Triangular scurve, 50% accel, 95% jerk
output
time
output
0
20
40
60
80
100
120
140
160
1
68
135
202
269
336
403
470
537
604
671
738
805
872
939
1006
1073
1140
1207
1274

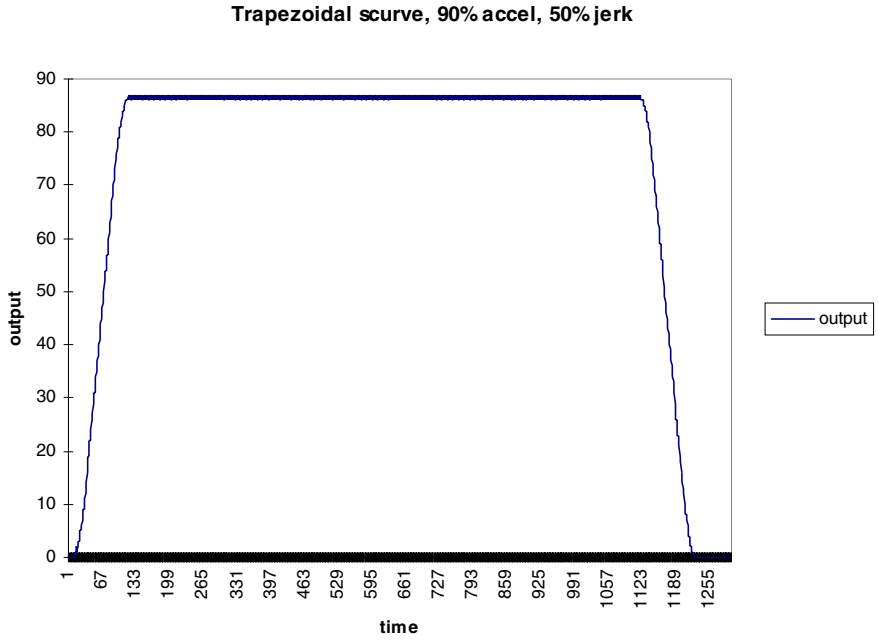

Trapezoidal no scurve, 50% accel
120
100
80
60
40
20
0
output
1
68
135
202
269
336
403
470
537
604
671
738
805
872
939
1006
1073
1140
1207
1274
time
output

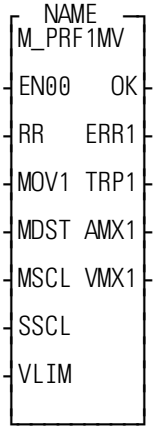

Trapezoidal scurve, 50% accel 5% jerk
120
100
80
60
40
20
0
output
1
68
135
202
269
336
403
470
537
604
671
738
805
872
939
1006
1073
1140
1207
1274
time
output


Trapezoidal scuve, 50% accel 50% jerk
120
100
80
60
40
20
0
output
1
68
135
202
269
336
403
470
537
604
671
738
805
872
939
1006
1073
1140
1207
1274
time
output




Trapezoidal scurve, 10% accel 50% jerk
160
140
120
100
80
60
40
20
0
output
1
68
135
202
269
336
403
470
537
604
671
738
805
872
939
1006
1073
1140
1207
1274
time
output


Trapezoidal scurve, 90% accel, 50% jerk
90
80
70
60
50
40
30
20
10
0
output
1
67
133
199
265
331
397
463
529
595
661
727
793
859
925
991
1057
1123
1189
1255
time
output



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.

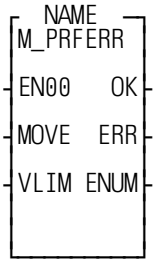
The M_PRF1MV function block sets up a single slave move in the master cycle.

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.

M_PRFERR

Check for profile errors **USER/M_PROFL**



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.

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

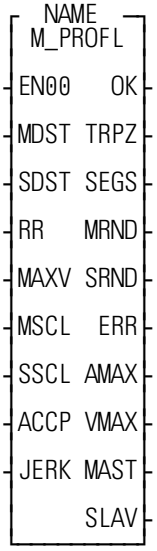
If any of the values are invalid, then a fault is flagged (ERR). All faults found are coded into a byte which is passed to the calling ladder. Use BYT2BOOL to decode faults. Only the first 6 bits are set in this function block. The upper 2 bits are used and set in the calling ladder. The Error Definitions 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.

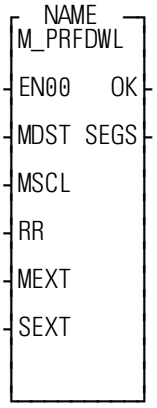
The M_PROFL function sets up one move

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.

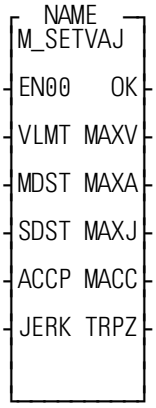
SEXT (DINT) - Extra slave feedback units.

Outputs: OK (BOOL) - execution completed without error

SEGS (USINT) - Total number of structures used for move.

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 acccel 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.

The M_SETVAJ function block calculates the acceleration, maximum velocity, and jerk to be used for this move.

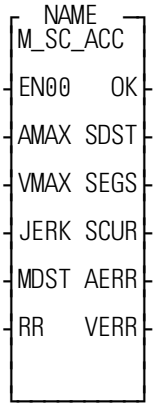
The M_SETVAJ ASFB calculates the acceleration, maximum velocity, and jerk to be used for this move. Separate calls can be used if acceleration and deceleration are to be different. The distance that the master will move during the acceleration (and deceleration) is also calculated. If Accel is not the 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.



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.

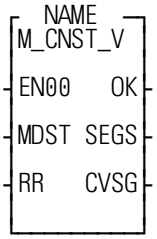
SCUR (BOOL) - If set, acceleration uses an S curve move.

AERR (BOOL) - If set, acceleration equals 0.

VERR (BOOL) - If set, velocity equals 0..

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.



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.

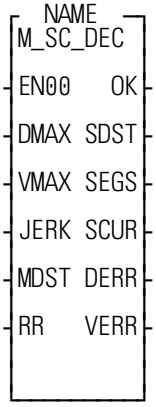
SEGS (USINT) - Total number of segments used for profile.

CVSG (BOOL) - A valid Constant Velocity structure was used flag.

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.



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.

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.

Index

A

Acceleration Scale Text A-4
Acceleration Scale Value A-4
ADDCKSUM 2-6
Alternative Mode Status A-17
Analog Input Acceleration Limit A-15
Analog Input Acceleration Limits Enable A-16
Analog Input Deceleration Limit A-15
Analog Offset A-12
Analog Output A-22
Analog Output Configuration Register A-12
Analog Scale A-13
Arm Triggering A-24
ASFB 1-1
  using 1-2
At Speed Limit A-6
Autotune Maximum Current A-17
Autotune Maximum Distance A-17
Average Current A-23
Average Motor Current A-23
Average Time Constant A-9

B

Back EMF Constant A-8
Boot Firmware Version A-3
BRAKE Active Delay A-11
BRAKE Inactive Delay A-11
Bus Voltage A-23
BYTE2HEX 2-7

C

Change Direction Flag A-16
Channel 1 Source A-24
Channel 2 Source A-24
CHKCKSUM 2-8
Collected Data A-25
COMMAND Input A-22
Command Source A-15
COMMAND Torque Offset A-12
COMMAND Torque Scale A-12
COMMAND Velocity Offset A-12
COMMAND Velocity Scale A-12
Commands, Common Product Line A-3
Commutation Type A-9
Continuous Current Limit A-7
Current Command A-23
Current Feedforward A-9
Current Negative Peak A-23
Current Positive Peak A-23

D

Digital Input Configuration Register A-11
Digital Input States A-21
Digital Output Configuration Register A-11
Digital Output States A-21
Digital Output Write Mask A-11
Drive Mode A-15
Drive Name A-4
DWOR2HEX 2-9

E

Encoder Alignment Offset A-17
Encoder Lines A-8
Encoder Output Configuration Register A-15

F

Fault Status A-20
Field Current A-23
Field Voltage Command A-23
Firmware, Main Version A-3

G

Gear Ratio A-5

H

Hall Offset A-9
HEX2BYTE 2-10
HEX2DWORD 2-11
HEX2WORD 2-12

I

Index Offset A-9
Installation 1-1
Integrator Zone A-5

L

Low Pass Filter Bandwidth A-7
Low Pass Filter Enable A-7

M

M_CHK1 2-13
M_CHK101 2-14
M_CHK109 2-15
M_CHK49 2-16
M_CHK57 2-17
M_CHK65 2-18
M_CHK73 2-19
M_CHK9 2-20
M_CLOS1 2-21
M_CLOS9 2-23
M_CLS101 2-25
M_CLS109 2-27
M_CNST_V B-27
M_CRSFIN 2-29
M_DATCAP 2-31
M_DATCPT 2-36
M_DNJOGC 2-40
M_DNPOSC 2-41
M_DNSTAT 2-43
M_DW2BOO 2-51
M_ERROR 2-52
M_FHOME 2-53
M_INCPTR 2-56
M_JOG 2-57
M_LHOME 2-58
M_LINCIR 2-61
M_PRF1MV B-17
M_PRF2MV B-6
M_PRFDWL B-23
M_PRFERR B-18
M_PROFL B-21
M_PRTCAM 2-65
M_PRTREL 2-66
M_PRTSLP 2-67
M_RATREL 2-68
M_RATSLP 2-69
M_RDTUNE 2-71
M_RGSTAT 2-72
M_RSET49 2-74
M_RSET57 2-75
M_RSET65 2-76
M_RSET73 2-77
M_SC_ACC B-26
M_SC_DEC B-28
M_SCRVLC 2-78
M_SETVAJ B-24
M_SRCMON 2-84
M_SRCPRC 2-86
M_SRCRDL 2-88
M_SRCWT 2-90
M_SRCWTL 2-92
M_STATUS 2-98
M_WTTUNE 2-100
Manual Tune Position Period A-17
Manual Tune Position Step A-17
Manual Tune Velocity Period A-17
Manual Tune Velocity Step A-17
Master Encoder Resolution A-18
Master Index Position A-18
Master Position A-22
Maximum Motor Speed A-8
Motor Continuous Current A-8
Motor Encoder Resolution A-18
Motor Forward Direction Flag A-10
Motor ID A-8
Motor Index Position A-18
Motor Peak Current A-8
Motor Position A-22
Motor Table Information A-9
Motor Table Record Size A-9
Motor Table Version A-9
Motor Velocity A-22

N

Negative Current Limit Input A-22

O

Operating Mode A-17
Over Speed Limit A-6
Override Analog Outputs A-13
Override Digital Output A-11

P

Packed Drive Status A-20
Pole Count A-9
Position Command A-22
Position Error A-22
Position Error Limit A-5
Position Error Time A-5
Position Loop Derivative Gain A-5
Position Loop Feedforward Gain A-5

Position Loop Integral Gain A-5
Position Loop Proportional Gain A-5
Position Negative Peak Error A-22
Position Positive Peak Error A-22
Position Scale Text A-4
Position Scale Value A-4
Position Window Size A-5
Position Window Time A-5
Positive Current Limit Input A-22
Powerup Status A-3
Preset Acceleration Limits Enable A-16
Preset Input Acceleration Limit A-15
Preset Input Deceleration Limit A-15
Press Transfer ASFBS Introduction B-1
Product Type A-3

R

Reset Drive A-19
Reset Faults A-19
Reset Peaks A-22
Reset Personality NVRAM A-4
revision
    history 1-1
    range 1-2
Rotor Inertia A-8
R-Phase Current A-23
Run State A-21

S

S_CLOS1 2-102
S_CLOS9 2-104
S_ERRORC 2-106
S_FHOME 2-108
S_IO_C 2-110
S_LHOME 2-112
Save Alignment Offset A-18
Serial Port Baud Rate A-14
Serial Port Frame Format A-14
Setpoint Acceleration A-19
Setpoint Control A-19
Slew Enable A-5
Slew Rate A-5
Software Drive Enable/Disable A-19
Software Drive ID A-14
Software Negative Current Limit A-7
Software Positive Current Limit A-7
Speed Window Size A-6

T

Thermal Time Constant Enable A-9
Thermostat Flag A-8
Timebase A-24
Torque Constant A-8
Torque Current A-23
Torque Preset A-15
Torque Scale Text A-4
Torque Scale Value A-4
Torque Setpoint A-19
Torque Voltage Command A-23
T-Phase Current A-23
Trigger Mode A-24
Trigger Source A-24
Trigger Status A-24
Trigger Threshold A-24
Tuning Direction Flag A-15

V

Velocity Command A-22
Velocity Error A-23
Velocity Loop Derivative Gain A-6
Velocity Loop Integral Gain A-6
Velocity Loop Proportional Gain A-6
Velocity Loop Update Period A-6
Velocity Preset A-15
Velocity Scale Text A-4
Velocity Scale Value A-4
Velocity Setpoint A-19

W

Winding Inductance A-8
Winding Resistance A-8
WORD2HEX 2-115

Z

Zero Speed Limit A-6

NOTES