

SV

NAME: Save Tuning Parameters
SYNTAX: <a>**SV**
DEFAULT: None
RANGE: None
VALID:
RESPONSE: None
TYPES: Programming
ATTRIBUTES: Immediate, Device Address Optional, Not Saved
DESCRIPTION: The **SV** command saves only the tuning parameters in nonvolatile memory.

Those parameters requiring the Save (**SV**) command are:

Integral Gain (**CIG, BCIG**)
 Integral Maximum (**CIM, BCIM**)
 Proportional Gain (**CPG, BCPG**)
 Proportional Maximum (**CPM, BCPM**)
 Differential Gain (**CDG, BCDG**)
 Differential Maximum (**CDM, BCDM**)
 Integral Sum Limit (**CIL, BCIL**)

SEE ALSO: None

EXAMPLES:

Command
 > **CIG50**
 > **SV**

Description

Set integral gain to 50% of the maximum value
 Unit saved all the changes in nonvolatile memory

T

NAME: Time
SYNTAX: <a>**Tn**
DEFAULT: None
RANGE: n = 0.01 to 999.99 seconds
VALID:
RESPONSE: None
TYPE: Programming
ATTRIBUTES: Buffered, Device Address Optional, Savable in Sequence
DESCRIPTION: The Time (**T**) command causes the JSI 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.

SEE ALSO: None

EXAMPLES:

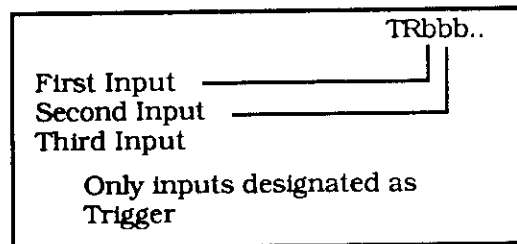
Command
 > **MN**
 > **A5**
 > **V5**
 > **D25000**
 > **T10**
 > **G**
 > **T5**
 > **G**

Description

Sets mode to normal
 Sets acceleration to 5 rps²
 Sets velocity to 5 rps
 Sets distance to 25,000 steps
 Pauses for 10 seconds
 Executes the move (Go)
 Pauses for 5 seconds after the move ends
 Executes the move (Go)

TR

NAME: Wait for Trigger
SYNTAX: <a>TRb
DEFAULT: None
RANGE: b = 0, 1, or x
VALID:
RESPONSE: None
TYPE: Programming
ATTRIBUTES: Buffered, Device Address Optional, Never Saved
DESCRIPTION: When the **TR** command is used, the JSI will get to this command and wait until the input (configured as triggers with the **IN** command) match the pattern.. It will not scrutinize those inputs not designated as triggers, before going on to the next command. The voltage level that represents an active state is defined with the **INL** command. The order of comparison as designated trigger inputs from Input 1 to Input 13 (as labeled on the unit).



Triggers are used to synchronize JSI operations with external events. They can be used to implement a handshaking function with other devices. Three characters are used for bbbbb variables are listed below:

1 = Input active(ON)
 0 = Input inactive (OFF)
 x = Don't care

SEE ALSO:
EXAMPLES:

D, TS, IN, INLCommandDescription

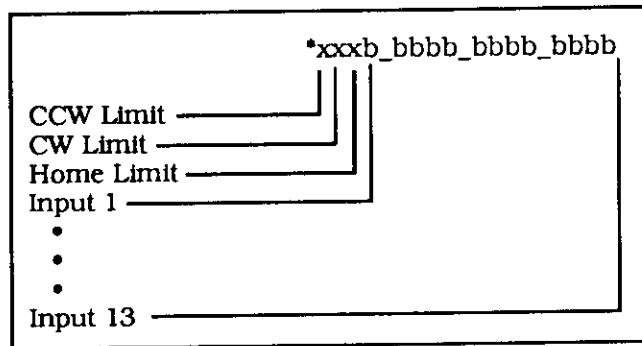
> IN2A	Configure input 2 as a trigger
> IN3A	Configure input 3 as a trigger
> IN7A	Configure input 7 as a trigger
> TR1X0	Wait for input 2 to be active and input 7 to be inactive before going on to the next command. Input 3 is ignored.
> A10	Sets acceleration to 10 rps ²
> V5	Sets velocity to 5 rps
> D25000	Sets distance to 25,000 steps
> G	Executes the move (Go)

TS

NAME: Trigger Input Status
 SYNTAX: a**TS**
 DEFAULT: None
 RANGE: b = 0, 1, x
 VALID:
 RESPONSE: a**TS** *xxx**b**_bbbb_bbbb_bbbb[cr]
 TYPE: Status
 ATTRIBUTES: Immediate, Device Address Required, Not Saved
 DESCRIPTION: This command reports the state of the trigger inputs. Response is in the form "xxx**b**_bbbb_bbbb_bbbb[cr]" where

b= 1 (Input Active)
 b = 0 (Input Inactive)
 b = x (Input not configured as a trigger)

TS command 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, you can check with **TS** command.



SEE ALSO: **TR, IN, INL**
 EXAMPLES: **Command**
 > **ITS**

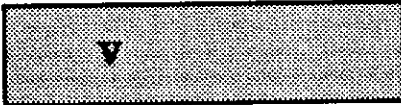
Description
 *xxx1_xxxx_11xx_xx00[cr]
 Inputs 1, 6, 7, 12 and 13 are configured as triggers with 1, 6 and 7 active and 12 and 13 inactive.

U

NAME: Pause and Wait for Continue
 SYNTAX: <a>**U**
 DEFAULT: None
 RANGE: None
 VALID:
 RESPONSE: None
 TYPE: Programming
 ATTRIBUTES: Immediate, Device Address Optional, Never Saved
 DESCRIPTION: This command causes the JSI to complete the command in progress, then wait until it receives a Continue (C) to resume processing. Since the buffer is saved, the JSI continues to execute the program (at the point where it was interrupted). The JSI continues processing when it receives the **C** command. This command is typically used to stop a machine while it is unattended.

SEE ALSO:	PS, C	
EXAMPLES:	<u>Command</u>	<u>Description</u>
	> MN	Sets move to Normal mode
	> A5	Sets acceleration to 5 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 JSI receives the Continue command.

This command string pauses at the point where the **U** command is entered. A Continue (**C**) command causes execution to resume at the point where it was paused. In this example, the loop stops at the end of a move, and resumes when the JSI 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.



NAME:	Velocity
SYNTAX:	<a> V n
DEFAULT:	n = 1 rps
RANGE:	n = 0 to 99.999 rps
VALID:	
RESPONSE:	a V * V n[cr]
TYPE:	Motion
ATTRIBUTES:	Buffered, Device Address Optional, Savable in Sequence
DESCRIPTION:	The velocity (V) command defines the maximum speed at which the motion will occur when given the Go (G) command. The value is stored in nonvolatile memory. Entering a velocity that is beyond the capabilities of your system will generate a following error. If the following error exceeds the CPE value, the JSI will fault and disable its control output.

SEE ALSO:	A, D, G	
EXAMPLES:	<u>Command</u>	<u>Description</u>
	> MC	Sets move to continuous
	> A5	Sets acceleration to 5 rps ²
	> V5	Sets velocity to 5 rps
	> G	Go (Begin motion)

In preset mode, Mode Normal (**MN**) the maximum velocity may also be limited when the resulting move profile is triangular. In Mode Continuous (**MC**), a Go (**G**) command is completed—the JSI moves on to the next command in the buffer—once the specified velocity is reached.

VRD

NAME: Read Velocity Value from Parallel Input/Output
SYNTAX: <a>**VRD**
DEFAULT: None
RANGE:
VALID:
RESPONSE: None
TYPE: Programming
ATTRIBUTES: Buffered, Device Address Optional, Savable in Sequence
DESCRIPTION: This command initiates the controller to read velocity values from the parallel inputs. You must set up the inputs as Data Inputs using the **IN** command. You must also set up outputs as strobe outputs. Typically, you would set up 8 inputs as DATA Inputs and 5 Outputs as strobe outputs using the **OUT** command. You must also set up the strobe output delay time (typically greater than a PLC Scan Time if using a PLC, or a minimal debounce time if using thumbwheel switches) so that the device you are getting data from will know when the JSI is ready for data. Sequence of events take place when we enter the **VRD** command:

- Step 1** The lowest Output bit designated as strobe output will go on and stay on for strobe output delay time defined by **STR** command.
- Step 1A** If an input is designated as a data valid input, then the strobe output will stay active until the data valid input is active.
- Step 2** The JSI will read the parallel inputs designated as DATA inputs. The lowest inputs will be the least significant bit. The JSI reads 8 bits (2 BCD digits) and stores them into two digits on the velocity register.
- Step 3** Next Strobe output bit designated as strobe output will go on and stay on for delay time defined by **STR** command.
- Step 3A** If an input is designated as a data valid input, then the strobe output will stay active until the data valid input is active.
- Step 4** The JSI will read the parallel inputs designated as DATA inputs are placed in the lowest digits on the velocity register. Previously entered values are shifted two positions to the left.
- Step 5** Steps 3 through 4 are repeated as many times as the strobe outputs are available. You can use up to 5 different outputs as strobe outputs. The last two digits read are the least significant digit of the loop value.

The velocity range you can enter via parallel interface is from .00001 to 99.99999.

SEE ALSO: **STR, IN, OUT**
EXAMPLE: The following example shows how the JSI reads the velocity value of 35.89721. We will use inputs 1 through 8 for DATA inputs and outputs 1 through 4 for Strobe Outputs.

- Step 1** Type the following:

<u>Command</u>	<u>Description</u>
OUT1J	Set Output 1 as Strobe Output
OUT2J	Set Output 2 as Strobe Output
OUT3J	Set Output 3 as Strobe Output
OUT4J	Set Output 4 as Strobe Output
IN1N	Set Input 1 as a Least Significant DATA input
IN2N	Set input 2 as a DATA input
IN3N	Set input 3 as a DATA input
IN4N	Set input 4 as a DATA input
IN5N	Set input 5 as a DATA input
IN6N	Set input 6 as a DATA input
IN7N	Set input 7 as a DATA input
IN8N	Set input 8 as a the Most significant DATA input
STR500	Set strobe output delay time to 500 msec.
A10	Sets acceleration to 10 rps ²
V5	Set velocity to 5 rps
D2000	Set Distance to 2,000 steps

Step 2 As soon as you enter the Read Velocity via Parallel Input (VRD) command, the following should happen

Type the following:

Command	Description
> VRD	Start reading velocity from Parallel Input
> A20	Set acceleration to 20 rps ²
> D100000	Set distance to 100,000
> G	Execute the move (Go)

Step 3 The JSI will turn on OUT1 for 500 msec.

Step 4 You must have your inputs 1 through 8 to the following settings.

IN8	IN7	IN6	IN5	IN4	IN3	IN2	IN1
OFF	OFF	OFF	OFF	OFF	OFF	ON	ON
						0	3

The current setting of the velocity is
00.00003 rps

Step 5 After the JSI reads the input, OUT1 turns off and OUT2 will automatically go on for 500 msec.

Step 6 You must have your inputs 1 through 8 set to the following settings.

IN8	IN7	IN6	IN5	IN4	IN3	IN2	IN1	
OFF	ON	OFF	ON	ON	OFF	OFF	OFF	
				5				8

Previously entered value of 03 shifts 2 locations to the left and 58 is placed on the 2 lowest position. The current setting of the velocity value is:
00.00358 rps

Step 7 OUT2 turns off and OUT3 automatically goes on for 500 msec.

Step 8 You must have your inputs 1 through 8 set to the following settings:

IN8	IN7	IN6	IN5	IN4	IN3	IN2	IN1	
ON	OFF	OFF	ON	OFF	ON	ON	ON	
			9					7

Previously entered values 0358 shifts 2 locations to the left and 97 is placed on the two lowest positions. The current setting of the velocity value is:
00.35897 rps

Step 9 OUT 3 turns off and OUT4 automatically goes on for 500 msec.

Step 11 You must have your inputs 1 through 8 set to the following settings.

IN8	IN7	IN6	IN5	IN4	IN3	IN2	IN1
OFF	OFF	ON	OFF	OFF	OFF	OFF	ON
		2					1

Previously entered values 035897 shifts two locations tooth left and 21 is placed on the two lowest positions. The final setting of the velocity value becomes
35.89721 rps

Step 12 Since the fifth strobe output is not defined, the JSI now goes to the next command in the buffer and executes the move at a velocity of 35.89721 rps.

W1

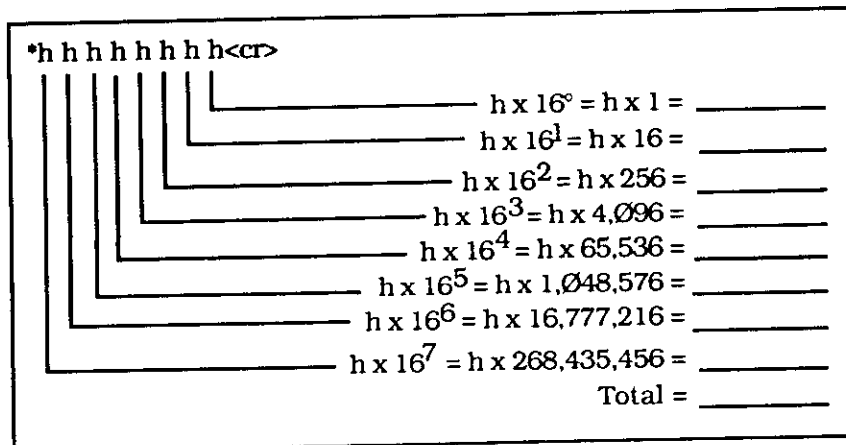
NAME: Signed Binary Position Report
 SYNTAX: aW1
 DEFAULT: None
 RANGE: b = 8 bit binary number
 VALID:
 RESPONSE: aW1 bbbb[space]
 TYPE: Status
 ATTRIBUTES: Immediate, Device Address Required, Never Saved
 DESCRIPTION: Report back gives immediate binary representation of position relative to start of the current move. The format of the response is a four character response (bbbb) 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. Move in negative direction will report back negative number (bit 31 is set to 1).

The transmission is not preceded with an asterisk (*) to maximize response time. Do not use the W1 command when multiple JSI units are daisy-chained.

SEE ALSO: W2, W3, PR

W2

NAME: Hexadecimal Position Report
 SYNTAX: aW2
 DEFAULT: None
 RANGE: h = hexadecimal number 0 - 9 and A - F
 VALID:
 RESPONSE: aW2 *hhhhhhh<cr>
 TYPE: Status
 ATTRIBUTES: Immediate, Device Address Required, Never Saved
 DESCRIPTION: The immediate hexadecimal character position report back (while motion is occurring) indicates position relative to start of current move. The format of the response is eight (8) hexadecimal characters. Your host computer needs to convert these characters into useable format. This command does not indicate the direction of the movement; you will only receive an unsigned number.



If the first digit of the response is an "F"

Fhhhhhh

then response represents a "two's complement" negative number Any response but Fhhhhhh should be interpreted per the **W2** command.

To interpret a negative number (starting with Fhhhhhh)

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: Subtract the hexadecimal number from 16^8 (2^{32}) (4,294,967,296).

The easy way:

1. Chop off all the leading "F"s, and convert to decimal
2. Convert and subtract the next largest power of 16.

Example: the JSI responds to W3 as follows:

*FFFF9E58

- | | | | |
|-------------------------|-----------|---|---------|
| 1. Chop off the Fs: | 9E58 hex | = | 40,536 |
| 2. Subtract from 16^4 | 10000 hex | = | 65,536 |
| | Results | = | -25,000 |

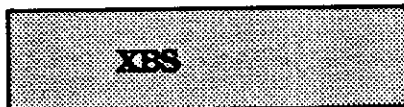
SEE ALSO:
EXAMPLES:

W1, PR

Command
> **IW3 *FFFA19C**

Description

In the current move, you are at -24163 steps from initiation of move.



NAME:
SYNTAX:
DEFAULT:
RANGE:
VALID:
RESPONSE:
TYPE:
ATTRIBUTES:
DESCRIPTION:

Sequence Memory Available Report

aXBS
None
 $n = 0 - 6000$ $x = 0 - 100$

aXBS *n_OF_8000_BYTES_(X%)_SEQUENCE_MEMORY_REMAINING[CR]
Status

Immediate, Device Address Required
This command reports the remaining amount of memory that can be used for sequence storage. The total space available for sequence storage is 8000 bytes (characters). This command is useful to find out how much more programming can be done on the JSI, after defining several programs.

This command reports both number of bytes available, and the percentage (%) of memory available. The status of the memory using the **B** and **BS** commands do not tell you the status of the sequence buffer.

SEE ALSO:
EXAMPLES:

B, BS
Command
1XBS

Description

***4000_of_8000_Bytes_(50%)_Sequence_Memory_Remaining**

There are 4000 bytes (50%) of the sequence buffer remaining for you to program.

XC

NAME: Sequence Checksum Report
SYNTAX: aXC
DEFAULT: None
RANGE: n = 0 - 255
VALID:
RESPONSE: aXC *n[cr]
TYPE: Status
ATTRIBUTES: Buffered, Device Address Required, Savable in Sequence
DESCRIPTION: 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.

SEE ALSO: **XD, XE**
EXAMPLES:

<u>Command</u>	<u>Description</u>
> IXC	Response = *00149

XD

NAME: Sequence Definition
SYNTAX: <a>XDn
DEFAULT: None
RANGE: n = 1 to 100
VALID:
RESPONSE: None
TYPE: Programming
ATTRIBUTES: Buffered, Device Address Optional, Savable in Sequence
DESCRIPTION: This command begins sequence definition for a specific sequence. All the commands between the **XD** command and the Sequence Termination (**XT**) command 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.

SEE ALSO: **XT, XE, XR**
EXAMPLES:

<u>Command</u>	<u>Description</u>
> XE1	Erase sequence #1
> XD1	Define sequence #1
> MN	Sets mode normal
> A10	Sets acceleration to 10 rps
> 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

NAME: Sequence Directory
 SYNTAX: **aXDIR**
 DEFAULT: None
 RANGE: None
 VALID:
 RESPONSE: See Description
 TYPE: Status
 ATTRIBUTES: Immediate, Device Address Required, Never Saved
 DESCRIPTION: This command reports the sequence number (sequence 1 to 100), and the amount of memory used by each sequence. The response is in the following format:

```

XDIR *No_Sequences_Defined - If there are no sequences
defined
XDIR *Sequence_n_uses_x_bytes
*Sequence_n_uses_x_bytes
    
```

If sequence exists, it would list the sequence number and the number of bytes used by the sequence.

SEE ALSO: **XT, XE, XD, XBS**

EXAMPLE:	XDIR	<u>Description</u> *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

NAME: Sequence Erase
 SYNTAX: **<a>XEn**
 DEFAULT: None
 RANGE: n = 1-100
 VALID:
 RESPONSE: None
 TYPE: Programming
 ATTRIBUTES: Buffered, Device Address Optional, Savable in Sequence
 DESCRIPTION: 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 as the sequence is irretrievable

SEE ALSO:	XD, XT, XR	
EXAMPLES:	> XE1	<u>Description</u> Deletes Sequence 1

XFK

NAME: Set Fault or Kill Sequence
SYNTAX: <a>**XFK**n
DEFAULT: n = 0
RANGE: n = 0-99
VALID:
RESPONSE: a**XFK** *nn[cr]
TYPE: Setup
ATTRIBUTES: Buffered, Device Address Optional;
DESCRIPTION: This command selects the sequence that will be executed is 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 fault conditions are listed below:

LED Display Code	Condition
20	EXCESSIVE POSITION ERROR
23	DRIVE ENABLE NOT ACTIVE
30	EEPROM CHECKSUM ERROR
41	CW LIMIT SWITCH ENGAGED
42	CCW LIMIT SWITCH ENGAGED
43	CW SOFTWARE LIMIT ENGAGED
44	CCW SOFTWARE LIMIT ENGAGED
66	USER FAULT

The Kill condition exists is you issue a Kill (**K**) command over the RS-232C interface or parallel interface.

SEE ALSO; **XR, K**
EXAMPLE:

Command	Description
> XFK5	Execute sequence 5 when fault or kill condition exists
> XES	Erase Sequence 5
> XD5	Define Sequence 5
" FAULT_OR_KILL "	Send the message
> XT	End Sequence Definition

Whenever a Fault or Kill occurs, the Controller will display "FAULT_OR_KILL"

XG

NAME: Goto Sequence
SYNTAX: <a>**XG**n
DEFAULT: None
RANGE: n = 1-99
VALID:
RESPONSE: None
TYPE: Programming
ATTRIBUTES: Buffered, Device Address Optional, Savable in Sequence
DESCRIPTION: This command will jump to a designated sequence for execution. Once you jump to a sequence using the **XG** command, you can not return to the sequence from which the **XG** originated (unless another **XG** command is

executed). To jump to a sequence and return (GOSUB operation), you must use the XR command. There are no limitations on the number of XG commands as there is no nesting involved.

SEE ALSO;
EXAMPLES:

<u>Command</u>	<u>Description</u>
> 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
> IPR	Absolute Position Report
> XT	End sequence 5 definition
> XR1	Execute sequence #1

The motor will move 10000 steps, then the controller will jump to sequence 5 and indicate the absolute position.



NAME: Sequence Interrupted Run Mode
SYNTAX: <a>XQb
DEFAULT: b = 0
RANGE: b = 0, 1
VALID:
RESPONSE: aXQ *0__INTERRUPTED_RUN_MODE_OFF[cr] or *1__INTERRUPTED_RUN_MODE_ON[cr]
TYPE: Setup
ATTRIBUTES: Buffered, Device Address Optional, Savable in Sequence
DESCRIPTION: Set interrupted run mode (b = 1)
 Clear interrupted run mode (b = 0)
 This command can only be used in conjunction with continuous mode (SSJ), and external sequence select lines. If XQ1 is executed, the JSI will not accept a sequence select inputs, until all sequence select lines have been brought inactive. After all lines have simultaneously been brought inactive, the JSI will then read the sequence select lines and execute the sequence whose number appears there. This interrupted mode will continue until an XQ0 command is executed. You may use S or K command to stop sequence execution.

SEE ALSO:
EXAMPLES:

<u>Command</u>	<u>Description</u>
> XE100	Erase sequence #100
> XD100	Define sequence #100
> LD3	Disable CW & CCW limits
> SSJ1	Sets continuous mode
> XQ1	Sets interrupted mode
> XT	End Sequence #100

When the JSI powers up, sequence 100 will be executed. Interrupted run mode will be set. Sequence select input lines all need to go inactive before selecting any sequences

XR

NAME: Run a Sequence
SYNTAX: <a>**XR**n
DEFAULT: None
RANGE: n = 1 to 100
VALID:
RESPONSE: None
TYPE: Programming
ATTRIBUTES: Buffered, Device Address Optional, Savable in Sequence
DESCRIPTION: This command executes the commands contained within the designated sequence.

An **XR** command can be used within one sequence to start execution of another sequence. In this respect an **XR** acts like a GOSUB as sequence execution will return to the calling sequence. The maximum number of nested **XR** commands is 16.

SEE ALSO: **XE, XD, XT, XRP, XG**

<u>Command</u>	<u>Description</u>
> XE1	Erase sequence #1
> XD1	Define sequence #1
> 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	End defining sequence #1
> XR1	Execute sequence #1

Sequence 1 is defined and executed using **XD1** and **XR1** commands respectively

XRD

NAME: Read Sequence via Parallel Input/Output
SYNTAX: <a>**XRD**
DEFAULT: None
RANGE: None
VALID:
RESPONSE: None
TYPE: Programming
ATTRIBUTES: Buffered, Device Address Optional, Savable in Sequence
DESCRIPTION: This command initiates the controller to read the sequence value from the parallel inputs from within a sequence. You must set up the inputs as Data input using the **IN** command. You must also set up outputs as strobe outputs. Typically, you would set up 8 inputs as DATA Inputs and 5 Outputs as strobe outputs using the **OUT** command. However if only **XRD** is used, then only 1 strobe is required.

You must also set up the strobe output delay time (typically greater than a **PLC** Scan Time if using a PLC, or a minimal debounce time if using thumbwheel switches) so that the device you are getting data from will know when the JSI is ready form data. Sequence of events take place when we enter the **XRD** command

- Step 1** The lowest Output bit designated as strobe output will go on and stay on for strobe output delay time defined by **STR** command.
- Step 1A** If an input is designated as a data valid input, then the strobe output will stay active until the data valid input is active.
- Step 2** The JSI will read the parallel inputs designated as DATA inputs. The lowest inputs will be the least significant bit. The JSI reads 8 bits (2 BCD digits) and stores them into two digits on the sequence register.
- Step 3** Next Strobe output bit designated as strobe output will go on and stay on for delay time defined by **STR** command.
- Step 3A** If an input is designated as a data valid input, then the strobe output will stay active until the data valid input is active.
- Step 4** The JSI will read the parallel inputs designated as DATA inputs are placed in the lowest digits on the sequence register. Previously entered values are shifted two positions to the left.
- Step 5** Steps 3 through 4 are repeated as many times as the strobe outputs are available. You can use up to 5 different outputs as strobe outputs. The last two digits read are the least significant digit of the sequence register.

In most cases you would use **SSJ** command to execute sequences from remote inputs. However, **SSJ** lets you select sequences only when the controller is not currently executing a sequence. You must use **XRD** command if you wish to execute a sequence remotely from within a sequence.

SEE ALSO:
EXAMPLE:

STR, IN, OUT

The following example shows how the JSI reads the sequence value of 21. We will use inputs 1 through 8 for DATA inputs and outputs 1 through 4 for Strobe Outputs.

- Step 1** Type the following:

<u>Command</u>	<u>Description</u>
> XE21	Erase sequence #21
> XD21	Define sequence #21
A20	Sets acceleration to 20 rps ²
V5	Sets acceleration to 5 rps
D25000	Sets distance to 25,000 steps
G	Executes the move (Go)
> XT	End defining sequence #1
OUT1J	Set Output 1 as Strobe Output
OUT2J	Set Output 2 as Strobe Output
OUT3J	Set Output 3 as Strobe Output
OUT4J	Set Output 4 as Strobe Output
IN1N	Set Input 1 as a Least Significant DATA input
IN2N	Set input 2 as a DATA input
IN3N	Set input 3 as a DATA input
IN4N	Set input 4 as a DATA input
IN5N	Set input 5 as a DATA input
IN6N	Set input 6 as a DATA input
IN7N	Set input 7 as a DATA input
IN8N	Set input 8 as a the Most significant DATA input
STR500	Set strobe output delay time to 500 msec.
A10	Sets acceleration to 10 rps ²
V5	Set velocity to 5 rps

Step 2 As soon as you enter the Read Distance via Parallel Input (**XRD**) command, the following should happen

Type the following:

<u>Command</u>	<u>Description</u>
XRD	Start reading from Parallel input

Step 3 The JSI will turn on OUT1 for 500 msec.

Step 4 You must have your inputs 1 through 8 to the following settings.

IN8	IN7	IN6	IN5	IN4	IN3	IN2	IN1
OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF
		Ø				Ø	

The current setting of the sequence is
ØØØØØØØØØØ

Step 5 After the JSI reads the input, OUT1 turns off and OUT2 will automatically go on for 500 msec.

Step 6 You must have your inputs 1 through 8 set to the following settings.

IN8	IN7	IN6	IN5	IN4	IN3	IN2	IN1
OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF
		Ø				Ø	

Previously entered value of ØØØØ shifts 2 locations to the left and ØØ is placed on the 2 lowest positions. The current setting of the sequence value is still
ØØØØØØØØØØ

Step 7 OUT2 turns off and OUT3 automatically goes on for 500 msec.

Step 8 You must have your inputs 1 through 8 set to the following settings:

IN8	IN7	IN6	IN5	IN4	IN3	IN2	IN1
OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF
		Ø				Ø	

Previously entered values Ø shifts 2 locations to the left and another Ø is placed on the two lowest positions. The current setting of the sequence value is still:
ØØØØØØØØØØ

Step 9 OUT3 turns off and OUT4 automatically goes on for 500 msec.

Step 10 You must have your inputs 1 through 8 set to the following settings.

IN8	IN7	IN6	IN5	IN4	IN3	IN2	IN1
OFF	OFF	ON	OFF	OFF	OFF	OFF	ON
		2				1	

Previously entered values ØØØØØØØØ shifts two locations to the left and 21 is placed on the two lowest positions. The final setting of the sequence value becomes
21

Step 11 The JSI now checks to see if there is a fifth strobe output. Since we do not have a fifth output, the JSI controller executes sequence 21 by moving the motor 25,000 steps.

XRP

NAME: Sequence Run With Pause
 SYNTAX: <a>**XRP**n
 DEFAULT: None
 RANGE: n = 1 to 100
 VALID: None
 RESPONSE: None
 TYPE: Programming
 ATTRIBUTES: Buffered, Device Address Optional, Savable in Sequence
 DESCRIPTION: 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 JSI 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. An **XRP** command can be used within one sequence to start execution of another sequence (in this respect an **XRP** acts like a **GOSUB**). The maximum number of nested **XRP** commands is 16.

SEE ALSO: **XR, XD, XT, XE, C**

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	JSI executes Sequence #5

Upon issuing **XRP5**, Sequence #5 is entered into the command buffer, but is not executed. You must issue a Continue (**C**) command to execute Sequence #5.

XSD

NAME: Sequence Status Definition Report
 SYNTAX: a**XSD**
 DEFAULT: None
 RANGE: n = 0, 1, 2
 VALID: None
 RESPONSE: a**XSD** *n[cr]
 TYPE: Status
 ATTRIBUTES: Buffered, Device Address Required, Savable in Sequence
 DESCRIPTION: This command reports back the status of previous sequence definition (**XD...XT**). The response is in the form *n[cr]. The valid values and descriptions of n are shown below:

- 0 = Download O.K.
- 1 = A sequence already exists with the number you have specified.
- 2 = Out of memory. The sequence buffer is full.

The **XSD** command is useful for verifying that the last sequence definition attempt was successful.

SEE ALSO: **XD, XE, XT**

Command	Description
> 1XSD	*1 When you issued the XSD command to device #1, the response was 1. This means that you need to erase the existing sequence to define that specific sequence.

XSR

NAME: Sequence Status Run Report
SYNTAX:
DEFAULT: None
RANGE: n = 0, 1, 2
VALID:
RESPONSE: a**XSR** *n[cr]
TYPE: Status
ATTRIBUTES: Immediate, Device Address Required, Never Saved
DESCRIPTION: This command allows you to check whether or not the last sequence you issued was started successfully. The valid values and descriptions for n are shown below.

0 = Last attempt to run the sequence was
 2 = Invalid sequence number was requested

SEE ALSO: **XR, XRP**
EXAMPLES:

<u>Command</u>	<u>Description</u>
> XR2	Runs Sequence 2
> LXSR	*Ø[cr] Sequence started OK

XSS

NAME: Sequence Status Report
SYNTAX: a**XSS**n
DEFAULT: None
RANGE: n = 1-100, x = 0, 1, 2
VALID:
RESPONSE: a**XSS**n *x[cr]
TYPE: Status
ATTRIBUTES: Buffered, Device Address Required, Savable in Sequence
DESCRIPTION: This command reports whether the sequence is empty, has a bad checksum, or is OK.

The valid values and descriptions of x are shown below:

0 = Empty
 1 = Bad Checksum
 2 = OK.

This command is useful to see if the particular sequence exists and if that portion of memory has been corrupted or not.

SEE ALSO: **XD, XT, XE**
EXAMPLES:

<u>Command</u>	<u>Description</u>
> LXSS1	*Ø[cr] Nothing programmed in sequence 1

XST

NAME: Single Step Mode
SYNTAX: <a>**XST**n
DEFAULT: n = 0
RANGE: n = 0, 1, x = ACTIVE, INACTIVE
VALID:
RESPONSE: a**XST** *n_STEP_MODE_X[cr]
TYPE: Setup
ATTRIBUTES: Buffered, Device Address Optional, Savable in Sequence
DESCRIPTION: This command sets the controller into a single step mode. This command can only be used with the Step (#) command. When you are running a sequence with the single step mode active, every time you issue a Step (#) command, the controller will execute one command in the sequence buffer.

XST1 Single Step Mode active
XST0 Single 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 the Trace (**XTR**) mode, the controller will display one command every time you enter the # 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 the sequence execution.

SEE ALSO:
EXAMPLES:

<u>Command</u>	<u>Description</u>
> 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
> XITR1	Enable trace mode
> XR1	Execute sequence 1
> #	Execute the first command
*SEQUENCE_001_COMMAND_A5	Displays the first command executed
> #	Execute the second command
*SEQUENCE_001_COMMAND_V2	Displays the second command executed
> #	Execute the third command
*SEQUENCE_001_COMMAND_D10000	Displays the third command executed
> #	Execute the fourth command
*SEQUENCE_001_COMMAND_G	Displays the fourth command executed
	Motor should have moved 10,000 steps.
> #	Execute the fifth command
*SEQUENCE_001_COMMAND_XT	Displays the last command executed

XT

NAME: Sequence Termination
SYNTAX: <a>**XT**
DEFAULT: None
RANGE: None
VALID:
RESPONSE: None
TYPE: Programming
ATTRIBUTES: Buffered, Device Address Optional, Savable in Sequence
DESCRIPTION: The **XT** command 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.

SEE ALSO: **XD, XE, XR**
EXAMPLES:

<u>Command</u>	<u>Description</u>
> XD1	Define sequence #1
> MN	Sets move to mode normal
> 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

NAME: Set Trace Mode
SYNTAX: <a>**XTRn**
DEFAULT: n=0
RANGE: n = 0, 1, x=ACTIVE, INACTIVE
VALID:
RESPONSE: a**XTR** *n_TRACE_MODE_x
TYPE: Setup
ATTRIBUTES: Buffered, Device Address Required, Savable in Sequence
DESCRIPTION: This command transmits the command that is being executed from the JSI to the host via RS-232C interface. This command only works if you are running a sequence.

aXTR1 Enables Trace Mode
aXTR0 Disables Trace Mode

Enabling trace mode transmits the commands and the sequence number being executed. If you have a Loop (**L**) command in a sequence, it will also display the loop count.

This command is useful if the user wishes to see where you are in the program as the program is being executed.

SEE ALSO:
EXAMPLES:

<u>Command</u>	<u>Description</u>
> 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
> X1TR1	Enable trace mode
> XR1	Execute sequence 1

After turning on the trace mode, as you run the sequence (XR1), the controller will display the current command being executed. The 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
```

KU

NAME: Upload Sequence
SYNTAX: aXUn
DEFAULT: None
RANGE: n = 1 to 100
VALID:
RESPONSE: aXUn uploads the sequence stored
ATTRIBUTES: Buffered, Device Address Required, Savable in Sequence
DESCRIPTION: Upload sequence. The sequence n will be sent to the host, via RS-232C interface preceded with 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, then *_[cr] is transmitted to the host.

SEE ALSO: XD, XE, XT
EXAMPLES:

<u>Command</u>	<u>Description</u>
> 1XU1	Upload sequence #1 from unit #1

Y

NAME: Stop Loop
SYNTAX: <a>Y
DEFAULT: None
RANGE: None
VALID:
RESPONSE: None
TYPE: Programming
ATTRIBUTES: Immediate, Device Address Optional, Never Saved
DESCRIPTION: The Stop Loop (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 JSI reaches the last command of the current loop. At that time, the JSI 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 Loop (L) and End Loop (N) commands.

SEE ALSO: L, N, LRD
EXAMPLES:

<u>Command</u>	<u>Description</u>
> L	Loops indefinitely
> A10	Sets acceleration to 10 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

The loop requires the motor to move 25,000 steps CW and then wait for 2 seconds. The loop terminates at the end of the loop cycle it is executing when it receives the Y command.

**Z**

NAME: Reset
SYNTAX: <a>**Z**
DEFAULT: None
RANGE: None
VALID:
RESPONSE: None
TYPE: Programming
ATTRIBUTES: Immediate, Device Address Optional, Never Saved
DESCRIPTION: The Reset (**Z**) command is equivalent to cycling AC power to the JSI. 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 Reset command, the JSI is busy for 2,500 milliseconds and ignores all commands.

Any changes to the tuning parameters that you do not save before issuing this command will be lost.

This command sets all position counters to zero.

SEE ALSO: **S, K**
EXAMPLES: None

