ACCU ELECTRIC MOTORS INC

USA: (888) 932-9183 CANADA: (905) 829-2505

- Over 100 years cumulative experience
- 24 hour rush turnaround / technical support service
- Established in 1993

The leading independent repairer of servo motors and drives in North America.

Visit us on the web:

www.servo-repair.com

www.servorepair.ca

www.ferrocontrol.com
www.sandvikrepair.com
www.accuelectric.com

Scroll down to view your document!

For 24/7 repair services :

USA: 1 (888) 932 - 9183
Canada: 1 (905) 829 -2505

Emergency After hours: 1 (416) 624 0386

Servicing USA and Canada

Compumotor

SX Indexer/Drive Software Reference Guide

Compumotor Division
Parker Hannifin Corporation
p/n 88-011871-01E **October 1996**

Parker
Motion & Control

IMPORTANT

User Information

To ensure that the equipment described in this user guide, as well as all the equipment connected to and used with it, operates satisfactorily and safely, all applicable local and national codes that apply to installing and operating the equipment must be followed. Since codes can vary geographically and can change with time, it is the user's responsibility to identify and comply with the applicable standards and codes. WARNING: *Failure to comply with applicable codes and standards can result in damage to equipment and/or serious injury to personnel.*

Personnel who are to install and operate the equipment should study this user guide and all referenced documentation prior to installation and/or operation of the equipment.

In no event will the provider of the equipment be liable for any incidental, consequential, or special damages of any kind or nature whatsoever, including but not limited to lost profits arising from or in any way connected with the use of this user guide or the equipment.

Since Parker Compumotor constantly strives to improve all of its products, we reserve the right to change this user guide and equipment mentioned therein at anyime without notice.

Technical Assistance *Contact your local automation technology center (ATC) or distributor, or ...*

North America and Asia:
Parker Hannifin
Compumotor Division
5500 Business Park Drive
Rohnert Park, CA 94928
Telephone: (800) 358-9070
Fax: (707) 584-3793
FaxBack System: (800) 936-6939
BBS: (707) 584-4059
E-Mail: tech_help@cmotor.com

Europe (non-German speaking):
Parker Digiplan
21 Balena Close
Poole, Dorset
England BH17 7DX
441-202-690-911
441-202-600-820

Germany, Austria, Switzerland:
Hauser Elektronik GmbH
Robert-Bosch-Str. 22
D-77656 Offenburg
Germany
49-781-509-300
49-781-509-176

Parker
Motion & Control

Software Reference Guide Change Summary

The following is a summary of the primary changes to this software reference guide since the last version was released. This software reference guide, version 88-011871-01**E**, superseded version 88-011871-01**D**.

The following commands were added to this software reference guide

[ABS]—Absolute Encoder Comparison
[AND]—Boolean AND Operator
BRK—Break Command
CIT—Configure In Position Time
DCLR—Clear Display
DCNT—Enable/Disable Pause and Continue
DLED—Turn RP240 LEDs On/Off
DPC—Position Cursor
DPI—Display Position Indexer
DSTP—Enable/Disable Stop
DTXT—Display Text Data on RP240 LCD
DVA—Display Actual Velocity
DVO—Display Variable Data on RP240 LCD
[ER]—Error Flag Operator
[FEP]—Following Encoder Position Comparison
[FL]—User Flag Operator
FSG—Stop Position Maintenance Move on Limit Encountered
FSH—Abort Position Maintenance on Limit Encountered
[IN]—Input Flag Operator
INA—CW Limit Status
INB—CCW Limit Status
INC—Home Limit Status
MV—Maximum Correction Velocity
[OR]—Boolean OR Operator
OSA—Define Active State of Limit Switch/Sensor
OSI—Save Sequence Scan Mode on Stop
OSJ—Configure Z-Channel Search Mode
[POS]—Position Counter Comparison
RVV—Report RevisionVerbose
SPA—Set Position Zero
SSC—Enable End of Move In-Position Window
SSP—Clear All Position Offsets with PZ, PFZ
VARN=FUN—Enable and Read Function Keys
VARN=NUM—Enable and Read Numeric Keypad

Command Listing

MPP—Mode Position Profile
MR—Configure Motor Resolution
MSP—Maximum Synchronization Percent
MV—Maximum Correction Velocity
MW—Set Motor Waveform

N—End of Loop
NG—End Position Profile
NIF—End of IF
NWHILE—End of While

O—Output
OFF—Off
ON—On
[OR]—Boolean OR Operator
OS—Function Set-Up Report
OSA—Define Active State of Limit Switch/Sensor
OSB—Backup to Home Switch
OSC—Define Active State of Home Switch
OSD—Enable Encoder Z Channel Input
OSE—Jog Enable
OSF—Acknowledge STOP & KILL Inputs On Power-up
OSG—Final Homing Direction
OSH—Reference Edge of Home Switch
OSI—Save Sequence Scan Mode on Stop
OSJ—Configure Z-Channel Search Mode
OUT—Output Functions
OUTL—Set Active Output Level
OUTP—Output on Position

PF—Follower Position Report
PFZ—Set Follower Counter to Zero
PHZ—Zero Motor Phase
[POS]—Position Counter Comparison
PR—Absolute Position Report
PS—Pause
PU—Configure Square Wave Output
PUL—Activate Square Wave Output
PX—Report Encoder Position
PZ—Set Absolute Counter to Zero

Q0—Exit Velocity Profiling Mode
Q1—Enter Velocity Profiling Mode

R—Request SX Status
RA—Limit Switch Status Report
RB—Loop, Pause, Shutdown, Trigger Status Report
REG—Configure Registration Move
REPEAT—Repeat
RG—Go Home Status Report
RIFS—Return Indexer to Factory Settings
RM—Rate Multiplier in Velocity Streaming Mode
RS—Report Status Sequence Execution
RSE—Report Servo Errors
RSIN—Set Variables Interactively
RV—Revision Level
RVV—Report RevisionVerbose

S—Stop
SCR—Set Standby Current Reduction
SFL—Set User Flag
SL—Software Limits
SLD—Software Limits Disable
SN—Scan Delay Time
SP—Set Position Absolute
SPA—Set Position Zero
SS—Function Set-Up Report
SSA—RS-232 Echo Control
SSC—Enable End of Move In-Position Window
SSG—Clear/Save the Command Buffer on Limit
SSH—Clear/Save the Command Buffer on Stop
SSI—Enable/Disable Interactive Mode
SSJ—Enable/Disable Continuous Scan Mode
SSL—Enable Resume Execution
SSN—Set Message Mode
SSP—Clear All Position Offsets with PZ, PFZ
SSR—Enable/Disable Fault On Shutdown
ST—Shutdown
STOP—Stop
STR—Set Strobe Output Delay Time

T—Time
TD—Set Input Debounce Time
TDR—Set Registration Input Debounce Time
TEST—Test
TF—Set Following Time
TM—Move Time Report
TR—Wait for Trigger
TRD—Read Timer from Parallel I/O
TS—Trigger Input Status
TW—Thumbwheel Input Mode
TX—Transmit Variable and String

U—Pause and Wait for Continue
UNTIL—Until

V—Velocity
VAR—Variables
VARD—Read Variables via Parallel I/O
VARN=FUN—Enable and Read Function Keys
VARN=NUM—Enable and Read Numeric Keypad
VRD—Read Velocity via Parallel I/O
VS—Start/Stop Velocity

W1—Signed Binary Position Report
W2—Hexadecimal Position Report
W3—Signed Hexadecimal Position Report
WHEN—Set When Condition
WHILE—While

XBS—Sequence Memory Available Report
XC—Sequence Checksum Report
XD—Sequence Definition
XDIR—Sequence Directory
XE—Sequence Erase
XEALL—Erase All Sequences
XFK—Set Fault or Kill Sequence
XG—GOTO Sequence
XQ—Sequence Interrupted Run Mode
XR—Run a Sequence
XRD—Read Sequence via Parallel I/O
XRP—Sequence Run with Pause
XS—Sequence Execution Status
XST—Sequence Step Mode
XT—Sequence Termination
XTR—Set Trace Mode
XU—Upload Sequence
XWHEN—Set When Sequence

Y—Stop Loop

Z—Reset

Objectives

The information in this user guide will enable you to:

- ❏ Identify the four types of commands in Compumotor's X-Series Language
- ❏ Use this book as a reference for the function, range, default, and sample use of each command

Command Descriptions

The SX advances Compumotor's X-language by offering high-level programming commands. Software structures include **`IF THEN-ELSE`** statements, **`WHILE`** loops, **`REPEAT UNTIL`** loops, subroutines, and **`GOTO`** statements. Math functions with variables can be performed with this command language. You can execute complex decision-making routines in the software structures. Multiple expressions can be evaluated with the **`IF`**, **`REPEAT`**, and **`WHILE`** statements. You can define and execute up to 99 sequences. With the added programming complexity, a Trace mode has been provided to trace step-by-step through sequences. This allows you to test your application and debug any potential problems. You can customize program messages to prompt the user. In addition, a fault sequence is available that will execute a sequence when several different error conditions exist. I/O activation can be simulated to help test program flow. A special sequence can be run that will execute only when certain conditions occur such as input states, variable comparisons, etc.

① **A**	**Acceleration**		
② Type	Motion	⑦ Valid	Software Version A
③ Syntax	<a>A<n>	⑧ Units	n = rps^2
④ Range	0.01 to 9999.99	⑨ Default	100
⑤ Attributes	Buffered, Savable in Sequence	⑩ See Also	D, V, G, AD, LAD, GHA, GHAD
⑥ Response	None		

① ***Mnemonic Code***

This field contains the command's mnemonic code and full command name.

② ***Type***

This field contains the command's type. The four command types are listed below.

Status—Status commands respond (report back) with information.

Set-Up—Set-Up commands define set-up conditions for the application. Set-Up commands include the following types of commands:

- ❏ Homing (Go home acceleration and velocity, etc.)
- ❏ Input/Output (Limits, scan time, in-position time, etc.)
- ❏ Tuning (Position tracking)
- ❏ General (Set switches, return to factory settings, etc.)

Programming—Programming commands affect programming and program flow. For example, trigger, output, and all sequence commands, quote, time delays, pause and continue, enable, loop and end loop, line feed, carriage return, backspace, and Evaluation commands..

Motion—Motion commands affect motor motion (i.e., acceleration, velocity, distance, go home, stop, direction, mode, etc.)

③ ***Syntax***

The proper syntax for the command is shown here. The specific parameters associated with the command are also shown. Definitions of the parameters are described below.

a This indicates that a device address must accompany the command. Only the device specified by this parameter receives and executes the command.

n This represents an integer. You may use an integer to specify a variety of values (acceleration, velocity, etc.)

<> If any parameters appear within brackets, <a>, it is optional.

④ ***Range***

This is the range of valid values that you can specify for n (or any other parameter specified).

⑤ ***Attributes***

This command indicates if the command is ***immediate*** or ***buffered***. The system executes immediate commands as soon as it receives them. Buffered commands are executed in the order that they are received with other buffered commands. You can store buffered commands in a sequence. Immediate commands cannot be saved in a sequence.

This field also explains how you can save the command.

❏ Savable in Sequence
❏ Never Saved

A command that is never saved is never placed in the system's permanent memory.

⑥ ***Response***

The systems Response to a command is shown here. This information will only be provided if the system provides a Response to the command.

⑦ ***Valid Revision Level***

This field contains the revision history of the command. It includes the revision of software when the command was added or modified. If the revision level of the software you are using is equal to or greater than the revision level listed in this field, you are using the proper version of the software to use that command. If not, contact your local distributor about purchasing a software upgrade.

⑧ ***Units***

This field describes what unit of measurement the parameter in the command syntax represents.

⑨ ***Default***

The default setting for the command is shown here. A command will perform its function with the default setting if you do not provide a value.

⑩ ***See Also***

Commands that are related or similar to the command described are listed here.

Note: All commands need to be followed by a carriage return or a space character. These are both legal delimiters. The command will not be recognized without a delimiter.

; Comment Field

Type	Programming	Valid	Software Version A
Syntax	<a>**;**	Units	None
Range	None	Default	None
Attributes	Buffered	See Also	None

The **;** designates a comment within a sequence. The characters after the **;** are not executed and allow adding descriptions of programs to sequences.

Command	Description
> XD1	Defines sequence #1
A20	Sets acceleration to 20 rps^2
AD30	Sets deceleration to 30 rps^2
V5	Sets velocity to 5 rps
D50000	Sets distance to 50,000
;MOVE_SLIDE_IN	Comment not executed
G	Executes the move (Go)
XT	Ends defining sequence #1
>	

The underscores shown in the example are required in place of spaces, since the space character is a legal command delimiter.

Step Sequence

Type	Programming	Valid	Software Version A
Syntax	<a>#<n>	Units	n = number of commands to be executed
Range	1 to 200	Default	1
Attributes	Immediate, Never Saved	See Also	XST, XTR

This command controls the execution of a sequence when the Single-Step mode (**XST**) is enabled. Each time you enter the #n command followed by a delimiter (carriage return or space), n commands in the sequence buffer will be executed. Only a **#** followed by the delimiter causes one command to be executed. You can run in Single-Step mode only if you have an RS-232C interface connected to a host. If you issue a Kill (**K**) command, while you are in Single-Step mode, the sequence execution will be aborted, but Single-Step mode is retained. When you cycle power, the indexer will no longer be in Single-Step mode.

Command	Description
> XE1	Erases sequence #1
> XD1	Defines sequence #1
A5	Sets acceleration to 5 rps^2
V2	Sets velocity to 2 rps
D10000	Sets distance to 10,000 steps
G	Executes the move (Go)
XT	Ends sequence definition
> 1XTR1	Enables Trace Mode
> XST1	Enables Single-Step mode
> XR1	Executes sequence #1
#	Execute the 1st command
*SEQUENCE_001_COMMAND_A5	Displays the 1st command executed
#	Execute the 2nd command
*SEQUENCE_001_COMMAND_V2	Displays the 2nd command executed
#	Execute the 3rd command
*SEQUENCE_001_COMMAND_D10000	Displays the 3rd command executed
#	Execute the 4th command
*SEQUENCE_001_COMMAND_G	Displays the 4th command executed motor should have moved 10,000 steps
#	Execute the 5th command
*SEQUENCE_001_COMMAND_XT	Displays the last command executed
>	

A Acceleration

Type	Motion	Valid	Software Version A
Syntax	<a>**A**<n>	Units	n = rps^2
Range	0.01 to 9999.99	Default	100
Attributes	Buffered, Savable in Sequence	See Also	D, V, G, AD, LAD, GHA, GHAD

The Acceleration (**A**) command specifies the acceleration rate to be used upon executing the next Go (**G**) command. The acceleration remains set until you change it and is stored in nonvolatile memory. You do not need to reissue this command for subsequent Go (**G**) commands.

Upon powering up, if the Deceleration (**AD**) command has not been modified, any modifications to Acceleration (**A**) will also affect the deceleration. This prevents having to access both parameters when the desired acceleration and deceleration rates are the same. Once the acceleration has been modified, changing the deceleration simply requires the **AD** command.

Command	Description
> MN	Sets to Normal mode(preset moves)
> A5	Sets acceleration to 5 rps^2
> AD10	Sets deceleration to 10 rps^2
> V10	Sets velocity to 10 rps
> D10000	Sets distance to 10,000 steps
> G	Executes the move (Go)

[ABS] Absolute Encoder Comparison

Type	Programming/Evaluation	Valid	Software Version F
Syntax	see below	Units	None
Range	None	Default	None
Attributes	Buffered, Savable in Sequence	See Also	FEP, ER, SPA, SSP

This evaluation operator is the present value of the absolute encoder counter. A **PZ** command will generate an internal offset so that the [**ABS**] value will differ from the **PX** Report. The **SSPI** command can be used to clear this offset.

Command	Description
> IF(ABS>16348)	If the present value of the absolute encoder counter is greater than 16348 counts, print ABSOLUTE
> 1"ABSOLUTE	
> NIF	

AD Deceleration

Type	Motion	Valid	Software Version A
Syntax	<a>**AD**<n>	Units	n = rps^2
Range	0.01 to 9999.99	Default	10 rps^2
Attributes	Buffered, Savable in Sequence	See Also	D**,** V**,** G**,** A, LAD, GHA, GHAD
Response	***AD**n		

The Deceleration (**AD**) command specifies the deceleration rate to be used upon executing the next Go (**G**) command. The deceleration rate remains set until it is changed and is stored in nonvolatile memory. You do not need to reissue this command for subsequent **G** commands.

Upon powering up, if the Deceleration (**AD**) command has not been modified, any modifications to Acceleration (**A**) will also affect the deceleration. This prevents having to access both parameters when the desired acceleration and deceleration rates are the same. Once the deceleration rate is modified with the **AD** command, the **A** command will no longer have any effect on deceleration.

Command	Description
> MN	Sets to Normal mode
> A5	Sets acceleration to 5 rps^2
> AD10	Sets deceleration to 10 rps^2
> V3	Sets velocity to 3 rps
> D10000	Sets distance to 10,000 steps
> G	Executes the move

[AND] Boolean AND Operator

Type	Programming/Evaluation	Valid	Software Version A
Syntax	see below	Units	None
Range	None	Default	None
Attributes	Immediate	See Also	OR, IF, REPEAT, WHILE

This command is used inside evaluation statements to **AND** conditions together. For an evaluation statement to evaluate true, both conditions must be true.

Command	Description
>VAR1=2	Set variable 1 = 2
>VAR2=5	Set variable 2 = 5
>IF(VAR1=2_AND_VAR2=5)	This statement would evaluate true
>NIF	
>IF(VAR1=2_AND_VAR2=10)	This statement would evaluate false
>NIF	

B Report Buffer Status

Type	Status	Valid	Software Version A
Syntax	a**B**	Units	None
Range	None	Default	None
Attributes	Immediate	See Also	BS, R
Response	***B or ***R		

The Buffer Status (**B**) command reports the status of the command buffer. If the command buffer is empty or less than 90% full, the indexer will respond with a ***R[cr]**. A ***B[cr]** Response will be issued if less than 10% of the command buffer is free. The command buffer is 2,000 bytes long.

This command is commonly used when a long series of commands will be loaded remotely. If the buffer size is exceeded, the extra commands will never be received by the indexer.

***R** = More than 10% of the buffer is free

***B** = Less than 10% of the buffer is free

Command	Response
> 1B	*R [cr] (more than 10% of the buffer is free)

BCPE Buffered Configure Position Error

Type	Set-Up	Valid	Software Version A
Syntax	<a>BCPE<n>	Units	n = motor steps
Range	0 - 336544320	Default	4000
Attributes	Buffered, Savable in Sequence	See Also	CPE, FSD, ER, FS
Response	*MAXIMUM_POSITION_ERROR=nnnnnnnnnn_STEPS		

This command is identical to **CPE** except that it is buffered. It is useful when the position error must be changed in a sequence.

If a value is supplied, the specified value will become the new value for the maximum allowable position (or following) error before a stall condition is flagged. The buffered **BCPE** command differs from the **CPE** command only in that it can be executed sequentially.

Command	Description
> A5	Sets Acceleration to 5 units/sec^2
> V10	Sets Velocity to 10 units/sec
> BCPE4000	Sets maximum following error to 4,000 steps
> D32000	Sets Distance to 32,000
> G	Executes the move (Go)
> BCPE6000	Sets maximum following error to 6,000 steps
> G	Executes the move (Go)

The maximum following error is set to 4,000 steps before the first 32,000-step move and to 6,000 steps before the second 32,000-step move.

BCPG Buffered Configure Proportional Gain

TYPE	Set-Up	Valid	Software Version A
Syntax	<BCPG<n>	Units	n = %
Range	0 - 99	Default	10
Attributes	Buffered, Savable in Sequence	See Also	BCPM, CPG, CPM, FSC, FSB, FS, ER,
Response	*PROPORTIONAL_GAIN=n_PERCENT		MV, DPE

This command defines the proportional gain to be used for position maintenance (**FSC**). When you use the Buffered Configure Proportional Gain (**BCPG**) command, you are actually setting a percentage of the proportional gain maximum (**CPM**). This command is useful in tuning out the proportional error after a move. This command is identical to **CPG** except that it is buffered. It is useful when the positional gain needs to be changed in a sequence.

Command	Description
> BCPG20	Sets proportional gain to 20 percent

BCPM Buffered Configure Proportional Maximum

TYPE	Set-Up	Valid	Software Version A
Syntax	<a>BCPM<n>	Units	n = maximum proportional gain
Range	1 - 32,767	Default	50
Attributes	Buffered, Savable in Sequence	See Also	CPE, CPM, BCPG**,** FSC, FSB, FS, ER, MV, DPE
Response	*PROPORTIONAL_GAIN_MAXIMUM=n		

This command defines the maximum buffered proportional gain for position maintenance (**FSC**). If a valid number is entered it will become the new maximum proportional gain. When you set the proportional gain (**CPG**), you will actually set a percentage of the Maximum Proportional gain.

This command is identical to the Configure Proportional Maximum (**CPM**) command except it is buffered. It is useful when the proportional maximum must be changed within a sequence.

The position maintenance correction velocity is determined by the equation: Velocity = **CPG*CPM*DPE** with the maximum correction velocity set with the **MV** command.

Command	Description
> BCPM2500	Set proportional maximum to 2,500

BL Backlash

Type	Motion	Valid	Software Version A
Syntax	<a>BL<n>,<m>	Units	n, m = pulses
Range	0 - 65,535	Default	0
Attributes	Buffered, Savable in Sequence	See Also	FSB
Response	*CCW_BACKLASH_n *CW_BACKLASH_m		

The **BL** command sets the number of pulses added to any moves when changing direction to correct for any mechanical backlash. The first field sets the number of pulses added when changing from the CCW to CW. The second field sets the number of pulses added when changing from CW to CCW. This command is only active when in Motor Step (**FSB0**) Mode.

Command	Description
> BL50,25	50 pulses are added to motion when changing from CCW to CW, while 25 pulses are added when changing from CW to CCW.

BRK Break Command

Type	Motion	Valid	Software Version A
Syntax	aBRK	Units	None
Range	None	Default	None
Attributes	Immediate, Never Saved	See Also	XFK

BRK command is used just like the **K** command to immediately kill any motion without deceleration and dump the command buffer. It also resets the **XFK** sequence to zero which prevents the **XFK** sequence from automatically running. This allows program editing, downloading, uploading, troubleshooting, and buffered command execution. The **XFK** sequence needs to be re-enabled after execution of the **BRK** command.

BS Buffer Status Report

Type	Status	Valid	Software Version A
Syntax	aBS	Units	None
Range	None	Default	None
Attributes	Immediate, Never Saved	See Also	B, R, RA, RB, RSE, RG, RS
Response	*N		

The Buffer Status Report (**BS**) command reports the number of characters (including delimiters) that can be loaded into the command buffer. When entering long string commands, check the buffer status to be sure that buffer space is available.

If you are using a computer interface, you can load the program lines to the indexer faster than the indexer can execute them. Therefore, the commands would get stored in the buffer. This command tells you how many more characters can be downloaded to the indexer before the buffer overflows.

Command	Response
> 1BS	*100 (Space is available for 100 characters in the command buffer)

C Continue

Type	Programming	Valid	Software Version A
Syntax	<a>C	Units	None
Range	None	Default	None
Attributes	Immediate, Never Saved	See Also	PS, SSL, U, S, STOP

The Continue (**C**) command ends a pause state. It enables your indexer to continue executing buffered commands. After you initiate a pause with the Pause (**PS**) command or the Pause and Wait for Continue (**U**) command, you can clear it with a Continue command. This command is useful when you want to transmit a string of commands before you actually need to execute them.

If you are using the resume function (**SSL1**), the Continue (**C**) command can also be used to resume execution of a move or a sequence after the program is halted by the Stop (**S**) command. Upon **C**, the program or the move will be started from the point where it stopped.

*Helpful Hint: To pause and continue a preset move without entering a **C** command over the RS232C interface, try the following: define a Stop input, **INnD**, and a Pause/Continue input, **INnF**, and connect them both to the same switch. Enter **SSH1** to Save*

the Command Buffer on Stop, and ***SSL1*** *to Resume Execution. In this manner, the stop input will stop the motion when the stop switch transitions to active, and the pause/continue input will resume the movement when the pause/continue input transitions to deactive. As an alternative, these two inputs may be wired together into one switch, and then motion will pause when the switch is activated, and resume when the switch is deactivated. This will not work with a Mode Continuous move.*

Command	Description
> PS	Pauses execution until the indexer receives a C command
> MN	Sets to Normal mode
> A5	Sets acceleration to 5 rps^2
> AD10	Sets deceleration to 10 rps^2
> V5	Sets velocity to 5 rps
> D25000	Sets distance to 25,000 steps
> G	Executes the move
> C	Starts executing commands in buffer, beginning with MN

CEW Configure Error Window

Type	Set-Up	Valid	Software Revision F2
Syntax	<a>CEW<n>	Units	n=steps (Motor=FSB0, Encoder=FSB1)
Range	$\pm 2^{31}$-1	Default	0
Attributes	Buffered, Savable in Sequence	See Also	CIT, SSC
Response	*END_OF_MOVE_IN_POSITION_WINDOW=n_STEPS		

CEW defines the end of move in-position error window. The value of the position error (difference between commanded position and actual position) must be less than or equal to the **CEW** value for the **CIT** interval in order for buffered command execution to continue. If at any time after completion of the move in progress, the position error exceeds the **CEW** value, a timer is started that pauses buffered command execution until the position error is within the **CEW** window for **CIT** time.

NOTE: If this command is used in motor mode, the **CEW** is defined in motor steps. The SX will use the defined motor to encoder ratio (**MR** and **ER** ratio) to determine how many encoder counts are needed. For an MR11 (25000 steps/rev) and ER4000, the motor to encoder ratio is 6.25. If the **CEW** is set below 6, the SX will wait until it is within zero encoder counts and may never execute the next buffered command. To use the End of Move in Position Window, the **SSC** command must be set to 1.

CIT Configure In Position Time

Type	Set-Up	Valid	Software Revision F2
Syntax	<a>CIT<n>	Units	n=minimum settling time in ms
Range	0 - 32000	Default	0
Attributes	Buffered, Savable in Sequence	See Also	CEW, SSC
Response	*END_OF_MOVE_SETTLING_TIME=n_MILLISECONDS		

CIT sets the minimum time (in milliseconds) required for the end of move position to be within the **CEW** window, before allowing buffered command execution to continue.

Note: To use the end of move in position window, the ***SSC*** *command must be set to 1.*

CPE Configure Position Error

Type	Set-Up	Valid	Software Version A
Syntax	<a>CPE<n>	Units	n = motor steps
Range	0 to 336,544,320	Default	4,000
Attributes	Immediate, Never Saved	See Also	BCPE, FSD, ER, FS
Response	*MAXIMUM_POSITION_ERROR=N_STEPS		

If a value is supplied, the specified value will become the new value for the maximum position error. If the position error exceeds the **CPE** value, a Position Error stall condition will exist if **FSD** is enabled.

Command	Description
> CPE100	Sets the maximum position error to 100 motor steps before a stall condition exists

CPG Configure Proportional Gain

Type	Set-Up	Valid	Software Version A
Syntax	<a>CPG<n>	Units	n = % of gain
Range	0 to 99	Default	10
Attributes	Immediate, Automatically Saved	See Also	BCPG, BCPM, CPM, FSC, FSB, FS, ER, MV, DPE
Response	*PROPORTIONAL_GAIN=n_PERCENT		

This command defines the proportional gain to be used for position maintenance. When you tune the drive using the Configure Proportional Gain (**CPG**) command, you are actually setting a percentage of the Proportional Gain Maximum (**CPM**).

Command	Description
> CPG4	Configure the proportional gain to 4% of the maximum value

CPM Configure Proportional Maximum

Type	Set-Up	Valid	Software Version A
Syntax	<a>CPM<n>	Units	n = maximum
Range	1 - 32767	Default	50
Attributes	Immediate, Never Saved	See Also	BCPG, BCPM, CPG, FSC, FSB, FS, ER
Response	*PROPORTIONAL_GAIN=n_MAXIMUM		MV, DPE

This command defines the maximum proportional gain you can use for Proportional Maximum position maintenance. If a valid number is entered, it will become the new maximum proportional gain. When you set the proportional gain (**CPG**), you are actually setting a percentage of the maximum proportional gain.

Command	Description
> CPM250	Sets proportional gain maximum to 250

CR Carriage Return

Type	Programming	Valid	Software Version A
Syntax	aCR	Units	None
Range	None	Default	None
Attributes	Buffered, Savable in Sequence	See Also	LF
Response	(carriage return)		

The Carriage Return (**CR**) command can determine when the indexer has reached a particular point in the execution buffer. When the indexer reaches this command in the buffer, it responds by issuing a carriage return (ASCII 13) over its interface back to the host computer. If you place the **CR** command after a Go (**G**) command, it will indicate when a move is complete. If you place the **CR** command after a Trigger (**TR**) command, it will indicate when the trigger condition is met.

Command	Description
> MPA	Sets to Absolute Position mode
> A50	Sets acceleration to 50 rps^2
> AD50	Sets deceleration to 50 rps^2
> V5	Sets velocity to 5 rps
> D5000	Sets distance to 5,000 steps
> G	Executes the move (Go)
> 1CR	Sends a carriage return after the move is complete

D Distance

Type	Motion	Valid	Software Version A
Syntax	<a>D<s><n>	Units	s = sign, n = steps
Range	±2,147,483,647	Default	25,000
Attributes	Buffered, Savable in Sequence	See Also	A, G, MN, MPA, MPI, MPP, V

The Distance (**D**) command defines motion in either the number of steps it will move or the absolute position it will seek after a Go (**G**) command is entered. In Incremental mode (**MPI**), the value set with the Distance (**D**) command will be the distance motion will travel on all subsequent **G** commands. In Absolute mode (**MPA**), the distance moved by the motor will be the difference between the current motor position and the position (referenced to the zero position) set with the **D** command. A valid distance must be defined with the **D** command before a preset move can be executed. The **D** command has no effect on continuous moves (**MC**). The **D** value is stored in nonvolatile memory. If in **MPP** mode, another **D** command will supersede previous **D** commands.

Command	Description
> MN	Sets to Normal (preset) mode
> A5	Sets acceleration to 5 rps^2
> V10	Sets velocity to 10 rps
> D50000	Sets distance to 50,000 steps
> G	Executes the move (Go)

DCLR Clear Display

Type	Programming	Valid	Software Version C2
Syntax	<a>DCLRn	Units	line number
Range	0, 1, 2	Default	None
Attributes	Buffered	See Also	DPC, DTXT, DVO

The Clear Display (**DCLR**) command clears a specified line of the RP240 display, and repositions the cursor to the beginning of the line.

n = 0 Clear all lines of the display
n = 1 Clear line 1 of the display
n = 2 Clear line 2 of the display

Command	Description
> DCLR1	Clear line one (1) of the display

DCNT Enable/Disable Pause and Continue

Type	Programming	Valid	Software Version C2
Syntax	<a>DCNTb	Units	None
Range	0, 1	Default	0
Attributes	Buffered	See Also	SSH, SSL

This command enables or disables the PAUSE and CONTINUE keys on the RP240.

DCNT0: Disable PAUSE and CONTINUE keys

DCNT1: Enable PAUSE and CONTINUE keys

When the **PAUSE** and **CONTINUE** keys are enabled (**DCNT1**), pressing the **PAUSE** key will cause the RP240 to send a Stop (**S**) command to the SX. Pressing the **CONTINUE** key will cause a Continue (**C**) command to be sent. To have the SX pause motion, the Save Command Buffer on Stop (**SSH1**) and the Resume Execution Enable (**SSL1**) must be enabled.

Once you have activated the **PAUSE** and **CONTINUE** keys, they will remain active at all times, except when numeric or function key information has been requested (**VARn=NUM** or **VARn=FUN**).

Command	Description
>XE100	Erase sequence #100
>XD100	Begin defition of sequence #100
SSH1	Enable save command buffer on stop
SSL1	Enable resume execution
DCNT1	Enable PAUSE and CONTINUE keys
DSTP1	Enable STOP key
XT	End definition of sequence #100
>	

DFS Display Flags for Drive Parameters

Type	Status	Valid	Software Version A
Syntax	aDFS	Units	None
Range	None	Default	None
Attributes	Immediate, Never Saved	See Also	R, RSE, DXF
Response	*bbbb_bbbb_bbbb_bbbb_bbbb_bbbb_bbbb_bbbb*		

The Display Flags of Drive Parameters (**DFS**) command returns all the drives status flags as 32-bit responses. Each bit refers to a status flag as shown on the chart below. The associated number for each flag corresponds to each binary (1 or 0) response is in the following order: *31, 30, 29, 28, . . . 3, 2, 1, 0*

Bit		
31 27 23 19 12 7 30 26 22 18 11 6 29 25 21 17 10 1 28 24 20 13 9	These bits are all reserved for future use. They all return zero's (0)	
16	Under voltage	no = 0, yes = 1
15	Motor Fault	no = 0, yes = 1
14	User Fault	no = 0, yes = 1
8	Failed non-volatile memory checksum	no = 0, yes = 1
5	Max position error exceeded	no = 0, yes = 1
4	Shutdown Active	no = 0, yes = 1
3	Drive Off	no = 0, yes = 1
2	Overtemperature Fault	no = 0, yes = 1
0	Commanded shutdown	no = 0, yes = 1

DFX Display Flags for Indexer Status

Type	Status	Valid	Software Version A
Syntax	aDFX	Units	None
Range	None	Default	None
Attributes	Immediate, Never Saved	See Also	MN, MPA, MPI, R, RA, RB, DFS, DR
Response	aDFX IS *bbbb_bbbb_bbbb_bbbb_bbbb_bbbb_bbbb_bbbb*		

The Display Flags of Indexer (**DFX**) command returns the indexer status flag as a 32-bit Response. Each bit refers to a status flag as shown on the chart below. The number for each flag corresponds to each binary (1 or 0) Response is in the following order: *31,30,29,28, . . . 3, 2, 1, 0*

Bit			
31	28	25	These bits are all reserved for
30	27	24	Future use. They all return zero's (0).
29	26	23	

Bit	Function
24	Mode Profile: no = 0, yes = 1
21	Hit a software CCW limit: no = 0; yes = 1
20	Hit a software CW limit: no = 0; yes= 1
19	Home limit: not found = 0; found = 1
18	Jogging: no = 0; yes = 1
17	Queued for RM mode: no = 0; yes = 1
16	Run sequence on power up: no=0; yes=1
15	U command: 0=not waiting; 1=waiting for continue
14	Waiting for a trigger: no = 0; yes = 1
13	Searching for encoder Z channel: no = 0; 1 = yes
12	Back up to home limit: 0 = no; 1 = yes
11	High-speed portion of home move: no = 0; in process = 1
10	Execute a sequence: no = 0; yes = 1
9	Wait on a timer: no = 0; yes = 1
8	Hit a CCW limit: no = 0; yes = 1
7	Hit a CW limit: no = 0; yes = 1
6	PS: 0 = not waiting; 1 = waiting for continue
5	Absolute move direction (Set only while moving): 0 = CW 1 = CCW
4	Incremental/absolute: 0 = MPI; 1 = MPA
3	Mode: preset = 0; Continuous = 1
2	Commanded move direction: 0 = CW; 1 = CCW
1	Preset move in progress: 0 = not moving; 1 = moving
0	Continuous move: 0 = not moving; 1 = moving

DIN Disable Inputs

Type	Set-Up	Valid	Software Version A
Syntax	<a>DIN<b>	Units	b = input
RANGE	0, 1, E	Default	E
Attributes	Immediate, Never Saved	See Also	DOUT, INR
Response	aDIN IS *nnnn_nnnn_nnnn		

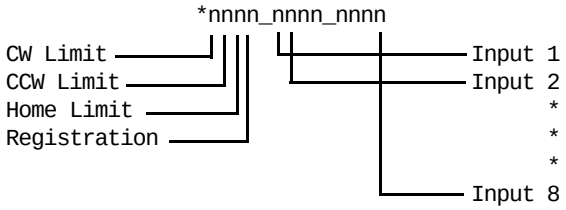
The **DIN** command allows you to disable inputs from their logic state and set them as active or inactive. This command is used for troubleshooting and start-up testing. It allows you to simulate input operations without wiring.

CAUTION

Re-enable the inputs (especially the limits) after using this command.

You can provide one of the following values for b.
1 = Active, 0 = Inactive, E = Enable

*Helpful Hint: **1DIN** Response*



Command	Description
> DIN1100	Disable the CW and CCW limits on—Home and Registration off
> DINEEEE	Re-enable the input functions

DLED Turn RP240 LEDs On/Off

Type	Programming	Version	C2
Syntax	<a>DLEDn	Units	None
Range	0, 1, X	Default	0
Attributes	Buffered	See Also	O

The **DLED** command controls the state of the 8 LEDs on the RP240. A one will turn an LED on, a zero will turn an LED off, and an X will leave the LED unchanged from its last state.
The command example below reads from left to right and corresponds to the LEDs from top to bottom.

Command	Description
> DLED1100XX11	Turn LEDs 1,2,7, and 8 on, LEDs 3 and 4 off, and leave LEDs 5 and 6 unchanged

DOUT Disable Outputs

Type	Set-Up	Valid	Software Version A
Syntax	<a>DOUT<b>	Units	b = output
Range	0, 1, E	Default	E
Attributes	Immediate, Never Saved	See Also	DIN
Response	*bbbbb		

The **DOUT** command allows you to disable any of the outputs from their configured function and set them on or off. (Except the fault output—it will report the status of a fault output when a **1DOUT** is executed, but won't allow you to disable it.) This command is used for troubleshooting and initial start-up testing. It allows you to simulate output operations and bypass their configured functions.

CAUTION

You must re-enable the outputs after using this command.

You can provide one of the following values for n.

1 = Active, 0 = Inactive, E = Enable

Command	Description
> DOUT11	Disable outputs 1 and 2 on
> DOUTEE	Re-enable outputs 1 and 2

DP Distance Point

Type	Motion	Valid	Software Version A
Syntax	<a>DPn	Units	n = steps
Range	±2,147,483,647	Default	None
Attributes	Buffered, Savable in Sequence	See Also	MPP, NG, FP

When in Mode Position Profile (**MPP**) commands subsequent to a **G** command are processed. Upon encountering a **DP** command, command processing stops until the distance matches the value set by the **DP** command. The **DP** value is absolute or incremental depending if **MPA** or **MPI** is active.

DP value Encoder Steps = **FSB1** (Encoder mode)
DP value Motor Steps = **FSB0** (Motor mode)

Command	Description
> MPI	Set Incremental mode
> A5	Set acceleration to 5 rps^2
> AD10	Set deceleration to 10 rps^2
> V2	Set velocity to 2 rps
> D200000	Set distance to 200,000 steps
> MPP	Set Position Profile mode
> G	Execute the move (Go)
> DP50000	Wait until motor has traveled 50,000 steps
> V4	Set velocity to 4 rps—motor accelerates to 4 rps
> DP20000	Wait until motor has traveled another 20,000 steps
> O11	Turn on outputs 1 and 2
> DP50000	Wait until motor has traveled another 50,000 steps
> V2	Set velocity to 2 rps
> NG	End Position Profiling

DPA Display Actual Position

Type	Status	Valid	Software Version A
Syntax	<a>DPA<n>	Units	n = display mode
Range	1 or null	Default	None
Attributes	Immediate, Never Saved	See Also	MN, MPA, MPI, FS, DPI, PR, FX, SPA, SSP
Response	*ACTUAL_POSITION=n_STEPS		

The Display Position Actual (**DPA**) command displays the actual position in feedback steps. The function is cancelled by sending any single ASCII character to the SX. The position reported does not include any offsets introduced by the **PZ** command.

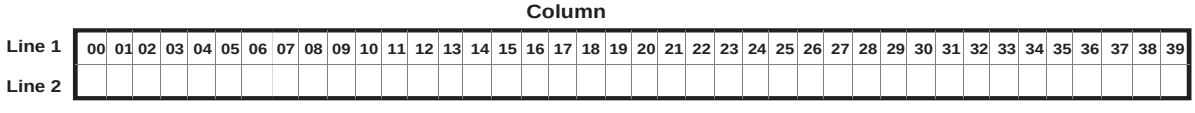
aDPA1: Reports the actual position once without descriptive text.
aDPA: Reports the actual position until a character is sent to the SX.

DPC Position Cursor

Type	Programming	Version	C2
Syntax	<a>DPCnxx	Units	n = line number, x = column number
Range	n = 1, 2, x = 00_39	Default	None
Attributes	Buffered	See Also	DCLR**,** DTXT**,** DVO

The Position Cursor (**DPC**) command places the cursor at line **n**, column **xx**. The lines are numbered from top to bottom, 1 to 2. The columns are numbered from left to right, 00 to 39.

You must use 00, 01, 02,...,09 instead of 0, 1, 2,..., 9 for the column number (i.e., DPC208, not DPC28).



Once the cursor has been placed, all succeeding text (**DTXT**) or variable data (**DVO**) will be displayed beginning at the current

cursor location. All numeric data entered using the **VARn=NUM** command will also be displayed at the current cursor location.

Command	Description
> DPC205	Position the cursor on line 2, column 5
> DTXTCOMPUMOTOR_DEMO_PROGRAM	Place message **COMPUMOTOR DEMO PROGRAM** at current cursor position

DPE Display Position Error

Type	Status	Valid	Software Version A
Syntax	<a>DPE<n>	Units	n = display mode
Range	1 or null	Default	None
Attributes	Immediate, Never Saved	See Also	PR, CPE
Response	*POSITION_ERROR=n_STEPS		

The Display Position Error (**DPE**) command reports the difference between the position setpoint and the current position.

aDPE: Continually reports the position error value until any key is pressed.
aDPE1: Report Position error value 1 time only without descriptive text.

Command	Description
> aDPE	Continually reports the position error value until any key is pressed
> aDPE1	Report position error value 1 time only without descriptive text

DPI Display Position Indexer

Type	Status	Valid	Software Version G
Syntax	aDPI<n>	Units	none = continuous, 1 = single report
Range	None, 1	Default	None
Attributes	Immediate, Never Saved	See Also	DPA, PR, PX, SPA, SSP
Response	*+(-)nnnnnnnn		

The Display Position Indexer (**DPI**) command displays the position of the indexer. It is similar to the **DPA** command, but reports the indexer position instead of the feedback position.

DR Display Parameters

Type	Status	Valid	Software Version A
Syntax	aDR	Units	None
Range	None	Default	None
Attributes	Immediate, Never Saved	See Also	None
Response	See Example		

This command transmits the current SX configuration and parameter settings. This allows a single command to display most of the important information about the SX configuration. See the example for the transmit format.

Command	Description
>1 DR	The following is transmitted:

```
*SX_SETTINGS*
*CONFIGURED_AS_A_INDEXER
*MOTION_MODE:_INCREMENTAL_PRESET
*0__CW_AND_CCW_LIMITS_ENABLED
*3__NO_SOFTWARE_TRAVEL_LIMITS_ENABLED
*MOTION_PARAMETERS:*A10.0 *AD10.0 *V1.0 D+25000
*SET_UP_PARAMETERS:*MR11      25000_STEPS_PER_REV
   *ER4000
      *SOFTWARE_SWITCHES
      *ABCD_EFGH_IGKL_MNOP_QRST
*SS*0000_0000_0000_0100
*FS*0000_0000_0000_0000
*OS*0000_0000
```

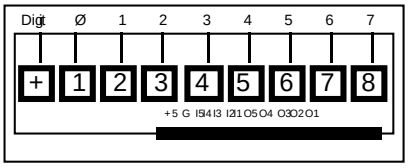
DRD Read Distance via Parallel Input/Output

Type	Programming	Valid	Software Version B
Syntax	<a>DRD<c><d><e>	Units	c, d = digit selector, e = scaling factor
Default	None		Range c, d = 0 - 7, e = 0 - 9
Attributes	Buffered, Savable in Sequence	See Also	IN, OUT, STR, FRD, LRD, TRD, VRD, XRD

This command instructs the SX to read distance values from Compumotor's *TM8 Module User Guide* (refer to Chapter 3, Installation for information on the TM8 Module).

Inputs 1-4 must be configured as data inputs. The inputs must be configured as active low (**INL0**) and outputs 1-3 must be configured as strobe outputs to use the TM8 Module.

Helpful Hint: <c> & <d> values represent the TM8 Module digits:



The command syntax allows for digit range selection through the optional <c><d> fields. The <c> field is used to signify the start of the digit range to be read from the TM8 Module. The raw data format is XXXXXXXXXX, the maximum value is 2147483600. Any larger number will result in the SX giving a ? Response. The <d> field represents the end of the digit range to be read. The values of these fields can range from 0 to 7 with the <c> field always being less than or equal to the <d> field value.

The <e> field is used to scale the distance value by 10^{e}. If the <c> and <d> fields are used, the <e> field must be used. If the <c>, <d>, and <e> fields are not used, the **DRD** command will read all the digits of the TM8 Module. If you are using the TM8 Module, the Output Strobe Delay Time must be set at a value of 10 or greater.

You may use a PLC with the **DRD** command. The **DRD** command uses a multiplexed I/O scheme. The outputs strobe through a BCD sequence at the Set Strobe Output Delay Time (**STR**) command rate and reads one BCD digit at a time. The outputs and inputs must be configured as in the TM8 Module read case. A **DRD** command will issue the following strobe sequence:

Voltage Level			
01	02	03	Data In (Active Low)
low	low	low	MSD (Digit 1)
high	low	low	Digit 2
low	high	low	Digit 3
high	high	low	Digit 4
low	low	high	Digit 5
high	low	high	Digit 6
low	high	high	Digit 7
high	high	high	LSD (Digit 8)

The PLC, while reading the output strobe, must place the appropriate data at the SX inputs 1 - 4. Input 1 is the digit's LSB and input 4 the MSB. Configuring an input as a data valid line would allow the PLC to control the rate at which the outputs will strobe through the above sequence.

Turn the TM8 Module's thumbwheels to display: +1 2 3 4 5 6 7 8

Type the following commands:

Command	Description
> DRD	Request all thumbwheel values
> 1D	
> *D+12345678	

DSTP Enable/Disable Stop

Type	Programming	VERSION	C2
Syntax	<a>DSTPb	Units	None
Range	0, 1	Default	0
Attributes	Buffered	See Also	SSH, SSL

The **DSTP** command enables or disables the STOP key on the RP240.

DSTP0 Disable **STOP** key
DSTP1 Enable **STOP** key

When the **STOP** key is enabled (**DSTP1**), pressing the **STOP** key will cause the RP240 to send a Kill (**K**) command to the Model 500, SX, or ZX.

CAUTION

Using the **STOP** key (**K** command) causes motion in progress to be stopped with the maximum deceleration rate. Once you have activated the **STOP** key, it will be active at all times.

Command	Description
>XE100	Erase sequence 100
>XD100	Begin definition of sequence 100
SSH1	Enable save command buffer on stop
SSL1	Enable resume execution

DCNT1	Enable PAUSE and CONTINUE keys
DSTP1	Enable STOP key
XT	End definition of sequence 100

DTXT Display Text Data on RP240 LCD

Type	Programming	Version	C2
Syntax	<a>DTXTtext_data	Units	None
Range	None	Default	None
Attributes	Buffered	See Also	DCLR, DPC, DVO

This command places the text string, text_data, beginning at the current cursor location. The text string can be any alpha character from A to Z, any numeric character from **0** to **9**, or the following characters:

‘,!,@,#,$,%,^,&,(,),-,+,=,{,},[,],|,:,”,’,<,>,?,,,.,/

An underscore (_) is used to separate words. The underscore will be displayed as a space on the RP240 display. The asterisk (*), semicolon (;), backslash (\), and tilde (~) are illegal characters to use with the **DTXT** command. If the text string is too long, the text string will wrap around to the next line.

Command	Description
>DPC205	Position the cursor on line 2, column 5
>DTXTCOMPUMOTOR_DEMO_PROGRAM_1	Place message **COMPUMOTOR DEMO PROGRAM 1** at current cursor position

DVA Display Actual Velocity

Type	Status	Valid	Software Version A
Syntax	aDVA<n>	Units	n = display mode
Range	1 or null	Default	None
Attributes	Immediate, Never Saved	See Also	DVS
Response	*ACTUAL_VELOCITY		

This number is reported in encoder steps per second. This value is the actual velocity being read from the feedback device.

aDVA: Reports the actual velocity continuously until any key is pressed
aDVA1: Report the actual velocity once, without descriptive text

DVO Display Variable Data on RP240 LCD

Type	Programming	Version	C2
Syntax	<DVO>n,n,n,b	Units	None
Range	See Below	Default	None
Attributes	Buffered	See Also	DPC, DTXT, DCLR

The **DVO** command is used to display a variable at the current cursor location. Any of the fifty variables available in the extended X products can be displayed.

1st n = Variable number (Range = 1 to 50)
2nd n = Number of whole digits displayed, digits to left of decimal point (Range = 0 to 15)
3rd n = Number of fractional digits displayed, digits to the right of the decimal point (Range = 0 to 5)
b = Sign bit, 0 = no sign displayed, 1 = display plus or minus sign

Command	Description
>DPC205	Position the cursor on line 2, column 5
>DVO2,2,1,1	Place variable 2 at current cursor position. If variable 2 contained the number 53.23, then +53.2 is displayed on the RP240 LCD starting at column 5, line 2

DVS Display Velocity Setpoint

Type	Status	Valid	Software Version A
Syntax	aDVS<n>	Units	n = display mode
Range	1 or null	Default	None
Attributes	Immediate, Never Saved	See Also	DVA
Response	*ACTUAL_VELOCITY		

This number is reported in motor steps per second and can be reported either once or continually. This value is the velocity being calculated by the indexer.

aDVS: Reports the velocity setpoint continuously until any key is pressed
aDVS1: Report the velocity setpoint once, without descriptive text

DW Dead Band Window

Type	Set-Up	Valid	Software Version A
Syntax	<a>DW<n>	Units	n = Encoder pulses
Range	0 - 65,535	Default	0
Attributes	Buffered, Savable in Sequence	See Also	FSB1, FSC1, CPG, CPM
Response	*POSITION_MAINTENANCE_WINDOW=n_STEPS		

The Dead Band Window (**DW**) command sets the number of encoder pulses the encoder may be in error, before position maintenance corrects for that position error. If the number is set too small, the motor may oscillate about its correct position.

Command	Description
> DW20	The motor may have an error of 20 encoder pulses before position maintenance will attempt to correct that error

E Enable Communications Interface

Type	Programming	Valid	Software Version A
Syntax	<a>E	Units	None
Range	None	Default	None
Attributes	Immediate, Never Saved	See Also	F, SSA, SSI, SSN

The **E** command allows the drive to accept commands over the RS-232C interface. You can re-enable the communications interface with this command if you had previously disabled the interface with the **F** command. If several units are using the same communications interface, the **E** and **F** commands can help streamline programming.

Command	Description
> F	Disables all units on the RS-232C interface
> 1E	Enables Device 1
> 4E	Enables Device 4
> G	Executes the move (only axes 1 and 4 will move)
> E	Enables communication on all devices

ELSE Else

Type	Programming	Valid	Software Version A
Syntax	<a>ELSE	Units	None
Range	None	Default	None
Attributes	Buffered, Savable in Sequence	See Also	IF, NIF

This command is used in conjunction with the **IF** and **NIF** commands to provide conditional program flow. If the **IF** condition is true, the commands between the **IF** and **ELSE** commands are executed, with the commands after the **ELSE** command ignored until the **NIF** command is encountered. If the **IF** condition is false, the commands between the **ELSE** and **NIF** commands are executed. The **ELSE** command is optional and does not have to be included in the **IF** statements.

<**IF**> (condition) commands <**ELSE**> commands <**NIF**>

Command	Description
>IF(INXXXX1_OR_VAR1>20)	If input status is XXXX1 or variable 1 is greater than 20, execute the **GD1** command. If not, execute the **GD2** command
>GD1	Execute predefined move #1
>ELSE	Else
>GD2	Execute predefined move #2
>NIF	End of IF

ER Configure Encoder Resolution

Type	Set-Up	Valid	Software Version A
Syntax	<a>ER<n>	Units	n = pulses/revolution
Range	1 - 65536	Default	4000
Attributes	Buffered, Savable in Sequence	See Also	MR
Response	*ERn		

If an encoder is being used for position purposes, this command configures the number of pulses the encoder produces per revolution. The number of pulses entered is post-quadrature (the number of encoder lines multiplied times 4). A 1,000-line encoder produces 4,000 pulses. You would enter **ER4000**.

Command	Description
> ER8000	Configure encoder resolution to 8,000 as 8,000 pulses are produced by the encoder per revolution.

[ER] Error Flag Operator

Type	Programming/Evaluation	Valid	Software Version A
Syntax	see below	Units	None
Range	None	Default	None
Attributes	Buffered, Savable in Sequence	See Also	RSE, AND, OR, IF, REPEAT, WHILE, FL, IN

The **ER** operator is used to make a comparison against a set of error flags.

Bit #	Flag (1=error present, 0=no error)
1	CCW Hardware Limit
2	CW Hardware Limit
3	System Fault
4	CW Software Limit
5	CCW Software Limit
6	Position Error
7	Input User Fault
8	Absolute Encoder Rollover

Command	Description
IF(ER01100010)	If the CW Limit, System Fault, and Input User Fault were tripped, the **IF** statement would evaluate true and **ERROR** would be printed to the screen.
1"ERROR	
> NIF	

F Disable Communications Interface

Type	Programming	Valid	Software Version A
Syntax	<a>F	Units	None
Range	None	Default	RS-232 Enabled
Attributes	Immediate, Never Saved	See Also	E, SSA, SSI, SSN

Use the **F** command when you are programming multiple units on a single interface. Units that are not intended to process global commands should receive device specific **F** commands. This allows you to program other units without specifying a device identifier on every command.

Command	Description
> 1F	Disables the communications interface on units with device address 1
> 3F	Disables the communications interface on units with device address 3
> G	All of the indexers (except devices 1 and 3) will execute the move (Go)

FAC Set Following Synchronization Rate

Type	Set-Up	Valid	Software Version A
Syntax	<a>FAC<n>	Units	n = synchro rate (5)
Range	0.01 - 99.99	Default	1.0
Attributes	Buffered, Savable in Sequence	See Also	FEN, FSF
Response	*FACn		

When the Enable Following Synchronized Acceleration (**FSF**) command is enabled (set to 1), the value of the **FAC** command sets the change in following ratio per encoder counts (refer to the **FEN** command) to attain an **FOL** value. Specifying a **FAC** value is like setting an acceleration rate, but **FAC** allows the SX to synchronize its position during acceleration to the primary encoder frequency.

Command	Description
> FOL100	Sets following to 100%
> FAC4	Sets the synchronization rate to 4% per FEN value
> FEN10	Sets the synchronization count to 10 encoder counts
> FSF1	Enables synchronized acceleration

FBS Following Base

Type	Set-Up	Valid	Software Version B-DX
Syntax	<a>FBS<n>	Units	n = encoder pulses
Range	1 - 2551	Default	0
Attributes	Buffered, Savable in Sequence	See Also	FSK, FSL, FIN, FC, MSP
Response	None		

This command sets the following base value, which is used in the following Self-Correction mode (**FSL**). The **FBS** value normalizes a count deviation from an expected count that is used to adjust the following value.

Following adjustment = (preset count - normal count) Following Base (**FBS**) ***** **FIN** (Following increment)

*Note: As of software revision E, the Synchronization algorithm for following adjustment no longer uses the **FBS** and **FIN** commands. See the **FSL** command for the algorithm used with revision E and later.*

FC Following Learn Count

Type	Set-Up	Valid	Software Version B
Syntax	<a>FC<n>	Units	n = # of counts
Range	1 - 2147483647	Default	1
Attributes	Buffered, Savable in Sequence	See Also	FSK, FSL, IN
Response	*FCn		

When using the synchronization feature, the Following Learn Count (**FC**) command is the number of encoder counts that are expected between the parts. The parts are indicated by a registration mark that goes to a synchronization input. This number can be learned by the SX by using the Following Learn mode (**FSK**). If the number of expected counts between registration marks is known it can be manually entered.

Command	Description
> 1FC4000	Sets the expected number of counts between parts to 4000 counts

FEN Set Following Synchronization Count

Type	Set-Up	Valid	Software Version B
Syntax	<a>FEN<n>	Units	n = encoder pulses
Range	1 - 50000	Default	4
Attributes	Buffered, Savable in Sequence	See Also	FAC, FSF
Response	*FENm		

When **FSF** is enabled (**FSF1**), the **FEN** value sets the encoder counts that specify the following ratio change, according to the following synchronization rate (refer to **FAC**). This allows the SX to synchronize its position during accelerations to the primary axis' position.

Synchronized Acceleration = **FAC** (%FOL change) **FEN** (primary master counts)

Command	Description
> FOL100	Sets following to 100%
> FEN4	Sets the synchronization count to 4 encoder counts
> FAC1	Sets the synchronization rate to 1%
> FSF1	Enables synchronized acceleration

[FEP] Following Encoder Position Comparison

Type	Programming/Evaluation	Valid	Software Version F
Syntax	see below	Units	None
Range	None	Default	None
Attributes	Buffered, Savable in Sequence	See Also	POS, ER

This evaluation operator is the following encoder position counter.

Command	Description
>IF(FEP>25000)	If the following encoder position is greater than 25000 counts, print FOLLOW
>1"FOLLOW	
>NIF	

FIN Following Increment

Type	Motion	Valid	Software Version B-DX
Syntax	<a>FIN<n>	Units	n = following percent
Range	0.01 - 99.99	Default	1.0
Attributes	Buffered, Savable in Sequence	See Also	FSI, FSK, IN, TD, FS, FBS, MSP
Response	*FINn		

When in Encoder Following mode, the following increment is the amount by which the following percent will change if increased or decreased using the **IN** command to define an input for increasing/decreasing the following percent. Each update time (1-ms sample period) that either the increase or decrease input is active, the following value will be increased or decreased by the value set with the **FIN** command. **FIN** is also used in the synchronization feature to determine the new following ratio correction to maintain synchronization.

*Note: As of software revision E, the Synchronization algorithm for following adjustment no longer uses the **FBS** and **FIN** commands. See the **FSL** command for the algorithm used with revision E and later.*

[FL] User Flag Operator

Type	Programming/Evaluation	Valid	Software Version A
Syntax	see below	Units	None
Range	None	Default	None
Attributes	None	See Also	SFL, AND, OR, IF, REPEAT, WHILE, ER, IN

The **FL** operator is used to make a comparison against a set of user flags. The user flags are set with the **SFL** command.

bit# = 1, Set ; bit# = 0, Not set

Command	Description
>IF(IN1XX1XXXX)	If bits 7 and 5 are set with the SFL1XX1XXXX command, the statement evaluates true, and USER is printed to the screen
>1"USER	
>NIF	

FOL Following Percent

Type	Motion	Valid	Software Version B
Syntax	<a>FOL<n>	Units	n = following %
Range	0.0 - 5000.0	Default	100.0
Attributes	Buffered, Savable in Sequence	See Also	FIN,FOR, FSI, TF
Response	*FOLLOWER=n_PERCENT		

The **FOL** value establishes a percentage of the following ratio that the SX determines by the encoder frequency to produce its step output frequency.

Command	Description
> FOR6.25	Set motor to encoder pulse ratio to 6.25
> FOL100.0	Set following % to 100.0 (the SX produces a step output frequency equal to the encoder input frequency *6.25)
> FOL50.0	Set following % to 50.0 (the SX produces a step output frequency 50% or half of the encoder input frequency *6.25)

FOR Following Ratio

Type	Motion	Valid	Software Version B
Syntax	<a>FOR<n>	Units	n = following ratio
Range	0.001 - 99.999	Default	6.25
Attributes	Buffered, Savable in Sequence	See Also	FIN,FOR, FSI, TF
Response	*MOTION_RATIO=n		

FOR configures the motor resolution to the following encoder resolution ratio used to produce the SX step output frequency. The **FOR** value is multiplied by a percentage to determine the following value.

If the encoder resolution is 4,000 counts per revolution and the motor resolution is 25,000 pulses per revolution, the following ratio is 25,000/4,000 or 6.25. **FOR6.25** should be entered for proper following operation.

FP Following Encoder Point

Type	Motion	Valid	Software Version B
Syntax	<a>FPn	Units	n = encoder pulses
Range	±2147483647	Default	None
Attributes	Buffered, Savable in Sequence	See Also	MPP, NG, DP

When in Position Profile mode (**MPP**), commands subsequent to a **G** command are processed. Upon encountering **FP**, command processing is paused until the following encoder incremental count matches the value set by the **FP** command.

Command	Description
> MPI	Set to Incremental mode
> A5	Set acceleration to 5 rps^2
> AD10	Set deceleration to 10 rps^2
> V2	Set velocity to 2 rps
> D200000	Set distance to 200,000 steps
> MPP	Set to Position Profile mode
> G	Execute the move (Go)
> FP10000	Wait until following encoder has traveled 10,000 steps
> V4	Set velocity to 4 rps—motor accelerates to 4 rps
> FP20000	Wait until following encoder has traveled another 20,000 steps
> O11	Turn on outputs 1 and 2
> FP10000	Wait until following encoder has traveled another 10,000 steps
> V2	Set velocity to 2 rps—motor decelerates to 2 rps
> NG	End Position Profile

FPA Following Encoder Absolute Point

Type	Motion	Valid	Software Version B
Syntax	<a>FPAn	Units	n = encoder pulses
Range	±2147483647	Default	None
Attributes	Buffered, Savable in Sequence	See Also	MPP, NG, DP, FP
Response	None		

When in Position Profile mode (**MPP**), commands subsequent to a **G** command are processed. When the SX reaches an **FPA** command, processing stops until the absolute position of the following encoder counter matches the value set by **FPA**.

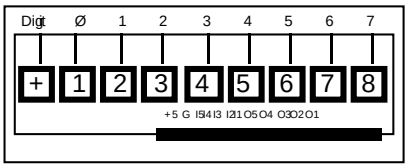
Command	Description
> MPI	Set to Incremental mode
> D200000	Set distance to 200,000 steps
> MPP	Set to Position Profile mode
> G	Execute the move (Go)
> FPA11000	Wait until following encoder's absolute position is 11,000 steps
> O11	Turns on outputs 1 and 2
> NG	End position profile

FRD Read Following Value via Parallel Programming Input/Output

Type	Programming	Valid	Software Version B
Syntax	<a>FRD<c><d><e>	Units	c, d = digit selector, e = scaling factor
Range	c, d = 0 - 7, e = 0 - 4	Default	None
Attributes	Buffered, Savable in Sequence	See Also	STR, IN, OUT, FOL, DRD, LRD, TRD, VRD, XRP

FRD instructs the SX to read Following Ratio values from the TM8 Module (refer to the SX Indexer/Drive User Guide for more on the TM8 Module). Inputs 1 - 4 must be configured as data inputs. Input 5 May be configured as a sign input. The inputs must be configured as active low (**INL0**) and outputs 1 - 3 must be configured as strobe outputs to use the TM8 Module.

Helpful Hint: <c> & <d> values represent the TM8 Module digits:



The command syntax allows for digit range selection through the optional <c><d> fields. The <c> field is the start of the digit range and the <d> field is the end of the digit range that the TM8 Module reads. The raw data format is XXXX.X, the maximum value = 5000.0. Any larger number will result in a ? Response. The <c> field is always less than or equal to the <d> field value.

The <e> field scales the distance value by 10^e. If the <c> and <d> fields are used, the <e> field must be used. If the <c>, <d>, and <e> fields are not used, **FRD** will read all the TM8 Module's digits. If you use the TM8 Module, the Output Strobe Delay Time (**STR**) must be set to 10 or greater.

You may use a PLC with **FRD** to enter data into the SX. Upon execution of a **FRD** command, the outputs strobe through a BCD sequence at the Set Strobe Output Delay Time (**STR**) command rate and read one BCD digit at a time. The I/O must be configured as in the TM8 Module. A **FRD** command will issue the following strobe sequence:

SX Output Voltage Level			
01	02	03	SX Data In (Active Low) (I1-I4)
low	low	low	MSD (Digit 1)
high	low	low	Digit 2
low	high	low	Digit 3
high	high	low	Digit 4
low	low	high	Digit 5
high	low	high	Digit 6
low	high	high	Digit 7
high	high	high	LSD (Digit 8)

The PLC, while reading the output strobe, must place the proper data at SX inputs 1 - 4. Input 1 is the digit's LSB and input 4 the MSB. Configuring an input as a data valid line allows the PLC to control the rate at which the outputs will strobe through the above sequence.

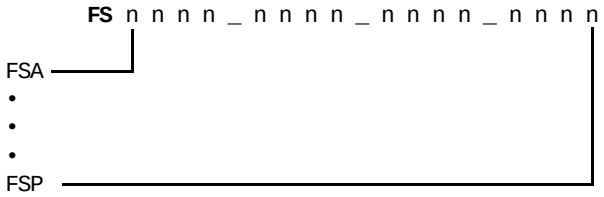
Turn the TM8's thumbwheel to the following digits: + 1 2 3 4 5 6 7 8 and enter the following commands.

Command	Description
> FRD040	Request ratio from digits 0-4
> 1FOL	
> FOLLOWER-1234.5_PERCENT	
> FRD330	Request digit 3
> 1FOL	
> FOLLOWER-0000.4_PERCENT	
> FRD351	Request digit range 3 through 5 multiplied by 10^1
> 1FOL	
> FOLLOWER-00456.0_PERCENT	

FS Encoder Function Report

Type	Status	VALID	Software Version A
Syntax	<a>FS	Units	None
Range	None	Default	None
Attributes	Immediate, Never Saved	See Also	OS, SS
Response	*nnnn_nnnn_nnnn_nnnn		

This command reports the status of the **FS** command (**FSA** - **FSP**) settings.



FSA Enable Following Mimic Mode

Type	Set-Up	Valid	Software Version B
Syntax	<a>FSAn	Units	n = mode
Range	0 = off, 1 = on	Default	0
Attributes	Buffered, Savable in Sequence	See Also	FSF, FSI, FSK, FSL, FSN, FSP

When **FSA** is set to 1, the SX does not accelerate or decelerate between speeds. For every sample period, the SX sends the **FOL** ratio of motor pulses for all encoder pulses seen during the sample period. The SX instantly accelerates if the speed changes from one sample period to the next. This mode works well when the master axis starts from zero speed and accelerates to its final speed. Encoder and motor pulses will track exactly.

Command	Description
> FSA1	Turns on Following Mimic mode
> FSA0	Turns off Following Mimic mode

FSB Motor/Encoder Step Mode

Type	Set-Up	Valid	Software Version A
Syntax	aFSBn	Units	n = mode
Range	0 = motor steps, 1 = encoder steps	Default	0
Attributes	Buffered, Savable in Sequence	See Also	ER, D, MR, PR

The Motor/Encoder Step mode (**FSB**) command sets the indexer to perform moves in either motor steps or encoder steps.

FSB0: Motor steps
FSB1: Encoder steps

In Motor Step mode, the Distance command (**D**) defines moves in motor steps. In Encoder Step mode, the distance command defines moves in encoder steps.

You must set up the indexer for the correct encoder resolution. The Encoder Resolution (**ER**) command defines encoder resolution.

Enabling Encoder Step mode does not guarantee that your moves will position to the exact encoder step commanded. Position maintenance (**FSC**) must be enabled to activate closed-loop servoing.

Command	Description
> ER4000	Set up encoder. 4,000 encoder pulses (1,000 lines) are produced per revolution of the encoder.
> FSB1	Set moves to encoder step mode
> A10	Set acceleration to 10 rps^2
> AD15	Set deceleration to 15 rps^2
> V5	Set velocity to 5 rps
> D4000	Set distance to 4,000 encoder steps
> G	Executes a one revolution move

FSC Enable/Disable Position Maintenance

Type	Set-Up	Valid	Software Version A
Syntax	<a>FSCn	Units	n = mode
Range	0 = disable, 1 = enable	Default	0
Attributes	Buffered, Savable in Sequence	See Also	CPG, CPM, DW, ER, FSB

FSC1: Enable Position Maintenance
FSC0: Disable Position Maintenance

Enabling position maintenance causes the indexer to servo the motor until the correct encoder position is achieved. This occurs at the end of a move (if the final position is incorrect) or any time the indexer senses a change in position while the motor is at zero velocity. You must have an encoder connected to enable position maintenance. Position maintenance can only be enabled if Encoder Step mode is selected (**FSB1**).

Command	Description
> ER2000	Set encoder resolution to 2,000 counts/revolution
> FSB1	Set Encoder Step mode
> FSC1	Enable position maintenance

FSD Stop on Stall

Type	Set-Up	Valid	Software Version A
Syntax	<a>FSCn	Units	n = mode
Range	0 = off, 1 = on	Default	0
Attributes	Buffered, Savable in Sequence	See Also	BCPE, CPE, ER, FS

If you enter **FSD0**, the indexer will attempt to finish the move when a stall is detected, even if the load is jammed. A stall occurs when the position error exceeds the maximum position error (**CPE**). If you enter **FSD1**, the indexer will stop the move in progress when it detects a stall. The move is stopped with a controlled deceleration.

*Note: Once a stall is detected and the motor stops, the **DPE** value is reset to 0. Stall Detect can be used in both motor step mode (**FSB0**) and encoder step mode (**FSB1**).*

Command	Description
> CPE100	Set allowable position error to 100 motor steps
> FSD1	Enable stop on stall

FSF Enable Following Synchronized Acceleration

Type	Set-Up	Valid	Software Version B
Syntax	<a>FSFn	Units	n = mode
Range	0 = off, 1 = on	Default	0
Attributes	Buffered, Savable in Sequence	See Also	FSA, FSI, FSK, FSL, FSN, FSP

FSF1: Acceleration is enabled
FSF0: Acceleration is disabled

The SX accelerates to a **FOL** value by a preset change in ratio (**FAC** command) per primary encoder count (**FEN** command).

Synchronized Acceleration = **FAC** (FOL% change)**FEN** (primary master counts)

Command	Description
> FSF1	Enables Following Synchronized Acceleration
> FSF0	Disables Following Synchronized Acceleration

FSG Stop Position Maintenance Move on Limit Encountered

Type	Set-Up	Valid	Firmware Revision E2
Syntax	<a>FSG<n>	Units	n - mode
Range	0 = disabled, 1 = enabled	Default	0
Attributes	Buffered, Savable in Sequence	See Also	FSB, FSC, FSH

The **FSG** command is used to stop a position maintenance move when a limit is hit. This will also safeguard a failed encoder, or runaway system.

FSG1: Stop position maintenance when a limit has been encountered.
FSG0: Ignore Limit while making a position maintenance correction

*Note: This command requires that Position Maintenance (**FSC1**) be enabled prior to issuing the **FSG** command.*

FSH Abort Position Maintenance Move on Limit Encountered

Type	Set-Up	Valid	Firmware Revision E2
Syntax	<a>FSH<n>	Units	n = mode
Range	0 = disabled, 1 = enabled	Default	0
Attributes	Buffered, Savable in Sequence	See Also	FSB, FSC, FSG

The **FSH** command is used to abort position maintenance when a limit is hit. Position Maintenance (**FSC**) and Abort Position Maintenance on Limit (**FSH**) will need to be reissued after a limit is cleared. This will also safeguard a failed encoder, or runaway system.

FSH1: Abort Position Maintenance when a limit is encountered during a position maintenance move.
FSH0: Ignore Limit while making a position maintenance correction move.

*Note: This command requires that Position Maintenance (**FSC1**) be enabled prior to issuing the **FSH** command.*

FSI Enable/Disable Following Mode

Type	Set-Up	Valid	Software Version B
Syntax	<a>FSIn	Units	n = mode
Range	0 = off, 1 = on	Default	0
Attributes	Buffered, Savable in Sequence	See Also	FOL, FOR, FSN

The **FSI** command sets the SX velocity and position to be dependent on a primary axis input. Encoder inputs or pulse and direction is input to the SX and sampled to calculate an output step frequency. The value of the step frequency is determined by the **FOL** command which sets the percentage of the primary axis that the SX outputs.

FSK Set Following Learn Mode

Type	Set-Up	Valid	Software Version B
Syntax	<a>FSKn	Units	n = mode
Range	0 = off, 1 = on	Default	0
Attributes	Buffered, Savable in Sequence	See Also	FIN, FSL, FSB

The **FSK** command sets the SX to learn an expected encoder count based on an input. An input must be defined as a synchronization input. When the input turns on, a counter begins counting encoder pulses, and then the input should turn off. When the input turns on again, the counter ceases counting and contains the expected number of pulses between input activations. This value identifies the pulse count between parts. Self-Correction mode (**FSL**) adjusts the following value based on deviations from the expected count to maintain part consistency.

Command	Description
> FSK1	Turn on Following Learn mode

FSL Set Following Self-Correction Mode

Type	Set-Up	Valid	Software Version A
Syntax	<a>FSLn	Units	n = mode
Range	0 = off, 1 = on	Default	0
Attributes	Buffered, Savable in Sequence	See Also	FIN, FSK, FSB, MSP, FC

This command sets the SX's Following function into a Self-Correction mode in which the following ratio continually adjusts its following value based on an input providing information on part placement. When the input turns on, a counter begins counting and then the input turns off. When the input is turned on again, the counter ceases counting. This value is compared to a normal count obtained either during the Following Learn mode (**FSK**) or with the **FC** command to adjust the following value to maintain part consistency.

Adjustment to following value = (present count - normal count) **FBS** (Following Base) * **FIN** (Following Increment)

*Note: As of software revision E, the Synchronization algorithm for following adjustment no longer uses the **FBS** and **FIN** commands as stated above. The synchronization algorithm used with revision E and after is as follows:*

Error = (# of encoder steps since last input)—(FC value)
Adjustment to Following Value = FC Value/[(FC value)—Error]

Command	Description
> FSL1	Turn on Following Self-Correction mode

FSM Set Absolute Encoder

Type	Set-Up	Valid	Software Version B
Syntax	<a>FSMn	Units	n = mode
Range	0, 1	Default	0
Attributes	Buffered, Savable in Sequence	See Also	ER, FSB, Z, RVV

This command sets the encoder interface to be either an absolute or incremental encoder.

FSM0: SX is set for an incremental encoder
FSM1: SX is set for an absolute encoder

The SX is compatible with Compumotor's AR-C absolute encoder.

You must reset the SX if the **FSM** mode is changed with the **Z** command.

Note: Effective January 1, 1995, the SX/SXF will no longer have the absolute encoder interface capability as a standard feature. Therefore, the standard SX/SXF will not be compatible with the AR-C absolute encoder. T'he SX/SXF absolute encoder interface will be a standard option. Refer to the **RVV** command for information on how to distingush between units with the option and units without the option.

FSN Set Pulse Following

Type	Set-Up	Valid	Software Version B
Syntax	<a>FSNn	Units	n = mode
Range	0 = off, 1 = on	Default	0
Attributes	Buffered, Savable in Sequence	See Also	FS

The **FSN** command sets the SX to follow a pulse and direction input rather than a quadrature encoder input when in Following mode. The table below shows the 9-pin encoder connector functions:

Pin #	**FSN0**	**FSN1**
3	Channel A+	Step+
4	Channel A-	Step-
5	Channel B+	Direction+
6	Channel B-	Direction-

FSP Set Tracking Mode

Type	Set-Up	Valid	Software Version B
Syntax	aFSPn	Units	n = mode
Range	0 = off, 1 = on	Default	0
Attributes	Buffered, Savable in Sequence	See Also	FS, H

FSP allows the SX to continually track a primary encoder input's direction and velocity. The SX must be in Continuous mode to be in Tracking mode. The SX's direction can be the same as the primary encoder (H+) or be inverted (H-).

G Go

Type	Motion	Valid	Software Version A
Syntax	<a>G	Units	None
Range	None	Default	None
Attributes	Buffered, Savable in Sequence	See Also	A, AD, D, MC, MN, MPA, MPI, S, V

The **G** command initiates motion based on previous motion parameters. You do not have to re-enter the Acceleration (**A**), Deceleration (**AD**), Velocity (**V**), Distance (**D**), or current mode (**MN** or **MC**) with each **G** command. The **A**, **AD**, **V**, and **D** values are saved in nonvolatile memory.

In Incremental Preset mode (**MPI**), **G** will move the motor the number of steps specified with the **D** command. A **G** in Absolute Preset mode (**MPA**) will not cause motion unless you change the Distance (**D**) value first (from current position). In Continuous mode (**MC**), you only need to enter the **A**, **AD**, and **V** commands prior to the **G**. The SX ignores distance in this mode. No motion will occur until you enter **G** in both the Normal (**MN**) and Continuous (**MC**) modes. If motion does not occur, a software travel limit or end-of-travel limit switch may be on.

Command	Description
> MN	Sets to Normal (preset) mode
> MPI	Set to Incremental Position mode
> A5	Sets acceleration to 5 rps^2
> AD10	Sets deceleration to 10 rps^2
> V10	Sets velocity to 10 rps
> D25000	Sets distance to 25,000 steps
> G	Executes the move (Go)
> A1	Sets acceleration to 1 rps^2
> G	Executes a second move (Go) with parameters A1, AD10, V10, D25000

GD Go Defined

Type	Motion	Valid	Software Version A
Syntax	<a>GDn	Units	n = defined sequence
Range	1 - 16	Default	None
Attributes	Buffered, Savable in Sequence	See Also	GDEF, G

The Go Defined (**GD**) command executes a move that has been predefined and precalculated with the Move Definition (**GDEF**) command. The command sequence **A**n **AD**n **V**n (or **FOL**n) **D**n **G** is compressed to **GD**, where the acceleration, deceleration, velocity (or following ratio), and distance have been previously defined and stored in nonvolatile memory with the **GDEF** command.

Command	Description
> GDEF10,A25,V5,D10000,AD5	Define and calculate move #10
> GDEF16,A2,AD5,V1,D20000	Define and calculate move #16
> GD10	Execute predefined move #10
> GD16	Execute predefined move #16

GDEF Move Definition

Type	Motion	Valid	Software Version A
Syntax	<a>GDEFn,Am,ADm,Vm(or FOLm),Dm	Units	n = move sequence #
Range	1 - 16	Default	None
Attributes	Buffered, Savable in Sequence	See Also	GD

The **GDEF** command defines move number n with parameters **A** (acceleration), **AD** (deceleration), **V** (velocity), **FOL** (following ratio), and **D** (distance). The move is precalculated for immediate execution with the Go Defined (**GD**) command. You can enter up to 16 predefined moves (the values are stores in nonvolatile memory). You can use the moves any number of times in any sequence.

Command	Description
> GDEF1,A5,AD10,V5,D250000	Define & calculate move #1
> GDEF2,A20,AD25,V10,D100000	Define & calculate move #2
> GDEF3,A20,AD25,FOL100,D100000	Define & calculate move #3
> GD1	Execute predefined move #1
> GD2	Execute predefined move #2

GH Go Home

Type	Motion	Valid	Software Version A
Syntax	<a>GH<n>	Units	n = rps
Range	0.00001 - 50.0000	Default	1
Attributes	Buffered, Savable in Sequence	See Also	GHA, GHAD, GHF, GHV, IN, INL, OSB, OSC, OSD, OSG, OSH

The Go Home (**GH**) command instructs the SX to search for the home limit switch in the positive or negative direction. It defines home as the position where the home limit signal changed states nearest the edge selected with the **OS** command. When the selected edge is detected (usually via a load activated switch) the motor decelerates to 0 velocity.

The SX then positions the motor on the outside of the selected edge. The motor will creep at final go home velocity (defined by the **GHF** command) in the direction of the home active region. The **GH** command will not zero the position counters when the home position is reached.

The Enable Encoder Z Channel (**OSD**) command can be used to enable the SX's encoder Z channel. When **OSD** is enabled the SX will look for the Z channel. The Z channel pulse should be enveloped by the home limit input's active region.

The indexer will reverse direction if an end-of-travel limit is activated while searching for home. However, if a second end-of-travel limit is encountered in the new direction, the go home procedure will stop and the operation will be aborted. The **RG** command will indicate if the operation was successful. If the Backup To Home Switch (**OSB**) function is disabled by the **OS** command, the motor will be considered at home if the home limit input is still active at the end of the deceleration, following the encounter of the selected edge of the home switch. Refer to the SX/SXF Indexer/Drive User Guide for a detailed description of the homing function.

If you set the go home velocity using the **GH** command, the motor will move in the direction and velocity specified by the **GH** command and the value is stored in the **GHV** command. If you do not specify the velocity, the homing will be executed using the velocity and direction defined by the **GHV** command.

Position Profile mode is temporarily disabled when homing.

Command	Description
> GHA10	Sets go home acceleration to 10 rps^2
> GHAD12	Sets go home deceleration to 12 rps^2
> GHV2	Sets go home velocity to 2 rps
> INL0	Remote go home will be executed when the go home input is low (closed)
> GH	Go home in the direction and velocity specified by GHV using the GHA acceleration
> PZ	Clears position counter

GHA Go Home Acceleration

Type	Motion	Valid	Software Version A
Syntax	<a>GHA<n>	Units	n = rps^2
Range	0.01 - 9999.99	Default	10.0
Attributes	Buffered, Savable in Sequence	See Also	GH, GHAD, GHV , GHF, OS
Response	GHAn		

The Go Home Acceleration (**GHA**) command sets the acceleration value that will be used during any subsequent Go Home (**GH**) moves. The value is stored in nonvolatile memory. The go home velocity is set using the Go Home Velocity (**GHV**) command. You must use the **IN** command to specify the remote go home input.

Command	Description
> GHA10	Sets go home acceleration to 10 rps^2
> GHAD12	Sets go home deceleration to 12 rps^2
> GHV2	Sets go home velocity to 2 rps
> IN1S	Sets input 1 as the remote go home input
> INL0	The remote go home will be executed when go home is low (closed)
> GH	Go home in the direction and velocity specified by **GHV**, using the **GHA** acceleration and **GHAD** deceleration

GHAD Go Home Deceleratаion

Type	Motion	Valid	Software Version A
Syntax	<a>GHAD<n>	Units	n = rps^2
Range	0.01 - 9999.99	Default	10.0
Attributes	Buffered, Savable in Sequence	See Also	GH, GHA, GHV , GHF, OS
Response	GHADn		

The Go Home Deceleration (**GHAD**) command sets the deceleration value that will be used during any subsequent go home (**GH**) moves. The value is stored in nonvolatile memory. Go home velocity is set with the Go Home Velocity (**GHV**) command. You must use the **IN** command to specify the remote Go Home input.

Command	Description
> GHA10	Sets go home acceleration to 10 rps^2
> GHAD12	Sets go home deceleration to 12 rps^2
> GHV2	Sets go home velocity to 2 rps
> IN1S	Sets input 1 as the remote go home input
> INL0	The remote go home will run when go home is low (closed)
> GH	Go home in the direction and velocity specified by GVH, using the GHA acceleration and GHAD deceleration

GHF Go Home Final Velоctiy

Type	Motion	Valid	Software Version A
Syntax	<a>GHF<n>	Units	n = rps^2
Range	0.0001 - 50.00000	Default	10.0
Attributes	Buffered, Savable in Sequence	See Also	GH, GHA, GHAD, GHV, OS
Response	GHFn		

The Go Home Final Velocity (**GHF**) command sets the velocity for the final approach in the go home sequence. The value is stored in nonvolatile memory.

Command	Description
> GHF.1	The velocity of the final approach of the next go home move will be 0.1 rps
> GH2	Execute go home at 2 rps (CW)

GHV Go Home Velocity

Type	Motion	Valid	Software Version A
Syntax	<a>GHV<n>	Units	n = rps
Range	±0.00001 - 50.00000	Default	1.0
Attributes	Buffered, Savable in Sequence	See Also	GH, GHA, GHF, GHAD, OS
Response	*GHVn		

The GHV sets the velocity and direction used for remote homing or for subsequent Go Home (**GH**) commands. It is stored in nonvolatile memory. When you turn on the remote go home input, or issue a Go Home (**GH**) command over RS-232C interface, the motor will start moving in the direction and velocity specified by **GHV**. You can define the acceleration of the homing process with the **GHA** command and the deceleration with the **GHAD** command. You must specify the remote go home input with the **IN** command.

Command	Description
> GHA10	Sets go home acceleration to 10 rps^2
> GHAD12	Sets go home deceleration to 12 rps^2
> GHV2	Sets go home velocity to 2 rps
> IN1S	Sets input 1 as the remote go home input
> INL0	The remote go home will be executed when the go home input is low (closed)
> GH	Go home in the direction and velocity specified by the GHV command, using the GHA acceleration and GHAD deceleration.

GOSUB GOSUB Sequence

Type	Programming	Valid	Software Version A
Syntax	<a>GOSUBn	Units	n =sequence number
Range	1 -100	Default	None
Attributes	Buffered, Savable in Sequence	See Also	XR, XG, GOTO

GOSUB jumps to the designated sequence to continue program execution. If an **XT** command is reached in the called sequence, program execution returns to the originating sequence. The maximum number of nested **GOSUB** commands is 16. Nesting is a sequence calling a sequence that also contains a **GOSUB** command. When the **GOSUB** routine is completed, program control returns to the statement following the **GOSUB** command.

Command	Description
> XE5	Erase sequence #5
> XD5	Define sequence #5
A2	Set acceleration to 2 rps^2
AD2	Set deceleration to 2 rps^2
V5	Set velocity to 5 rps
D25000	Set distance to 25,000 steps
GOSUB2	Call sequence #2 as subroutine
D50000	Set distance to 50,000 steps
GOSUB2	Call sequence #2 as subroutine
XT	End defining sequence #5
> XE2	Erase sequence #2
> XD2	Define sequence #2
G	Execute move
1PR	Report position counter
XT	End defining sequence #2
> XR5	Execute sequence #5

GOTO GOTO Sequence

Type	Programming	Valid	Software Version A
Syntax	<a>GOTOn	Units	n = sequence number
Range	1 - 100	Default	None
Attributes	Buffered, Savable in Sequence	See Also	XG, XR, GOSUB

This command transfers program control to a designated sequence to continue program execution. Once you jump to a sequence using the **GOTO** command, the program will not automatically return to the originating sequence as a **GOSUB** operation would. There are no limitations on the number of **GOTO** commands as there is no nesting involved.

Command	Description
>XE5	Erase sequence #5
>XD5	Define sequence #5
A2	Set acceleration to 2 rps^2
AD2	Set deceleration to 2 rps^2
V5	Set velocity to 5 rps
D25000	Set distance to 25,000 steps
GOTO2	Goto sequence #2

XT	End defining sequence #5
>XE2	Erase sequence #2
>XD2	Define sequence #2
G	Execute move
1PR	Report position counter
XT	End defining sequence #2
>XR5	Execute sequence #5

^H Backspace

Type	Programming	Valid	Software Version A
Syntax	^H (CTRL + H) ASCII 8	Units	None
Range	None	Default	None
Attributes	Immediate, Never Saved	See Also	None

This command allows you to delete the last character that you entered. The **^H** command will not prevent execution of an immediate command. A new character may be entered at that position to replace the existing character. (**^H** indicates that the Ctrl key is held down when the H key is pressed.) This command prompts the SX to backup one character in the command buffer, regardless of what appears on the terminal. On some terminals, Ctrl and the left arrow (¨) keys produce the same character.

H Set Direction

Type	Motion	Valid	Software Version A
Syntax	<a>H<s>	Units	s = direction
Range	+ = CW, - = CCW	Default	+
Attributes	Buffered, Savable in Sequence	See Also	D, MPA

The Set Direction (**H**) command changes or defines the direction of the next move that the system will execute. This command does not affect moves already in progress.

H+ = Sets move to CW
H– = Sets move to CCW
H = Changes direction from the previous setting

Distance (**D**) command entered after the Set Direction (**H**) command overrides the **H** command.

Command	Description
> MN	Sets to Normal mode
> A5	Sets acceleration to 5 rps^2
> AD10	Sets deceleration to 10 rps^2
> V5	Sets velocity to 5 rps
> D25000	Sets distance to 25,000 steps
> G	Executes the move (Go) in CW direction
> H	Reverses direction
> G	Executes the move (Go) in CCW direction

HALT Halt

Type	Programming	Valid	Software Version A
Syntax	<a>HALT	Units	None
Range	None	Default	None
Attributes	Buffered, Savable in Sequence	See Also	K

The **HALT** command stops program execution and clears the command buffer.

Command	Description
> XE1	Erase sequence #1
> XD1	Define sequence #1
A2	Set acceleration to 2 rps^2
AD20	Set acceleration to 20 rps^2
V2	Set velocity to 2 rps
D100000	Set distance to 100,000 steps
L25	Loop 25 times
G	Execute move (Go)
IF(IN10)	If input 1 on and input 2 off execute all commands until NIF encountered
HALT	Terminate program execution
NIF	End of If statement
N	End of loop
XT	End defining sequence #1

ID Immediate Distance

Type	Motion	Valid	Software Version A
Syntax	<a>IDn	Units	n = steps
Range	±2147483647	Default	None
Attributes	Immediate, Never Saved	See Also	IV

During motion, the travel distance can be set to a new value with the **ID** command. A change in direction is not allowed.

Command	Description
> MN	Set to Normal mode
> A10	Set acceleration to 10 rps^2
> AD15	Set deceleration to 15 rps^2
> V5	Set velocity to 5 rps
> D500000	Set distance to 500,000 steps
> G	Execute the move (Go)
> ID750000	Set distance to 750,000 steps—motor travels 750,000 steps instead of 500,000 steps

IF IF

Type	Programming	Valid	Software Version A
Syntax	<a>IFe	Units	e = evaluation commands
Range	Evaluation commands	Default	None
Attributes	Buffered, Savable in Sequence	See Also	ELSE, NIF, REPEAT, WHILE, [ER], [IN], [FL], AND, OR

This command is used in conjunction with the **ELSE** and **NIF** commands to provide conditional program flow. If the **IF** condition is true, the commands between **IF** and **ELSE** are executed. Commands after **ELSE** are ignored until **NIF** is encountered. If the **IF** condition is false, commands between the **IF** and **ELSE** are ignored, and commands between the **ELSE** and **NIF** are executed. The **ELSE** command is optional and does not have to be included in **IF** statements.

IF(condition)...commands...**ELSE**...commands...**NIF**

*Note: **IF** commands may not be nested, this means an **IF** command may not be placed in another **IF** statement.*

Command	Description
> IF(VAR1=5_AND_INXXXX11)	If variable 1 is 5 and inputs 1 and 2 are active, execute next command, else execute command following ELSE
> GD1	Execute predefined move 1 if above conditions are true
> ELSE	Else
> GD2	Execute predefined move 2 if above conditions are false
> NIF	End of IF

IN Input Functions

Type	Set-Up	Valid	Software Version A
Syntax	<a>IN<n><x>	Units	n = input #, x = function
Range	n = 1-8, x = A-Z	Default	x = A
Attributes	Buffered, Savable in Sequence	See Also	INL, INR, IS, [IN]
Response	See Below		

The Set Input Functions (**IN**) command configures the function for each of the 8 inputs on the SX. The on and off state of all 8 inputs can be defined using the Input Polarity (**INL**) command. Using this command, you can set your input to be active high or active low. You can configure each input to perform one of the following functions:

Function A — Trigger Input - Used with the **TR** command as a comparison input.

- ❏ INPUT ON - Trigger active
- ❏ INPUT OFF - Trigger inactive

Function B — Sequence Select Input - Executes predefined sequences from remote inputs.

- ❏ INPUT ON - Sequence input selected
- ❏ INPUT OFF - Sequence input not selected

Function C — Kill Input - Immediately halts execution of a move and dumps the sequence or command buffer. Same as Kill (**K**) command.

- ❏ INPUT ON - Kill initiated
- ❏ INPUT OFF - Kill not initiated

Function D — Stop Input - Decelerates the motor to a stop using the preset deceleration (**AD**) rate. Same as Stop (**S**) command.

- ❏ INPUT ON - Stop initiated
- ❏ INPUT OFF - Stop not initiated

Function E — Command Enable Input - Enables and disables the drive. Same as **ON** and **OFF** commands.

- ❏ INPUT ON - Shutdown active (motor windings off)
- ❏ INPUT OFF - Shutdown Inactive (motor windings on)

Function F — Pause/Continue Input - Pauses and continues command execution. Same as Pause (**U**) and Continue (**C**) commands.

- ❏ INPUT ON - Pause Execution
- ❏ INPUT OFF - Continue Execution

Function G — Go Input - Initiates a move. This is entered as a Go (**G**) command. To make the motor move, you must bring the input from the off to the on state.

- ❏ INPUT ON - Execute the move (Go)
- ❏ INPUT OFF - Do not execute the move

Function H — Direction Input - Used to toggle the direction of the motor. Same as entering the H command. You must toggle your input from off to on to change the direction.

Function I — Synchronization Input - Used to adjust the following rate based on the rate of this input turning on (see **FSK** and **FSL** commands).

Function J — Jog CW Input - This input initiates jogging in the CW or positive direction. To use this input, you must enable jogging using the **OSE1** command and set both High-Jog Velocity (**JVH**) and Low-Jog Velocity (**JVL**).

- ❏ INPUT ON - Start Jogging in CW direction
- ❏ INPUT OFF - Stop Jogging

Function K — Jog CCW Input - This input initiates jogging in the CCW direction. To use this input, you must enable jogging using the **OSE1** command and define both High-Jog Velocity (**JVH**) and Low-Jog Velocity (**JVL**).

- ❏ INPUT ON - Start Jogging in CCW direction
- ❏ INPUT OFF - Stop Jogging

Function L — Jog Speed Select Input - Selects two different velocities for jogging. High-Jog Velocity is defined by **JVH** command and Low-Jog Velocity defined by the **JVL** command. If not used, jogging defaults to the low-jog velocity.

- ❏ INPUT ON - High Jog Velocity selected
- ❏ INPUT OFF - Low Jog Velocity selected

Function M — Terminate Loop Input - Terminates the loop after finishing the current pass. Same as **Y** command.

- ❏ INPUT ON - Terminate Loop
- ❏ INPUT OFF - Terminate Loop not activated

Function N — Data Input - Loads parallel bytes of data into the SX. You can use this input to strobe data (i.e., distance, velocity, loop counts, and sequence inputs). Inputs 1 - 4 should only be defined as data inputs. See the **DRD**, **FRD**, **LRD**, **TRD**, **VARD**, **VRD**, **XRD**, and command descriptions for a detailed explanation of the data inputs.

- ❏ INPUT ON - Data input true
- ❏ INPUT OFF - Data input false

Function O — Reserved

Function P — Memory Lock Input - Prevents sequence editing and prevents the use of some commands. This input is useful if you do not want others to access your program or modify parameters.

- ❏ INPUT ON - Lock out sequence editing and commands
- ❏ INPUT OFF - Do not lock out commands

Commands Locked Out: **CPE**, **CPG**, **CPM**, **RIFS**, **XE**, **XD**
The indexer will report its setting, but you cannot change or use those commands while you are in a locked-out mode.

Function Q — Reserved

Function R — Reset Input - Invokes a software reset of the SX. Same as Reset (**Z**) command.

- ❏ INPUT ON - Reset the SX

❏ INPUT OFF - Do not reset the SX

Function S Go Home Input - Initiates a Go Home (**GH**) command. Same as executing **GH** command. You must define Go Home Velocity (**GHV**), Go Home Acceleration (**GHA**), and Go Home Deceleration (**GHAD**) before executing homing.

❏ INPUT ON - Execute Go Home
❏ INPUT OFF - Do not execute Go Home

Function T Position Zero Input - Sets the present position as the absolute zero position. Same as **PZ** command.

❏ INPUT ON - Zero the absolute position counter
❏ INPUT OFF - Do not zero the absolute position counter

Function U User Fault Input - Indicates, from other devices, that a fault condition exists. This input might be a pushbutton, PLC, or a sensor input.

❏ INPUT ON - Fault condition exists

❏ INPUT OFF - No fault condition exists

When **User Fault** is activated, this fault input is latched. To clear the fault, you must reset the drive, cycle power, or issue an **ON** command through the RS-232C interface.

Function V Data Valid Input - Used with **STR** command to handshake the data transfer from the interface to the SX's input line. It is used to input distance (**DRD**), velocity (**VRD**), loop count (**LRD**), following ratio (**FRD**), time delay (**TRD**), variables (**VARD**), and the sequence number (**XRD**). When you enable one of the inputs as data valid, the data will not be read until a data valid line becomes active. If an input is not assigned to data valid, the data loading occurs synchronously according to the Strobe (**STR**) command.

❏ INPUT ON - Data is valid
❏ INPUT OFF - Data is not valid

Function W Data Sign Input - Input used with the **DRD** command to indicates the sign of the distance value being loaded. If an input is not set to a W function, the sign defaults to a + sign.

❏ INPUT ON - Sign is negative
❏ INPUT OFF - Sign is positive

Function X Increase Following Ratio Input- Input to increase the following value when following is enabled.

Function Y Decreased Following Ratio Input - Input to decrease the following value when following is enabled.

Function Z No function

Command	Description
> IN1C	Set up input 1 as Kill (K) input
> IN2G	Set up input 2 as Go (G) input
> IN3J	Set up input 3 as CW Jog input
> IN4K	Set up input 4 as CCW Jog input
> IN5L	Set up input 5 as jog velocity Select input
> OSE1	Enable Jogging
> JVH10	Set high speed jog velocity to 10 rps
> JVL1	Set low speed jog velocity to 1 rps
> 1IN	Status of all the inputs are reported *0A__CW_LIMIT_(STATUS_ON) *0B__CCW_LIMIT_(STATUS_ON) *0C__HOME_LIMIT_(STATUS_OFF) *0R__REGISTRATION_INPUT (STATUS_OFF) *01_C_KILL_INPUT_(STATUS_OFF) *02_G_GO_INPUT_(STATUS_OFF) *03_J_JOG_CW_INPUT_(STATUS_OFF) *04_K_JOG_CCW_INPUT_(STATUS_ON) *05_L_JOG_SPEED_SELECT_INPUT_(STATUS_ON) *06_A_TRIGGER_INPUT_(STATUS_ON) *07_A_TRIGGER_INPUT_(STATUS_ON) *08_A_TRIGGER_INPUT_(STATUS_ON)
> 1IN1	Status of only input 1 is reported back *01_C_KILL_INPUT_(STATUS_OFF)

[IN] Input Flag Operator

Type	Programming/Evaluation	Valid	Software Version A
Syntax	see below	Units	None
Range	None	Default	None
Attributes	Buffered, Savable in Sequence	See Also	IN, AND, OR, IF, REPEAT, WHILE, ER, FL

The **IN** operator is used to make a comparison against the current state of the inputs.

```
IN 1 2 3 4 5 6 7 8 9 10 11 12
```

Bit #	Flag (1 = Active, 0 = Inactive)		
1	CW Limit	7	Input 3
2	CW Limit	8	Input 4
3	Home	9	Input 5
4	Registration	10	Input 6
5	Input 1	11	Input 7
6	Input 2	12	Input 8

Command	Description
IF(INXX1XXX1X1XX1)	If the Home, 3, 5, and 8 inputs are active, the statement evaluates true, and INPUT is printed to the screen
1"INPUT	
NIF	

INA CW Limit Status

Type	Status	Valid	Software Version D2
Syntax	<a>INA	Units	None
Range	None	Default	None
Attributes	Buffered, Savable in Sequence	See Also	IN, INB, INC
Response	*A____CW_LIMIT_____(STATUS_ON)		

This command reports the status of the CW Hardware limit.

INB CCW Limit Status

Type	Status	Valid	Software Version D2
Syntax	<a>INB	Units	None
Range	None	Default	None
Attributes	Buffered, Savable in Sequence	See Also	IN, INA, INC
Response	*B____CW_LIMIT_____(STATUS_ON)		

This command reports the status of the CCW Hardware limit.

INC HomeLimit Status

Type	Status	Valid	Software Version D2
Syntax	<a>INC	Units	None
Range	None	Default	None
Attributes	Buffered, Savable in Sequence	See Also	IN, INB, INA
Response	*C____CW_LIMIT_____(STATUS_ON)		

This command reports the status of the Home limit input.

INL Set Active Programmable Input Level

Type	Set-Up	Valid	Software Version A
Syntax	<a>INL<n>	Units	n = active level
Range	1 = high, 0 = low	Default	0
Attributes	Buffered, Savable in Sequence	See Also	IN, OUTL, OSA, OSC
Response	*0_INPUT_SIGNAL_ACTIVE_LOW or *1_INPUT_SIGNAL_ACTIVE_HIGH		

This command configures the voltage level the SX considers to be an active programmable input signal.

INL0: Sets a low level (closed) as an active signal
INL1: Sets a high level (open) as an active signal

This command allows you to use either normally-closed (**INL1**) or normally-open (**INL0**) switches. The maximum voltage for inputs are 24VDC, and it is current limited.

Command	Description
> INL1	Sets a high level (open) as an active signal
> A10	Sets acceleration to 10 rps^2
> V5	Sets velocity to 5 rps
> D5000	Sets distance to 5,000 steps
> 1INL	*1_INPUT_SIGNAL_ACTIVE_HIGH

The SX responds with the active level of the inputs after entering these commands.

INR Enable/Disable Registration Input

Type	Set-Up	Valid	Software Version A
Syntax	<a>INR<n>	Units	n = input state
Range	None	Default	None
Attributes	Buffered, Savable in Sequence	See Also	IN, DIN, REG, FSL

This command allows you to disable the registration input and set it as active or inactive. Registration interrupt occurs only when the registration input is enabled and it makes an inactive to active transition. Disabling the registration input to the active state will never cause a registration interrupt.

The input is enabled or disabled to a set state by providing one of the following values for n.

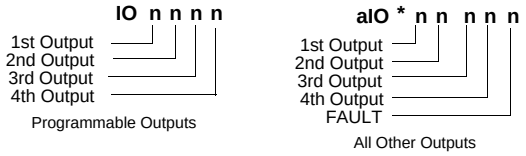
E = Enabled 0 = Disabled—Inactive
1 = Disabled—Active L= Synchronization Input

Command	Description
> INR0	Disables registration, sets registration input inactive (TR or IN commands)
> INRE	Re-enables registration
> INRL	Changes function of the registration input to a synchronization input

IO Immediate Output

Type	Programming	Valid	Software Version A
Syntax	<a>IO<n>	Units	n = output state
Range	0 = low, 1 = high, x = don't care	Default	None
Attributes	Immediate, Never Saved	See Also	O, OUT, OUTL
Response	*nnnnn		

This command immediately sets the output bits configured as programmable outputs as specified in the **IO** command. It does not affect the output state of outputs not configured as programmable. You can only use this command with a host that has an RS-232C interface on-line with the indexer. This command is used to turn a peripheral device on or off during the machine process.



The programmable output bits are configured by the **OUT** command. The voltage level represented as an active state is defined with **OUTL**. The order of the output pattern is programmable outputs from 1 - 4.

Command	Description
> OUT1B	Sets output 1 as moving/not moving output.
> OUT2A	Sets output 2 as a programmable output
> OUT3A	Sets output 3 as a programmable output
> MN	Sets to Preset mode
> A5	Sets acceleration to 5 rps^2
> AD10	Sets deceleration to 10 rps^2
> V10	Sets velocity to 10 rps
> D25000	Sets distance to 25,000 steps
> IO1X	Sets Programmable Output 1 (labeled as Output 2) on and leaves Programmable Output 2 (labeled as Output 3) unaffected
> G	Executes the move (Go)
> IO01	Sets Programmable Output 1 (labeled as Output 2) off and leaves Programmable Output 2 (labeled as Output 3) on

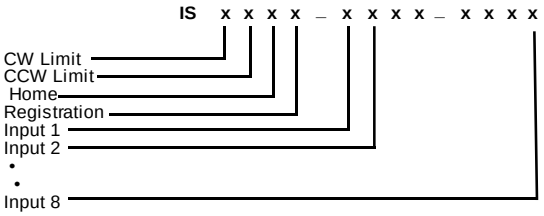
If **IO01** is issued while the motor is moving, O1 turns off, and O2 turns on. This differs from the Output (**O**) command. **O** is buffered and executes after the move is finished.

IS Input Status Report

Type	Status	Valid	Software Version A
Syntax	aIS	Units	None
Range	None	Default	None
Attributes	Immediate, Never Saved	See Also	O, OUT, OUTL, IN, [IN]
Response	*nnnn_nnnn_nnnn (n = 0 or 1)		

This command reports the status of the inputs. The actual function of the input is determined by the **IN** command. The **INL** command sets the active voltage level. This is not a software status report. It reports the actual hardware status of the inputs. **IS** can help you troubleshoot an application.

Helpful Hint: ***1IS*** *Response*



Command	Response
> 1IS	*1000_1000_0000

IV Immediate Velocity

Type	Motion	Valid	Software Version A
Syntax	<a>IVn	Units	n = rps
Range	0.0000 to 50.0000	Default	None
Attributes	Immediate, Never Saved	See Also	MPP, NG, ID

During motion, the velocity at which the motor is traveling can be changed to a new value set by the **IV** command. The SX will accelerate or decelerate to the new velocity and still travel the commanded number of pulses. The **IV** command is used via RS-232C communications to modify the velocity. To alter the velocity at predetermined breakpoints, see the **MPP** command.

Command	Description
> MN	Set to Normal mode
> A10	Set acceleration to 10 rps^2
> AD10	Set deceleration to 10 rps^2
> V5	Set velocity to 5 rps^2
> D500000	Set distance to 500,000 steps
> G	Execute the move (Go)
> IV2	Change velocity to 2 rps—motor decelerates to 2 rps and maintains position setpoint

JA Jog Acceleration

Type	Motion	Valid	Software Version A
Syntax	<a>JA<n>	Units	n = rps^2
Range	0.01 - 999.99	Default	10.0
Attributes	Buffered, Savable in Sequence	See Also	JAD, JVH, JVL, OSE
Response	*JAn		

The **JA** command allows you to specify the acceleration rate that will be applied to reach constant jog velocity. The value is stored in nonvolatile memory.

Command	Description
> JA30	Sets jog acceleration to 30 rps^2
> JVL2	Sets low jog velocity to 2 rps

JAD Jog Deceleration

Type	Motion	Valid	Software Version A
Syntax	<a>JAD<n>	Units	n = rps^2
Range	0.01 - 999.99	Default	10.0
Attributes	Buffered, Savable in Sequence	See Also	JVL, JVH, OSE, JA
Response	*JADn		

The **JAD** command allows you to specify the deceleration rate that will be applied to reach zero velocity from jog velocity. The value is stored in nonvolatile memory.

Command	Description
> JA30	Sets jog acceleration to 30 rps^2
> JAD35	Sets jog deceleration to 35 rps^2
> JVL2	Sets low jog velocity to 2 rps

JVH Jog Velocity High

Type	Set-Up	Valid	Software Version A
Syntax	<a>JVH<n>	Units	n = rps
Range	0.0001 - 50.0000	Default	10.000
Attributes	Buffered, Savable in Sequence	See Also	IN, INL, JA, JAD, JVL, OSE
Response	*JVHn		

The **JVH** defines the speed at which the motor will run when the jog input has been turned on and the high jog velocity is selected. You can define two different jogging velocities:

JVL: Low-speed jog
JVH: High-speed jog

Typically, you will use a high-speed jog to move a load to a desired vicinity. Then you can use the low-speed jog to position your load accurately. You must specify the jog speed (low or high) and the jog inputs (Jog CW or CCW) with the **IN** command. You must specify the acceleration rate of the jog with the **JA** command. Use **JAD** to specify the jog's deceleration rate. Low-jog velocity (**JVL**) is the jog speed default.

Command	Description
> IN1L	Set input 1 as the jog velocity select (high or low)
> IN2J	Set input 2 as Jog input in the positive direction
> IN3K	Set input 3 as Jog input in the negative direction
> JA10	Jog acceleration is set to 10 rps^2
> JAD20	Jog deceleration is set to 20 rps^2
> JVL.1	Set low jog velocity to 0.1 rps
> JVH10	Set high jog velocity to 10 rps
> INL0	Set the active level of the jog input to low

JVL Jog Velocity Low

Type	Set-Up	Valid	Software Version A
Syntax	<a>JVL<n>	Units	n = rps
Range	0.0001 - 50.0000	Default	1.000
Attributes	Buffered, Savable in Sequence	See Also	IN, INL, JA, JAD, JVH, OSE
Response	*JVLn		

This command defines the speed at which the motor will run when the jog input has been turned off and the low-speed jog velocity is selected. You can define two different jogging velocities:

JVL: Low-speed jog
JVH: High-speed jog

Typically, you will use high-speed jog to move the load to the vicinity of where you want to be. You can use the low-speed jog to position your load accurately. You must specify the jog speed (low or high) and the jog inputs (Jog CW or CCW) with the **IN** command. You must specify the acceleration rate of the jog with the **JA** command. The deceleration rate of the jog is specified with the **JAD** command. The default for the jog speed is low-jog velocity (**JVL**).

Command	Description
> IN1L	Set input 1 as the jog velocity select (high or low)
> IN2J	Set input 2 as Jog input in the positive direction
> IN3K	Set input 3 as Jog input in the negative direction
> JA10	Jog acceleration is set to 10 rps^2
> JAD20	Jog deceleration is set to 20 rps^2
> JVL.1	Set low jog velocity to 0.1 rps
> JVH10	Set high jog velocity to 10 rps
> INL0	Set the active level of the jog input to low

K Kill

Type	Motion	Valid	Software Version A
Syntax	<a>K	Units	None
Range	None	Default	None
Attributes	Immediate, Never Saved	See Also	S

The Kill (**K**) command is an emergency stop command and should only be used as such. If the **K** command causes the motor to lose torque (i.e., large loads at high speed), the load could be driven past limit switches and cause damage to the mechanism and possibly to the operation.

The **K** command will also terminate a loop, end a time delay, clear the command buffer, and terminate **RM** mode.

Command	Description
> A5	Sets acceleration to 5 rps^2
> V2	Sets velocity to 2 rps
> MC	Sets mode to continuous
> G	Executes the move (Go)
•	
•	
> K	Stops the motion instantly

L Loop

Valid	Software Version A	Type	Programming
Syntax	<a>L<n>	Units	n = # of loops
Range	0 - 99,999,999	Default	None
Attributes	Buffered, Savable in Sequence	See Also	C, N, U, Y

When you combine the Loop (**L**) command with the End-Of-Loop (**N**) command, all of the commands between **L** and **N** will be repeated the number of times indicated by n. If you enter the **L** command without a value specified for n, or with a 0, subsequent commands will be repeated continuously. Loops may be nested up to 16 levels deep.

The Stop Loop (**Y**) command terminates a loop at the end of the current repetition. The Immediate Pause (**U**) command allows you to temporarily halt loop execution. Use the Continue (**C**) command to resume loop execution.

Command	Description
> A5	Sets acceleration to 5 rps^2
> AD10	Sets deceleration to 10 rps^2
> V10	Sets velocity to 10 rps
> D10000	Sets distance to 10,000 steps
>L5	Loops five times
> G	Executes the move (Go)
> N	Specifies the 10,000-step move to be repeated 5 times

LAD Limit Deceleration

Type	Set-Up	Valid	Software Version A
Syntax	<a>LAD<n>	Units	rps^2
Range	0.01 - 9,999.99	Default	900.00
Attributes	Buffered, Savable in Sequence	See Also	A, AD, LD, SLD
Response	*LADn		

The Limit Deceleration (**LAD**) command allows you to define the deceleration rate that should be used when an end-of-travel limit or software travel limit is encountered. The value is stored in nonvolatile memory. This command is useful if you do not want an abrupt stop upon encountering a limit. However, you should specify a deceleration rate that will stop the load before it can do any damage. Limit switches should be placed so that the load safely decelerates to a stop.

Command	Description
> LAD50	Motion decelerates at 50 rps^2 when it encounters an end-of-travel limit

LD Limit Disable

Type	Set-Up	Valid	Software Version A
Syntax	<a>LD<n>	Units	n = Type of limit enabled
Range	0 - 3 (see below)	Default	0
Attributes	Buffered, Savable in Sequence	See Also	IN, IS, OSA
Response	*0_CW_AND_CCW_ LIMITS ENABLED *1_CCW_ LIMIT ENABLED *2_CW_LIMIT ENABLED *3_NO LIMITS ENABLED		

The Limit Disable (**LD**) command allows you to enable/disable the end-of-travel limit switch protection. If the limit inputs are not installed properly, **LD0** prohibits motion. If you want motion without wiring the limits, use **LD3**. The table below shows the valid n values (which you select) and their functions.

n	**Function**
0	Enable CCW and CW limits (Default)
1	Disable CW limits
2	Disable CCW limits
3	Disable CCW and CW limits

Command	Description
> 1LD0	Enables CW & CCW limits. Motion will occur only if the limit inputs are grounded or connected to limit switches.
> 1LD3	Allows you to make any move, regardless of the limit input state.

LF Line Feed

Type	Programming	Valid	Software Version A
Syntax	aLF	Units	None
Range	None	Default	None
Attributes	Buffered, Savable in Sequence	See Also	CR
Response	[linefeed]		

When you issue the Line Feed (**LF**) command, the system transmits a line feed character (ASCII code 10) over the communications link. This command is useful when you send messages to control a display screen, or to signal a move completion.

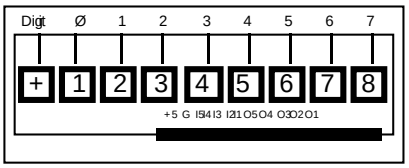
Command	Description
> A5	Sets acceleration to 5 rps^2
> AD20	Sets deceleration to 20 rps^2
> V5	Sets velocity to 5 rps
> D15000	Sets distance to 15,000 steps
> G	Executes the move (Go)
> 1LF	Transmits a line feed character over the communications link after the move is completed

LRD Read Loop Count via Parallel Input/Output

Type	Programming	Valid	Software Version A
Syntax	<a>LRD<c><d><e>	Units	c, d = digit selector, e = scaling factor
Range	c, d = 0 - 7, e = 0 - 7	Default	None
Attributes	Buffered, Savable in Sequence	See Also	DRD, FRD, TRD, VRD, XRD, IN, OUT, STR
Response	None		

This command instructs the SX to read loop count values from Compumotor's TM8 Module. Inputs 1-4 must be configured as data inputs. The inputs must be configured as active low and outputs 1-3 must be configured as strobe outputs to use the TM8 Module.

Helpful Hint: <c> & <d> values represent the TM8 Module digits:



The command syntax allows for digit range selection through the optional <c><d> fields. The <c> field is used to signify the start of the digit range to be read from the TM8 Module. The raw data format is XXXXXXXX, the maximum value is 99,999,999. Any larger number will result in the SX giving a ? Response. The <d> field represents the end of the digit range to be read. The values of these fields can range from 0 to 7 with the <c> field always being less than or equal to the <d> field value.

The <e> field is used to scale the distance value by 10^e. If the <c> and <d> fields are used, the <e> field must be used.

If the <c>, <d>, and <e> fields are not used, the **LRD** command will read all the digits of the TM8 Module. If you are using the TM8 Module, the Output Strobe Delay Time must be set at a value of 10 or greater.

You may use a PLC with the **LRD** command. **LRD** uses a multiplexed I/O scheme. The outputs strobe through a BCD sequence at the Set Strobe Output Delay Time (**STR**) command rate and reads one BCD digit at a time. The outputs and inputs must be configured as in the TM8 Module read case. An **LRD** command will issue the following strobe sequence:

Voltage Level			
01	02	03	Data In (Active Low)
low	low	low	MSD (Digit 1)
high	low	low	Digit 2
low	high	low	Digit 3
high	high	low	Digit 4
low	low	high	Digit 5
high	low	high	Digit 6
low	high	high	Digit 7
high	high	high	LSD (Digit 8)

The PLC, while reading the output strobe, must place the proper data at the SX inputs I1 - I4. I1 is the digit's LSB and I4 is the MSB. Configuring an input as a data valid line would allow the PLC to control the rate at which the outputs will strobe through the above sequence.

Turn the TM8 Module's thumbwheels to display +1 2 3 4 5 6 7 8 and type the following command.

Command	Description
> LRD	Loop count is 12,345,678

MC Mode Continuous

Type	Motion	Valid	Software Version A
Syntax	<a>MC	Units	None
Range	None	Default	None
Attributes	Buffered, Savable in Sequence	See Also	A, AD, MN, MPP, T, TR, V, IV

The Mode Continuous (**MC**) command causes subsequent moves to ignore any distance parameter and move continuously. You can clear the **MC** command with the Mode Normal (**MN**) command.

The SX uses the Acceleration (**A**), Deceleration (**AD**) and Velocity (**V**) commands to perform the motion profile. Using the Time Delay (**T**), Trigger (**TR**), and Velocity (**V**) commands (while in **MPP** mode), you can achieve basic velocity profiling.

Command	Description
> MC	Sets to Continuous mode
> AD5	Sets deceleration to 5 rps^2
> A5	Sets acceleration to 5 rps^2
> V5	Sets velocity to 5 rps
> G	Executes the move (Go)

MN Mode Normal

Type	Motion	Valid	Software Version A
Syntax	<a>MN	Units	None
Range	None	Default	None
Attributes	Buffered, Savable in Sequence	See Also	A, D, MC, MPA, MPI, MPP, V, AD, G

The Mode Normal (**MN**) command sets the Positioning mode to preset. In Normal mode, the motor will move the distance specified with the Distance (**D**) command. To define the complete move profile, you must define Acceleration (**A**), Deceleration (**AD**), Velocity (**V**), and Distance (**D**). The **MN** command is used to change the mode of operation from Mode Continuous (**MC**).

Command	Description
> MN	Set to Positioning Preset mode
> MPI	Set to Incremental Positioning mode
> A5	Set acceleration to 5 rps^2
> AD10	Set deceleration to 10 rps^2
> V5	Set velocity to 5 rps
> D1000	Set distance to 1,000 steps
> G	Executes the move (Go)

MPA Mode Position Absolute

Type	Set-Up	Valid	Software Version A
Syntax	<a>MPA	Units	None
Range	None	Default	None
Attributes	Buffered, Savable in Sequence	See Also	D, MN, MPI, MPP, PZ, SP, FR, PX

This command sets the positioning mode to absolute. In this mode, all move distances are referenced to absolute zero. In Mode Normal (**MN**) and Mode Position Absolute (**MPA**), giving two consecutive go (**G**) commands will cause the motor to only move once, since the motor will have achieved its desired absolute position at the end of the first move.

MPA is useful in applications that require moves to specific locations. You can set the absolute counter to zero by cycling power or issuing a Position Zero (**PZ**) command.

Command	Description
> MN	Set to Normal (preset) mode
> MPA	Set to Position Absolute mode
> A5	Set acceleration to 5 rps^2
> AD10	Sets deceleration to 10 rps^2
> V10	Set velocity to 10 rps
> D25000	Set distance to 25,000 steps.
> G	Motor will move to absolute position 25,000
> D12500	Set distance to 12,500 steps
> G	Motor will move to absolute position 12,500 steps

MPI Mode Position Incremental

Type	Set-Up	Valid	Software Version A
Syntax	<a>MPI	Units	None
Range	None	Default	None
Attributes	Buffered, Savable in Sequence	See Also	D, MN, MPA, MPP

MPI sets the SX to Incremental Positioning mode. In this mode, all move distances specified with the Distance (**D**) command are referenced to the current position. Incremental mode is useful in applications that require repetitive movements (e.g., feed-to-length applications).

Command	Description
> MN	Set to Normal (preset) mode
> MPI	Set to Position Incremental mode
> A5	Sets acceleration to 5 rps^2
> AD10	Sets deceleration to 10 rps^2
> V10	Sets velocity to 10 rps
> D10000	Sets distance of move to 10,000 steps
> G	Move 10,000 steps CW
> G	Move another 10,000 steps CW

MPP Mode Position Profile

TYPE	Set-Up	VALID	Software Version A
SYNTAX	<a>MPP	UNITS	None
RANGE	None	DEFAULT	None
ATTRIBUTES	Buffered,Savable in Sequence	SEE ALSO	DP, FP, NG, IV, ID, MC

MPP sets an execution mode where after a **G** command is executed, subsequent commands are processed to provide complex motion profiles, input and output operations during motion and other commands. A **DP** command correlates events with the motor position and causes command processing to pause until the distance matches the value set by the **DP** command. An **NG** command marks the end of a position profile and turns off **MPP** mode. **MPP** mode is not valid while homing.

Command	Description
> MPI	Set to Incremental Position mode
> A5	Set acceleration to 5 rps^2
> AD10	Set deceleration to 10 rps^2
> V2	Set velocity to 2 rps
> D200000	Set distance to 200,000 steps
> MPP	Set to Position Profile mode
> G	Execute move(Go)
> DP50000	Wait until motor has traveled 50,000 steps
> O11	Turn on outputs 1 and 2
> NG	End position profile
> O00	Turn off outputs 1 and 2 once motion is complete

MR Configure Motor Resolution

Type	Set-up	Version	Software Version A
Syntax	<a>MR<n>	Units	n=resolution selection
Range	1 - 16 or any of the 16 values	Default	25000 (11)
Attributes	Buffered, Savable in Sequence	See Also	ER
Response	*MRn____x_STEPS_PER_REV		

This command configures the number of pulses the drive requires to produce one revolution of the motor. This value determines the step frequency as the velocities and accelerations are entered in units of motor revolutions. a**MR**? will display a list of the following resolutions:

1—200	5—5000	9—20000	13—25600
2—400	6—10000	10—21600	14—36000
3—1000	7—12800	11—25000	15—50000
4—2000	8—18000	12—25400	16—50800

Where n is the current selection and x is the corresponding resolution.

When this command is issued the amplifier is reset and torque to the motor is lost. Changing motor resolution causes the amplifier to turn off for 1 second, the counter is not reset. ***The original motor position will not be maintained.***

Command	Description
> MR7	Configure the motor resolution to 12800 so the drive requires 12,800 pulses to produce one motor revolution

MSP Maximum Synchronization Percent

Type	Set-Up	Valid	Software Version E
Syntax	<a>MSP<n>	Units	n = maximum following %
Range	0.0 — 5000.0	Default	200.0
Attributes	Buffered, Savable in Sequence	See Also	FOL, FOR, FSL, FC
Response	*MSPn		

The **MSP** command is used to set the maximum percentage of the following ratio that the slave will use in the synchronization following mode (**FSL**). Every time a synchronization input is triggered, a new following percentage is calculated using the equations below:

$$\text{Error} = (\#\text{ of encoder steps since last input}) - (\text{FC value})$$

$$\text{Percentage} = \frac{\text{FC value}}{(\text{FC value} - \text{Error})}$$

The calculated percentage approaches infinity as the error approaches the FC value. The **MSP** command was implemented to allow the user to limit the following percentage to a specified maximum.

Note: The **MSP** value should always be greater than the **FOL** value. The synchronization correction will not function correctly if the **MSP** value is less than or equal to the **FOL** value.

MV Maximum Correction Velocity

Type	Motion	Version	Software Version A
Syntax	<a>MV<n>	Units	n = rps
Range	.00001 to 50.0	Default	1
Attributes	Buffered, Savable in Sequence	See Also	FS, DW, ER
Response	*MVn		

This command sets the maximum correction velocity which is the maximum velocity the motor can possibly travel during position maintenance. This will prevent the motor from stalling during a position maintenance correction.

MW Set Motor Waveform

Type	Set-Up	Version	Software Version A
Syntax	<a>MW<n>	Units	n = waveform selection
Range	1—5 or ?	Default	-4% 3rd harmonic
Attributes	Immediate, Never Saved	See Also	None
Response	*MWn____-X%_3RD_HARMONIC		

This command selects the amount of 3rd harmonic correction that is used in the waveform that is output to the motor. A waveform change will be executed immediately. If the waveform is changed while the motor is running you will notice a difference in motor waveforms.

In the response, n is the current selection number and x is the corresponding 3rd harmonic correction percentage (see table below). If you type a**MW**?, the following is displayed:

1____-0%_3RD_HARMONIC
2____-2%_3RD_HARMONIC
3____-4%_3RD_HARMONIC
4____-6%_3RD_HARMONIC
5____-10%_3RD_HARMONIC

Command	Description
> 1MW3	Selects -4% 3rd harmonic correction

N End of Loop

Type	Programming	Version	Software Version A
Syntax	<a>N	Units	None
Range	None	Default	None
Attributes	Buffered, Savable in Sequence	See Also	C, L, PS, Y

This command marks the end of loop. You can use this command in conjunction with the Loop (**L**) command. All buffered commands that you enter between **L** and **N** are executed as many times as the number that you enter following the **L** command.

Command	Description
> PS	Pauses the execution until the indexer receives a C
> MN	Sets to Normal mode
> A5	Sets acceleration to 5 rps^2
> AD10	Sets deceleration to 10 rps^2

> V5	Sets velocity to 5 rps
> D10000	Sets move distance to 10,000 steps
> L5	Loops the Go command five times with the above series of parameters
> G	Executes the move (Go)
> N	Ends the loop
> C	Clears pause and executes all buffered commands

NG End Position Profile

Type	Motion	Version	Software Version A
Syntax	<a>NG	Units	None
Range	None	Default	None
Attributes	Buffered, Savable in Sequence	See Also	MPP, DP

This command turns off Position Profile mode (**MPP**). In Position Profile mode, commands subsequent to **G** are processed until an **NG** command is encountered. Command processing pauses until motion is complete, then command execution resumes.

Command	Description
> MPI	Set to Incremental mode
> A5	Set acceleration to 5 rps^2
> AD10	Set deceleration to 10 rps^2
> V2	Set velocity to 2 rps
> D200000	Set distance to 200000 steps
> MPP	Set to Position Profile mode
> G	Execute the move (Go)
> DP50000	Wait until motor has traveled 50000 steps
> V5	Set velocity to 5 rps—motor accelerates to 5 rps
> O10	Set output #1 on and output #2 off
> NG	End position profile
> 1PR	Report position counter at end of move

NIF End of IF

Type	Programming	Version	Software Version A
Syntax	<a>NIF	Units	None
Range	None	Default	None
Attributes	Buffered, Savable in Sequence	See Also	ELSE, IF, UNTIL, NWHILE

This command marks the end of an **IF** statement. When using the **IF** command, **NIF** must be used to identify the end of the **IF** statement.

```
IF(condition)... commands...  ELSE... commands... NIF
```

Command	Description
> IF(VAR5=7)	If variable #5 = #7, execute next command, if not, execute command following NIF
> GD1	Execute predefined move #1 (only if variable 5 = 7)
> NIF	End of IF statement
> GD2	Execute predefined move #2

NWHILE End of WHILE

Type	Programming	Version	Software Version A
Syntax	<a>NWHILE	Units	None
Range	None	Default	None
Attributes	Buffered, Savable in Sequence	See Also	WHILE, NIF, ELSE, UNTIL

NWHILE marks the end of the **WHILE** statement. **WHILE** is evaluated, and if it is true all commands between the **WHILE** and **NWHILE** commands are executed. **NWHILE** then redirects program flow back to the **WHILE** for another evaluation check. Commands between **WHILE** and **NWHILE** will continue to execute as long as the **WHILE** command is true; when the **WHILE** command is false, program flow jumps to the command after **NWHILE**.

WHILE(condition) . . . commands . . . **NWHILE**

Command	Description
> WHILE(XXXX1)	While input #1 is active, execute commands between WHILE & NWHILE
> GD1	Execute predefined move #1
> T2.0	Time delay of two seconds
> NWHILE	End of WHILE

O Output

Type	Programming	Version	Software Version A
Syntax	<a>O<n>	Units	n = output
Range	0 = inactive (OFF), 1 = active (ON), X = don't care	Default	0
Attributes	Buffered, Savable in Sequence	See Also	IO, OUT, OUTL
Response	*nnnnn		

The **O** command turns the output bits configured as programmable outputs on and off. It does not affect the state of the outputs not configured as programmable. This is used for signaling remote controllers, turning on LEDs, or sounding alarms. The output indicates the motor is in position, about to begin its move, or is at constant velocity, etc. The programmable output bits are configured by the **OUT** command. The voltage level represented as an active state is defined with **OUTL**. The order of the output pattern is programmable outputs from O1-O4 (labeled on the unit).

Command	Description
> OUT1B	Set output #1 as a moving/not moving
> OUT2C	Set output #2 as a sequence in progress
> OUT3A	Set output #3 as a programmable output #1
> OUT4A	Set output #4 as a programmable output #2
> A10	Set acceleration to 10 rps^2
> AD20	Sets deceleration to 20 rps^2
> V5	Sets velocity to 5 rps
> D20000	Set move distance to 20,000 steps
> O01	Set programmable output 1 (labeled output 3) off and output 2 (labeled output 4) on
> G	Executes the move (Go)
> OX0	After the move, turn off programmable output 2 (labeled as output 4)

OFF Off

Type	Programming	Version	Software Version A
Syntax	<a>OFF	Units	None
Range	None	Default	None
Attributes	Immediate, Never Saved	See Also	ON, ST1, ST0, RSE

This command turns off the drive. When you issue an **OFF** command, the fault will go active to indicate that the drive is shut-down. You must issue an **ON** command to command the drive to energize the motor and clear the fault. The contents of the buffer will be cleared when you execute this command. This command is similar to the **ST1** command. However, this command is immediate, and **ST1** is buffered.

Command	Description
> OFF	Turns off the drive

ON On

Type	Programming	Version	Software Version A
Syntax	<a>ON	Units	None
Range	None	Default	None
Attributes	Immediate, Never Saved	See Also	OFF, ST1, ST0

This command turns on the drive and clears any faults. The **ON** command stops motion. It is similar to the **ST0** command. The **ON** command, however, is immediate and **ST0** is buffered.

Command	Description
> ON	Turns on the drive—clears faults

[OR] Boolean OR Operator

Type	Programming/Evaluation	Version	Software Version A
Syntax	None	Units	None
Range	None	Default	None
Attributes	Immediate, Never Saved	See Also	AND, IF, REPEAT, UNTIL

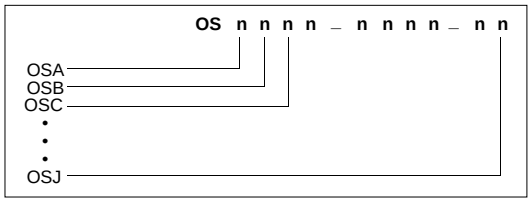
This command is used inside of evaluation statements to **OR** conditions together. For an evaluation statement to evaluate true, only one of the conditions must be satisfied.

Command	Description
VAR1=2	Set VAR1 = 2
VAR2=5	Set VAR2 = 5
IF(VAR1=2_OR_VAR2=10)	This statement would evaluate true
NIF	
IF(VAR1=10_OR_VAR2=20)	This statement would evaluate false
NIF	

OS Function Set-Up Report

Type	Status	Version	Software Version A
Syntax	aOS	Units	None
Range	None	Default	None
Attributes	Buffered, Savable in Sequence	See Also	FS, SS
Response	*OSnnnn_nnnn_nn		

This command reports the status of **OS** command (**OSA**-**OSJ**) settings.



OSA Define Active State of Limit Switch/Sensor

Type	Set-Up	Version	Software Version D2
Syntax	<a>OSAn	Units	n = mode
Range	0 = active low, 1 = active high	Default	0
Attributes	Buffered, Savable in Sequence	See Also	LD, OSC, INL

OSA0: Hardware limit inputs configured for normally closed switches/sensors
OSA1: Hardware limit inputs configured for normally open switches/sensors

This command sets the active state of the hardware limit inputs. In the default setting (**OSA0**) the inputs are configured for switches/sensors normally closed to ground. To configure the inputs for normally open switches/sensors, use **OSA1**.

Note: To obtain motion with the SX the hardware limits must be either active or disabled with the **LD3** command. To activate the limits they must be closed to ground with **OSA0**, or left open with **OSA1** (pulled-up to OPTO1 voltage).

OSB Backup to Home Switch

Type	Set-Up	Version	Software Version A
Syntax	<a>OSBn	Units	n = mode
Range	0 = off, 1 = on	Default	0
Attributes	Buffered, Savable in Sequence	See Also	GH, OSC, OSD, OSG, OSH, OS

OSB0: Do not back up to home switch
OSB1: Back up to home switch

If you do not enable this function, the SX will consider the motor at home, if the home input is active at the end of deceleration after reaching the active edge of the home region. If you enable this function, the SX will decelerate the motor to a stop after reaching the active edge of the home region, and move the motor in the opposite direction of the initial go home move at the **GHF** velocity until the active edge of the home region is encountered. The SX will then consider the motor at home. This occurs regardless of whether or not the home input is active at the end of the deceleration of the initial go home move.

OSC Define Active State of Home Switch

Type	Set-Up	Version	Software Version A
Syntax	<a>OSCn	Units	n = mode
Range	0 = off, 1 = on	Default	0
Attributes	Buffered, Savable in Sequence	See Also	GH, OS, OSB, OSF, OSG, OSH, OSA, INL

OSC0: Active low level state of home input
OSC1: Active high level state of home input

This command sets the active state of the home input. It enables you to use either a normally closed or a normally open switch for homing. **OSC0** requires a low-level signal to activate the home limit input, requiring a normally open switch. **OSC1** requires a high-level signal to activate the home limit input, requiring a normally closed switch.

Command	Description
> OSC1	Sets home input's active state to a high state

OSD Enable Incremental Encoder Z-Channel

Type	Set-Up	Version	Software Version A
Syntax	<a>OSDn	Units	n = mode
Range	0 = disable, 1 = enable	Default	0
Attributes	Buffered, Savable in Sequence	SEE ALSO	GH, GHA, GHF, GHV, GHAD, OSB, OSC, OSJ

OSD0: Disable Z channel function during home
OSD1: Recognizes Z channel after encountering the home

The Encoder Z Channel is used (in conjunction with a load-activated switch connected to home limit) to determine the home

position. The switch determines the home region. The Z-Channel determines the exact position in that region that will be used as the home reference. If **OSD1** is selected **OSB1** must also be selected.

Command	Description
> OSD0	Disable Z channel function

OSE Jog Enable

Type	Set-Up	Version	Software Version A
Syntax	<a>OSEn	Units	n = mode
Range	0 = disable, 1 = enable	Default	0
Attributes	Buffered, Savable in Sequence	SEE ALSO	IN, JVL, JA, JVH, JAD, INL

The jogging functions are configured with the **IN** command, and are enabled with the **OSE** command.

OSE0: Jog Disable
OSE1: Jog Enable

OSF Acknowledge STOP & KILL Inputs On Power-up

Type	Set-Up	Version	Software Version E
Syntax	<a>OSFn	Units	n = mode
Range	0 = off, 1 = on	Default	0
Attributes	Buffered, Savable in Sequence	See Also	IN, JVL, JA, JVH, JAD, INL

The **OSF** command determines whether or not active inputs will be responded to when power is applied to the SX/SXF.

OSF0: Programmable input states (e.g., **STOP, KILL, RESET**) are ignored during power-up unless specific commands (e.g., **IF**, **TRIG**, and other conditional commands) are used in the sequence.

OSF1: All programmable input states are recognized (and the associated function is executed) during power-up.

Command	Description
> 1OSF1	Responds to inputs when power is cycled. If an input is configured for **STOP** or **KILL**, no motion will be permitted if the input is active on power-up.

OSG Final Homing Direction

Type	Set-Up	Version	Software Version A
Syntax	<a>OSEn	Units	n = mode
Range	0 = CW, 1 = CCW	Default	0
Attributes	Buffered, Savable in Sequence	See Also	OSH, OSB

OSG0: Sets the final home approach direction to be CW
OSG1: Sets the final home approach direction to be CCW

OSG allows you to approach home from any direction yet stop at home repeatedly on the same edge of the switch.

OSH Reference Edge of Home Switch

Type	Set-Up	Version	Software Version A
Syntax	<a>OSHn	Units	n = mode
Range	0 = off, 1 = on	Default	0
Attributes	Buffered, Savable in Sequence	See Also	OSG, OSB

OSH0: Selects CW side of home signal as the *edge* where the final approach stops
OSH1: Selects CCW side of home signal as the *edge* where the final approach stops

The CW edge of the home switch is defined as the first switch transition seen by the SX when traveling from the CW limit in the CCW direction. If n = 1, the CCW edge of the home switch will be referenced as the home position. The CCW edge of the home switch is the first switch transition seen by the SX when traveling off of the CCW limit in the CW direction.

OSI Save Sequence Scan Mode On Stop

Type	Set-Up	Version	Software Version A
Syntax	<a>OSIn	Units	n = mode
Range	0 = off, 1 = on	Default	0
Attributes	Buffered, Savable in Sequence	See Also	SSJ, XQ

OSI1 preserves sequence scan mode (**SSJ1**) during and after execution of a stop command. If **OSI0** is issued, the **SSJ** bit will be cleared after execution of a stop. Use **OSI1** carefully to ensure that no sequences are unintentionally executed after a stop. The **XQ** command can help with this problem.

OSJ Configure Z-Channel Search Mode

Type	Set-Up	Version	Software Version A
Syntax	<a>OSJn	Units	n = mode
Range	0 or 1	Default	0
Attributes	Buffered, Savable in Sequence	See Also	OSD

OSJ determines how the SX will search for the Z-pulse during a go home move. If **OSJ0** is set selected, the SX will search for a Z-pulse for only one revolution. Setting **OSJ1** causes the SX to search for the Z-pulse beyond one revolution.

Command	Description
> OSJ1	Search for Z-pulse until received
> OSJ0	Search for Z-pulse one revolution only

OUT Output Functions

Type	Programming	Version	Software Version A
Syntax	<a>OUT<n><x>	Units	n = bits, x = functions
Range	n = 1 - 4, x = A - Z	Default	x = A
Attributes	Buffered, Savable in Sequence	See Also	IN, OUTL, OUTP, O
Response	See Example		

This command sets the functions for each of the four outputs. All the outputs can be configured for different functions. The factory setting for all four outputs are programmable output bits. The following table lists the functions available for each output bit.

A	*Programmable Output* - Used with **O** and **IO** commands
B	*Moving/Not Moving* - On when motion is occurring
C	*Sequence in Progress* - On when a sequence is being run
D	*At Limits* - On when the motor reaches a software or hardware limit
E	*At Position Zero* - On when the absolute position counter is 0
F	*Fault* - On when a fault has occurred (see **RSE** command)
G	*Reserved*
H	*Shutdown Command* - Indicates when you turn off the drive using OFF or ST1
I	*Reserved*
J	*Strobe* - Turns on to load bytes of parallel data. The SX is ready for data
K	*Command Error* - On when RS-232C or BCD input has invalid data
L	*Position Error Fault* - On when a stall condition is detected
M	*Reserved*
N	*CW Software Limit Reached* - CW software limit set with SL command has activated
O	*Reserved*
P	*CCW Software limit reached* - CCW software limit set with **SL** has activated
Q	*Reserved*
R	*CW Hardware limit activated*
S	*CCW hardware limit activated*
T	*Output based on position* - Goes on when a condition specified by OUTP exists
U	*Pulse Output* - A square wave is output based on the PU and PUL commands
V	*No Function*
W	*No Function*
X	*No Function*
Y	*No Function*
Z	*No Function*
OTHERS	*No Function*

Outputs 1 - 4 can be configured as any one of the functions described above.
The **OUT** command also reports the status of each output bit.

aOUT1 *1_A_PROGRAMMABLE_OUTPUT_(STATUS_ON)

By not specifying the output bit number, the SX reports the status of all the output bits.

```
aOUT
*1_A_PROGRAMMABLE_OUTPUT_(STATUS_ON)
*2_A_PROGRAMMABLE_OUTPUT_(STATUS_ON)
*3_A_PROGRAMMABLE_OUTPUT_(STATUS_ON)
*4_A_PROGRAMMABLE_OUTPUT_(STATUS_ON)
*5_F_FAULT_INDICATOR_(STATUS_OFF)
```

Command	Description
> OUT3E	Sets output #3 as an at position zero indicator

OUTL Set Active Output Level

Type	Set-Up	Version	Software Version A
Syntax	<a>OUTL<n>	Units	n = active level
Range	0 = low, 1 = high	Default	0
Attributes	Buffered, Savable in Sequence	See Also	INL, OUT, O
Response	*0_OUTPUT_SIGNAL_LEVEL_ACTIVE_LOW *0_OUTPUT_SIGNAL_LEVEL_ACTIVE_HIGH		

This command configures the voltage level that the SX considers to be an active output signal.

OUTL0: The output transistor is normally not conducting when in the off state (0V when active)
OUTL1: The output transistor is normally conducting when in the off state (5-24V when active)

Output #'s 1 through 4 are open collector outputs. Therefore, you must supply power (5-24VDC) to run the outputs. The output is able to sink up to 30 mA of current. You must configure the outputs using the **OUT** command if you wish to use the outputs for anything other than programmable outputs.

This command will not affect the fault output's active level.

Command	Description
> OUTL0	Set a low-level (0V) active signal
> OUT1A	Set output 1 as a programmable output
> 1OUTL	*0_OUTPUT_SIGNAL_LEVEL_ACTIVE_LOW

OUTP — Output on Position

Type	Set-Up	Version	Software Version A
Syntax	<a>OUTP<x><n>,<t>	Units	x = mode, n = steps, t = ms
Range	x = A or I, n = 838,860,800, t = 0 - 5000	Default	A, 0, 5
Attributes	Buffered, Savable in Sequence	See Also	OUT
Response	*OUT_ON_X_POSITION_N *POSITION_OUTPUT_ON_TIME_T_MILLISECONDS		

An output can be configured to activate based on the position with the **OUT** command. The comparison position is entered as either an incremental or absolute position.

OUTP is configured as **OUTP<x><n>,<t>** where x is **A** for Absolute or **I** for Incremental mode, n is distance, and **t** is the time in milliseconds the output is active (only in Incremental mode).

In *Absolute mode*, the output turns on when the current position is greater than the entered distance. Time is ignored in this mode. In *Incremental mode*, the output turns on each time the distance is traveled and stays activated for the entered time. The sign for the distance will convert to a plus sign as the distance is an absolute value.

*Helpful Hint: To review the **OUTP** command's algorithm*

The accuracy of the SX outputs based on **OUTP** is ±1 ms. For example, if a motor is commanded to move at 250,000 pulses/second, the output could actually turn on at ±250 pulses from the commanded position (0.001 seconds • 250,000 pulses/sec = 250 pulses). The motor's speed determines the possible position error. During acceleration and deceleration, motor speed changes, hence repeatability is difficult to predict.

Command	Description
> OUT2T	Set Output #2 to activate based on position
> OUTPA1000	When absolute position is greater than +1,000, the output turns on

Command	Description
> OUT2T	Set Output #2 to activate based on position
> OUTPI5000,500	The output will activate for 500 ms every 5,000 steps

PF — Follower Position Report

Type	Status	Version	Software Version B
Syntax	aPF	Units	n = None
Range	±2,147,483,647	Default	None
Attributes	Buffered, Savable in Sequence	See Also	PFZ, PR
Response	*n (n = Encoder steps ±2,147,483,647)		

Reports the absolute position of the encoder used for the following function. The counter is in respect to the power-up position or the last time the **PFZ** command was issued.

PFZ — Set Follower Counter to Zero

Type	Set-Up	Version	Software Version B
Syntax	<a>PFZ	Units	None
Range	None	Default	None
Attributes	Buffered, Savable in Sequence	See Also	PF, PZ

This command sets the absolute position of the following encoder to zero.

Command	Description
> PFZ	Sets following encoder count to zero
> 1PF	Reports following encoder position (response = *+0000000000)

PHZ Zero Motor Phase

Type	Programming	Version	Software Version A
SYNTAX	<a>PHZ	Units	None
Range	None	Default	None
Attributes	Buffered, Savable in Sequence	See Also	Z

This command sets the amplifier board to its power-up state, with one phase on and one phase off.

Command	Description
> PHZ	Reset the amplifier to the power-up state

[POS] Position Counter Comparison

Type	Programming/Evaluation	Version	Software Version F
Syntax	See below	Units	None
Range	None	Default	None
Attributes	Buffered, Savable in Sequence	See Also	FEP, ER

This evaluation operator is the present value of the position counter.

Command	Description
IF(POS>25000) 1"POSITION NIF	If the present value of the position counter is greater than 25000 counts, print **POSITION**

PR Absolute Position Report

Type	Status	Version	Software Version A
Syntax	aPR	Units	None
Range	None	Default	None
Attributes	Buffered, Savable in Sequence	See Also	D, MN, MPI, MPA, PZ, SP, PX
Response	*n (n = Motor steps 2,147,483,647) or *snnnnnnnnnn[cr]		

Reports motor position with respect to power-up position or last **SP** or **PZ** command. The absolute position counter can track up to 2,147,483,647 steps. If the counter is overrun in the Relative Position mode (by running the motor continuously for long periods of time, 24 hours at 20 rps and 5,000 steps per rev), the absolute position will be invalid. The response is in motor steps when in **FSB0** and in encoder steps when in **FSB1** and is the actual encoder position.

Command	Description
> PZ	Resets the absolute counter to zero
> LD3	Disable both CW & CCW limits
> A10	Set acceleration to 10 rps^2
> AD20	Sets deceleration to 20 rps^2
> V5	Set velocity to 5 rps
> D5000	Set move distance to 5,000 steps
> G	Executes the move (Go)
> 1PR	Request absolute position report—Response = *+0000005000

PS Pause

Type	Motion	Version	Software Version A
SYNTAX	<a>PS	Units	None
Range	None	Default	None
Attributes	Buffered, Savable in Sequence	See Also	C, IN

This command pauses execution of a command string or sequence following the Pause (**PS**) command until the SX receives a Continue (**C**) command. This command is useful if you need to enter a complete string of commands before you can execute your other commands.

*Helpful Hint: To pause and continue a preset move without entering a **C** command over the RS232C interface, try the following: define a Stop input, **INnD**, and a Pause/Continue input, **INnF**. Enter **SSH1** to Save the Command Buffer on Stop, and **SSL1** to Resume Execution. In this manner, the stop input will stop the motion when the stop switch transitions to active, and the pause/continue input will resume the movement when the pause/continue input transitions to deactive. As an alternative, these two inputs may be wired together into one switch, and then motion will pause when the switch is activated, and resume when the switch is deactivated. This will not work with a Mode Continuous move.*

This command is used for interactive tests and in synchronizing multiple indexers that have long command strings.

Command	Description
> PS	Pauses execution until the indexer receives a C
> A5	Sets acceleration to 5 rps^2
> AD5	Sets deceleration to 5 rps^2
> V5	Sets velocity to 5 rps

> D25000	Sets move distance to 25,000 steps
> G	Executes the move (Go)
> T2	Delays the next move for 2 seconds
> G	Executes the move (Go)
> C	Continues Execution

PU Configure Square Wave Output

Type	Programming	Version	Software Version A
Syntax	<a>PU<n><m-1>	Units	n = pulses, m = ms
Range	n = 0 to 2147483647, m = 0 to 5000	Default	n = 0, m = 0
Attributes	Buffered, Savable in Sequence	See Also	PUL, OUT
Response	*OUTPUT_PULSE_NUMBER_n[cr] *OUTPUT_PULSE_RATE__m_MILLISECONDS		

You can configure an output with the **OUT** command to produce a square wave output. The **PU** command sets the square wave characteristics. The first field determines the number of pulses and the second field determines the time in milliseconds (ms) the pulse is active. A **PU5,200** command configures a square wave of 5 pulses, which is on for 200 ms and off for 200 ms. The **PUL** command initiates the square wave.

Command	Description
> OUT2U	Set output 2 as the square wave output
> PU1000,2	Set a square wave of 1,000 pulses with a 2 ms on/off time
> PUL	Initiate square wave on output 2

PUL Activate Square Wave Output

Type	Programming	Version	Software Version A
Syntax	<a>PUL	Units	None
Range	None	Default	None
Attributes	Buffered, Savable in Sequence	See Also	OUT, PU

An output can be configured with the **OUT** command to produce a square wave output whose characteristics are set with the **PU** command. The **PUL** command initiates the square wave output. The output can possibly be used to control another axis of motion.

Command	Description
> OUT5U	Set output 5 as the square wave output
> PU20,5	Set a square wave of 20 pulses with a 5-ms on/off time
> PUL	Initiate square wave on output 5

PX Report Encoder Position

Type	Status	Version	Software Version A
Syntax	aPX	Units	None
Range	None	Default	None
Attributes	Buffered, Savable in Sequence	See Also	PR, PZ, D, MN, MPI, MPA
Response	*n (n = encoder steps ±2147483647)		

This command transmits a value indicating the absolute position of the incremental or absolute encoder. Whether in Motor Step mode or Encoder Step mode, the position is reported in encoder steps.

PZ Set Absolute Counter to Zero

Type	Set-Up	Version	Software Version A
Syntax	<a>PZ	Units	None
Range	None	Default	None
Attributes	Buffered, Savable in Sequence	See Also	PR, PX, SP, D, MN, MPI, MPA

This command sets the absolute position counter (**PR**) and the absolute encoder position counter (**PX**) to zero.

Command	Description
> MPA	Set to Absolute Position mode
> A10	Set acceleration to 10 rps^2
> AD20	Sets deceleration to 20 rps^2
> V5	Set velocity to 5 rps
> D2500	Set move distance to 2500 steps
> G	Executes the move (Go)
> 1PR	Report Absolute Position (Response = *+0000002500)
> PZ	Zero the absolute counter
> 1PR	Report absolute position (Response = *+0000000000)

" Quote

Type	Programming	Version	Software Version A
SYNTAX	a"x	Units	x = ASCII or number
Range	ASCII33—ASCII126	Default	None
Attributes	Buffered, Savable in Sequence	See Also	CR, LF
Response	x		

Any characters entered after the quotation marks (**"**) will be transmitted, exactly as they were entered, over the RS-232C link. A space entered by the space bar indicates the end of the command. A space is always sent after the last character in the string. This command is used during buffered moves or sequences, or to command other devices to move. The ASCII range of characters accepted by the command is 33 - 126 (decimal). Each quote command may be followed by a maximum of 50 characters.

Command	Description
> MN	Set to Normal mode (Preset Moves)
> A10	Set acceleration to 10 rps^2
> AD10	Set deceleration to 10 rps^2
> V5	Set velocity to 5 rps
> D12500	Set distance to 12,500 steps
> G	Executes the move (Go)
> 1"MOVE_DONE	After motor finished the move, the Compumotor SX will send the message **MOVE_DONE** out from the RS-232C port

With two SX's in a daisy chain the following commands will execute a move on axis #1. After completion of the move have axis #1 command a move on axis #2 using a quote.

Command	Description
> MN	Set to Normal mode (Preset Moves)
> A10	Set acceleration to 10 rps^2
> AD10	Set deceleration to 10 rps^2
> V5	Set velocity to 5 rps
> D12500	Set distance to 12,500 steps
> 1G	Executes the move (Go) on Axis #1
> 1"2XG1	Once the move is done, GOTO Sequence 1 is commanded on a unit with device address 2 from Axis #1

Q0 Exit Velocity Profiling Mode

Type	Set-Up	Version	Software Version A
Syntax	<a>Q0	Units	None
Range	None	Default	None
Attributes	Buffered, Savable in Sequence	See Also	Q1, RM

The **Q0** command exits Velocity Profiling mode. Motion will stop when **Q0** is issued.

Q1 Enter Velocity Profiling Mode

Type	Set-Up	Version	Software Version A
Syntax	<a>Q1	Units	None
Range	None	Default	None
Attributes	Buffered, Savable in Sequence	See Also	Q0, RM, K

The **Q1** command enters the SX into Velocity Profiling mode. The command buffer is cleared and any motion is killed. Subsequent **RM** commands will cause an immediate change in motor velocity. Use **Q0** to exit this mode.

Command	Description
> Q1	Enter Velocity Profiling mode
> RM0100	Go to RM velocity of 14,061 steps per second
> RM0500	Go to RM velocity of 70,313 steps per second
> RM1000	Go to RM velocity of 225,005 steps per second
> RM0500	Go to RM velocity of 70,313 steps per second
> RM0100	Go to RM velocity of 14,061 steps per second
> Q0	Exit Velocity Profiling mode

R Report SX Status

Type	Status	Version	Software Version A
Syntax	aR	Units	None
Range	None	Default	None
Attributes	Immediate, Never Saved	See Also	RA, RB, RG, RS, RSE
Response	*n (n = R, S, B, C, Z, K, W)		

The Request SX Status (**R**) command can be used to indicate the general status of the SX. Possible responses are:

Response Character	Definition
*R	Ready
*S	Ready, Attention Needed
*B	Busy

*C	Busy, Attention Needed
*Z	Busy, Auto Run mode
*K	Kill Input Active
*W	Waiting for SSC Window

The following conditions cause a response indicating that the SX is busy:

- ❏ Performing a preset move
- ❏ Waiting on a Trigger
- ❏ Performing a continuous move
- ❏ In Jog mode
- ❏ A time delay in progress (**T** command)
- ❏ Going Home
- ❏ In RM mode
- ❏ In Power-on sequence mode
- ❏ Paused
- ❏ Running a sequence
- ❏ Executing a loop

The following conditions will cause an error response:

- ❏ A feedback error condition exists
- ❏ Limit has been encountered
- ❏ Go home failed
- ❏ Sequence execution was unsuccessful

When attention is required, more details on the error condition are available with the **RA**, **RB**, **RS**, **RG**, or **RSE** commands. It is not recommended that you use **R** in tight polling loops that may result in microprocessor overload. Inserting time delays can alleviate this problem. **R** is not intended to determine if a move is complete. Rather, it should be used after the move is complete to determine if there are errors or faults. Use a buffered status request command or a programmable output to indicate move completion.

Command	Response
> 1R	*R (SX ready to accept a command—no error conditions require attention)

RA Limit Switch Status Report

Type	Status	Version	Software Version A
Syntax	aRA	Units	None
Range	None	Default	None
Attributes	Immediate, Never Saved	See Also	IS, INA, INB
Response	*x (x = characters @, A-O)		

RA responds with the status of the end-of-travel limits during the last move as well as the present condition. This is done by responding with one of 16 characters representing the conditions listed below.

Response Character	Last Move Ended By		Limit Switch Active	
	CW Limit	CCW Limit	CW Limit	CCW Limit
*@	NO	NO	NO	NO
*A	YES	NO	NO	NO
*B	NO	YES	NO	NO
*C	YES	YES	NO	NO
*D	NO	NO	YES	NO
*E	YES	NO	YES	NO
*F	NO	YES	YES	NO
*G	YES	YES	YES	NO
*H	NO	NO	NO	YES
*I	YES	NO	NO	YES
*J	NO	YES	NO	YES
*K	YES	YES	NO	YES
*L	NO	NO	YES	YES
*M	YES	NO	YES	YES
*N	NO	YES	YES	YES
*O	YES	YES	YES	YES

RA is useful when the motor will not move in either or both directions. The report back will indicate whether or not the last move was terminated by one or both end-of-travel limits.

RB Loop, Pause, Shutdown, Trigger Status Report

Type	Status	Version	Software Version A
Syntax	aRB	Units	None
Range	None	Default	None
Attributes	Immediate, Never Saved	See Also	L, PS, R, RA, ST, TR
Response	*x (x = @, A-O)		

This command reports the status of four functions within the SX:

- ❏ Loop in Progress
- ❏ Pause in Progress
- ❏ Trigger Wait Active
- ❏ Shutdown Active.

Response Character	Loop Active	Pause Active	Shutdown Active	Trigger Active
*@	NO	NO	NO	NO
*A	YES	NO	NO	NO
*B	NO	YES	NO	NO
*C	YES	YES	NO	NO
*D	NO	NO	YES	NO
*E	YES	NO	YES	NO
*F	NO	YES	YES	NO
*G	YES	YES	YES	NO
*H	NO	NO	NO	YES
*I	YES	NO	NO	YES
*J	NO	YES	NO	YES
*K	YES	YES	NO	YES
*L	NO	NO	YES	YES
*M	YES	NO	YES	YES
*N	NO	YES	YES	YES
*O	YES	YES	YES	YES

Command	Response
> 1RB	*A The SX is currently executing a loop

REG Configure Registration Moves

Type	Motion	Version	Software Version A
Syntax	<a>REG1,Am,ADm,Vm (or FOLm),Dm	Units	None
Range	None	Default	None
Attributes	Buffered, Savable in Sequence	See Also	IN, INR
Response	*REG1,Am,ADm,Vm(or FOLm),Dm		

The **REG** command defines registration move n with parameters **A** (Acceleration), **AD** (Deceleration), **V** (Velocity), or **FOL** (Following) and **D** (Distance). One registration move may be entered (the values are stored in nonvolatile memory). The move is calculated for immediate execution upon receipt of an active registration input. Only the registration input can produce registration moves.

Command	Description
> REG1,A5,AD10,V5,D200000	Define registration move

REPEAT Repeat

Type	Programming	Version	Software Version A
Syntax	<a>REPEAT	Units	None
Range	None	Default	None
Attributes	Buffered, Savable in Sequence	See Also	UNTIL, WHILE, IF, [ER], [IN], [FL], AND, OR

The **REPEAT** command in conjunction with the **UNTIL** command provides a means of conditional program flow. The **REPEAT** command marks the beginning of the conditional statement. The commands after the **REPEAT** are executed until the **UNTIL** statement is encountered. The **UNTIL** command is then evaluated. If it is false, the program flow is redirected to the **REPEAT** command, otherwise a true evaluation causes the command after the **UNTIL** command to execute as the **REPEAT . . . UNTIL** loop is exited. The commands between the **REPEAT** and **UNTIL** are always executed at least once. Up to 16 levels of **REPEAT** commands may be nested.

REPEAT . . . commands . . . **UNTIL**(condition)

*Helpful Hint: The input conditions will not be evaluated until the **UNTIL** command is evaluated. Hence, the statements/moves between **REPEAT** & **UNTIL** will not be interrupted.*

Command	Description
> REPEAT	Repeat command
> GD1	Execute predefined move #1
> T2	Delay 2 seconds
> UNTIL(INXXXX1)	If input #1 active, do next command, if not, execute GD1 again

RG — Go Home Status Report

Type	Status	Version	Software Version A
Syntax	aRG	Units	None
Range	None	Default	None
Attributes	Immediate, Never Saved	See Also	GH, R, RA, RB
Response	*x (x = @, A)		

Go Home Status (**RG**) request responds with either *@ or *A indicating success or failure of last go home attempt.

Response	Successful
*@	NO
*A	YES

RIFS — Return to Factory Settings

Type	Set-Up	Version	Software Version A
Syntax	<a>RIFS	Units	None
Range	None	Default	None
Attributes	Immediate, Never Saved	See Also	DR

RIFS will cause the indexer parameters to return to factory default settings.

RM — Rate Multiplier in Velocity Streaming Mode

Type	Motion	Version	Software Version A
Syntax	<a>RMn	Units	n = digits
Range	00000 to FFFFF	Default	None
Attributes	Buffered, Savable in Sequence	See Also	Q0, Q1, IV

The **RM** command followed by 5 hexadecimal digits represents a velocity. The velocity change is almost immediate. There is no acceleration/deceleration ramp between velocities. A limit switch-closure stops movement in Velocity Profiling mode, but does not exit the SX from Velocity Streaming mode. Profiling (**RM**) mode is unidirectional. The direction will be the last activated direction either from an actual move or from a **D** or **H** command. Bidirectional moves can be made in this mode by returning to velocity zero, switching off **RM** mode, changing direction, and re-enabling **RM** mode. This extra overhead should be acceptable given the need to return to velocity zero when changing directions in real situations. The velocity value is determined with the following formula:

steps/sec = (decimal equivalent of RM value) • (54.93164)

The value of 54.93164 is replaced by a different constant when using a different resolution (see **MR** command).

Command	Description
> Q1	Enter Velocity Streaming mode
> RM00100	Go to velocity of 14,061 steps/sec
> RM00500	Go to velocity of 70,313 steps/sec
> T1	Delay for one second
> RM00100	Go to velocity of 714.061 steps/sec
> RM00000	Go to velocity of 0 steps/sec
> Q0	Exit Velocity Streaming mode

Velocity Profiling Mode

Situations requiring non-linear accelerations may use the **Q0**, **Q1** and **RM** commands. **Q1** is used to enter Velocity Profiling mode, and **Q0** is used to exit. While in this mode, the **RM** command generates velocity values that are implemented during the move (immediately). The **RM** command must be sent to the SX at the time the change in velocity is required. This creates a stair-step effect in velocity change. By implementing many small instantaneous velocity changes, a smooth, non-linear acceleration ramp can be achieved.

RS — Report Status of Sequence Execution

Type	Status	Version	Software Version A
Syntax	aRS	Units	None
Range	None	Default	None
Attributes	Immediate, Never Saved	See Also	XR, XRP, XDIR, L, N
Response	*x (x = characters @, A, B, C, D)		

Bit 1: Sequence Ended

Response Character	Sequence Started	Sequence Ended	Bad Loop
*@	NO	NO	NO
*A	YES	NO	NO
*B	NO	YES	NO
*C	YES	YES	NO
*D	None	None	YES

Whenever a sequence is started, the sequence start bit is set and the sequence end bit is cleared (this only occurs if the sequence is valid and is actually run). Whenever a sequence is ended, the start bit is cleared and the end bit is set. Any abrupt move termination (e.g., limit activation), or a **K** or S command clears both bits.

***D** is reported when there is an unbalanced number of loops and loop terminators inside a sequence. Starting a loop in one sequence and terminating it in another sequence is not allowed. Nested loops require complete closure before execution will begin.

Sequence Started is true when an **XR**, **XRP** or a power-up successfully starts a sequence. It is false when a **STOP** or a **KILL** command is received, or limits are hit.

Sequence Ended is *true* when an **XT** is encountered, when a **STOP** or **KILL** is executed, or when an end-of-travel limit is encountered. Sequence Ended is *false* when a sequence is successfully started.

RSE Report System Errors

Type	Status	Version	Software Version A
Syntax	aRSE	Units	None
Range	None	Default	None
Attributes	Immediate, Never Saved	See Also	R, RA, RB, ST1, STØ, IN, XC
Response	See below		

This command reports error or warning conditions. Use ***RSE*** to start troubleshooting the unit when a fault condition exists. The possible messages are:

```
*NO_ERRORS
*INDEXER_DISABLED_BY_CW_HARDWARE_LIMIT
*INDEXER_DISABLED_BY_CCW_HARDWARE_LIMIT
*INDEXER_DISABLED_BY_CW_SOFTWARE_LIMIT
*INDEXER_DISABLED_BY_CCW_SOFTWARE_LIMIT
*ABSOLUTE_ENCODER_FAULT
*ERROR__BATTERY_BACKUP_RAM_CORRUPTED
*SX_DISABLED_BY_ST1_OR_OFF_COMMAND_BROWNOUT_CONDITION
*SX_DISABLED_BY_ST1_OR_OFF_COMMAND_MOTOR_FAULT
*SX_DISABLED_BY_ST1_OR_OFF_COMMAND_USER_INPUT_FAULT
*SX_BUSY_IN_AUTORUN_MODE
*SX_DISABLED_BY_ST1_OR_OFF_COMMAND
*SX_DISABLED_BY_ST1_OR_OFF_COMMAND_AMPLIFIER_OVERHEATING
*INITIALIZING
*SX_DISABLED_BY_ACTIVE_KILL_INPUT
```

Command	Response
> ON	
> 1RSE	*NO_ERRORS

RSIN Set Variable Interactively

Type	Programming	Version	Software Version A
SYNTAX	<a>VARn = RSIN	Units	n = variable number
Range	1 - 50	Default	None
Attributes	Buffered, Savable in Sequence	See Also	VAR

The **RSIN** command is used in conjunction with the variables to allow the variables to be loaded with data from the RS-232C port during sequence execution. When the command is executed, the data is transmitted to the SX with **!nnnnnnnnnn.nnnnn**. Command processing pauses with the **RSIN** command and resumes when the data is transmitted to the SX.

VARn=RSIN data is to be loaded into VARn
!nnnnnnnnnn.nnnnn = load data into VARn

Command	Description
> XE1	Erase sequence #1
> XD1	Define sequence #1
1"ENTER_DATA	Transmit message
1CR	Transmit carriage return
1LF	Transmit linefeed
VAR2=RSIN	Data is to be entered for variable #2
XT	End defining sequence #1
> XR1	Execute sequence #1
> ENTER_DATA	Message transmitted
> !12.34	Variable 2 gets loaded with 12.34
>1VAR2	*+00000000000012.34000

RV Revision Level

Type	Status	Version	Software Version A
Syntax	aRV	Units	None
Range	None	Default	None
Attributes	Immediate, Never Saved	See Also	RVV
Response	See Below		

The Revision (**RV**) command responds with the software part number and its revision levels. The part number (92-011640-01) identifies the installed software. The suffixes (xx) identify the revision level. Record this information, it is required if you consult Compumotor's Applications Department.

Command	Response
> 1RV	*92-011640-01xx

RVV Report Revision Verbose

Type	Status	Version	Software Version A
Syntax	<n>RVV	Units	None
Range	None	Default	None
Attributes	Buffered, Savable in Sequence	See Also	RV, FSM
Response	*92-011640/012039-01XX ABSOLUTE_ENCODER_INTERFACE_(NOT)PRESENT		

The **RVV** responds with the current software revision number, and whether the SX is equipped with the Absolute Encoder Option.

Note: In order for this command to detect the absolute encoder option, ***FSM1*** *must be set.*

S Stop

Type	Motion	Version	Software Version A
Syntax	<a>S	Units	None
Range	None	Default	None
Attributes	Immediate, Never Saved	See Also	AD, C, K, Q0, SSH, SSL, STOP

This command decelerates motion to a stop using the last defined Deceleration (**AD**) command. This command normally clears any remaining commands in the command buffer, unless prevented from doing so by the Clear/Save the Command Buffer On Stop (**SSH1**) command. When the **SSH1** command is present, the S command stops only the current move. The SX then executes the next command in the buffer. The Stop (**S**) command does not stop motion in Velocity Streaming or Rate Multiplier (**RM**) mode. If you are in the **RM** mode, issue an Exit Velocity Profiling Mode (**Q0**) command to stop the motion.

If the **SSL** command is enabled, issuing a Continue (**C**) command after a **S** command will resume execution by completing the interrupted move and continuing sequence or buffer execution.

Command	Description
> MC	Sets to Continuous mode
> AD5	Sets deceleration to 5 rps^2
> A1	Sets acceleration to 1 rps^2
> V10	Sets velocity to 10 rps
> G	Executes the move (Go)
> S	Stops (motor decelerates) to 0 rps at 5 rps^2

SCR Set Standby Current Reduction

Type	Set-Up	Version	Software Version A
Syntax	<a>SCR<n>	Units	n = standby current level
Range	1-4 or ?	Default	1
Attributes	Buffered, Savable in Sequence	SEE ALSO	None
Response	*SCRn_X%_CURRENT_REDUCTION_ON_STANDBY		

This command selects the amount of current in the motor that will be reduced when the motor is at rest. Reducing current to the motor when it is at rest helps to cool it. When the motor resumes motion, the current is restored to values selected by the current DIP switches. When current is reduced, holding torque is reduced, which may cause the position of the motor to change slightly. If position maintenance is enabled, standby current is set to 0%. Response to <a>**SCR**? is:

```
1_ _ _ _0%_CURRENT_REDUCTION_ON_STANDBY
2_ _ _ _25%_CURRENT_REDUCTION_ON_STANDBY
3_ _ _ _50%_CURRENT_REDUCTION_ON_STANDBY
4_ _ _ _75%_CURRENT_REDUCTION_ON_STANDBY
```

Command	Description
> 1SCR2	Sets current reduction to 25% when the motor is at rest for one second

SFL Set User Flag

Type	Programming	Version	Software Version A
Syntax	<a>SFL<n>	Units	n = bits
Range	0, 1, or x	Default	0
Attributes	Buffered, Savable in Sequence	See Also	IF, NIF, WHILE, REPEAT, [FL]
Response	*SFLn		

This command sets a condition of 8 different bits. You can define the state of bits 0 - 7 with this command (0, 1, or X). 0 clears the corresponding bit, a 1 sets the corresponding bit, and an X retains the bit's present value. Not all the bits need to be defined with **SFL**, **SFL11** sets bits 6 and 7 while leaving the remaining bits unaltered.

Once you set bits 0—7, you can use IF to do an **IF_ELSE_NIF** operation or **REPEAT_UNTIL** and **WHILE_NWHILE** to do conditional looping.

Command	Description
> SFL1010	Set bits 5 and 7, and clear bits 6 and 4
> IF(FL1010)	If user flag bits 5 and 7 are set, then issue the following commands
> A10	Set acceleration to 10 rps^2
> AD15	Set deceleration to 15 rps^2
> V5	Set velocity to 5 rps
> D25000	Set distance to 25,000 steps
> G	Executes the move (Go)
> NIF	End of IF command

SL Software Limits

Type	Set-Up	Version	Software Version A
Syntax	<a>SL<n>,<n>	Units	n = steps
Range	± 2,147,483,647	Default	0
Attributes	Buffered, Savable in Sequence	See Also	LAD, OUT, SLD
Response	*SLn,n		

This command defines the CCW and CW software end-of-travel limits. Once you define the CCW and CW software limits, the motor will not be able to exceed those positions, unless the software limits are disabled by the **SLD3** command. You may use the **OUT** command to configure 1 or more outputs to signal if a software end of travel limit has been encountered. When the motor reaches the software limit, it will come to a stop, using the deceleration (**LAD**) specified.

Command	Description
> SL-10000,26944	CCW software limit -10,000 steps—CW software limit 26,944 steps
> SLD0	Enable both CW and CCW software limits
> PZ	Set absolute position to zero
> A10	Set acceleration to 10 rps^2
> AD15	Set deceleration to 15 rps^2
> V5	Set velocity to 5 rps
> D40000	Set distance to 40,000 steps
> G	Executes the move (Go), move will decelerate to a stop after 26944 steps

SLD Software Limit Disable

Type	Set-Up	Version	Software Version A
Syntax	<a>SLD<n>	Units	n = limits enabled/disabled
Range	0 - 3	Default	3
Attributes	Buffered, Savable in Sequence	See Also	OUT, SL, LD
Response	See Below		

This command enables or disables the software end-of-travel limits defined by the **SL** command.

SLD0: Enables both CW and CCW software limits. Motion will not go past the limits.
SLD1: Disables CW software limit. Can travel past CW software limit.
SLD2: Disables CCW software Limit. Can travel past CCW software Limit.
SLD3: Disable both CW and CCW software limits. Motor will ignore limits.

This command is similar to the Hardware Limit Disable (**LD**) command, except it uses software limits. You can use the **OUT** command to configure an output to indicate that the software limit has been reached. However, if you disable the limits, the output will not be activated even if motion goes past the software limit. The possible responses to the **SLD** commands are given below:

0_CW_CCW_SOFTWARE_TRAVEL_LIMITS_ENABLED
1_CCW_SOFTWARE_TRAVEL_LIMIT_ENABLED
2_CW_SOFTWARE_TRAVEL_LIMIT_ENABLED
3_NO_SOFTWARE_TRAVEL_LIMITS_ENABLED

Command	Description
> SL10000,10000	Set CCW software limit to 10,000 and CW software limit to 10,000
> PZ	Set absolute position to zero
> A10	Set acceleration to 10 rps^2
> V5	Set velocity to 5 rps
> D25000	Set distance to 25,000 steps
> SLD0	Enable both CW and CCW software limits
> G	Executes the move (Go)

SN Scan Time Delay

Type	Set-Up	Version	Software Version A
Syntax	<a>SN<n>	Units	n =milliseconds
Range	1 to 1,000	Default	50
Attributes	Buffered, Savable in Sequence	See Also	TD, TDR
Response	*SCAN_TIME=n_MILLISECONDS		

The Scan (**SN**) command allows you to define the debounce time (in milliseconds) for external sequence selection inputs. The debounce time is the amount of time that the sequence inputs must remain constant for a proper reading from a remote controller, such as a programmable logic controller (PLC). If you are using a PLC, you should change the debounce time to be greater than the scan time of the PLC to assure that the correct data is present when the data is read.

Command	Description
> SN150	Sets scan time of sequence select inputs to 150 ms

SP Set Position Absolute

Type	Set-Up	Version	Software Version A
Syntax	<a>SPn	Units	n = steps
Range	2,147,483,647	Default	0
Attributes	Buffered, Savable in Sequence	See Also	D, MPA, MPI, PR, PX

This command allows you to set the absolute counter (value n). The **SP** command is useful for labeling a position value at a certain point. For example, you can set the zero reference point (home) to some location other than that of the physical hardware home. If you have a cut-off saw, for example, you may not be able to mount the home switch at the cut point. However, by mounting the home switch at a known distance away, and resetting the reference point with the **SP** command, you can make the system function as if the home switch were at the cut point.

SP will alter both the **PR** and **PX** counts. You can use **SPA** to zero the position report offset. The indexer returns to the original coordinate system (similar to a **Z** command or power-up).

Command	Description
> MN	Sets to Normal mode
> A2	Sets acceleration to 2 rps^2
> GH2	Instructs the motor to go home at 2 rps
> PZ	Sets absolute position counter to zero
> 1PR *+0000000000	Reads the absolute position counter
> SP5000	Sets absolute counter to 5,000
> 1PR *+0000005000	Reads the absolute position counter to verify the absolute position
> SPA	Set to absolute position
> 1PR *+0000000000	Reads the absolute position counter

SPA Set Position Zero

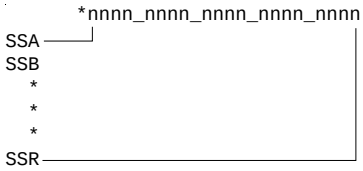
Type	Set-Up	Version	Software Version A
SYNTAX	<a>SPA	Units	None
Range	None	Default	None
Attributes	Buffered, Savable in Sequence	See Also	SP, PR, PX, D, MPA, MPI, SSP, Z

The **SPA** command will zero the position report offset introduced into the **PR** and **PX** counters with the **SP** command. The indexer returns to the original coordinate system (similar to a **Z** command or power-up).

SS Function Set-Up Report

Type	Status	Version	Software Version A
Syntax	aSS	Units	None
Range	None	Default	None
Attributes	Buffered, Savable in Sequence	See Also	FS, OS
Response	*nnnn_nnnn_nnnn_nnnn_nnnn		

Reports the status of the SS command (SSA - SSR) settings.



SSA RS-232C Echo Control

Type	Set-Up	Version	Software Version A
Syntax	<a>SSAn	Units	n = mode
Range	0 = on, 1 = off	Default	0
Attributes	Buffered, Savable in Sequence	See Also	SS, E, F

This command turns the RS-232C echo (the retransmission to the host device of characters received from the remote device by the SX) on and off.

SSA0: = Echo on
SSA1: = Echo off

In the Echo On (**SSA0**) mode, characters that are received by the SX are echoed automatically. In the Echo Off (**SSA1**) mode, characters are not echoed from the SX. This command is useful only if your computer can handle echoes. In a daisy chain, you must have the echo turned on (**SSA0**) to allow SX's further down the chain to receive commands.

Command	Description
> SSA1	Turns echo off (characters sent to the SX are not echoed back to the host)

SSC Enable End of Move in Position Window

Type	Set-Up	Version	Software Version A
Syntax	<a>SSC<n>	Units	n = mode
Range	0 = off, 1 = on	Default	0
Attributes	Buffered, Savable in Sequence	See Also	CIT, CEW

The **SSC** command enables the end of move in-position window. This window, defined by **CIT** and **CEW**, defines the *settling time* required for the current move in progress. Once the **SSC** is enabled, buffered command execution will only continue when the in-position requirements are satisfied. An immediate stop (**S**) or kill (**K**) will set **SSC** to 0, and thereby allow for subsequent buffered commands to execute.

NOTE: The SSC command requires an encoder.

SSG Save the Command Buffer on Limit

Type	Set-Up	Version	Software Version
Syntax	<a>SSGn	Units	n = mode
Range	0 = off, 1 = on	Default	0
Attributes	Buffered, Savable in Sequence	See Also	LAD, LD, SS, SLD

In most cases, it is desirable that upon activating an end-of-travel limit input, all motion should cease until the problem causing the over-travel is rectified. This will be assured if all commands pending execution in the command buffer are cleared when hitting a limit, this is the case if **SSG0** is specified. If **SSG1** is specified and a limit is activated, the current move is aborted, but the remaining commands in the buffer continue to be executed.

Command	Description
> SSG1	Save buffer on limit
> A10	Set acceleration to 10 rps^2
> AD5	Set deceleration to 5 rps^2
> V5	Set velocity to 5 rps
> D25000	Set distance to 25,000 steps
> G	Executes the move (Go), if a limit is encountered, the outputs will still turn on
> 011	Activate programmable outputs #1 and #2

SSH Save the Command Buffer on Stop

Type	Set-Up	Version	Software Version A
Syntax	<a>SSHn	Units	n = mode
Range	0 = off, 1 = on	Default	0
Attributes	Buffered, Savable in Sequence	See Also	SS, IN, S

SSH0 = Clears command buffer
SSH1 = Saves command buffer

In normal operation (**SSH0**) the Stop (**S**) command or a dedicated stop input will cause any commands in the command buffer to be cleared. If you select the Save Command Buffer On Stop (**SSH1**) command, a remote stop input or Stop (**S**) command will only stop execution of a move in progress. It will not stop execution of any commands that remain in the buffer.

Command	Description
> SSH0	Clears buffer on stop
> A10	Sets acceleration to 10 rps^2
> AD20	Sets deceleration to 20 rps^2
> V5	Sets velocity to 5 rps
> D25000	Sets distance to 25,000 steps
> L50	Loops 50 times
> G	Executes the move (Go)
> T.5	Pauses for 500 ms
> N	Ends loop
> S	Stops execution of the move in progress and clears the loop

SSI Enable/Disable Interactive Mode

Type	Set-Up	Version	Software Version A
Syntax	<a>SSIn	Units	n = mode
Range	0 = enable, 1 = disable	Default	0
Attributes	Buffered, Savable in Sequence	See Also	SS, SSA, SSN, E, F

When **SSI0** is enabled and the device address is set to 1, the SX responds with a (**>**) when it understands a command and a (**?**) when it does not. Both responses are preceded with a line feed, carriage return sequence. In Interactive mode, the SX will transmit a ***READY** and a (**>**) when energized or when a Reset (**Z**) command is executed.

If you try to define a loop command, you will not get back a (**>**) until you finish defining the loop. **SSI1** disables the Interactive mode. You will not receive (**>**) or (**?**) from the SX.

SSJ Enable/Disable Continuous Scan Mode

Type	Set-Up	Version	Software Version A
Syntax	<a>SSJn	Units	n = mode
Range	0 = off, 1 = on	Default	0
Attributes	Buffered, Savable in Sequence	See Also	XR, XD, XT, IN, INL, XQ, SN, SS

SSJ1: This command *enables* Continuous Scan mode. The SX continuously scans the inputs designated as sequence select inputs by the IN command and executes the sequence represented by the BCD number on the inputs. The SX will not check the inputs while the sequence is still running. If Interrupted Run mode (**XQ**) is active, all the sequence input lines must go inactive prior to scanning the next sequence. A stop command discontinues continuous sequence scanning , unless **SSH1** is used.

SSJ0: This command *disables* Continuous Scan mode. It does not scan the BCD numbers for sequence execution. In this mode, you can execute sequences using the RS-232C interface.

SSL Resume Execution

Type	Set-Up	Version	Software Version A
Syntax	<a>SSLn	Units	n = mode
Range	0 = off, 1 = on	Default	0
Attributes	Buffered, Savable in Sequence	See Also	S, C, PS

This command allows a sequence or move to resume execution, after being stopped, by entering a **C** command.

SSL1: Resume execution of the sequence or commands in the buffer when Continue (C) command is entered
SSL0: Disable resume feature

You can stop a program or move using Stop (**S**) or the remote stop input (**INnD**). If **SSL1** is enabled, the move will resume when you enter **C**.

*Helpful Hint: To pause and continue a preset move without entering a **C** command over the RS232C interface, try the following: define a Stop input, **INnD**, and a Pause/Continue input, **INnF**. Enter SSH1 to save the Command Buffer on Stop input, and SSL1 to Resume Execution. In this manner the stop input will stop motion when the stop switch transitions to active, and the pause/continue input will resume the movement when the pause/continue input transitions to deactive. As an alternative, these two inputs may be wired together into one switch, and then motion will pause when the switch is activated, and resume when the switch is deactivated. This will not work with a Mode Continuous move.*

Command	Description
> SSL1	Enable resume function
> XE1	Erase sequence #1
> XD1	Define sequence #1

A1	Set acceleration to 1 rps^2
AD5	Set deceleration to 5 rps^2
V1	Set velocity to 1 rps
D200000	Set distance to 200,000 steps
G	Execute the move (Go)
T2	Wait 2 seconds
G	Execute the move (Go)
XT	End sequence definition
> XR1	Run sequence #1
> S	Stops sequence
> C	Resumes sequence

SSN Set Message Mode

Type	Set-Up	Version	Software Version A
Syntax	<a>SSNn	Units	n = mode
Range	0 = off, 1 = on	Default	1
Attributes	Buffered, Savable in Sequence	See Also	SS, SSI

When the Interactive (**SSI**) and Message modes are active, an error message accompanies the prompt (**?**) to identify the error.

SSN0 = Turn off Message mode
SSN1 = Turn on Message mode

Command	Description
> SSI0	Set to Interactive Active mode
> SSN1	Set to Message Active mode
> PR	The command is incorrect and the response is: DEVICE_ADDRESS_REQUIRED

SSP Clear All Position Offsets with PZ, PFZ

Type	Set-Up	Version	Firmware Version E
SYNTAX	<a>SSP<n>	Units	n = mode
Range	0 = off, 1 = on	Default	0
Attributes	Buffered, Savable in Sequence	See Also	PZ, PFZ, SPA, Z

The **SSP** command allows **PZ** to clear not only the reported position, but the actual internal counter offsets resulting from previous **PZ** commands.

SSP1: Allow **PZ**, **PFZ** to clear all internal offsets and positions (Equivalent to issuing a Reset (**Z**) command, in order to clear counters)
SSP0 Restrict **PZ**, **PFZ** to clear reported positions

SSR Enable/Disable Fault On Shutdown

Type	Set-Up	Version	Software Version A
Syntax	<a>SSRn	Units	n = mode
Range	0 = off, 1 = on	Default	0
Attributes	Buffered, Savable in Sequence	See Also	RSE

SSR0: The dedicated fault output of the SX only goes active when there is a true fault condition. The output does not go active if the user or a sequence turns off the drive amplifier via the **OFF** or **ST1** commands.

SSR1: The dedicated fault output of the SX goes active any time the drive amplifier is shutdown. Either a fault condition or a user shutdown will activate the output. The **RSE** command can be used to determine the cause of the shutdown.

Command	Description
> SSR1	Sets fault output to active upon shutdown
> OFF	Turns off drive amplifier—fault output will go active

ST Shutdown

Type	Programming	Version	Software Version A
Syntax	<a>STn	Units	n = mode
Range	0 = off, 1 = on	Default	0
Attributes	Buffered, Savable in Sequence	See Also	OFF, ON

The Shutdown (**ST1**) command activates the shutdown output to the drive to command the drive to de-energize the motor. The system ignores move commands that you issue after the **ST1** command.

The **ST0** command turns off the shutdown output allowing the drive to energize the motor. Once you enable the drive, you can execute moves.

This command alleviates motor heating and allows you to manually position the load, it does not reset the absolute counter.

Command	Description
> ST1	Commands the drive to de-energize the motor

STOP Stop

Type	Programming	Version	Software Version A
Syntax	<a>STOP	Units	None
Range	None	Default	None
Attributes	Buffered, Savable in Sequence	See Also	S, SSH, SSL

This command decelerates motion to a stop using the last defined deceleration (**AD**) value. This command normally clears any remaining commands in the command buffer, unless Clear/Save the Command Buffer on Stop (**SSH**) is set to 1, then only the current command is interrupted from completion and execution proceeds.

If the Resume Execution (**SSL**) command is set to 1, a Continue (**C**) command may be issued to complete the interrupted move and continue sequence or buffer execution. This command is similar to the **S** command, except that the **STOP** command is buffered and can be saved in a sequence.

STR Set Strobe Output Delay Time

Type	Set-Up	Version	Software Version A
Syntax	<a>STR<n>	Units	n = ms
Range	10 - 5,000	Default	1000
Attributes	Buffered, Savable in Sequence	See Also	VRD, LRD, DRD, XRD, IN, OUT, TRD,
Response	`*STROBE_TIME_n_MILLISECONDS`		FRD, VARD

This command defines the amount of time each strobe output (defined by the **OUT** command) is active. These delay and strobe outputs are used when loading parallel BCD data via remote inputs. The data transferred from the remote inputs are:

Velocity	**VRD**
Distance	**DRD**
Loop Counts	**LRD**
Sequence Numbers	**XRD**
Time	**TRD**
Following	**FRD**
Variables	**VARD**

If used with a PLC, the Output Delay Time should be greater than the PLC Scan Time (to ensure that the data is present during a read) or set to a minimal debounce time if thumbwheel switches are used. The strobe output indicates that the SX is ready for parallel input. The **STR** value must be no less than `10`.

T Time

Type	Programming	Version	Software Version A
Syntax	<a>Tn	Units	n = seconds
Range	0.001 to 999.999	Default	None
Attributes	Buffered, Savable in Sequence	See Also	TRD

The Time (**T**) command causes the SX to wait the number of seconds that you specify before it executes the next command in the buffer. This command is useful whenever you need to dwell within a sequence.

Command	Description
> MN	Sets to Normal mode
> A5	Sets acceleration to 5 rps^2
> AD5	Sets deceleration to 5 rps^2
> V5	Sets velocity to 5 rps
> D25000	Sets distance to 25,000 steps
> T10	Pauses for 10 seconds
> G	Executes the move (Go)
> T5	Pauses for 5 seconds after the move ends
> G	Executes the move (Go)

TD Set Inputs Debounce Time

Type	Set-Up	Version	Software Version C2
Syntax	<a>TD<n>	Units	n = milliseconds
Range	1 to 1000	Default	2
Attributes	Buffered, Savable in Sequence	See Also	TDR, SN

This command determines how long (in milliseconds) an input must remain activated to be recognized as true. The value n should be set to avoid problems with bouncy switches or noisy environments. It affects all inputs except the registration input.

Command	Description
> 1TD50	Sets debounce time to 50 ms
> 1TD	Reports debounce status: DEBOUNCE TIME 50 MILLISECONDS

CAUTION

TD affects all inputs (including CW & CCW limits), with the exception of the registration input. If the TD value is too large and the limit inputs are active for a period less than the TD value, the inputs may not be recognized.

TDR Set Registration Input Debounce Time

Type	Set-Up	Version	Software Version A
Syntax	<a>TDR<n>	Units	n = milliseconds
Range	1 - 255	Default	10
Attributes	Buffered, Savable in Sequence	See Also	IN, REG, TD, SN
Response	*TDRn		

This command sets the debounce time (in milliseconds) for the registration input. The value should be set to avoid problems with bouncy switches or in noisy environments.

TEST Test

Type	Motion	Version	Software Version A
Syntax	<a>TEST	Units	None
Range	None	Default	None
Attributes	Buffered, Savable in Sequence	See Also	None

This command rotates the motor one revolution in each direction at a velocity of 1 rps. It is designed to test the SX drive interface. The command ignores the limit switches and software settings that would normally preclude such motion to test the interface. ***Use this command with extreme caution***.

Command	Description
> MR25000	Sets motor resolution to 25,000 pulses/rev
> TEST	SX sends 25,000 pulses to the drive in each direction (CW & CCW)

TF Set Following Time

Type	Set-Up	Version	Software Version B
Syntax	<a>TF<n>	Units	n = milliseconds
Range	1 - 32	Default	4
Attributes	Buffered, Savable in Sequence	See Also	None
Response	*FOLLOWING_TIME_n_MILLISECONDS		

TF sets the time (in ms) that the encoder uses in the following function to obtain the SX step output frequency. The **TF** value should be adjusted to obtain maximum motor shaft smoothness. For example, **TF4** sets the sample time at 4 ms, every 4 ms the SX outputs a new step frequency based on the accumulation of encoder pulses during those 4 ms.

TM Move Time Report

Type	Status	Version	Software Version A
Syntax	aTM	Units	None
Range	None	Default	None
Attributes	Buffered, Savable in Sequence	SEE ALSO	None
Response	*MOVE_TIME_n_MILLISECONDS		

TM reports the length of time (in milliseconds) the previous move required for completion. The timer begins at the end of the **G** command and runs until the move is complete. The timer indicates the time (ms) that was required to output the designated number of pulses.

Command	Description
> MN	Sets to Normal mode
> A10	Sets acceleration to 10 rps^2
> AD10	Sets deceleration to 10 rps^2
> V1	Sets velocity to 1 rps
> D25000	Sets distance to 25,000 steps
> G	Executes the move
> 1TM	Reports the time the move required

TR Wait for Trigger

Type	Programming	Version	Software Version A
Syntax	<a>TRn	Units	n = state of input
Range	0, 1, or x (see below)	Default	None
Attributes	Buffered, Savable in Sequence	See Also	IN, INL, TS

TR prompts the SX to wait until the inputs (configured as triggers with **IN**) match the pattern, before going on to the next command. It will not check inputs not designated as triggers. The active state voltage level is defined with **INL**. The order of comparison of the inputs designated as trigger inputs ranges from inputs 1 to 8 (as labeled on the unit).

Triggers synchronize SX operations with external events. They can implement a handshaking function with other devices. The n variables and their functions are listed below:

1 = Input active(ON)
0 = Input inactive (OFF)
X = Don't care

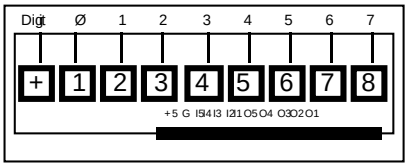
Command	Description
> IN1B	Configure input 1 as a Sequence Select Input
> IN2A	Configure input 2 as a trigger
> IN3A	Configure input 3 as a trigger
> IN4C	Configure input 4 as a Kill Input
> IN5D	Configure input 5 as a Stop Input
> IN6E	Configure input 6 as a Command Enable Input
> IN7A	Configure input 7 as a trigger
> TR1X0	Wait for input 2 to be active and input 7 to be inactive before running next command—Trigger input 3 ignored
> A10	Sets acceleration to 10 rps^2
> AD20	Sets deceleration to 20 rps^2
> V5	Sets velocity to 5 rps
> D25000	Sets distance to 25,000 steps
> G	Executes the move (Go)

TRD — Read Timer via Parallel Input/Output

Type	Programming	Version	Software Version B
Syntax	<a>TRD<c><d><e>	Units	c,d =digit selector, e = scaling factor
Range	c,d = 0 - 7, e = 0 - 5	Default	None
Attributes	Buffered, Savable in Sequence	See Also	IN, OUT, STR, T, DRD, FRD, LRD, VRD, XRD

This command instructs the SX to read Timer values from Compumotor's TM8 Module (refer to *Chapter 3, Installation* for information on the TM8 Module). Inputs 1-4 must be configured as data inputs. The inputs must be configured as active low (**INL0**) and outputs 1-3 must be configured as strobe outputs to use the TM8 Module.

Helpful Hint: <c> & <d> values represent the TM8 Module digits:



The command syntax allows for digit range selection through the optional <c><d> fields. The <c> field is used to signify the start of the digit range to be read from the TM8 Module. The raw data format is **XXX.XXX**, the maximum value is **999.999**. Any larger number will result in the SX giving a **?** response. The <d> field represents the end of the digit range to be read. The values of these fields can range from 0 to 7 with the <c> field always being less than or equal to the <d> field value.

The <e> field is used to scale the distance value by 10^e. If the <c> and <d> fields are used, the <e> field must be used.

If the <c>, <d>, and <e> fields are not used, the **TRD** command will read all the digits of the TM8 Module. If you are using the TM8 Module, the Output Strobe Delay Time must be set at a value of 10 or greater.

You may use a PLC with the **TRD** command. **TRD** uses a multiplexed I/O scheme. The outputs strobe through a BCD sequence at the Set Strobe Output Delay Time (**STR**) command rate and reads one BCD digit at a time. The outputs and inputs must be configured as in the TM8 Module read case. A **TRD** command will issue the following strobe sequence:

Voltage Level 01	02	03	Data In (Active Low)
low	low	low	MSD (Digit 1)
high	low	low	Digit 2
low	high	low	Digit 3
high	high	low	Digit 4
low	low	high	Digit 5
high	low	high	Digit 6
low	high	high	Digit 7
high	high	high	LSD (Digit 8)

The PLC, while reading the output strobe, must place the appropriate data at the SX inputs 1 - 4. Input 1 is the digit's LSB and input 4 the MSB. Configuring an input as a data valid line would allow the PLC to control the rate at which the outputs will strobe through the above sequence.

Turn the TM8's thumbwheels to display the following digits: **+ 0 0 1 2 3 4 5 6**

Enter the following command:

Command	Description
> 1TRD	Sets a time delay of 123.456 seconds

TS Trigger Input Status

Type	Status	Version	Software Version A
Syntax	<a>TS	Units	None
Range	None	Default	None
Attributes	Immediate, Never Saved	See Also	IN, INL, TR
Response	*xxxx_nnnn_nnnn (n = 0, 1, or X)		

This command reports the state of the trigger inputs. Response is in the form **xxxx_nnnn_nnnn** where

n = **1** (Input active)

n = **0** (Input inactive)

n = **x** (Input not configured as a trigger)

The response to **1TS** is: *The first four digits of the response always have the value **x** (input not configured as a trigger) to denote that they cannot be configured as triggers.*

TS is useful for checking the status of the trigger inputs when it appears as though execution is being halted by a **TR** command. To make sure that your trigger pattern is met use **TS**.

Command	Response
> 1TS	*xxxx_1xxx_x110

Inputs 1, 6, and 7 are configured as triggers and are active.

TW Select Thumbwheel Input Mode

Type	Set-Up	Version	Software Version B
Syntax	<a>TW<n>	Units	n = mode
Range	1, 2, 3	Default	1
Attributes	Immediate, Savable	See Also	DRD, FRD, LRD, TRD, VARD, VRD, XRD

The parallel read commands (**DRD**, **FRD**, **LRD**, **TRD**, **VARD**, **VRD**, and **XRD**) can be used in one of three modes that are selected by the **TW** command. A description of each mode is given below:

TW0 N This mode reads two BCD digits at a time over the input pins. It requires that the eight inputs be defined as data inputs, and at least one output be defined as a strobe output. The number of pairs of BCD digits read is equal to the number of outputs defined as strobe outputs. When a parallel read command is received, the SX will activate each strobe output pin individually for the amount of time specified by the Strobe Output Delay (**STR**) command. During this time, the proper BCD data should be placed on the data input pins.

TW1 N This mode was designed to be compatible with Compumotor's Thumbwheel Interface Module (TM8). It reads one BCD digit at a time over the input pins. Four of the eight inputs must be defined as data inputs, and three of the four outputs must be defined as strobe outputs. When a parallel read command is requested the strobe outputs will toggle in a binary pattern as follows:

Strobe #1	Strobe #2	Strobe #3
0	0	0
1	0	0
0	1	0
1	1	0
0	0	1
1	0	1
0	1	1
1	1	1

0 = Inactive, 1 = Active

The strobe outputs will remain in each state for the amount of time specified by the Strobe Output Delay (**STR**) command. You can also define an input as a data valid input to allow for external strobe rates. **TW1** mode allows you to select a range of digits to be read and to provide a scale factor for the data.

TW2 The **TW2** mode is identical to **TW0** except that only one BCD digit is read at a time rather than two. Therefore, only four inputs need to be defined as data inputs.

TX — Transmit Variable & String

Type	Programming	Version	Software Version C2
Syntax	<a>TXn,m,p,x	Units	see below
Range	n = 1 - 50, m = 0 or 1, p = 0 - 5, x = ASCII	Default	None
Attributes	Buffered, Savable in Sequence	See Also	VAR, "

This command transmits via the serial port (RS-232C) the value of a user variable and a string to compose a command for another indexer. The command format is **TXn**,**m**,**p**,**x**, where:

n = The variable number
m = 1 if the sign is to be sent, 0 if the sign is not to be transmitted
p = The number of digits to be sent after the decimal point
x = The ASCII string that is transmitted prior to the variable contents

Command	Response
> VAR1=987.12345	Set variable #1
> TX1,0,3,3V	3V987.123 is transmitted via the serial port (RS-232C)
> TX1,1,0,2D	2D+987 is transmitted via the serial port (RS-232C)
> TX1,0,5,2VAR3=	2VAR3=987.12345 is transmitted via the serial port (RS-232C)

U — Pause and Wait for Continue

Type	Programming	Version	Software Version A
Syntax	<a>U	Units	None
Range	None	Default	None
Attributes	Immediate, Never Saved	See Also	C, PS

This command causes the SX to complete the command in progress, then wait until it receives a **C** (Continue) to resume processing. Since the buffer is saved, the SX continues to execute the program (at the point where it was interrupted). The SX continues processing when it receives the **C**. This command is typically used to stop a machine while it is unattended.

Command	Description
> MN	Sets to Normal mode
> A5	Sets acceleration to 5 rps^2
> AD10	Sets deceleration to 10 rps^2
> V5	Sets velocity to 5 rps
> L0	Loops indefinitely
> D25600	Sets distance to 25,600 steps
> G	Executes the move (G)
> T10	Waits 10 seconds after the move
> N	Ends loop
> U	Halts execution until the SX receives a C command

The loop stops at the end of a move, and resumes when the SX receives the **C** command. There may be a 10-second delay before motion resumes after the **C** command is executed, depending on when the Pause and Wait for Continue (**U**) command is completed.

UNTIL — Until

Type	Programming	Version	Software Version A
Syntax	<a>UNTILe	Units	e=evaluation conditions
Range	Evaluation commands	Default	None
Attributes	Buffered, Savable in Sequence	See Also	REPEAT, WHILE, NWHILE, IF

The **UNTIL** command marks the end of the **REPEAT** command. The **UNTIL** command is evaluated and if it is false, program flow is redirected to the **REPEAT** command, where the commands between the **REPEAT** and **UNTIL** commands are executed again. Those commands will continue to execute until the **UNTIL** command is true, then the commands after the **UNTIL** command are executed. The commands between **REPEAT** and **UNTIL** are always executed at least once.

REPEAT...commands...**UNTIL**(condition)

Command	Description
> REPEAT	Repeat
> GD1	Execute predefined move #1
> T1	Time delay of one seconds
> UNTIL(INXXXX10_OR_INXXXX01)	If input #1 on and input #2 off, or input #1 off and input #1 on, do next command, otherwise execute from command following REPEAT

V Velocity

Type	Motion	Version	Software Version A
Syntax	<a>V<n>	Units	n = rps
Range	0 - 50.00000	Default	1
Attributes	Buffered, Savable in Sequence	See Also	A, AD, D, G, IV
Response	*Vn		

The Velocity (**V**) command defines the maximum speed at which motion will occur. The value is stored in nonvolatile memory. The maximum velocity is 50 rps for all resolutions except MR50000 and MR50800. The maximum velocity for those resolutions is 35 rps.

Command	Description
> MC	Sets to Continuous mode
> A5	Sets acceleration to 5 rps^2
> AD10	Sets deceleration to 10 rps^2
> V5	Sets velocity to 5 rps
> G	Execute the move (Go) at 5 rps

In Normal mode (**MN**) the maximum velocity may also be limited when the resulting move profile is triangular.

VAR Variables

Type	Programming	Version	Software Version A
Syntax	<a>VARn	Units	n = variables
Range	1 - 50	Default	0
Attributes	Buffered, Savable in Sequence	See Also	TX, POS, FEP, ABS
Response	*snnnnnnnnnnnnnn.nnnnn		

Fifty variables (**VAR1**—**VAR50**) can be used to perform mathematical operations and then be used for selected data fields or in evaluation statements. You can substitute variables for data fields for the following commands:

XG(VARn)	Goto sequence number contained in VARn
XR(VARn)	Run sequence number contained in VARn
XRP(VARn)	Run with pause sequence number contained in VARn
GOTO(VARn)	Goto sequence number contained in VARn
GOSUB(VARn)	Gosub to sequence number contained in VARn
L(VARn)	Load loop count with value in VARn
V(VARn)	Load velocity with value in VARn
D(VARn)	Load distance with value in VARn
A(VARn)	Load acceleration with value in VARn
AD(VARn)	Load deceleration with value in VARn
DP(VARn)	Load distance point with value in VARn
FP(VARn)	Load following point with value in VARn
T(VARn)	Load and execute a timer with value in VARn
FOL(VARn)	Load following with value in VARn

Command	Description
VAR1=50	Load variable #1 with 50
XR(VAR1)	Execute sequence #50

You can use variables in mathematical operations to obtain new values:

Addition: **VARn=VARn+VARn**
Division: **VARn=VARn/VARn**
Subtraction: **VARn=VARn-VARn**
Multiplication: **VARn=VARn*VARn**

A constant, **POS** (the present value of the position counter), **FEP** (Following Encoder Position), or **ABS** (Encoder Position) can be substituted for the operands:

- ❑ **VAR1=VAR2+20**
- ❑ **VAR5=VAR7**
- ❑ **VAR8=POS+50000**
- ❑ **VAR9=20.62**

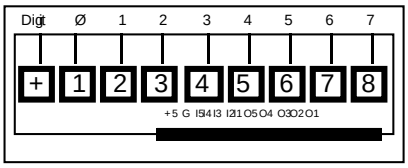
Command	Description
VAR1=20	Load variable #1 with 20
VAR2=5	Load variable #2 with 5
VAR3=VAR1-VAR2	Variable #3 now contains 15

VARD Read Variable Via Parallel Input/Output

Type	Programming	Version	Software Version B
Syntax	<a>VARD<b>,<c><d><e> (b = # of variable)	Units	c,d =digit selector, e = scaling factor
Range	c,d = 0 - 7, e = 0 - 9	Default	None
Attributes	Buffered, Savable in Sequence	See Also	STR, IN, OUT, VAR

This command instructs the SX to read Variable values from Compumotor's TM8 Module (refer to *Chapter 3, Installation* for information on the TM8 Module). Inputs 1-4 must be configured as data inputs. The inputs must be configured as active low (**INL0**) and outputs 1-3 must be configured as strobe outputs to use the TM8 Module.

Helpful Hint: <c> & <d> values represent the TM8 Module digits:



The command syntax allows for digit range selection through the optional <c><d> fields. The <c> field is used to signify the start of the digit range to be read from the TM8 Module. Raw data is in the form **xxxxx.xxx** The maximum **VARD** number is **21474.836**. Any larger number will result in the SX giving a **?** response. The <d> field represents the end of the digit range to be read. The values of these fields can range from 0 to 7 with the <c> field always being less than or equal to the <d> field value.

The <e> field is used to scale the distance value by 10^e If the <c> and <d> fields are used, the <e> field must be used. If the <c>, <d>, and <e> fields are not used, **VARD** will read all the digits of the TM8 Module. If you are using the TM8 Module, the Output Strobe Delay Time must be set at a value of 10 or greater.

You may use a PLC with the **VARD** command. **VARD** uses a multiplexed I/O scheme. The outputs strobe through a BCD sequence at the Set Strobe Output Delay Time (**STR**) command rate and reads one BCD digit at a time. The outputs and inputs must be configured as in the TM8 Module read case. **VARD** will issue the following strobe sequence:

Voltage Level			
01	02	03	Data In (Active Low)
low	low	low	MSD (Digit 1)
high	low	low	Digit 2
low	high	low	Digit 3
high	high	low	Digit 4
low	low	high	Digit 5
high	low	high	Digit 6
low	high	high	Digit 7
high	high	high	LSD (Digit 8)

The PLC, while reading the output strobe, must place the appropriate data at the SX inputs 1—4. Input 1 is the digit's LSB and input 4 the MSB. Configuring an input as a data valid line would allow the PLC to control the rate at which the outputs will strobe through the above sequence.

Turn the TM8's thumbwheels to display the following digits: **+1 2 3 4 5 6 7 8**. Enter the following commands.

Command	Description
> 1VARD1	Read variable 1 from digits
> 1VAR1	*00000000000000123.45678

VARN=FUN Enable and Read Function Keys

Type	Programming	Version	C2
Syntax	<a>VARn=FUN	Units	n = variable number
Range	1 to 50	Default	None
Attributes	Buffered, Savable in Sequence	See Also	VAR, VARD

The **VARN=FUN** command is used to enable the function keys on the RP240 and retrieve the function key pressed. Once this command is encountered, command processing stops until the RP240 returns a number corresponding to the function key pressed. Function key 1 (**F1**) returns a 1, function key 2 (**F2**) returns a 2, etc. **MENU RECALL** returns a zero. The number that is returned is placed in the variable **n**.

The RP240 is actually sending an exclamation point (**!**) before the number returned, allowing the SX to interpret the value returned.

Note: If you are using a kill sequence (**XFK**), place the command **"*** at the beginning of the kill sequence.

Command	Description
>XE1	Erase sequence 1
>XD1	Begin definition of sequence 1
DPC205	Position the cursor on line 2, column 5
DTXTPRESS_F1	Place message PRESS F1 at current cursor location
VAR1=FUN	Retrieve function key pressed value and place in variable 1
L	Begin endless loop
IF(VAR1=1)	If variable 1 equals 1, do the commands between IF and NIF
XG2	Branch to sequence 2
NIF	End IF statement
N	End endless loop
XT	End definition of sequence 1
>	

VARN=NUM Enable and Read Numeric Keypad

Type	Programming	Version	C2
Syntax	<a>VARn=NUM	Units	n = Variable number
Range	1 to 50	Default	None
Attributes	Buffered, Savable in Sequence	See Also	VAR, VARD

The **VARN=NUM** command is used to enable the numeric keypad on the RP240 and retrieve the numeric information entered. Once this command is encountered, command processing stops until the RP240 returns a value entered on the numeric keypad. The **+/-** key changes the sign of the number entered. The **+/-** key must be pressed before the number is entered, or it will have no effect. The C/E key will clear the number you have entered and allow you to restart. The ENTER key sends the number entered to the SX. If no value was entered before the ENTER key was pressed, the ENTER key will have no effect.

The number that is returned by the RP240 is placed in variable **n** of the SX. If either the **+/-** key or the **.** key is entered by itself, the variable will be set to zero.

The RP240 is actually sending an exclamation point (**!**) before the number returned, thus allowing the SX to interpret the value returned.

Note: If you are using a kill sequence (**XFK**), place the command "* at the beginning of the kill sequence.

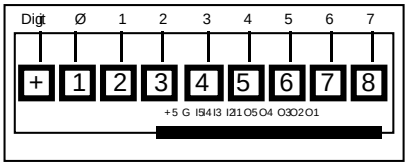
Command	Description
>XE1	Erase sequence 1
>XD1	Begin definition of sequence 1
DCLR0	Clear Display
DPC105	Position the cursor on line 1, column 5
DTXTENTER_BAG_COUNT	Place message ENTER BAG COUNT at current cursor location
DCLR2	Clear Display
DPC121	Position the cursor on line 1, column 21
VAR1=0	Initialize variable 1
WHILE(VAR1<1_OR_VAR1>10)	Do statements between WHILE and NWHILE until 0<VAR1<11
DCLR2	Clear line 2 of RP240 display
DPC121	Position cursor on line 1, column 21
VAR1=NUM	Retrieve numeric value and place in variable 1
IF(VAR1<1)	If variable 1 < 1 do the commands between IF and NIF
DPC205	Position the cursor on line 2, column 5
DTXTBAG_COUNT_TOO_LOW	Place message BAG COUNT TOO LOW at current cursor location
T2	Time delay 2 seconds
NIF	End IF statement
IF(VAR1>10)	If variable 1 > 10 do the commands between IF and NIF
DPC205	Position the cursor on line 2, column 5
DTXTBAG_COUNT_TOO_HIGH	Place message BAG COUNT TOO HIGH at current cursor location
T2	Time delay 2 seconds
NIF	End IF statement
NWHILE	End WHILE statement
MN	Set to Normal mode
LD3	Disable limits (if not connected)
A10	Set acceleration to 10 rps^2
V2	Set velocity to 10 rps
D25000	Set distance to 25000 steps
L(VAR1)	Loop as many times as specified by variable 1
G	Initiate motion
N	End loop
XT	End definition of sequence 1

VRD Read Velocity Value from Parallel Input/Output

Type	Programming	Version	Software Version B
SYNTAX	<a>VRD<c><d><e>	Units	c,d =digit selector, e = scaling factor
Range	c,d = 0 - 7, e = 0 - 5	Default	None
Attributes	Buffered, Savable in Sequence	See Also	IN, OUT, STR, V, DRD, FRD, TRD, XRD

This command instructs the SX to read Velocity values from Compumotor's TM8 Module (refer to *Chapter 3, Installation* for information on the TM8 Module). Inputs 1-4 must be configured as data inputs. The inputs must be configured as active low (**INL0**) and outputs 1 - 3 must be configured as strobe outputs to use the TM8 Module.

Helpful Hint: <c> & <d> values represent the TM8 Module digits:



The command syntax allows for digit range selection through the optional <c><d> fields. The <c> field is used to signify the start of the digit range to be read from the TM8 Module. The raw data format is **XX.XXXX**, the maximum value is 50.0000. Any larger number will result in the SX giving a **?** response. The <d> field represents the end of the digit range to be read. The values of these fields can range from 0 to 7 with the <c> field always being less than or equal to the <d> field value.

The <e> field is used to scale the distance value by 10^e. If the <c> and <d> fields are used, the <e> field must be used. If the <c>, <d>, and <e> fields are not used, the **VRD** command will read all the digits of the TM8 Module. If you are using the TM8 Module, the Output Strobe Delay Time must be set at a value of 10 or greater.

You may use a PLC with the **VRD** command. **VRD** uses a multiplexed I/O scheme. The outputs strobe through a BCD sequence at the Set Strobe Output Delay Time (**STR**) command rate and reads one BCD digit at a time. The outputs and inputs must be configured as in the TM8 Module read case. **VRD** will issue the following strobe sequence:

Voltage Level			
01	02	03	Data In (Active Low)
low	low	low	MSD (Digit 1)
high	low	low	Digit 2
low	high	low	Digit 3
high	high	low	Digit 4
low	low	high	Digit 5
high	low	high	Digit 6
low	high	high	Digit 7
high	high	high	LSD (Digit 8)

The PLC, while reading the output strobe, must place the appropriate data at the SX inputs 1 - 4. Input 1 is the digit's LSB and input 4 the MSB. Configuring an input as a data valid line would allow the PLC to control the rate at which the outputs will strobe through the above sequence.

Turn the TM8's thumbwheels to display the following digits: **+0 0 1 2 3 4 5 6** and type the following commands.

Command	Description
> 1VRD	Reads velocity value
> 1V	Response *V12.3456

VS Set Start/Stop Velocity

Type	Motion	Version	Software Version A
Syntax	<a>VS<n>	Units	n = rps
Range	0.00000-50.00000	Default	0
Attributes	Buffered, Savable in Sequence	See Also	V, MR, IV
Response	*VSn		

The motor will jump immediately to **VS** value before ramping to a **V** value. If the **VS** command value is greater than the **V** command value for a move, the **V** command value will be used regardless of the **VS** command. (It will jump up to the **V** command value immediately.) If you use the **STOP** command to end a move, the SX will ramp down to the acceleration value and ignore the stop velocity command

Command	Description
> MC	Sets to Continuous mode
> VS1	Start/Stop velocity set to 1 rps
> V5	Sets velocity to 5 rps.
> A20	Sets acceleration to 20 rps^2
> G	Executes the move (Go)

Motor will immediately reach 1 rps, then accelerate at 20 rps^2 to 5 rps.

W1 Signed Binary Position Report

Type	Status	Version	Software Version A
Syntax	aW1	Units	None
Range	None	Default	None
Attributes	Immediate, Never Saved	SEE ALSO	PR, W2, W3
Response	See below		

Report back gives immediate binary representation of position relative to start of the current move. The format of the response is a four character response (**nnnn**) that is interpreted as a 32-bit binary number. The number must then be interpreted by the host device to give a numerical position in steps. The format is in 2's complement. A move in the negative direction will report back a negative number (bit 31 is set to 1).

Interpreting Binary Position Reports

This format of the position report (**nnnnn**), consists of five bytes. The first will be the axis number, followed by four bytes that must be linked together to form a 32-bit binary number. A typical SX communications algorithm expects to handle characters rather than binary numbers and may have problems with this kind of response. Assume that a response equivalent to the ASCII characters **^@**, **#**, **0**, and / (**^** refers to the **CTRL** key and **@** an unprintable character) is given. The binary code for this response should be:

^@ # 0 /

```
00000000 00100011 00110000 00101111
```

This code has to be interpreted by the computer. The four characters must be converted to their ASCII code numbers and multiplied by the appropriate power of 256. The first character received is the most significant byte. Refer to the table below for a conversion technique for **^@ # 0 /**. The formula used for the binary conversion is:

ASCII Value • Character Multiplier = Character Value

Response	ASCII Value	Character Multiplier	Conversion (steps)		
^@	0	x	16,777,216 (256^3)	=	0
#	35	x	65,536 (256^2)	=	2,293,760
0	48	x	256 (256^1)	=	12,288
/	47	x	1 (256^0)	=	47
			Position Total:		2,306,095

The transmission is not preceded with an asterisk (*) to maximize response time. *Do not use the **W1** command when multiple SX units are daisy-chained.*

W2 Hexadecimal Position Report

Type	Status	Version	Software Version A
SYNTAX	aW2	Units	None
Range	None	Default	None
Attributes	Immediate, Never Saved	See Also	PR, W1, W3
Response	*nnnnnnnn (nnnnnnnn = steps)		

The immediate hexadecimal character position report back (during motion) indicates position relative to the start of current move. The format of the response is 8 hexadecimal characters. Your computer needs to convert these characters into a usable format. This command does not indicate direction. You will receive an unsigned number.

Digit	Digit Multiplier
h (**MSD**)	h •16^7 = h • 268,435,456= _____
h	h •16^6 = h • 16,777,216= _____
h	h •16^5 = h • 1,048,576= _____
h	h •16^4 = h • 65,536= _____
h	h •16^3 = h • 4,096= _____
h	h •16^2 = h • 256= _____
h	h •16^1 = h • 16= _____
h (**LSD**)	h •16^0 = h • 1= _____

The variable **h** can be one of the following values:

Decimal Value	Hexadecimal Value (h)	Decimal Value	Hexadecimal Value (h)
0	0	8	8
1	1	9	9
2	2	10	A
3	3	11	B
4	4	12	C
5	5	13	D
6	6	14	E
7	7	15	F

Command	Description
> 1W2 *0000FA04	A distance of FA04 (64,004 steps) has occurred since the start of the last move

W3 Signed Hexadecimal Position Request

Type	Status	Version	Software Version A
Syntax	aW3	Units	None
Range	None	Default	None
Attributes	Immediate, Never Saved	See Also	PR, W1, W2
Response	*nnnnnnnn (n = steps)		

The **W3** command provides a position report in signed hexadecimal while the motor is moving. The report indicates position relative to the start of the current move.

Interpreting Hexadecimal Position Reports

This form of position report (***nnnnnnnn**) is generated by the **W3** command. It consists of an asterisk followed by eight hexadecimal characters: 0—9 and A—F. The position report is followed by a carriage return.

The decimal value of the hexadecimal expression can be determined using the technique demonstrated in the example. The response is in two's complement notation reflecting direction. Negative numbers imply CCW motion.

Positive ***W3*** Response Interpretation

The system provides responses in the following format:

MSD LSD
***hhhhhhhh**

The first digit is the most significant digit (**MSD**). The last digit is the least significant digit (**LSD**). Refer to the table below for the value of each digit.

Digit	Digit Multiplier
h (**MSD**)	$h \cdot 16^7 = h \cdot 268{,}435{,}456 =$ _____
h	$h \cdot 16^6 = h \cdot 16{,}777{,}216 =$ _____
h	$h \cdot 16^5 = h \cdot 1{,}048{,}576 =$ _____
h	$h \cdot 16^4 = h \cdot 65{,}536 =$ _____
h	$h \cdot 16^3 = h \cdot 4{,}096 =$ _____
h	$h \cdot 16^2 = h \cdot 256 =$ _____
h	$h \cdot 16^1 = h \cdot 16 =$ _____
h (**LSD**)	$h \cdot 16^0 = h \cdot 1 =$ _____

The decimal (**h**) may have one of the values shown below.

Decimal Value	Hexadecimal Value	Decimal Value	Hexadecimal Balue
0	0	8	8
1	1	9	9
2	2	10	A
3	3	11	B
4	4	12	C
5	5	13	D
6	6	14	E
7	7	15	F

Using the previous tables, review the decimal value that would be calculated if the following hexadecimal response were given:
***000433AE**

Hexadecimal	Character Multiplier	Conversion (steps)
0	0 • 268,435,456	0
0	0 • 16,777,216	0
0	0 • 1,048,576	0
4	4 • 65,536	262,288
3	3 • 4,096	12,288
3	3 • 256	768
A (= 10)	10 • 16	160
E (= 14)	14 • 1	14
Total Steps:	275,374	

Negative ***W3*** Response Interpretation

If the first digit of the response is an **F**, the response represents a *two's complement* negative number. There are several ways to convert an 8-digit two's complement hexadecimal number to decimal.

The Binary Approach

① Convert the hexadecimal response to binary form.
② Complement the binary number.
③ Add 1 to the binary result.
④ Convert the binary result to decimal value with a minus sign placed ahead of the decimal value.

The Computer Approach
① Subtract the hexadecimal number from **16^8 (2^{32} or 4,294,967,296).**

The Easy Way
① Ignore all the leading **F**'s, then convert the hexadecimal number to decimal.
② Subtract the next largest power of 16.

The indexer responds to **W3** as follows: ***FFFF9E58**
① Delete the **F**'s: **9E58 hex = 40,536**
② Subtract from **16^4 10000 hex = 65,536**
③ Subtraction Result = **-25,000**

WHEN — Set WHEN Condition

Type	Set-Up	Version	Software Version A
Syntax	<a>WHENe	Units	e=evaluation commands
Range	See EVALUATION	Default	None
Attributes	Buffered, Savable in Sequence	See Also	XWHEN
Response	*(condition)		

The **WHEN** command allows you to continuously examine a set of conditions and if the condition is true, the SX will execute the sequence defined by the **XWHEN** command. The **WHEN** condition is saved in non-volatile RAM.

Note: The **XWHEN** sequence will not be executed if the SX is in motion unless **MPP** mode is active. If the SX is waiting intentionally on a time delay or a trigger, the **XWHEN** sequence will not be called regardless of the **MPP** mode setting.

Command	Description
> XWHEN2	Set WHEN sequence to 2
> WHEN(INXXXX1)	When input #1 becomes active, execute the WHEN sequence (#2)

WHILE — WHILE

Type	Programming	Version	Software Version A
SYNTAX	<a>WHILEe	Units	e = evaluation commands
Range	evaluation	Default	None
Attributes	Buffered, Savable in Sequence	See Also	NWHILE, REPEAT, UNTIL, IF, [ER], [IN], [FL], AND, OR

The **WHILE** command, in conjunction with the **NWHILE** command, provides a means of conditional program flow. The **WHILE** command marks the beginning of the conditional statement. If the **WHILE** is true, the commands between the **WHILE** and **NWHILE** commands are executed. However, if the **WHILE** is false, program execution jumps to the command after the **NWHILE** command. Up to 16 levels of **WHILE** commands may be nested.

WHILE(condition) . . . commands . . . **NWHILE**

Command	Description
> WHILE(INXXXX1)	While input #1 is active, run commands between WHILE & NWHILE
> GD1	Execute predefined move 1
> NWHILE	End of WHEN

XBS Sequence Memory Available Report

Type	Status	Version	Software Version A
Syntax	aXBS	Units	None
Range	None	Default	None
Attributes	Immediate, Never Saved	See Also	XDIR
Response	*n_OF_8000_BYTES_(x%)_SEQUENCE_MEMORY_REMAINING *n = bytes 0 - 8,000, x = % 0 - 100		

This command reports the remaining amount of memory that can be used for sequence storage. The total space available for sequence storage is 8,000 bytes (characters). **XBS** is useful for determining how much more programming can be done on the SX, after defining several programs. It reports both number of bytes available and the percentage (%) of memory available.

Command	Response
> 1XBS	*4000_OF_8000_BYTES_(50%)_ SEQUENCE_ MEMORY_REMAINING

There are 4,000 bytes (50%) available in the sequence buffer.

XC Sequence Checksum Report

Type	Status	Version	Software Version A
SYNTAX	aXC	Units	None
Range	None	Default	None
Attributes	Buffered, Savable in Sequence	See Also	XD, XE, RSE
Response	*n (n = byte sum 0 - 00255)		

This command reports the nonvolatile memory checksum. After the unit has been programmed, the response can be used for system error checking. The number reported does not indicate the number of bytes programmed. This response is designed to be used for comparison. As long as the sequences are not reprogrammed, the checksum response should always be the same. Use the **XC** command to remove a **30** error (bad battery-backed RAM checksum).

Command	Response
> 1XC	*00149

XD Sequence Definition

Type	Programming	Version	Software Version A
Syntax	<a>XDn	Units	n = sequence number
Range	1 - 100	Default	None
Attributes	Buffered, Never Saved	See Also	XE, XR, XT

This command begins sequence definition for a specific sequence. All the commands between **XD** and the Sequence Termination (**XT**) commands will be defined as a sequence. If a sequence you are trying to define already exists, you must erase that sequence before defining it. ***Immediate commands cannot be entered into a sequence***.

Command	Description
> XE1	Erase sequence #1
> XD1	Define sequence #1
MN	Sets to Normal mode
A10	Sets acceleration to 10 rps^2
AD20	Sets deceleration to 20 rps^2
V5	Sets velocity to 5 rps
D10000	Sets distance to 10,000 steps
G	Executes the move (Go)
XT	End defining sequence #1
> XR1	Execute sequence #1

The commands in sequence #1 are defined and executed.

XDIR Sequence Directory

Type	Status	Version	Software Version A
Syntax	aXDIR	Units	None
Range	None	Default	None
Attributes	Immediate, Never Saved	See Also	XBS
Response	See Below		

This command reports the sequence number (sequences #1 - #100), and the amount of memory used by each sequence. The response is in the following format:

```
1XDIR *NO_SEQUENCES_DEFINED - If no sequences are defined
1XDIR *SEQUENCE_N_USES_X_BYTES
      *SEQUENCE_N_USES_X_BYTES
```

If a sequence exists, it would list the sequence number and the number of bytes used by the sequence.

Command	Response
> 1XDIR	*SEQUENCE_5_USES_251_BYTES
	*SEQUENCE_39_USES_45_BYTES

Reports the number of bytes used by sequences 5 and 39. No other sequences are defined.

XE Sequence Erase

Type	Programming	Version	Software Version A
Syntax	<a>XEn	Units	n = sequence number
Range	1 - 100	Default	None
Attributes	Buffered, Savable in Sequence	See Also	XD, XR, XT, XEALL

This command allows you to delete a sequence. The sequence that you specify (**n**) will be deleted when you issue the command. **Caution should be used when executing this command. You cannot retrieve the sequence after it is deleted.**

Command	Description
> XE1	Deletes sequence #1

XEALL Erase All Sequences

Type	Programming	Version	Software Version A
SYNTAX	<a>XEALL	Units	None
Range	None	Default	None
Attributes	Buffered, Savable in Sequence	See Also	XD, XE, XR, XT

This command erases all defined sequences. ***It should be used with extreme caution.***

Command	Description
> XEALL	Erase all defined sequences

XFK Set Fault or Kill Sequence

Type	Motion	Version	Software Version A
Syntax	<a>XFK<n>	Units	n = sequence number
Range	0 - 100	Default	0
Attributes	Buffered, Savable in Sequence	See Also	K, XR
Response	*n		

This command selects the sequence that will be executed if a fault or kill condition occurs. A selection of 0 causes no sequences to be executed. This command is useful if you want to reset an output in case a fault or kill condition exists. You can also use this command to send a message through the RS-232C interface when a fault or kill condition exists.

The kill condition exists if you issue a Kill (**K**) command over the RS-232C interface or parallel interface. If user fault input becomes active then the **XFK** sequence will execute repetitively.

The following is a list of faults that will call the **XFK** sequence on the SX.

OVER-TEMPERATURE FAULT
MOTOR FAULT
LOW LINE FAULT (BROWNOUT)
COMMANDED MOTOR SHUTDOWN (ST1 or OFF with SSR1)
HARD END OF TRAVEL LIMITS
KILL INPUT
SOFT END OF TRAVEL LIMITS
EXCESSIVE POSITION ERROR
ABSOLUTE ENCODER ROLLOVER
USER FAULT INPUT
K COMMAND

Command	Description
> XFK5	Execute sequence #5 when fault or kill condition exists
> XE5	Erase sequence #5
> XD5	Define sequence #5
1"FAULT_OR_KILL	Send the message
XT	End sequence definition

XG GOTO Sequence

Type	Programming	Version	Software Version A
Syntax	<a>XGn	Units	n = sequence number
Range	1 - 100	Default	None
Attributes	Buffered, Savable in Sequence	See Also	XR, GOTO, GOSUB

This command will jump to a designated sequence for execution. Once you jump to a sequence using **XG**, you cannot return to the sequence from which **XG** originated (unless another **XG** command is executed). To jump to a sequence and return (**GOSUB** operation), you must use the **XR** or **GOSUB** commands. There are no limitations on the number of **XG** commands—no nesting is involved.

Command	Description
> XE1	Erase Sequence #1
> XD1	Define Sequence #1
A2	Sets acceleration to 2 rps^2
V5	Sets velocity to 5 rps
D10000	Sets distance to 10,000 steps
G	Executes the move (Go)
XG5	Go to sequence #5
XT	End defining sequence #1
> XE5	Erase Sequence #5
> XD5	Define sequence #5
1PR	Absolute position report
XT	End sequence #5 definition
> XR1	Execute sequence #1

XQ Sequence Interrupted Run Mode

Type	Set-Up	Version	Software Version A
Syntax	<a>XQ<n>	Units	n = mode
Range	0 = off, 1 = on	Default	0
Attributes	Buffered, Savable in Sequence	See Also	IN, SSJ, XD, XE
Response	*1_INTERRUPTED MODE_ON or *0_INTERRUPTED MODE_OFF		

XQ1: Set Interrupted Run mode
XQ0: Clear Interrupted Run mode

This command can only be used in conjunction with Continuous Scan mode (**SSJ**), and external sequence select lines. If **XQ1** is executed, the SX will not accept a sequence selected from the inputs, until all sequence select lines have become inactive. When all lines are inactive, the SX will then begin reading the sequence select lines and execute the sequence whose number appears there. This mode will continue until an **XQ0** command is executed. You may use the **S** or **K** commands to stop sequence execution.

Command	Description
> XE100	Erase sequence #100
> XD100	Define sequence #100
LD3	Disable CW & CCW limits
IN1B	Sets input as sequence select
SSJ1	Sets Continuous Scan mode
XQ1	Sets Interrupted mode
XT	End sequence #100

XR Run a Sequence

Type	Programming	Version	Software Version A
Syntax	<a>XRn	Units	n = sequence number
Range	1 - 100	Default	None
Attributes	Buffered, Savable in Sequence	See Also	XD, XE, XG, XRP, XT, GOSUB, GOTO

XR executes the commands contained within a designated sequence. **XR** can be used within one sequence to start execution of another sequence. In this respect, **XR** acts like a **GOSUB** as sequence execution will return to the calling sequence. The maximum number of nested **XR** commands is 16.

Command	Description
> XE1	Erase sequence #1
> XD1	Define sequence #1
A10	Sets acceleration to 10 rps^2
AD20	Sets deceleration to 20 rps^2
V5	Sets velocity to 5 rps
D10000	Sets distance to 10,000 steps
G	Executes the move (Go)
XT	End defining sequence #1
> XR1	Execute sequence #1

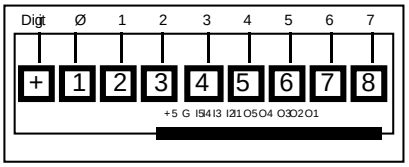
Sequence #1 is defined and executed using **XD1** and **XR1** commands respectively.

XRD Read Sequence via Parallel Input/Output

Type	Motion	Version	Software Version B
Syntax	<a>XRD<c><d><e>	Units	c,d =digit selector, e = scaling factor
Range	c,d = 0 - 7, e = 0 - 2	Default	None
Attributes	Buffered, Savable in Sequence	See Also	IN, OUT, STR, XR, DRD, FRD, LRD, TRD, VRD

This command instructs the SX to read sequence values from Compumotor's TM8 Module (refer to *Chapter 3, Installation* for information on the TM8 Module). Inputs 1-4 must be configured as data inputs. The inputs must be configured as active low (**INL0**) and outputs 1-3 must be configured as strobe outputs to use the TM8 Module.

Helpful Hint: <c> & <d> values represent the TM8 Module digits:



The command syntax allows for digit range selection through the optional <c><d> fields. The <c> field is used to signify the start of the digit range to be read from the TM8 Module. Raw data is in the form **XXX.** The maximum value is 100. The <d> field represents the end of the digit range to be read. The values of these fields can range from 0 to 7 with the <c> field always being less than or equal to the <d> field value.

The <e> field is used to scale the distance value by 10^e. If the <c> and <d> fields are used, the <e> field must be used.

If the <c>, <d>, and <e> fields are not used, the **XRD** command will read all the digits of the TM8 Module. If you are using the TM8 Module, the Output Strobe Delay Time must be set at a value of 10 or greater.

You may use a PLC with the **XRD** command. **XRD** uses a multiplexed I/O scheme. Only output 1 should be configured as a strobe output in this case. Inputs 1-4 still need to configured as data inputs. Output 1 strobes (on/off) through a BCD sequence at the Set Strobe Output Delay Time (**STR**) command rate and reads one BCD digit at a time. The outputs and inputs must be configured as in the TM8 Module read case. **XRD** will issue the following strobe sequence:

01	Data In (Active Low)
low	MSD (Digit 1)
high	LSD (Digit 8)

The PLC, while reading the output strobe, must place the appropriate data at the SX inputs 1 - 4. Input 1 is the digit's LSB and input 4 the MSB. Configuring an input as a data valid line would allow the PLC to control the rate at which the output 1 will strobe through the above sequence.

Example
Turn the TM8's thumbwheels to display the following digits: **+ 0 0 0 0 0 0 1 2**.
Type the following commands

Command	Description
> 1XRD	Run sequence 12

XRP Sequence Run With Pause

Type	Programming	Version	Software Version A
Syntax	<a>XRPn	Units	n = sequence number
Range	1 - 100	Default	None
Attributes	Buffered, Savable in Sequence	See Also	C, XD, XE, XR, XT, PS, GOSUB

This command is identical to the Sequence Run (**XR**) command, except that it automatically generates a pause condition. You must clear this condition with the Continue (**C**) command before the SX executes the command buffer. The pause condition is asserted only if the sequence is valid. This allows you to execute a sequence without the delay of buffering that sequence. **XRP** can be used within one sequence to start execution of another sequence (in this respect, **XRP** acts like **GOSUB**). The maximum number of nested **XRP** commands is 16.

Command	Description
> XE5	Erases sequence #5
> XD5	Defines sequence #5
A10	Sets acceleration to 10 rps^2
V5	Sets velocity to 5 rps
D10000	Sets distance to 10,000 steps

G	Executes the move (Go)
XT	Ends defining sequence #5
> XRP5	Runs sequence #5 with a pause
> C	SX executes sequence #5

XS Sequence Execution Status

Type	Status	Version	Software Version A
Syntax	aXS	Units	None
Range	None	Default	None
Attributes	Immediate, Never Saved	See Also	XTR
Response	*SEQUENCE_n_COMMAND_n or *SEQUENCE_EXECUTION_NOT_IN_PROGRESS		

This command transmits the sequence number and the command currently being executed to the host via the RS-232C interface. If a sequence is not being executed, a status message is transmitted. This command is useful to determine the progress of program execution.

Command	Description
> XE1	Erase sequence #1
> XD1	Define sequence #1
A10	Set acceleration to 10 rps^2
AD20	Set deceleration to 20 rps^2
V5	Set velocity to 5 rps
D200000	Set distance to 20,000 steps
G	Execute the move
XT	End defining sequence #1
> XR1	Execute sequence #1
> 1XS	Request sequence #1 execution status *SEQUENCE_1_COMMAND_V5 is transmitted as that command was being executed when the XS command was issued

XST Sequence Step Mode

Type	Set-Up	Version	Software Version A
Syntax	<a>XST<n>	Units	n = mode
Range	0 = off, 1 = on	Default	0
Attributes	Buffered, Savable in Sequence	See Also	#, XR, XTR, XS
Response	*n_STEP_MODE_ACTIVE or *n_STEP_MODE_INACTIVE		

This command sets the controller into a Sequence Step mode. This command can only be used with the Step (**#n**) command. When you are running a sequence with Sequence Step mode active, every time you issue a Step (**#n**) command, the controller will execute **n** commands in the sequence buffer.

XST1: Sequence Step Mode active

XST0: Sequence Step Mode inactive

Since you need to send a **#** command over the RS-232C interface, this command cannot be run in Stand Alone mode. You must be executing the sequence in RS-232C mode. You must enter a delimiter after the Step (**#**) command to execute the command. If you are in Trace mode (**XTR**) , the controller will execute and display **n** commands every time you enter the **#n** command. This command is useful for troubleshooting your program to see where you are in the program and what takes place with each command. You can use the Kill (**K**) command to abort sequence execution.

Command	Description
> XE1	Erase sequence #1
> XD1	Define sequence #1
A5	Sets acceleration to 5 rps^2
V2	Sets velocity to 2 rps
D10000	Sets distance to 10,000 steps
G	Executes the move (Go)
XT	End defining sequence #1
> XST1	Enable Single-Step mode
> 1XTR1	Enable Trace mode
> XR1	Execute sequence #1
> #	Execute the 1st command
*SEQUENCE_001_COMMAND_A5	Displays the 1st command executed
> #	Execute the 2nd command
*SEQUENCE_001_COMMAND_V2	Displays the 2nd command executed
> #	Execute the 3rd command
*SEQUENCE_001_COMMAND_D10000	Displays the 3rd command executed
> #	Execute the 4th command
*SEQUENCE_001_COMMAND_G	Displays the 4th command executed—motor should have moved 10,000 steps
> #	Execute the 5th command
SEQUENCE_001_COMMAND_XT	Displays the last command executed

XT Sequence Termination

Type	Programming	Version	Software Version A
Syntax	<a>XT	Units	None
Range	None	Default	None
Attributes	Buffered, Savable in Sequence	See Also	XD, XE, XR

XT is a sequence terminator. This command flags the end of the sequence currently being defined. Sequence definition is not complete until this command is issued.

Command	Description
> XE1	Erase sequence #1
> XD1	Define sequence #1
MN	Sets to Normal mode
A10	Sets acceleration to 10 rps^2
V5	Sets velocity to 5 rps
D25000	Sets distance to 25,000 steps
G	Executes the move (Go)
XT	End sequence definition

XTR Set Trace Mode

Type	Set-Up	Version	Software Version A
Syntax	aXTR<n>	Units	n = mode
Range	0 = off, 1 = on	Default	0
Attributes	Buffered, Savable in Sequence	See Also	XR, XST, XS
Response	*n_TRACE_MODE_ACTIVE or *n_TRACE_MODE_INACTIVE		

XTR transmits the command that is being executed from the SX to the host via RS-232C interface. This command only works if you are running a sequence.

XTR1: Enables Trace Mode
XTR0: Disables Trace Mode
Enabling Trace mode transmits the commands and the sequence number being executed. If you have a Loop (**L**), **REPEAT**, or **WHILE** command in a sequence, it will display the iteration count. **XTR** is useful if you wish to see where you are in the program as the program is being executed.

Command	Description
> XE1	Erase sequence #1
> XD1	Define sequence #1
A10	Sets acceleration to 10 rps^2
V5	Sets velocity to 5 rps
D25000	Sets distance to 25,000 steps
L2	Loop 2 times
G	Executes the move (Go)
N	End loop
XT	End defining sequence #1
> 1XTR1	Enable Trace mode
> XR1	Execute sequence #1

Trace mode output is shown below:

```
*SEQUENCE_001_COMMAND_A10
*SEQUENCE_001_COMMAND_V5
*SEQUENCE_001_COMMAND_D25000
*SEQUENCE_001_COMMAND_L2
*SEQUENCE_001_COMMAND_G_LOOP_COUNT_001
*SEQUENCE_001_COMMAND_N_LOOP_COUNT_001
*SEQUENCE_001_COMMAND_G_LOOP_COUNT_002
*SEQUENCE_001_COMMAND_N_LOOP_COUNT_002
*SEQUENCE_001_COMMAND_XT
```

XU Upload Sequence

Type	Status	Version	Software Version A
Syntax	aXU<n>	Units	n = sequence number
Range	1 - 100	Default	None
Attributes	Buffered, Savable in Sequence	See Also	XD, XE, XT, XDIR
Response	*_(sequence contents)		

XU sends sequence *n* to the host via the RS-232C interface preceded by an asterisk (*) and terminated by an extra **[cr]**. All command delimiters in the sequence will be sent out as underscores. If the sequence is empty, ***_ [cr]** is transmitted to the host.

Command	Description
> 1XU1	Upload sequence #1 from unit #1

XWHEN Set WHEN Sequence

Type	Set-Up	Version	Software Version A
Syntax	<a>XWHEN<n>	Units	n = sequence number
Range	0 - 100	Default	0
Attributes	Buffered, Savable in Sequence	See Also	WHEN
Response	*n		

XWHEN selects the sequence that will be executed if a **WHEN** condition is true. **XWHEN0** causes no sequences to be executed. The **WHEN** condition continually examines a set of conditions. If the condition is true, the **XWHEN** sequence is executed. Value saved in non-volatile RAM.

Command	Description
> XWHEN5	Execute sequence #5 if the WHEN condition is true

Y Stop Loop

Type	Programming	Version	Software Version A
Syntax	<a>Y	Units	None
Range	None	Default	None
Attributes	Immediate, Never Saved	See Also	L, LRD, N

The **Y** command takes you out of a loop when the loop completes its current pass. This command does not halt processing of the commands in the loop until the SX reaches the last command of the current loop. At that time, the SX executes the command that follows the End Loop (**N**) command. You cannot restart the command loop unless you enter the entire command structure, including the **L** and **N** commands.

Command	Description
> L	Loops indefinitely
> A10	Sets acceleration to 10 rps^2
> AD20	Sets deceleration to 20 rps^2
> V5	Sets velocity to 5 rps
> D25000	Sets distance to 25,000 steps
> T2	Waits 2 seconds
> G	Executes the move (Go)
> N	Ends loop
> Y	Stops loop

Z Reset

Type	Programming	Version	Software Version A
Syntax	<a>**Z**	Units	None
Range	None	Default	None
Attributes	Immediate, Never Saved	See Also	IN

The **Z** command is equivalent to cycling power to the SX. This command returns all internal settings to their power up values. It clears the command buffer. Like the Kill (**K**) command, the **Z** command immediately terminates motion. When you use the **Z** command, the SX is busy for about 2 seconds and ignores all commands. Command execution is delayed for approximately one second after a reset command.

Index