

MC Motion	Mode Continuous			VALID Software Version A
SYNTAX <a>MC	UNITS None	RANGE None	DEFAULT None	ATTRIBUTES Buffered Savable in Sequence
EXECUTION TIME		SEE ALSO A, AD, MN, MPP, T, TR, V		

Description

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 500 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.

Example

Command	Description
> MC	Sets mode to continuous
> MPP	Sets mode to profile
> 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)

The motor turns at 5 rps until it is halted by the Stop (S) command, Kill (K) command, a limit switch, or by a new velocity specification.

MN Motion	Mode Normal			VALID Software Version A
SYNTAX <a>MN	UNITS None	RANGE None	DEFAULT None	ATTRIBUTES Buffered Savable in Sequence
EXECUTION TIME		SEE ALSO A, D, MC, MPA, MPI, MPP, V, AD		

Description

The Mode Normal (MN) command sets the positioning mode to preset. In Mode Normal, 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 the Distance (D). The MN command is used to change the mode of operation from Mode Continuous (MC).

Example

Command	Description
> MN	Set positioning mode to preset
> 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)

Motor turns 1,000 steps in the CW direction after the G command is issued.

MPA Set-Up	Mode Position Absolute			VALID Software Version A
SYNTAX <a>MPA	UNITS None	RANGE None	DEFAULT None	ATTRIBUTES Buffered Savable in Sequence
EXECUTION TIME		SEE ALSO D, MN, MPI, MPP, PZ, SP		

Description

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

Mode Position Absolute (MPA) is most 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.

Example

<u>Command</u>	<u>Description</u>
> MN	Set mode normal (preset).
> MPA	Set position mode absolute.
> 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 Set-Up	Mode Position Incremental			VALID Software Version A
SYNTAX <a>MPI	UNITS None	RANGE None	DEFAULT None	ATTRIBUTES Buffered Savable in Sequence
EXECUTION TIME		SEE ALSO D, MN, MPA, MPP		

Description

This command sets the positioning mode to incremental. In incremental mode all move distances specified with the Distance (D) command will be referenced to the current position. Mode Position Incremental (MPI) is most useful in applications that require repetitive movements, such as feed to length applications.

Example

<u>Command</u>	<u>Description</u>
> MN	Set positioning mode normal (preset)
> MPI	Set positioning mode incremental
> 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

The motor moves 10,000 steps CW after each G command

MPP Set-Up	Mode Position Profile			VALID Software Version A
SYNTAX <a>MPP	UNITS None	RANGE None	DEFAULT None	ATTRIBUTES Buffered Savable in Sequence
EXECUTION TIME		SEE ALSO DP, FP, NG		

Description

The **MPP** command sets an execution mode where after a **G** command is executed, subsequent commands are processed to provide complex motion profiles and input and output operations during motion. A **DP** command provides a means to correlate events with the motor position as the command causes command processing to pause until the distance matches the value set by the **DP** command. An **NG** command will mark the end of a position profile and turn off the **MPP** mode.

Example

<u>Command</u>	<u>Description</u>
> 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 move
> 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

MR Set-up	Configure Motor Resolution			VALID Software Version A
SYNTAX <a>MR<n>,i	UNITS n = pulse/rev i = option	RANGE n = 200 - 1024000 i = 0, 1 or 2	DEFAULT n = 25000 i = 0	ATTRIBUTES Buffered Savable in Sequence
EXECUTION TIME		SEE ALSO ER		
RESPONSE TO aMR IS *n				

Description

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.

Some Compumotor drives, such as the PK2, PK3, & PK130, require a minimum pulse width of 10 μ s. To set the pulse width to 10 μ s, add a ,1 to the end of the MR command setting. For instance, if you are using a PK3 and axis #1 and you need a resolution of 200 pulses/rev, issue the 1MR200,1 command.

Leaving off the ,n modifier or adding a ,0 to the end of the MR command sets the pulse width to 5 μ s (the default setting).

To interface to a Compumotor C Drive with a 200 step/rev resolution, use the 1MR200,2 command. The ,2 modifier sets the pulse width to 600ms for use with the C Drive.

The following table delineates some constraints that are dependent on the motor resolution setting.

Motor Resolution (Pulses/rev)	Velocity Constant		RM Constant	
	Default	10 μ s Pulse*	Default	10 μ s Pulse*
200 - 400	50	100	3.0518	1.52588
401 - 1999	24	100	6.3578	1.52588
2000 - 2999	17	100	8.9758	1.52588
3000 - 4999	15	100	10.1725	1.52588
4000 - 5999	14	100	10.8991	1.52588
6000 - 7999	11	100	13.8716	1.52588
8000 - 10999	10	100	15.2588	1.52588
11000 - 15999	9	100	16.9542	1.52588
16000 - 19999	8	100	19.0735	1.52588
20000 - 24000	7	100	21.7983	1.52588
>25000	5	100	30.5176	1.52588

* Use this column if a ,1 modifier is used with the MR command .

The Maximum velocity is determined by the following formula:

$$\text{Maximum velocity (rps)} = \frac{10000000}{[\text{motor resolution}] * [\text{velocity constant}] * [2]}$$

The RM constant is used for the RM command to set the velocity.

The pulse width (in seconds) is determined by the following equation:

$$\text{Pulse Width} = \frac{\text{Velocity Constant}}{1000000}$$

Example

<u>Command</u>	<u>Description</u>
> MR12800	Configure the motor resolution to 12800 so that the drive requires 12,800 pulses to produce one motor revolution

MV Motion	Maximum Correction Velocity			VALID Software Version A
SYNTAX <a>MV	UNITS n = rps	RANGE 0.001 to 99.999	DEFAULT 1	ATTRIBUTES Buffered Savable in Sequence
EXECUTION TIME		SEE ALSO FS, DW, ER		
RESPONSE TO aMV IS *MVn				

Description 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.

N Programming	End of Loop			VALID Software Version A
SYNTAX <a>N	UNITS None	RANGE None	DEFAULT None	ATTRIBUTES Buffered Savable in Sequence
EXECUTION TIME		SEE ALSO C, L, PS, Y		

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

Example	<u>Command</u>	<u>Description</u>
	> PS	Pauses the execution of buffered commands until the 500 receives a Continue (c) command
	> MN	Sets move to mode normal
	> 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 above series of commands five times
	> G	Executes the move (Go)
	> N	Ends the loop
	> C	Clears pause and executes all the buffered commands

The motor makes a 10,000-step move 5 times.

NG Motion	End Position Profile			VALID Software Version A
SYNTAX <a>NG	UNITS None	RANGE None	DEFAULT None	ATTRIBUTES Buffered Savable in Sequence
EXECUTION TIME		SEE ALSO MPP, DP		

Description This command marks the end of a **MPP** position profile and turns off **MPP** mode. When in Mode Position Profile, commands subsequent to a **G** command are processed until an **NG** command is encountered; command processing pauses until motion is complete, then command execution resumes.

Example

<u>Command</u>	<u>Description</u>
> 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 200000 steps
> MPP	Set position profile mode
> G	Execute mode
> 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 Programming	End of If			VALID Software Version A
SYNTAX <a>NIF	UNITS None	RANGE None	DEFAULT None	ATTRIBUTES Buffered Savable in Sequence
EXECUTION TIME		SEE ALSO ELSE, IF, <i>Evaluation Commands</i>		

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

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

Example

<u>Command</u>	<u>Description</u>
> IF(VAR5=7)	If variable #5 is equal to variable #7, execute the next command, ELSE , execute the command following the NIF command.
> GD1	Execute predefined move #1
> NIF	End of IF
> GD2	Execute predefined move #2

NWHILE Programming	End of While			VALID Software Version A
SYNTAX <a>NWHILE	UNITS None	RANGE None	DEFAULT None	ATTRIBUTES Buffered Savable in Sequence
EXECUTION TIME		SEE ALSO WHILE, REPEAT, UNTIL, <i>Evaluation Commands</i>		

Description

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

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

Example

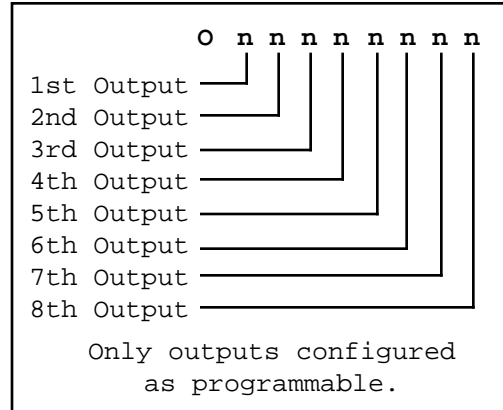
<u>Command</u>	<u>Description</u>
> WHILE (INXXX1)	While input #2 is active, continue to execute the commands between WHILE and NWHILE
> GD1	Execute predefined move #1
> T2.0	Time delay of two seconds
> NWHILE	End of while

O Programming	Output			VALID Software Version A
SYNTAX <a>O<n>	UNITS n = output status	RANGE 0 = low 1 = high X = don't care	DEFAULT None	ATTRIBUTES Buffered Savable in Sequence
EXECUTION TIME		SEE ALSO IO, OUT, OUTL		
RESPONSE TO aO IS *On				

Description

The Output (O) command turns the output bits configured as programmable outputs on and off. It does not affect the output state of the outputs not configured as programmable. This is used for signaling remote controllers, turning on LEDs, or sounding alarms. The output can indicate that 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 the **OUTL** command. The order of the output pattern is programmable outputs from output 1 to output 8 (as labeled on the unit) and then the relay.



Example

<u>Command</u>	<u>Description</u>
> OUT3A	Set output #3 as a programmable output
> OUT4A	Set output #4 as a programmable output
> 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 as Output 3) off and output 2 (labeled as Output 4) on
> G	Executes the move (Go)
> OX0	After the move ends, turn off programmable output 2 (labeled as Output 4)

OFF Programming	Off			VALID Software Version A
SYNTAX <a> OFF	UNITS None	RANGE None	DEFAULT None	ATTRIBUTES Immediate Never Saved
EXECUTION TIME		SEE ALSO ON, ST1, ST0		

Description

This command activates the shutdown output to the drive to command the drive to de-energize the motor. When you issue an **OFF** command, the fault LED light will be on to indicate that the command **OUT** output 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, except that this command is immediate and **ST1** is buffered.

Example

<u>Command</u>	<u>Description</u>
> OFF	Turns on the shutdown output to command the drive to de-energize the motor.

ON Programming	On			VALID Software Version
SYNTAX <a>ON	UNITS None	RANGE None	DEFAULT None	ATTRIBUTES Immediate Never Saved
EXECUTION TIME		SEE ALSO OFF, ST1, STØ		

Description This command turns off the shutdown output to command the drive to energize the motor. If you issue the **OFF** command to de-energize the motor, issuing an **ON** commands the drive to energize the motor.

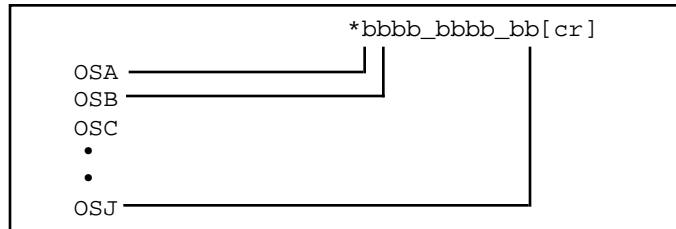
The **ON** command stops existing motion. This command is similar to the **STØ** command. The **ON** command, however, is immediate and **STØ** is buffered.

Example

<u>Command</u>	<u>Description</u>
> ON	Turns off shutdown output — commands drive to energize the motor

OS Status	Function Setup Report			VALID Software Version A
SYNTAX aOS	UNITS None	RANGE None	DEFAULT None	ATTRIBUTES Buffered Savable in Sequence
EXECUTION TIME		SEE ALSO OSA through OSJ		
RESPONSE TO aOS IS *OSb (b = 0 or 1)				

Description This command reports the status of OS command (OSA–OSJ) settings.



OSA Set-Up	Define Active State for End-of-Travel Limits			VALID Software Version D3
SYNTAX <a>OSAn	UNITS n = status	RANGE Ø (active low) or 1 (active high)	DEFAULT Ø	ATTRIBUTES Buffered Savable in Sequence
EXECUTION TIME		SEE ALSO IN, INL, OS, OSC		

Description This command sets the active state of the CW and CCW end-of-travel limit inputs. It enables you to use either a normally-closed or a normally-open switch for end-of-travel limits (default is **OSAØ**—normally-closed switches required). **NOTE:** The **INL** command no longer affects the active level of the end-of-travel limits.

Example

<u>Command</u>	<u>Description</u>
> 1OSA1	Configures CW and CCW end-of-travel limits for normally-open contacts
> 1INB	Check status of the CW end-of-travel limit

OSB Set-Up	Backup to Home Switch			VALID Software Version A
SYNTAX <a>OSBn	UNITS n = mode	RANGE Ø or 1	DEFAULT Ø	ATTRIBUTES Buffered Savable in Sequence
EXECUTION TIME		SEE ALSO GH, OSC, OSD, OSG, OSH, OS		
RESPONSE TO aOSB IS *OSBn (n = 0 or 1)				

Description

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

If you do not enable the function, the Model 500 will consider the motor at Home if the home input is active at the end of deceleration after encountering the active edge of the Home region. If you enable this function, the Model 500 will decelerate the motor to a stop after reaching the active edge of the Home region, and then move the motor in the opposite direction of the initial Go Home move with the **GHF** velocity until the active edge of the Home region is encountered. The Model 500 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 Set-Up	Define Active State of Home Switch			VALID Software Version A
SYNTAX <a>OScn	UNITS n = mode	RANGE Ø or 1	DEFAULT Ø	ATTRIBUTES Buffered Savable in Sequence
EXECUTION TIME		SEE ALSO GH, OS, OSB, OSF, OSG, OSH		

Description

OSCØ: Active state of home input is a low signal level
OSC1: Active state of home input is a high signal level

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. **OSCØ** requires a low-level signal to activate the home limit input. **OSC1** requires a high-level signal to activate the home limit input.

Example

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

OSD Set-Up	Enable Encoder Z Channel Input			VALID Software Version A
SYNTAX <a>OSDn	UNITS n = mode	RANGE Ø or 1	DEFAULT Ø	ATTRIBUTES Buffered Savable in Sequence
EXECUTION TIME		SEE ALSO GH, GHA, GHF, GHV, GHAD, OSB, OSC, OSE, OSF, OSJ		

Description

OSDØ: Disables Z channel function during home
OSD1: Recognizes Z channel after encountering the home limit switch

The 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.

NOTE: The OSD command is applicable only to the incremental encoder connected to the **ABS/INC ENCODER** interface. The **INC ENCODER** interface does not accept the Z channel input.

Example	<u>Command</u> > OSD0	<u>Description</u> Disables Z channel function
----------------	--------------------------	---

OSE Set-Up	Jog Enable			VALID Software Version A
SYNTAX <a>OSEn	UNITS n = mode	RANGE 0 or 1	DEFAULT 0	ATTRIBUTES Buffered Savable in Sequence
EXECUTION TIME		SEE ALSO IN, JVL, JA, JVH, JAD, INL		

Description

OSE0: Jog Disable
 OSE1: Jog Enable

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

OSF Set-Up	Acknowledge Programmable Inputs On Power Up			VALID Software Version D3
SYNTAX <a>OSFn	UNITS n = mode	RANGE 0 (ignore) or 1 (acknowledge)	DEFAULT 0	ATTRIBUTES Buffered Savable in Sequence
EXECUTION TIME		SEE ALSO IN, OS		

Description

The OSF command setting determines whether or not active programmable inputs are responded to when power is applied to the Model 500. After power up (during *normal operation*), programmable inputs affect the Model 500 only when they change state (off to on, on to off).

OSF0: On power up, the status of all programmable inputs (e.g., Stop, Kill, Reset, etc.) are ignored, unless specific commands like IF, TRIG, and other conditional commands are used in the power-up sequence (#100).

OSF1: On power up, the status of all programmable inputs are checked on power up and their functions (if active) are executed. **NOTE:** *If a programmable input is configured for Stop or Kill, no motion will occur if that input is active on power up. If an input is configured as a Reset input and it is active, the 500 will continually cycle in and out of the reset mode.*

Example	<u>Command</u> > 1OSF1	<u>Description</u> Respond to programmable inputs states when power is cycled
----------------	---------------------------	--

OSG Set-Up	Final Homing Direction			VALID Software Version A
SYNTAX <a>OSEn	UNITS n = mode	RANGE 0 or 1	DEFAULT 0	ATTRIBUTES Buffered Savable in Sequence
EXECUTION TIME		SEE ALSO OSB, OSH		

Description **OSG0:** Sets the final home approach direction to be CW
OSG1: Sets the final home approach direction to be CCW
This enables you to approach home from any direction yet stop at home repeatedly in the same edge of the switch

OSH Set-Up	Reference Edge of Home Switch			VALID Software Version A
SYNTAX <a>OSHn	UNITS n = mode	RANGE Ø or 1	DEFAULT Ø	ATTRIBUTES Buffered Savable in Sequence
EXECUTION TIME		SEE ALSO OSB, OSG		

Description **OSH0:** Selects the clockwise side of the home signal as the *edge* on which the final approach will stop
OSH1: Selects the counterclockwise side of the home signal as the *edge* on which the final approach will stop
The CW edge of the Home switch is defined as the first switch transition seen by the 500 when traveling off of 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 defined as the first switch transition seen by the 500 when traveling off of the CCW limit in the CW direction.

OSI Set-Up	Continue Sequence Scan after STOP			VALID Software Version A4
SYNTAX <a>OSIn	UNITS n = mode	RANGE 1 (enable) or Ø (disable)	DEFAULT Ø	ATTRIBUTES Buffered Savable in Sequence
EXECUTION TIME		SEE ALSO SSJ, SSH, STOP		

Description When in the sequence scan mode (entered via the **SSJ1** command), if a **STOP** command is used to stop motion within a sequence, the Model 500 will not exit the sequence scan mode if **OSI1** is issued.

If **OSI1** is used in conjunction with the **SSH1** command (Save command buffer on Stop), then the current sequence in which the **STOP** command is encountered (or stopped from the programmable inputs) will be completed and it will run the next valid sequence on the sequence select input lines.

Example

<u>Command</u>	<u>Description</u>
> SSJ1	Enables <i>sequence scan</i> mode
> SSH1	Enables <i>save buffer on stop</i> mode
> OSI1	Stays in sequence scan mode after a stop occurs on a move in a sequence

OSJ Set-Up	Select Z Channel Homing to Single or Indefinite Revs			VALID Software Version D3
SYNTAX <a>OSJn	UNITS n = mode	RANGE Ø (single rev) or 1 (indefinitely)	DEFAULT Ø	ATTRIBUTES Buffered Savable in Sequence
EXECUTION TIME		SEE ALSO OS, OSD		

Description **OSJØ:** During homing operation, the Z channel must be found within one revolution after activation of the Home limit input—if not, the motor will stop.

OSJ1: Seek the Z channel indefinitely.

Example

Command
> 1OSJ1

Description
Configure Z channel homing to seek the Z channel indefinitely

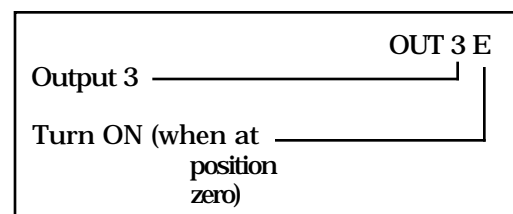
OUT Programming	Output Functions			VALID Software Version A
SYNTAX <a>OUT<n><x>	UNITS n = bits x = functions	RANGE n = 1 - 9 x = A - Z	DEFAULT None	ATTRIBUTES Buffered Savable in Sequence
EXECUTION TIME		SEE ALSO IN, OUTL, OUTP		
RESPONSE TO aOUT IS SEE EXAMPLE				

Description

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

<x>	Function
A	<i>Programmable Output</i> - Used with O and IO commands
B	<i>Moving/Not Moving</i> - On when motion is occurring
C	<i>Sequence in Progress</i> - On when a sequence is being executed.
D	<i>At Limits</i> - On when the motor has reached either a software or hardware limit
E	<i>At Position Zero</i> - On when the absolute position counter is zero
F	<i>Fault</i> - On when a fault has occurred. Error #16, 20, 30, 66 only
H	<i>Shutdown Command</i> - Indicates when you turn off the drive using OFF or ST1 command
I	<i>Reserved</i>
J	<i>Strobe</i> - Turns on to load bytes of parallel data. Indicates to the interface that the 500 is ready for data.
K	<i>Command Error</i> - On when RS-232C or BCD input has bad data
L	<i>Position Error Fault</i> - On when a stall condition is detected. This situation also creates an error 2Ø
M	<i>Reserved</i>
N	<i>CW Software Limit Reached</i> - CW Software limit set with SL command has activated
O	<i>Reserved</i>
P	<i>CCW Software limit reached</i> - CCW Software limit set with SL command has activated
Q	<i>Reserved</i>
R	<i>CW Hardware limit activated</i>
S	<i>CCW hardware limit activated</i>
T	<i>Output based on position</i> - This output goes on when a condition specified by OUTP command exists.
U	<i>Pulse Output</i> - A square wave is output based on the PU and PUL commands.
OTHERS	<i>No Function</i>

Outputs 1 through 8 and the relay can be configured as any one of the functions described above. For example, output #3 is set as an at position zero indicator:



Example 1 The **OUT** command also reports the status of each output bit.

```
aOUT1 *1_A_PROGRAMMABLE_OUTPUT_(STATUS_ON)
```

Example 2 By not specifying the output bit number, the Model 500 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_A_PROGRAMMABLE_OUTPUT_(STATUS_ON)
*6_A_PROGRAMMABLE_OUTPUT_(STATUS_ON)
*7_A_PROGRAMMABLE_OUTPUT_(STATUS_ON)
*8_A_PROGRAMMABLE_OUTPUT_(STATUS_ON)
*9_A_PROGRAMMABLE_OUTPUT_(STATUS_ON)
```

OUTL Set-Up	Set Active Output Level			VALID Software Version A
SYNTAX <a>OUTL<N>	UNITS n = active level	RANGE 0 = low 1 = high	DEFAULT 0	ATTRIBUTES Buffered Savable in Sequence
EXECUTION TIME		SEE ALSO INL, OUT		
RESPONSE TO aOUTL IS *Ø_OUTPUT_SIGNAL_LEVEL_ACTIVE_LOW *Ø_OUTPUT_SIGNAL_LEVEL_ACTIVE_HIGH				

Description This command configures the voltage level that the Model 500 considers to be an active output signal.

- OUTLØ : Sets a low level (0V) as an active signal
- OUTL1 : Sets a high level (5-30V) as an active signal

Outputs 1 through 8 are open collector outputs. Therefore, you must supply power (5-30VDC) to run the outputs. The output is able to sink up to 300 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.

Example	<table border="0"> <tr> <td><u>Command</u></td> <td><u>Description</u></td> </tr> <tr> <td>> OUTLØ</td> <td>Set a low level (0V) active signal</td> </tr> <tr> <td>> OUTL1A</td> <td>Set output 1 as a programmable output</td> </tr> <tr> <td>> 1OUTL</td> <td>*Ø_OUTPUT_SIGNAL_LEVEL_ACTIVE_LOW</td> </tr> </table>	<u>Command</u>	<u>Description</u>	> OUTLØ	Set a low level (0V) active signal	> OUTL1A	Set output 1 as a programmable output	> 1OUTL	*Ø_OUTPUT_SIGNAL_LEVEL_ACTIVE_LOW
<u>Command</u>	<u>Description</u>								
> OUTLØ	Set a low level (0V) active signal								
> OUTL1A	Set output 1 as a programmable output								
> 1OUTL	*Ø_OUTPUT_SIGNAL_LEVEL_ACTIVE_LOW								

OUTP Set-Up	Output on Position			VALID Software Version A
SYNTAX <a>OUTP<x><n><t>	UNITS x = mode n = steps t = ms	RANGE x = A or I n = ±2,147,483,647 t = 0 - 500	DEFAULT A, 0, 5	ATTRIBUTES Buffered Savable in Sequence
EXECUTION TIME			SEE ALSO OUT	
RESPONSE TO aOUTP IS *OUT_ON_X_POSITION_N *POSITION_OUTPUT_ON_TIME_T_MILLISECONDS				

Description

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

The command **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. The time is ignored in this mode.

In incremental mode, the output turns on each time the distance is traversed and stays activate for the entered time. The sign for the distance will convert to a plus sign as the distance is an absolute value.

Example 1

<u>Command</u>	<u>Description</u>
> OUT2T	Set Output #2 to activate based on position
> OUTPA1000	When the absolute position is > +1000, the output turns on

Example 2

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

PF Status	Follower Position Report			VALID Software Version A
SYNTAX aPF	UNITS n = encoder steps	RANGE ±2,147,483,647	DEFAULT None	ATTRIBUTES Buffered Savable in Sequence
EXECUTION TIME			SEE ALSO PFZ, PR	
RESPONSE TO aPF IS *n				

Description

This command 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-Up	Sets Follower Counter to Zero			VALID Software Version A
SYNTAX <a>PFZ	UNITS None	RANGE None	DEFAULT None	ATTRIBUTES Buffered Savable in Sequence
EXECUTION TIME		SEE ALSO PF, PZ		

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

Example

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

PR Status	Absolute Position Report			VALID Software Version A
SYNTAX aPR	UNITS None	RANGE None	DEFAULT None	ATTRIBUTES Buffered Savable in Sequence
EXECUTION TIME		SEE ALSO D, MN, MPI, MPA, PZ, SP		
RESPONSE TO aPR IS *n (n = ±2,147,483,647 steps)				

Description Absolute position request; response is *snnnnnnnnn[cr]. This command reports the motor's position with respect to the power-up position or the last time an SP or PZ command was issued. The absolute position counter can track up to +/- 2³¹ - 1, or 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.

Example

<u>Command</u>	<u>Description</u>
> 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. The response should be <i>*+0000005000</i>

PS Motion	Pause			VALID Software Version A
SYNTAX <a>PS	UNITS None	RANGE None	DEFAULT None	ATTRIBUTES Buffered Savable in Sequence
EXECUTION TIME		SEE ALSO C		

Description This command pauses execution of a command string or sequence following the Pause (PS) command until the 500 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.

This command is useful for interactive tests and in synchronizing multiple indexes that have long command strings.

Example

<u>Command</u>	<u>Description</u>
> PS	Pauses execution of following commands until the 500 receives the Continue (C) command
> 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 sec
> G	Executes the move (Go)
> C	Continues Execution

When the Model 500 receives the C command, the motor moves 25,000 steps twice with a 2 second delay.

PU Programming	Configure Square Wave Output			VALID Software Version A
SYNTAX <a>PU<n><m>	UNITS n = pulses m = ms	RANGE n = 0 to 2147483647 m = 0 to 5000	DEFAULT n = 0 m = 0	ATTRIBUTES Buffered Savable in Sequence
EXECUTION TIME		SEE ALSO PUL, OUT		
RESPONSE TO aPU IS * OUTPUT_PULSE_NUMBER_n[cr]*OUTPUT_PULSE_RATE__m_MILLISECONDS				

Description An output can be configured with the OUT command to produce a square wave output. The PU command sets the square wave characteristics with the first field determining the number of pulses and the second field determines the time in milliseconds the pulse is active. A PU5,200 command configures a square wave of 5 pulses which is on for 200 milliseconds and off for 200 milliseconds. The PUL command initiates the square wave.

Example

<u>Command</u>	<u>Description</u>
> OUT2U	Set output 2 as the square wave output
> PU1000,2	Set a square wave of 1000 pulses with a 2 ms on and off time
> PUL	Initiate square wave on output 2

PUL Programming	Activate Square Wave Output			VALID Software Version A
SYNTAX <a>PUL	UNITS None	RANGE None	DEFAULT None	ATTRIBUTES Buffered Savable in Sequence
EXECUTION TIME		SEE ALSO OUT, PU		

Description 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.

Example

<u>Command</u>	<u>Description</u>
> OUT5U	Set output 5 as the square wave output
> PU20,5	Set a square wave of 20 pulses with a 5 millisecond on and off time
> PUL	Commence square wave on output 5

PX Status	Report Encoder Position			VALID Software Version A
SYNTAX aPX	UNITS None	RANGE None	DEFAULT None	ATTRIBUTES Buffered Savable in Sequence
EXECUTION TIME		SEE ALSO PR		
RESPONSE TO aPX IS *n (n = ±2147483647 encoder steps)				

Description 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-Up	Set Absolute Counter to Zero			VALID Software Version A
SYNTAX <a>PZ	UNITS None	RANGE None	DEFAULT None	ATTRIBUTES Buffered Savable in Sequence
EXECUTION TIME		SEE ALSO D, MN, MPI, MPA, PR		

Description This command sets the absolute position counter to zero.

Example

<u>Command</u>	<u>Description</u>
> MPA	Make all preset moves with respect to absolute zero position
> 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)

" Programming	Quote			VALID Software Version A
SYNTAX a"	UNITS x = ASCII or number	RANGE 33-126 decimal any character	DEFAULT None	ATTRIBUTES Buffered Savable in Sequence
EXECUTION TIME		SEE ALSO CR, LF		
RESPONSE TO a" IS * "n				

Description 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 Compumotor devices to move. The ASCII range of characters accepted by the command is 33 - 126 (decimal).

Example 1

<u>Command</u>	<u>Description</u>
> MN	Set to mode normal (Preset Moves)
> A1Ø	Set acceleration to 10 rps ²
> AD1Ø	Set deceleration to 10 rps ²
> V5	Set velocity to 5 rps
> D125ØØ	Set distance to 12,500 steps
> G	Executes the move (Go)
> 1"MOVE_DONE	After motor finished the move, the Compumotor Model 500 will send the message MOVE_DONE out from the RS-232C port.

Example 2

<u>Command</u>	<u>Description</u>
> MN	Set to mode normal (Preset Moves)
> A1Ø	Set acceleration to 10 rps ²
> AD1Ø	Set deceleration to 10 rps ²
> V5	Set velocity to 5 rps
> D125ØØ	Set distance to 12,500 steps
> G	Executes the move (Go)
> 1"2XR1	Once the move is done, Run Sequence 1 is commanded on a unit with device address 2.

QØ Set-Up	Exit Velocity Profiling Mode			VALID Software Version A
SYNTAX <a>QØ	UNITS None	RANGE None	DEFAULT None	ATTRIBUTES Buffered Savable in Sequence
EXECUTION TIME		SEE ALSO Q1, RM		

Description The QØ command exits the velocity profiling mode. Motion will stop when QØ is issued.

Example See Q1 command below

Q1 Set-Up	Enter Velocity Profiling Mode			VALID Software Version A
SYNTAX <a>Q1	UNITS None	RANGE None	DEFAULT None	ATTRIBUTES Buffered Savable in Sequence
EXECUTION TIME		SEE ALSO QØ, RM		

Description The Q1 command enters the 500 in velocity profiling mode. Subsequent RM commands will cause an immediate change in motion velocity. Use QØ to exit this mode. The velocity in steps per second is determined by the following formula:

$$\text{steps/sec} = (\text{decimal equivalent of RM value}) \cdot 30.3184$$

Example

<u>Command</u>	<u>Description</u>
> Q1	Enter Velocity Profiling mode
> RMØ1ØØ	Go to RM velocity of 7,812 steps per second
> RMØ5ØØ	Go to RM velocity of 39,063 steps per second
> RM1ØØØ	Go to RM velocity of 125,003 steps per second
> RMØ5ØØ	Go to RM velocity of 39,063 steps per second
> RMØ1ØØ	Go to RM velocity of 7,812 steps per second
> QØ	Exit velocity profiling mode

Motion will stop when QØ command is entered.

R Status	Report 500 Status			VALID Software Version A
SYNTAX aR	UNITS None	RANGE None	DEFAULT None	ATTRIBUTES Immediate Never Saved
EXECUTION TIME		SEE ALSO RA, RB		
RESPONSE TO aR IS *x (x = R, S, B or C)				

Description The Request 500 Status (R) command can be used to indicate the general status of the Model 500. Possible responses are:

<u>Response Character</u>	<u>Definition</u>
*R	Ready
*S	Ready, Attention Needed
*B	Busy
*C	Busy, Attention Needed

The following conditions cause a response indicating that the Model 500 is busy:

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

The following conditions will cause an error response.

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

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

Example

<u>Command</u>	<u>Response</u>
> 1R	*R (500 ready to accept a command—no error conditions require attention.)

RA Status	Limit Switch Status Report			VALID Software Version A
SYNTAX aRA	UNITS None	RANGE None	DEFAULT None	ATTRIBUTES Immediate Never Saved
EXECUTION TIME		SEE ALSO		
RESPONSE TO aRA IS *x (x = characters @, A-0)				

Description

The Limit Switch Status Report (**RA**) command 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 CW Limit	Ended By CCW Limit	Limit Switch CW Limit	Active 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

The **RA** command 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 Status	Loop, Pause Shutdown, Trigger Status Report			VALID Software Version A
SYNTAX aRB	UNITS None	RANGE None	DEFAULT None	ATTRIBUTES Immediate Never Saved
EXECUTION TIME		SEE ALSO L, PS, R, RA, ST, TR		
RESPONSE TO aRB IS *x (x = @, A-O)				

Description

This command reports the status of four functions within the 500; Loop in Progress, Pause in Progress, Trigger Wait Active and Shutdown Active. The response is in the form *x[cr], where x = @ - O.

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

Example

<u>Command</u>	<u>Response</u>
> 1RB	*A (the 500 is currently executing a loop)

REG Motion	Configure Registration Moves			VALID Software Version A
SYNTAX <a>REGn, see below	UNITS n = registration number	RANGE 1 - 4	DEFAULT None	ATTRIBUTES Buffered Savable in Sequence
EXECUTION TIME		SEE ALSO IN		
RESPONSE TO aREG IS *REGn, Ax, ADx, Cx (or FOLx), Dx				

Description

Syntax: <a>REGn, Ax, ADx, Cx (or FOLx), Dx

The REGn command defines registration move n with parameters A (acceleration), AD (deceleration), v (velocity) or FOL (following) and D (distance). Up to 4 registration moves may be entered (the values are stored in nonvolatile memory). The move is calculated for immediate execution upon receipt of an active registration input. Inputs can be configured to produce registration moves by the IN command.

Example

<u>Command</u>	<u>Description</u>
> REG1, A5, AD10, V5, D200000	Define registration move #1
> REG4, A20, AD25, V10, D100000	Define registration move #4
> REG2, A10, AD5, FOL100, D250000	Define registration move #2

REPEAT Programming	Repeat			VALID Software Version A
SYNTAX <a>REPEAT	UNITS None	RANGE None	DEFAULT None	ATTRIBUTES Buffered Savable in Sequence
EXECUTION TIME		SEE ALSO UNTIL, WHILE, NWHILE, <i>Evaluation Commands</i>		

Description

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 true, the program flow is redirected to the **REPEAT** command; otherwise, a false 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)

Example

Command	Description
> REPEAT	Repeat Command
> GD1	Execute predefined move #1
> T2	Delay 2 seconds
> UNTIL (INXXX1)	If input #1 is active, do next command, else execute from command following REPEAT

NOTE: The input condition (INXXX1) will not be evaluated until the **UNTIL** command is evaluated. The current move will not be interrupted. Evaluation of the input condition is done at the end of the move in progress.

RG Status	Go Home Status Report			VALID Software Version A
SYNTAX aRG	UNITS None	RANGE None	DEFAULT None	ATTRIBUTES Immediate Never Saved
EXECUTION TIME		SEE ALSO GH, R, RA, RB		
RESPONSE TO aRG IS *x (x = @ or A)				

Description

This command reports back the status of the last Go Home attempt. The response is either *@ or *A, indicating failure or success.

Response	Go Home Successful
*@	NO
*A	YES

RIFS Set-Up	Return to Factory Settings			VALID Software Version A
SYNTAX <a>RIFS	UNITS None	RANGE None	DEFAULT None	ATTRIBUTES Immediate Never Saved
EXECUTION TIME		SEE ALSO None		

Description

Execution of this command will cause the following 500 parameters to return to factory default settings.

- Software Limits (SL) set to 0, 0
- Distance set to 25,000
- Jog Velocity High (JVH) set to 10 rps
- Jog Velocity Low (JVL) set to 1 rps
- Jog Acceleration (JA) set to 10 rps²
- Jog Deceleration (JAD) set to 10 rps²
- Limit Deceleration (LAD) set to 900 rps²
- Go Home Velocity (GHV) set to 1 rps
- Go Home Final (GHF) set to 0.1 rps
- Go Home Acceleration (GHA) set to 10 rps²
- Go Home Deceleration (GHAD) set to 10 rps²
- Velocity (V) set to 1 rps
- Acceleration (A) set to 10 rps²
- Deceleration (AD) set to 10 rps²
- Proportional Gain (PG) set to 10
- Proportional Maximum (PM) set to 500
- Position Error (CPE) set to 4000
- Max Correction Velocity (MV) set to 1 rps
- FS flags set to 0
- SS flags set to 0
- OS flags set to 0
- Encoder resolution (ER) set to 4000
- Motor resolution (MR) set to 25,000
- Configurable Inputs (IN) set to trigger inputs (A)
- Configurable Outputs (OUT) set to programmable outputs (A)
- All Sequences (1 - 100) are erased.

NOTE

The position counters are reset to zero when you issue the RIFS command.

RM Motion	Rate Multiplier in Velocity Streaming Mode			VALID Software Version A
SYNTAX <a>RMh	UNITS h = hex	RANGE 00000 to FFFFF	DEFAULT None	ATTRIBUTES Buffered Savable in Sequence
EXECUTION TIME		SEE ALSO Q0, Q1		
RESPONSE TO aRM IS *RMn				

Description

The **RM** command followed by 5 hexadecimal digits represent a velocity. The velocity change is essentially instantaneous; there is no acceleration/deceleration ramp between velocities. A limit switch closure will stop movement while in velocity profiling mode, but does not cause the 500 to exit velocity streaming mode. **RM** (profiling) mode is unidirectional. The direction will be the last activated direction either from an actual move or from a **D** or **H** command. Bi-directional moves using this mode can be made by returning to velocity zero, switching off **RM** mode, changing the direction, and re-enabling **RM** mode. This extra overhead should be acceptable given the need to attend to going to velocity zero when changing directions in real situations. The velocity value is determined with the following formula:

$$\text{steps/sec} = \text{decimal equivalent of RM value} \cdot 30.5184$$

Example

<u>Command</u>	<u>Description</u>
> Q1	Enter Velocity streaming mode
> RM00100	Go to velocity of 7,812 steps/sec
> RM00500	Go to velocity of 39,063 steps/sec
> T1	Delay for one second
> RM00100	Go to velocity of 7,812 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 the velocity profiling mode, and Q0 is used to exit. While in this mode the **RM** command is used to generate velocity values that are immediately implemented while the motor is moving. This means that the **RM** command must be sent to the Model 500 at the time the change in velocity is required. This creates a stair-step effect in velocity change. By implementing a large number of very small instantaneous velocity changes, a smooth, non-linear acceleration ramp can be achieved.

RS Status	Sequence Execution Status Report			VALID Software Version A
SYNTAX aRS	UNITS None	RANGE None	DEFAULT None	ATTRIBUTES Immediate Never Saved
EXECUTION TIME		SEE ALSO XR, XRP		
RESPONSE TO aRS IS *x (x = characters @, A, B, C or D)				

Description

Responds in the form *x[cr] where x = @ - D.
 Bit 0: Sequence started; 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 End-of-Travel limit is encountered.

Sequence ended is false when: A sequence is successfully started.

RSE Status	Report Errors			VALID Software Version A
SYNTAX aRSE	UNITS None	RANGE None	DEFAULT None	ATTRIBUTES Buffered Savable in Sequence
EXECUTION TIME		SEE ALSO None		
RESPONSE TO aRSE IS See Below				

Description

You can find out what error condition exists in the 500 using this command. If you see the two digit LED on the front panel flashing a code, you should start troubleshooting the unit using the **RSE** command. The error number and its description are reported. The possible error messages are:

Error #	Description
* 16	Shutdown active
* 20	Stall detected
* 30	CRC error in Battery Backed RAM
41	CW Hardware Limit
42	CCW Hardware Limit
43	CW Software Limit
44	CCW Software Limit
60	Shutdown commanded
* 66	User Fault
71	Absolute Encoder not connected
72	Bad Absolute Encoder reads

* These are errors that will activate an output defined as a fault output using the **OUTnF** command (see **OUT** command description).

Possible**Responses to RSE**

*NO_ERRORS[cr]
 *COMMAND_OUT_DISABLED_BY_x x = 16, 20, 23, 30, 60, 66
 *500_DISABLED_BY_x[cr] x = 41, 42, 43, 44

RSIN Programming	Set Variable Interactively			VALID Software Version A
SYNTAX <<a>VARn=RSIN	UNITS n = variable number	RANGE 1 - 150	DEFAULT None	ATTRIBUTES Buffered Savable in Sequence
EXECUTION TIME		SEE ALSO		

Description

The **RSIN** command is used in conjunction with the variables to allow the variables to be loaded with data during sequence execution. When the command is executed, the data is then transmitted to the Model 500 with **!nnnnnnnnnn.nnnnn**. Command processing pauses with the **RSIN** command and resumes when the data is transmitted to the Model 500.

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

Example

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

RV Status	Revision Level Report			VALID Software Version D3
SYNTAX aRV	UNITS None	RANGE None	DEFAULT None	ATTRIBUTES Immediate Not Saved
EXECUTION TIME		SEE ALSO None		
RESPONSE TO aRV IS See Below				

Description The Revision (RV) command responds with the software part number and its revision level. The response is in the form shown below:

*92-nnnnnn-nn<xn>[cr]

The part number identifies which product the software is written for, as well as any special features that the software may include. The revision level identifies when the software was written. You may want to record this information in your own records for future use. This type of information is useful when you consult Parker Compumotor's Applications Department.

Example

<u>Command</u>	<u>Response</u>
1RV	*92-011512-01D3 (The product is identified by 92-011512. The revision level is identified by 01D3.)

S Motion	Stop			VALID Software Version A
SYNTAX <a>S	UNITS None	RANGE None	DEFAULT None	ATTRIBUTES Immediate Not Saved
EXECUTION TIME		SEE ALSO AD, C, K, Q0, SSH, SSL, STOP		

Description 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 Model 500 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.

This command is similar to the STOP command, except that the STOP command is buffered and can be saved in a sequence.

Example

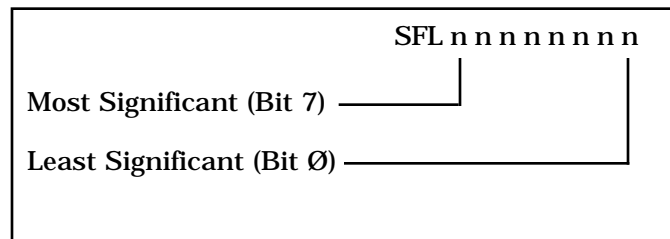
<u>Command</u>	<u>Description</u>
> MC	Sets move in continuous mode
> AD5	Sets deceleration to 5 rps ²
> A1	Sets acceleration to 1 rps ²
> V10	Sets velocity to 10 rps
> G	Executes the move (Go)
> S	Stops (motor decelerates) to 0 rps at 5 rps ²

The S command is not buffered since it is an immediate command. As soon as the Model 500 receives the S command, it stops motion.

SFL Programming	Set User Flag			VALID Software Version A
SYNTAX <a>SFL<n>	UNITS n = bits	RANGE 0, 1, or x	DEFAULT 0	ATTRIBUTES Buffered Savable in Sequence
EXECUTION TIME		SEE ALSO IF, NIF, UNTIL, WHILE		
RESPONSE TO aSFL IS *SFLn				

Description

This command sets a condition of 8 different bits. You can define the state of bits 0 through 7 using this command to be either a \emptyset , 1, or x modifier. A \emptyset 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 during an SFL command; SFL11 sets bits 6 and 7 while leaving the remaining bits unaltered.



Once you set bits 0 through 7, you can use the IF command to do an IF_THEN_ELSE operation or the REPEAT_UNTIL and WHILE_NWHILE commands to do conditional looping.

Example

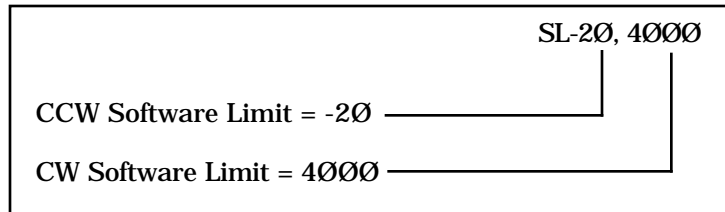
<u>Command</u>	<u>Description</u>
> SFL1010	Set bits 5 and 7, and clear bits 6 and 4
> IF (FL1010)	If user flag bits 5 and 7 are set, then issue the following commands
> A10	Set acceleration to 10 rps ²
> AD15	Set deceleration to 15 rps ²
> V5	Set velocity to 5 rps
> D25000	Set distance to 25,000 steps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

If the IF pattern matches the SFL pattern, the motor will move 25,000 steps.

SL Set-Up	Software End-of-Travel Limits			VALID Software Version A
SYNTAX <a>SL<n>, <n>	UNITS n = steps	RANGE ± 2,147,483,647	DEFAULT 0	ATTRIBUTES Buffered Savable in Sequence
EXECUTION TIME		SEE ALSO LAD, OUT, SLD		
RESPONSE TO aSL IS *SLn, n				

Description

This command defines the CCW and CW software end of travel limits respectively. 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.



Example

<u>Command</u>	<u>Description</u>
> SL-10000, 26944	Set CCW software limit to -10,000 steps. Set CW software limit to 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)

Motor will start moving and starts decelerating when it reaches 26,944 steps.

SLD Set-Up	Software Limit Disable			VALID Software Version A
SYNTAX <a>SLD<n>	UNITS None	RANGE 0 - 3	DEFAULT 3	ATTRIBUTES Buffered Savable in Sequence
EXECUTION TIME		SEE ALSO OUT, SL		
RESPONSE TO aSLD IS See Below				

Description

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 be allowed to go past the software 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 both CW and CCW software limits.

This command is very 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 the 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

```

Example

<u>Command</u>	<u>Description</u>
> SL0,0	Set CCW software limit to 0 and CW software limit to 0
> 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
> SLD3	Disable both CW and CCW software limits
> G	Executes the move (Go)

If SLD0 was entered instead of SLD3, motion would not have occurred, because the motor is sitting on the software limit.

SN Set-Up	Scan Time Delay			VALID Software Version A
SYNTAX <a>SN<n>	UNITS n = milliseconds	RANGE 1 to 1,000	DEFAULT 50	ATTRIBUTES Buffered Savable in Sequence
EXECUTION TIME		SEE ALSO None		
RESPONSE TO aSN IS *SCAN_TIME=n_MILLISECONDS				

Description 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 for the PLC to assure that the correct data is present when the data is read.

Example

<u>Command</u>	<u>Description</u>
> SN150	Sets scan time of sequence select inputs to 150 milliseconds.

SP Set-Up	Set Position Absolute			VALID Software Version A
SYNTAX <a>SPn	UNITS n = steps	RANGE ±2,147,483,647	DEFAULT 0	ATTRIBUTES Buffered Savable in Sequence
EXECUTION TIME		SEE ALSO D, MPA, MPI, PR		

Description 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.

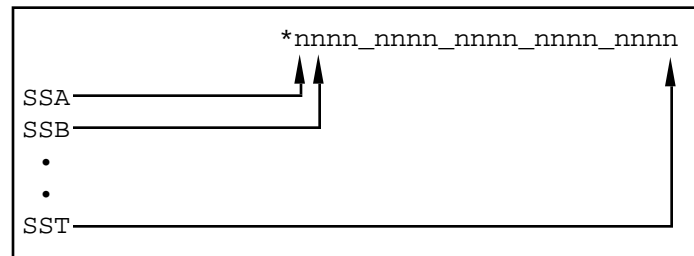
Example

<u>Command</u>	<u>Description/Response</u>
> MN	Sets mode to normal
> 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 (absolute position)
> SP5000	Sets absolute counter to 5,000
> 1PR	*+0000005000 (absolute position)

SS Status	Function Set-Up Report			VALID Software Version A
SYNTAX aSS	UNITS None	RANGE 0, 1	DEFAULT None	ATTRIBUTES Buffered Savable in Sequence
EXECUTION TIME		SEE ALSO SSA, SSD, SSG, SSH, SSI, SSJ, SSL, SSN, SSO, SSP, SSJ,		
RESPONSE TO aSS IS *nnnn_nnnn_nnnn_nnnn				

Description

Reports the status of the **SS** commands (**SSA** - **SST**) settings.



SSA Set-Up	RS-232C Echo Control			VALID Software Version A
SYNTAX <a>SSAn	UNITS n = mode	RANGE 0 or 1	DEFAULT None	ATTRIBUTES Buffered Savable in Sequence
EXECUTION TIME		SEE ALSO SS		

Description

This command turns the RS-232C echo (the re-transmission to the host device of characters received from the remote device by the Model 500) on and off.

SSA0: Echo on
SSA1: Echo off

In the Echo On (**SSA0**) mode, characters that are received by the 500 are echoed automatically. In the Echo Off (**SSA1**) mode, characters are not echoed from the 500. 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 500's further down the chain to receive commands.

Example

Command
 > SSA1

Description
 Turns echo off (Characters sent to the Model 500 are not echoed back to the host.)

SSD Set-Up	Alternate Mode Stop			VALID Software Version A
SYNTAX <a>SSDn	UNITS n = mode	RANGE 0 or 1	DEFAULT 0	ATTRIBUTES Buffered Savable in Sequence
EXECUTION TIME		SEE ALSO SS		

Description

This command sets the function of the Stop (s) command when issued during the Alternate Mode (MA). If SSD is set to 1, motion stops immediately. An SSD setting of 0 causes the stop command to stop motion at the end of the current cycle.

SSD1 = stop motion immediately
 SSD0 = stop motion at end of current cycle

Example

<u>Command</u>	<u>Description</u>
> SSD1	Set alternate mode stop to 1
> MA	Set Alternate mode
> G	Executes the move
> s	Stops motion immediately

SSG Set-Up	Clear/Save the Command Buffer on Limit			VALID Software Version A
SYNTAX <a>SSGn	UNITS n = mode	RANGE 0 or 1	DEFAULT 0	ATTRIBUTES Buffered Savable in Sequence
EXECUTION TIME		SEE ALSO LAD, LD, SS		

Description

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.

Example

<u>Command</u>	<u>Description</u>
> 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)
> 011	Activate programmable outputs #1 and #2

If a limit switch is encountered while executing the move, outputs #1 and #2 will still go on.

SSH Set-Up	Clear/Save the Command Buffer on Stop			VALID Software Version A
SYNTAX <a>SSHn	UNITS n = mode	RANGE 0, 1	DEFAULT 0	ATTRIBUTES Buffered Savable in Sequence
EXECUTION TIME		SEE ALSO SS		

Description **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.

Example

<u>Command</u>	<u>Description</u>
> 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 msec
> N	Ends Loop
> s	Stops execution

When you issue the **s** command, the Model 500 will clear the buffer and stop the move.

SSI Set-Up	Enable/Disable Interactive Mode			VALID Software Version A
SYNTAX <a>SSI n	UNITS n = mode	RANGE 0 or 1	DEFAULT 0	ATTRIBUTES Buffered Savable in Sequence
EXECUTION TIME		SEE ALSO SS, SSA, SSN		

Description When the interactive mode is enabled (**SSI0**), only the 500 at device address 1 responds with a prompt (>) when it understands a command and a question mark (?) when it does not. Both responses are preceded with a line feed, carriage return sequence. If in Interactive Mode, the 500 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 receive a (>) until you finish defining the loop.

The **SSI1** command disables the interactive mode. You will not receive (>) or (?) from the Model 500.

If you have multiple 500s on a daisy-chain, only the 500 with address #1 will respond with prompt, question marks, etc. Having only 500 #1 respond prevents confused information when all daisy-chained 500s respond at the same time. *Typically, you should disable the interactive mode when using a host computer with the 500.*

SSJ Set-Up	Enable/Disable Continuous Scan Mode			VALID Software Version A
SYNTAX <a>SSJn	UNITS n = mode	RANGE 0 or 1	DEFAULT None	ATTRIBUTES Buffered Savable in Sequence
EXECUTION TIME		SEE ALSO XR, XD, XT, IN, INL, XQ, SN, SS		

Description

SSJ1 enabled: The Model 500 continuously scans the inputs designated as sequence select inputs by the **IN** command and executes the sequence represented by the BCD number presented on the inputs. If Interrupted Run Mode (**XQ**) is active, then all the sequence input lines must go inactive prior to scanning the next sequence. A stop command discontinues continuous sequence scanning.

SSJ0 disabled: Does not scan the BCD numbers for sequence execution. In this mode, you could execute sequences using the RS-232C interface.

SSL Set-Up	Resume Execution Enable			VALID Software Version A
SYNTAX <a>SSLn	UNITS n = mode	RANGE 0 or 1	DEFAULT None	ATTRIBUTES Buffered Savable in Sequence
EXECUTION TIME		SEE ALSO S, C		

Description

This command will enable the controller to stop execution of commands, then resume execution using the Continue (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 the Stop (S) or a Remote Stop input. If **SSL1** is enabled, the move will resume as soon as you enter the Continue (C) command.

Example

<u>Command</u>	<u>Description</u>
> 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

While the motor is moving, enter a Stop (S) command. The motor will come to a stop. Now enter Continue (C) command, the motor would pick up where it stopped and complete the moves.

SSN Set-Up	Set Message Mode			VALID Software Version A
SYNTAX <a>SSNn	UNITS n = mode	RANGE 0,1	DEFAULT 0	ATTRIBUTES Buffered Savable in Sequence
EXECUTION TIME		SEE ALSO SS, SSI		

Description When the interactive mode is enabled with the `SSI0` command and the message mode is activated, an error message accompanies the prompt (?) to identify the error.

`SSN0`: Disable message mode

`SSN1`: Enable message mode

Example

<u>Command</u>	<u>Description</u>
> <code>SSI0</code>	Set Interactive Mode Active
> <code>SSN1</code>	Set Message Mode Active
> <code>PR</code>	The command is incorrect and the response is: <code>DEVICE_ADDRESS_REQUIRED</code>

SSO Set-Up	Set Sequence Select			VALID Software Version A
SYNTAX <a>SSOn	UNITS n = mode	RANGE 0 or 1	DEFAULT 0	ATTRIBUTES Buffered Savable in Sequence
EXECUTION TIME		SEE ALSO SS, CPB, SSP		

Description When the front panel pushbuttons are enabled (`CPB1`), the function of the data entry can be set to select and execute sequences if `SSO` is enabled. Activating `SSO` turns off `SSP` as they are mutually exclusive.

`SSO0` = sequence entry disabled

`SSO1` = enter and execute sequences from front panel pushbuttons

Example

<u>Command</u>	<u>Description</u>
> <code>CPB1</code>	Enable front panel pushbuttons
> <code>SSO1</code>	Enable sequence select and execution from the front panel pushbuttons

SSP Set-Up	Set Ratio Select			VALID Software Version A
SYNTAX <a>SSPn	UNITS n = mode	RANGE 0 or 1	DEFAULT 0	ATTRIBUTES Buffered Savable in Sequence
EXECUTION TIME		SEE ALSO SS, CPB, SSO		

Description When the front panel pushbuttons are enabled (`CPB1`), the function of the data entry can be set to select following values if `SSP` is enabled. Activating `SSP` turns off `SSO` as they are mutually exclusive.

`SSP0` = following value entry disabled

`SSP1` = enter following values from front panel pushbuttons

Example

<u>Command</u>	<u>Description</u>
> <code>CPB1</code>	Enable front panel pushbuttons
> <code>SSP1</code>	Enable following value entry from the front panel pushbuttons

SSQ Set-Up	Set Drive Fault Polarity			VALID Software Version A
SYNTAX <a>SSQn	UNITS n = mode	RANGE 0 or 1	DEFAULT 0	ATTRIBUTES Buffered Savable in Sequence
EXECUTION TIME		SEE ALSO SS		

Description This command sets the polarity (active high or active low) for the Drive Fault (15-pin D connector—pin #5) to become active. A Compumotor stepping drive will usually need **SSQ** set to 0, while a Compumotor servo drive requires **SSQ** to be set to 1.

SSQ0 = Drive fault active low
SSQ1 = Drive fault active high

Example

<u>Command</u>	<u>Description</u>
> SSQ1	Set drive fault active high

ST Programming	Shutdown			VALID Software Version A
SYNTAX <a>STn	UNITS n = mode	RANGE 0 or 1	DEFAULT 0	ATTRIBUTES Buffered Savable in Sequence
EXECUTION TIME		SEE ALSO OFF, ON		

Description 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. The absolute position counter is set to zero when you enter an **ST0** command.

Example

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

STOP Programming	Stop			VALID Software Version A
SYNTAX <a>STOP	UNITS None	RANGE None	DEFAULT None	ATTRIBUTES Buffered Savable in Sequence
EXECUTION TIME		SEE ALSO S, SSH, SSL		

Description

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 **SSH** (Clear/Save the Command Buffer on Stop) is set to 1, then only the current command is interrupted from completion and execution proceeds.

If the **SSL** (Resume execution) command is set to 1, a **C** (continue) command may be issued to complete the interrupted move and continue sequence or buffer execution.

This command is similar to the **s** (Stop) command, except that the **s** command is immediate and cannot be saved in a sequence.

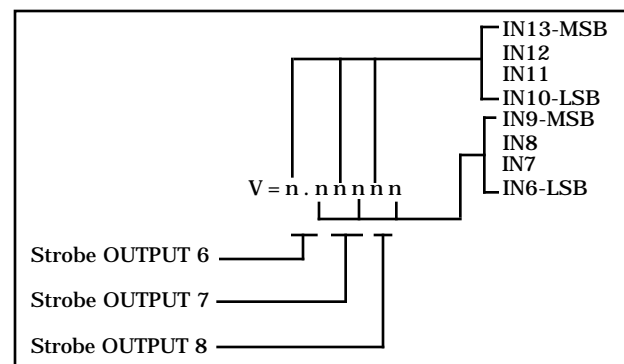
STR Set-Up	Set Strobe Output Delay Time			VALID Software Version A
SYNTAX <a>STRn	UNITS n = ms	RANGE 1 - 5,000	DEFAULT 1000	ATTRIBUTES Buffered Savable in Sequence
EXECUTION TIME		SEE ALSO VRD, LRD, DRD, XRD, IN, OUT, TRD, FRD, VARD		
RESPONSE TO aSTR IS *STROBE_TIME_n_MILLISECONDS				

Description

This command defines the amount of time each strobe output (defined by **OUT**) is active. This 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**), and Sequence Numbers (**XRD**), Time (**TRD**), Following (**FRD**) and Variables (**VARDn**).

The **STR** command would typically be greater than a PLC scan time (to ensure that the data is present during a read), if used with a PLC or would set to a minimal debounce time if using thumbwheel switches. The strobe output indicates that the Model 500 is ready for parallel input.

The data read by the inputs goes from the least significant digit to the most significant digit. The lowest input numbers are the least significant bit of the lower digit. For example, if we specify inputs 6 - 13 as the inputs, the inputs would receive the following values during a **VRD** operation:



If the data VALID input is used, when the strobe delay times out, the output strobe will remain active until the data valid input becomes active.

When you strobe the output, the first set of data read by inputs 6 through 13 will become the least significant digits. The next time the output is *strobed*, the previously read data will be shifted two digits to the left, and the new value read will become the least significant digits. This process continues until you turn on and off the number of strobe outputs you have configured (maximum of 5 strobe outputs available).

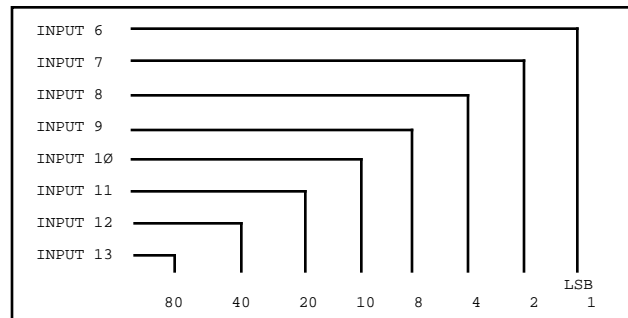
If you use a PLC or a thumbwheel to set the data, be sure to have a diode connected backwards between the inputs and the logic ground to prevent current from flowing. This will prevent the 500 indexer from reading false information.

Example

<u>Command</u>	<u>Description</u>
OUT6J	Set output 6 as strobe output
OUT7J	Set output 7 as strobe output
OUT8J	Set output 8 as strobe output
IN6N	Set input 6 as least significant bit data input
IN7N	Set input 7 as 2nd least significant bit data input
IN8N	Set input 8 as 3rd least significant bit data input
IN9N	Set input 9 as 4th least significant bit data input
IN10N	Set input 10 as 5th least significant bit data input
IN11N	Set input 11 as 6th least significant bit data input
IN12N	Set input 12 as 7th least significant bit data input
IN13N	Set input 13 as 8th least significant bit data input
STR500	Set strobe output delay time to 500 mseconds
VRD	Read parallel inputs as velocity data

The example above would cause the Model 500 to perform the following tasks as soon as it recognized the VRD command.

- Step 1 Turn on Output #6 for 500 msec.
- Step 2 Read the inputs 6 - 13 for the two digits.
- Step 3 Turn off Output #6 and turn on Output #7 for Model 500 msec.
- Step 4 Read two more digits.
- Step 5 Turn off Output #7 and turn on Output #8 for 500 msec.
- Step 6 Read two more digits.
- Step 7 Output 8 would then turn off



T Programming	Time			VALID Software Version A
SYNTAX <a>Tn	UNITS n = seconds	RANGE 0.01 to 999.99	DEFAULT None	ATTRIBUTES Buffered Savable in Sequence
EXECUTION TIME		SEE ALSO TRD		

Description The Time (T) command causes the Model 500 to wait the number of seconds that you specify with (n) before it executes the next command in the buffer. This command is useful whenever you need to dwell within a sequence.

Example	Command	Description
	> MN	Sets mode to normal
	> 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-Up	Set Inputs Debounce Time			VALID Software Version A11
SYNTAX <a>TDn	UNITS n = milliseconds	RANGE 1 to 1000	DEFAULT 2	ATTRIBUTES Buffered Savable in Sequence
EXECUTION TIME		SEE ALSO IN		
RESPONSE TO aTD IS *TDn				

Description This command sets the time (in milliseconds) that any of the inputs must remain activated to be recognized as true. Set the debounce value (n) to avoid problems with *bouncy* switches or electrically noisy environments.

WARNING

This command affects all inputs, including the end-of-travel limits. If the debounce value is too large and the end-of-travel limits are active for a period less than the TD value, **the limit inputs may be ignored by the Model 500.**

TDR Set-Up	Set Registration Debounce			VALID Software Version A
SYNTAX <a>TDRn	UNITS n = milliseconds	RANGE 1 to 255	DEFAULT 10	ATTRIBUTES Buffered Savable in Sequence
EXECUTION TIME		SEE ALSO IN, REG		
RESPONSE TO aTDR IS *TDRn				

Description This command sets the time in milliseconds that a registration input must be valid to initiate a registration move. The value should be set to avoid problems with *bouncy* switches or electrically noisy environments.

TEST Motion	Test			VALID Software Version A
SYNTAX <a>TEST	UNITS None	RANGE None	DEFAULT None	ATTRIBUTES Buffered Savable in Sequence
EXECUTION TIME		SEE ALSO None		

Description This command rotates the motor one revolution (as defined by the Motor Resolution [MR] command) in each direction at a velocity of 1 rps. It is designed to test the Model 500 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.**

Example

Command > MR2500 > TEST	Description Sets motor resolution to 25,000 pulses/rev. Model 500 sends 25,000 pulses to the drive in each direction (CW & CCW)
--------------------------------------	--

TF Set-Up	Set Following Time			VALID Software Version A
SYNTAX <a>TFn	UNITS n = milliseconds	RANGE 1 - 32	DEFAULT 4	ATTRIBUTES Buffered Savable in Sequence
EXECUTION TIME		SEE ALSO S		
RESPONSE TO aTF IS *FOLLOWING_TIME_n_MILLISECONDS				

Description This command sets the time (in milliseconds [ms]) that the encoder uses in the following function to obtain the Model 500 step output frequency. The TF value should be adjusted to obtain maximum smoothness of the motor shaft. For example, a TF4 command sets the sample time at 4 ms, every 4 ms the Model 500 outputs a new step frequency based on the accumulation of encoder pulses during those 4 ms.

TM Status	Move Time Report			VALID Software Version A
SYNTAX aTM	UNITS None	RANGE None	DEFAULT None	ATTRIBUTES Buffered Savable in Sequence
EXECUTION TIME		SEE ALSO		
RESPONSE TO aTM IS *MOVE_TIME_n_MILLISECONDS				

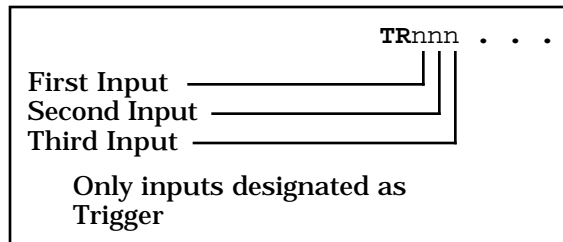
Description This command reports the length of time (in milliseconds) the previous move required for completion. The timer is initiated at the end of the Go (G) command and accumulates until the move is complete. The timer indicates the duration in milliseconds that was required to output the designated number of pulses.

Example	<u>Command</u>	<u>Description</u>
	> MN	Sets made to normal
	> A10	Sets acceleration to 10 rps ²
	> AD10	Sets deceleration to 10 rps ²
	> V1	Sets velocity to 1 rps
	> D25000	Sets distance to 25000 steps
	> G	Executes the move
	> 1TM	Reports the time the move required

TR Programming	Wait for Trigger			VALID 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
EXECUTION TIME		SEE ALSO IN, INL, TS		

Description

When the TR command is used, the Model 500 will get to this command and wait until the inputs (configured as triggers with the IN command) match the pattern, before going on to the next command. It will not scrutinize those inputs not designated as triggers. The voltage level that represents an active state is defined with the INL command. The order of comparison of the inputs designated as trigger inputs ranges from Input 1 to Input 13 (as labeled on the unit).



Triggers are used to synchronize Model 500 operations with external events. They can be used to implement a handshaking function with other devices. Three characters are used for nnnnn variables are listed below:

- 1 = Input active (ON)
- 0 = Input inactive (OFF)
- X = Ignore input

Example	<u>Command</u>	<u>Description</u>
	> 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
	> 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 Programming		Read Timer via Parallel Input/Output			VALID Software Version A
SYNTAX <a>TRD	UNITS None	RANGE None	DEFAULT None	ATTRIBUTES Buffered Savable in Sequence	
EXECUTION TIME			SEE ALSO IN, OUT, STR, T		

Description This command initiates the controller to read a time delay value from the parallel inputs and execute the delay. You must set up the inputs (typically using 8 inputs) as *data* inputs using the **IN** command. You must also set up outputs (typically using 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 Model 500 is ready for data. The following sequence of events takes place when you enter the **TRD** command:

- Step 1** The lowest output bit designated as a strobe output will go on and stay on for the strobe output delay time defined by the **STR** command.
- Step 1A** If an input is designated as data valid input, then the strobe output will stay active until the data valid input is active.
- Step 2** The 500 will read the parallel inputs designated as *data* inputs. The lowest inputs will be the least significant bit. The 500 reads 8 bits (2 BCD digits) and stores them into two digits in the timer 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 data valid input, the strobe output will stay active until the data valid input is active.
- Step 4** The Model 500 will read the parallel inputs designated as *data* inputs are placed in the lowest digits in the timer register. Previously entered values are shifted two positions to the left.
- Step 5** Steps 3, 3A, & 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 timer value.

Example The following example shows how the Model 500 reads the timer value of 7.51 seconds. We will use inputs 1 - 8 for data inputs and outputs 1 - 4 for strobe outputs.

Step 1 Type the following commands:

Command	Description
> OUT1J	See 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 the Most Significant DATA input
> STR500	Set strobe output delay time to 500 ms

Step 2 As soon as you enter the Read Timer via Parallel Input TRD command, the following should happen

Type the following command:

Command	Description
TRD	Start reading from Parallel input

Step 3 The Model 500 will turn on OUT1 for 500 ms.

Step 4 You must set inputs 1 - 8 to the following settings.

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

The current setting of the timer is 0000000000.

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

Step 6 You must set inputs 1 - 8 set to the following settings:

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

Previously entered value of 0 shifts 2 locations to the left and 00 is placed in the 2 lowest positions. The current setting of the timer value is: 0000000000.

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

Step 8 You must set inputs 1 - 8 set to the following settings:

IN8	IN7	IN6	IN5	IN4	IN3	IN2	IN1
OFF	OFF	OFF	OFF	OFF	ON	ON	ON
		0				7	

Previously entered value of 0000 shifts 2 locations to the left and 07 is placed in the two lowest positions. The current setting of the timer value is: 0.07.

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

Step 10 You must set inputs 1 - 8 set to the following settings:

IN8	IN7	IN6	IN5	IN4	IN3	IN2	IN1
OFF	ON	OFF	ON	OFF	OFF	OFF	ON
		5				1	

Previously entered valued 000007 shifts two locations to the left and 51 is placed in the two lowest positions. The final setting of the timer value becomes 000751.

Step 11 Since the fifth strobe output is not defined, the Model 500 now executes the time delay of 7.51 seconds.

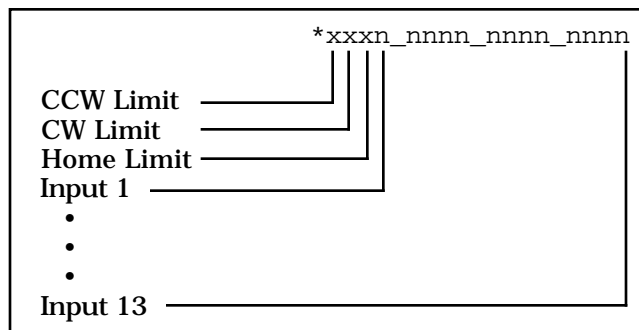
TS Status	Trigger Input Status			VALID Software Version A
SYNTAX aTS	UNITS None	RANGE None	DEFAULT None	ATTRIBUTES Immediate Not Saved
EXECUTION TIME		SEE ALSO IN, INL, TR		
RESPONSE TO aTS IS *xxxn_nnnn_nnnn_nnnn (n = 0, 1, or X)				

Description This command reports the state of the trigger inputs. Response is in the form xxxn_nnnn_nnnn_nnnn where

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

The first three 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 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 the TS command.



Example

<u>Command</u>	<u>Response</u>
> 1TS	*xxx1_xxxx_11xx_xx00 (Inputs 1, 6, 7, 12 and 13 are configured as triggers with 1, 6 and 7 are active and 12 and 13 are inactive.)

TW Programming	Select Thumbwheel Operation Mode			VALID Software Version B2
SYNTAX <a>TWn	UNITS n = mode	RANGE 0 or 1	DEFAULT 0	ATTRIBUTES Buffered Savable in Sequence
EXECUTION TIME		SEE ALSO IN, OUT, STR		

Description This command selects one of two possible mode of operation for the use of thumbwheels. When **TW1** is issued, the Model 500 is set up to use Compumotor's TM8 Thumbwheel Module. Refer to the *Thumbwheel Interface* section in Chapter 4 of the **Model 500 Indexer User Guide** for installation instructions. This mode uses four inputs for thumbwheel data.

The alternative method is to use your own thumbwheels. In this case, you must issue **TW0** (this is the default setting of the **TW** command). This mode requires the use of eight inputs as thumbwheel data inputs. Refer to the *Thumbwheel Interface* section in Chapter 4 of the **Model 500 Indexer User Guide** for installation instructions.

Example	<u>Command</u>	<u>Description</u>
	> TW1	Enables the use of Compumotor's TM8 thumbwheel module
	> TW0	Selects the mode for using your own thumbwheels

TX Programming	Transmit Variable and String			VALID Software Version B2
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
EXECUTION TIME		SEE ALSO VAR		

Description 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 or 0 if the sign is not 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

Example	<u>Command</u>	<u>Description</u>
	> 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

U Programming		Pause and Wait for Continue			VALID Software Version A
SYNTAX <a>U	UNITS None	RANGE None	DEFAULT None	ATTRIBUTES Immediate Never Saved	
EXECUTION TIME			SEE ALSO C, PS		

Description This command causes the Model 500 to complete the command in progress, then wait until it receives a Continue (C) to resume processing. Since the buffer is saved, the Model 500 continues to execute the program (at the point where it was interrupted). The Model 500 continues processing when it receives the C command. This command is typically used to stop a machine while it is unattended.

Example	<table border="0"> <tr> <td style="text-align: right;"><u>Command</u></td> <td style="text-align: left;"><u>Description</u></td> </tr> <tr> <td>> MN</td> <td>Sets move to Normal mode</td> </tr> <tr> <td>> A5</td> <td>Sets acceleration to 5 rps²</td> </tr> <tr> <td>> AD10</td> <td>Sets deceleration to 10 rps²</td> </tr> <tr> <td>> V5</td> <td>Sets velocity to 5 rps</td> </tr> <tr> <td>> L0</td> <td>Loops indefinitely</td> </tr> <tr> <td> D25600</td> <td>Sets distance to 25,600 steps</td> </tr> <tr> <td> G</td> <td>Executes the move</td> </tr> <tr> <td> T10</td> <td>Waits 10 seconds after the move</td> </tr> <tr> <td>> N</td> <td>Ends loop</td> </tr> <tr> <td>> U</td> <td>Halts execution until the Model 500 receives the Continue command.</td> </tr> </table>	<u>Command</u>	<u>Description</u>	> MN	Sets move 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	T10	Waits 10 seconds after the move	> N	Ends loop	> U	Halts execution until the Model 500 receives the Continue command.
<u>Command</u>	<u>Description</u>																						
> MN	Sets move 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																						
T10	Waits 10 seconds after the move																						
> N	Ends loop																						
> U	Halts execution until the Model 500 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 Model 500 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 Programming		Until			VALID Software Version A
SYNTAX <a>UNTIL(e)	UNITS e= evaluation commands	RANGE see below	DEFAULT None	ATTRIBUTES Buffered Savable in Sequence	
EXECUTION TIME			SEE ALSO UNTIL, WHILE, NWHILE, Evaluation Commands		

Description The UNTIL command marked the end of the REPEAT command. The UNTIL command is evaluated and if it is true, 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 evaluates false, then the commands after the UNTIL command are executed.

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

Example	<table border="0"> <tr> <td style="text-align: right;"><u>Command</u></td> <td style="text-align: left;"><u>Description</u></td> </tr> <tr> <td>> REPEAT</td> <td>Repeat</td> </tr> <tr> <td>> GD1</td> <td>Execute predefined move #1</td> </tr> <tr> <td>> T1</td> <td>Time delay of one seconds</td> </tr> <tr> <td>> UNTIL(INXXX10_OR_INXXX01)</td> <td>If input #1 on and input #2 off, or input #1 off and input #1 on, do next command, else execute from command following REPEAT</td> </tr> </table>	<u>Command</u>	<u>Description</u>	> REPEAT	Repeat	> GD1	Execute predefined move #1	> T1	Time delay of one seconds	> UNTIL(INXXX10_OR_INXXX01)	If input #1 on and input #2 off, or input #1 off and input #1 on, do next command, else execute from command following REPEAT
<u>Command</u>	<u>Description</u>										
> REPEAT	Repeat										
> GD1	Execute predefined move #1										
> T1	Time delay of one seconds										
> UNTIL(INXXX10_OR_INXXX01)	If input #1 on and input #2 off, or input #1 off and input #1 on, do next command, else execute from command following REPEAT										

V Motion	Velocity			VALID Software Version A
SYNTAX <a>Vn	UNITS n = rps	RANGE 0 - 99.99999	DEFAULT 1	ATTRIBUTES Buffered Savable in Sequence
EXECUTION TIME		SEE ALSO A, CPE, D, FSD, G		
RESPONSE TO aV IS *Vn				

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 non-volatile 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 Model 500 will fault and create a stall condition.*

Example	<u>Command</u>	<u>Description</u>
	> MC	Sets move to continuous
	> A5	Sets acceleration to 5 rps ²
	> AD10	Sets deceleration to 10 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.

VAR Programming	Variables			VALID Software Version A
SYNTAX <a>VARn	UNITS n = variables	RANGE 1 - 50	DEFAULT 0	ATTRIBUTES Buffered Savable in Sequence
EXECUTION TIME		SEE ALSO		
RESPONSE TO aVARn IS *snnnnnnnnnnn.nnnnn				

Description Fifty variables (VAR1 through 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

Examples	Command	Description
	<code>VAR1=50</code>	Load variable #1 with 50
	<code>XR(VAR1)</code>	Execute sequence #50

You can use variables in mathematical operations to obtain new values:

- Addition: `VARn=VARn+VARn`
- Subtraction: `VARn=VARn-VARn`
- Division: `VARn=VARn/VARn`
- Multiplication: `VARn=VARn*VARn`

A constant, the present value of the position counter (POS), the encoder's position (ABS), or the following encoder's position (FEP) can be substituted for the operands (see examples below).

Examples	Command	Description
	<code>> VAR5=VAR7</code>	Load variable #5 with variable #7
	<code>> VAR8=POS+50000</code>	Read position of motor (plus 50,000) into variable #8
	<code>> VAR4=FEP</code>	Read position of following encoder into variable #4
	<code>> VAR1=20</code>	Load variable #1 with 20
	<code>> VAR2=5</code>	Load variable #2 with 5
	<code>> VAR3=VAR1-VAR2</code>	Variable #3 now contains (20-5) or 15
	<code>> VAR1=VAR2+20</code>	Variable #1 now contains (5+20) or 25

VARD Programming	Read Variable Via Parallel Input/Output			VALID Software Version A
SYNTAX <a>VARDn	UNITS n = variable	RANGE 1 to 50	DEFAULT None	ATTRIBUTES Buffered Savable in Sequence
EXECUTION TIME		SEE ALSO STR, IN, OUT		

Description This command initiates the controller to read a variable value from the parallel inputs. You must set up the inputs as Data input using the **IN** command. You must also set up outputs as strobe outputs using the **OUT** command. Typically, you would set up 8 inputs as Data Inputs and 5 outputs as Strobe Outputs using the **OUT** command. If an input is designated as a Data Sign, then if it is active during any byte read, the sign becomes negative, otherwise the inactive condition (default) is a positive sign.

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 Model 500 is ready for data. The following sequence of events takes place when you enter the **VARD** 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 Model 500 will read the parallel inputs designated as data inputs. The lowest inputs will be the least significant bit. The Model 500 reads 8 bits (2 BCD digits) and stores them into two digits on the variable 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 500 will read the parallel inputs designated as Data inputs are placed in the lowest digits in the variable 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 variable value.

Example The following example shows how the 500 reads the value of 3589721 for variable 2. We will use inputs 1 through 8 for data inputs and outputs 1 through 4 for strobe outputs.

Step 1 Type the following commands:

Command	Description
> 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 Output 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 the Most significant DATA input
> STR500	Set strobe output delay time to 500 msec.

Step 2 As soon as you enter the Read Variable via Parallel Input (VARD) command for variable #2, the following should happen

Enter the following command:

Command	Description
> VARD2	Start reading from Parallel input

Step 3 The Model 500 will turn on OUT1 for 500 ms.

Step 4 You must set inputs 1 - 8 to the following settings.

IN8	IN7	IN6	IN5	IN4	IN3	IN2	IN1
OFF	OFF	OFF	OFF	OFF	OFF	OFF	ON
		0				2	

The current setting of variable #2 is 0000000001.

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

Step 6 You must set inputs 1 - 8 set to the following settings

IN8	IN7	IN6	IN5	IN4	IN3	IN2	IN1
OFF	OFF	OFF	OFF	OFF	OFF	ON	ON
		2				3	

The previously entered value of 01 shifts 2 locations to the left and 23 is placed in the two lowest positions. The current setting of the variable value is: 00000.00123.

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

Step 8 You must set inputs 1 - 8 set to the following settings:

IN8	IN7	IN6	IN5	IN4	IN3	IN2	IN1
OFF	ON	OFF	OFF	OFF	ON	OFF	ON
		4				5	

Previously entered values 0.00123 shifts 2 locations to the left and 45 is placed on the two lowest positions. The current setting of the variable value is: 00000.12345.

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

Step 10 You must set inputs 1 - 8 set to the following settings.

IN8	IN7	IN6	IN5	IN4	IN3	IN2	IN1
OFF	ON	ON	OFF	OFF	ON	ON	ON
		6				7	

Previously entered values 0.12345 shifts two locations to the left and 67 is placed in the two lowest positions. The final setting of the variable value becomes 12.34567.

Step 11 Since the fifth strobe output is not defined, the Model 500 now goes to the next command in the buffer.

VRD Programming	Read Velocity Value from Parallel Input/Output			VALID Software Version A
SYNTAX <a>VRD	UNITS None	RANGE None	DEFAULT None	ATTRIBUTES Buffered Savable in Sequence
EXECUTION TIME		SEE ALSO IN, OUT, STR		

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 Model 500 is ready for data.

The following sequence of events takes place when you 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 Model 500 will read the parallel inputs designated as data inputs. The lowest inputs will be the least significant bit. The Model 500 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 Model 500 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 velocity.

The velocity range you can enter via parallel interface is from 0.00001 to 99.99999.

Example The following example shows how the Model 500 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 Enter the following commands:

Command	Description
> 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 Type the following commands:

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)

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

Step 3 The Model 500 will turn on OUT1 for 500 ms.

Step 4 You must set inputs 1 - 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 Model 500 reads the input, OUT1 turns off and OUT2 will automatically go on for 500 ms.

Step 6 You must set inputs 1 - 8 set to the following settings.

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

The previously entered value of 03 shifts 2 locations to the left and 58 is placed in the two 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 ms.

Step 8 You must set inputs 1 - 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 in the two lowest positions. The current setting of the velocity value is: 0.35897 rps.

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

Step 10 You must set inputs 1 - 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 to the left and 21 is placed in the two lowest positions. The final setting of the velocity value becomes 35.89721 rps.

Step 11 Since the fifth strobe output is not defined, the Model 500 now goes to the next command in the buffer and executes the move at a velocity of 35.89721 rps.

VS Motion	Set Start/Stop Velocity			VALID Software Version A
SYNTAX <a>VSn	UNITS n = rps (revs/second)	RANGE 0.00000 - 99.99999	DEFAULT 0	ATTRIBUTES Buffered Savable in Sequence
EXECUTION TIME		SEE ALSO V, MR		

Description This command should only be used with low-resolution drives (200; 400; or 1,000 pulses per revolution). This is the minimum velocity employed by an MN or MC moves. The motor will jump immediately to vs value before ramping to a v value. It is typically used to prevent full-step motors from stalling at the resonance point. 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 move up to the v command value immediately).

Example

<u>Command</u>	<u>Description</u>
> MR200	Sets motor resolution to 200 pulses per rev
> MC	Sets mode continuous
> vs1	Start/Stop velocity set to 1 rps
> v5	Sets velocity to 5 rps
> A200	Sets acceleration to 20 rps ²
> G	Executes the move (Go)

Motor will immediately reach 1 rps, then accelerate at 20 rps² to 5 rps.

W1 Status	Signed Binary Position Report			VALID Software Version A
SYNTAX aW1	UNITS None	RANGE None	DEFAULT None	ATTRIBUTES Immediate Not Saved
EXECUTION TIME		SEE ALSO PR, W2, W3		
RESPONSE TO aW1 IS *n (n = steps)				

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 (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. Move in negative direction will report back negative number (bit 31 is set to 1).

Interpreting Binary Position Reports

This form of position report (nnnnn), consists of five bytes. The first will be the axis number, followed by four bytes that must be linked together (concatenated) to form a 32-bit binary number. A typical STD22 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 ^@, #, ø, and / (^@ refers to the CTRL key and @, an unprintable character) is given. The binary code for this response should be:

^@ # ø /
 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 ^@ # ø /. The formula used for the binary conversion is:

ASCII Value • Character Multiplier = Character Value

Response	ASCII Value	Character Multiplier	Conversion (steps)
^@	0	16,777,216 (256 ³)	47
#	35	65,536 (256 ²)	12,288
ø	48	256 (256 ¹)	2,293,760
/	47	1 (256 ⁰)	0
			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 Model 500 units are daisy-chained.

W2 Status	Hexadecimal Position Report			VALID Software Version A
SYNTAX <a>W2	UNITS None	RANGE None	DEFAULT None	ATTRIBUTES Immediate Not Saved
EXECUTION TIME		SEE ALSO PR, W1, W3		
RESPONSE TO aW2 IS *nnnnnnnn (nnnnnnnn = steps)				

Description

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
n (MSD)	$n \cdot 16^7 = n \cdot 268,435,456 = \underline{\hspace{2cm}}$
n	$n \cdot 16^6 = n \cdot 16,777,216 = \underline{\hspace{2cm}}$
n	$n \cdot 16^5 = n \cdot 1,048,576 = \underline{\hspace{2cm}}$
n	$n \cdot 16^4 = n \cdot 65,536 = \underline{\hspace{2cm}}$
n	$n \cdot 16^3 = n \cdot 4,096 = \underline{\hspace{2cm}}$
n	$n \cdot 16^2 = n \cdot 256 = \underline{\hspace{2cm}}$
n	$n \cdot 16^1 = n \cdot 16 = \underline{\hspace{2cm}}$
n (LSD)	$n \cdot 16^0 = n \cdot 1 = \underline{\hspace{2cm}}$

The variable n can be one of the following values:

Decimal	Hexadecimal
Value	Value (n)
Ø	Ø

W3 Status	Signed Hexadecimal Position Request			VALID Software Version A
SYNTAX aw3	UNITS None	RANGE None	DEFAULT None Never Saved	ATTRIBUTES Immediate
EXECUTION TIME <10 ms		SEE ALSO PR		
RESPONSE TO aw3 IS *nnnnnnnn (n = steps)				

Description

The Immediate Position Request (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 through 9 and A through 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
*nnnnnnnn

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
n (MSD)	$n \cdot 16^7 = n \cdot 268,435,456 = \underline{\hspace{2cm}}$
n	$n \cdot 16^6 = n \cdot 16,777,216 = \underline{\hspace{2cm}}$
n	$n \cdot 16^5 = n \cdot 1,048,576 = \underline{\hspace{2cm}}$
n	$n \cdot 16^4 = n \cdot 65,536 = \underline{\hspace{2cm}}$
n	$n \cdot 16^3 = n \cdot 4,096 = \underline{\hspace{2cm}}$
n	$n \cdot 16^2 = n \cdot 256 = \underline{\hspace{2cm}}$
n	$n \cdot 16^1 = n \cdot 16 = \underline{\hspace{2cm}}$
n (LSD)	$n \cdot 16^0 = n \cdot 1 = \underline{\hspace{2cm}}$

The decimal (n) may have one of the values shown below.

Decimal Value	Hexadecimal Value (n)
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	A
11	B
12	C
13	D
14	E
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 16⁸ (2³² 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.

Example

The indexer responds to w3 as follows: *FFFF9E58

1. Chop off the F's: 9E58 hex = 40,536
2. Subtract from 16⁴ 10000 hex = 65,536
3. Subtraction Result = -25,000

WHEN	Set When Condition			VALID
Set-Up				Software Version A
SYNTAX <a>WHEN(e)	UNITS e = evaluation commands	RANGE (see below)	DEFAULT None	ATTRIBUTES Buffered Savable in Sequence
EXECUTION TIME		SEE ALSO XWHEN, Evaluation Commands		
RESPONSE TO aWHEN IS *(condition)				

Description

The WHEN command allows you to continuously examine a set of conditions and if the condition evaluates true, the 500 will execute the sequence defined by the XWHEN command. The command currently in progress will finish, and then the XWHEN sequence executes.

Example

Command	Description
> XWHEN2	Set WHEN sequence to 2
> WHEN (INXXX1)	When input #2 becomes active, execute WHEN sequence #2

WHILE Programming	While			VALID Software Version A
SYNTAX <a> WHILE (e)	UNITS e = evaluation	RANGE evaluation commands	DEFAULT None	ATTRIBUTES Buffered Savable in Sequence
EXECUTION TIME		SEE ALSO NWHILE, REPEAT, UNTIL, Evaluation Commands		

Description

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** command evaluates true, then the commands between the **WHILE** and **NWHILE** commands are executed. However, if the **WHILE** command evaluates false, then program execution jumps to the command after the **NWHILE** command.

The **WHILE** condition is evaluated at the end of the current move, between the **WHILE** and **NWHILE** commands (see example below).

Up to 16 levels of **WHILE** commands may be nested.

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

Example

<u>Command</u>	<u>Description</u>
> WHILE (INXXX1)	While input #1 is active, continue to do the commands between WHILE and NWHILE
> GD1	Execute predefined move 1
> T2.0	Delay two seconds
> NWHILE	End of while

NOTE: The input condition (INXXX1) will be evaluated after the **GD1** command is executed.

XBS Status	Sequence Memory Available Report			VALID Software Version A
SYNTAX aXBS	UNITS None	RANGE None	DEFAULT None	ATTRIBUTES Immediate Not Saved
EXECUTION TIME		SEE ALSO XDIR		
RESPONSE TO aXBS IS *n_OF_8000_BYTES_(x%)_SEQUENCE_MEMORY_REMAINING n = bytes 0 - 8,000, x = % 0 - 100				

Description

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). This command is useful to find out how much more programming can be done on the Model 500, after defining several programs.

This command reports both number of bytes available, and the percentage (%) of memory available.

Example

<u>Command</u>	<u>Response</u>
> 1XBS	*4000_OF_8000_BYTES_(50%)_SEQUENCE_MEMORY_REMAINING (There are 4,000 bytes [50%] of the sequence buffer remaining for you to program.)

XC Status	Sequence Checksum Report			VALID Software Version A
SYNTAX aXC	UNITS None	RANGE None	DEFAULT None	ATTRIBUTES Buffered Savable in Sequence
EXECUTION TIME		SEE ALSO XD, XE		
RESPONSE TO aXC IS *n (n = byte sum 0 - 255)				

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. Use the XC command to remove a 3Ø error (bad battery-backed RAM checksum).

Example

<u>Command</u>	<u>Response</u>
> 1XC	*ØØ149

XD Programming	Sequence Definition			VALID Software Version A
SYNTAX <a>XDn	UNITS n = sequence #	RANGE 1 - 100	DEFAULT None	ATTRIBUTES Buffered Never Saved
EXECUTION TIME		SEE ALSO XE, XR, XT		

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.**

Example

<u>Command</u>	<u>Description</u>
> XE1	Erase sequence #1
> XD1	Define sequence #1
> MN	Sets mode normal
> A1Ø	Sets acceleration to 10 rps ²
> AD2Ø	Sets deceleration to 20 rps ²
> V5	Sets velocity to 5 rps
> D1ØØØØ	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 Status	Sequence Directory			VALID Software Version A
SYNTAX aXDIR	UNITS None	RANGE None	DEFAULT None	ATTRIBUTES Buffered Never Saved
EXECUTION TIME		SEE ALSO XBS		
RESPONSE TO aXDIR IS see below				

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:

```
1XDIR *NO_SEQUENCES_DEFINED - If there are no sequences 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.

Example

<u>Command</u>	<u>Response</u>
> 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 Programming	Sequence Erase			VALID Software Version A
SYNTAX <a>XEn	UNITS n = sequence #	RANGE 1 - 100	DEFAULT None	ATTRIBUTES Buffered Savable in Sequence
EXECUTION TIME		SEE ALSO XD, XR, XT		

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.**

Example

<u>Command</u>	<u>Description</u>
> XE1	Deletes Sequence #1

XFK Motion	Set Fault or Kill Sequence			VALID Software Version A
SYNTAX <a> XFK n	UNITS n = sequence #	RANGE 0 - 100	DEFAULT 0	ATTRIBUTES Buffered Savable in Sequence
EXECUTION TIME		SEE ALSO K, XR		
RESPONSE TO a XFK IS *n				

Description

This command selects the sequence that will be executed if a fault or kill condition occurs. A selection of Ø 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
16	NON-COMMANDED DRIVE OFF
20	STALL DETECTED
30	BATTERY BACKED RAM 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 (κ) command over the RS-232C interface or parallel interface.

Example

<u>Command</u>	<u>Description</u>
> XFK 5	Execute Sequence #5 when fault or kill condition exists
> XE 5	Erase Sequence #5
> XD 5	Define Sequence #5
1 " FAULT_OR_KILL	Send the message
> XT	End Sequence Definition

Whenever a Fault or Kill occurs, the indexer will transmit **FAULT_OR_KILL**.

XG Programming		GOTO Sequence			VALID Software Version A
SYNTAX <a>XGn	UNITS n = sequence #	RANGE 1 - 100	DEFAULT None	ATTRIBUTES Buffered Savable in Sequence	
EXECUTION TIME			SEE ALSO XR, GOTO, GOSUB		

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 or GOSUB commands. There are no limitations on the number of XG commands as there is no nesting involved.

Example	<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
	1PR	Absolute Position Report
	> XT	End Sequence #5 definition
	> XR1	Execute sequence #1

The motor will move 10,000 steps, then the controller will jump to Sequence #5 and indicate the absolute position.

XQ Set-Up		Sequence Interrupted Run Mode			VALID Software Version A
SYNTAX aXQn	UNITS n = mode	RANGE 0, 1	DEFAULT 0	ATTRIBUTES Buffered Savable in Sequence	
EXECUTION TIME			SEE ALSO IN, SSJ, XD, XE, XP, XZ		
RESPONSE TO aXQ IS *n_INTERRUPTED MODE_ON or *n_INTERRUPTED MODE_OFF					

Description 1XQ1 = Set interrupted run mode
1XQ0 = 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 Model 500 will not accept a sequence selected from the inputs, until all sequence select lines have been brought inactive. After all lines have simultaneously been brought inactive, the Model 500 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 commands to stop sequence execution.

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

When the Model 500 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 Programming	Run a Sequence			VALID Software Version A
SYNTAX <a>XRn	UNITS n = sequence #	RANGE 1 - 100	DEFAULT None	ATTRIBUTES Buffered Savable in Sequence
EXECUTION TIME		SEE ALSO XD, XE, XG, XRP, XT, GOSUB, GOTO		

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.

Example	<u>Command</u> > XE1 > XD1 A10 AD20 V5 D10000 G > XT > XR1	<u>Description</u> Erase Sequence #1 Define Sequence #1 Sets acceleration to 10 rps ² Sets deceleration to 20 rps ² Sets velocity to 5 rps Sets distance to 10,000 steps Executes the move (Go) End defining Sequence #1 Execute Sequence #1
----------------	---	---

Sequence #1 is defined and executed using XD1 and XR1 commands respectively.

XRD Motion	Read Sequence via Parallel Input/Output			VALID Software Version A
SYNTAX <a> XRD	UNITS None	RANGE None	DEFAULT None	ATTRIBUTES Buffered Savable in Sequence
EXECUTION TIME		SEE ALSO IN, OUT, SSJ, STR		

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 inputs using the **IN** command. You must also set up outputs as strobe outputs. Typically, you would set up 8 inputs as data inputs using the **IN** command, and 5 outputs as strobe outputs using the **OUT** command. However, if only **XRD** is used, 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 Model 500 is ready form data. The following sequence of events takes place when you 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 Model 500 will read the parallel inputs designated as data inputs. The lowest inputs will be the least significant bit. The Model 500 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 the **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 Model 500 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 number of strobe outputs that 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.

Example

The following example shows how the Model 500 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 Enter the following commands.

<u>Command</u>	<u>Description</u>
> XE21	Erase sequence #21
> XD21	Define sequence #21
A20	Sets acceleration to 20 rps ²
V5	Sets velocity 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
> 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
> STR50	Set strobe output delay time to 50 msec.
> A10	Sets acceleration to 10 rps ²
> V5	Set velocity to 5 rps

Step 2 Enter the Read Variable via Parallel Input (XRD) command:

<u>Command</u>	<u>Description</u>
> XRD	Start reading from the parallel input

Step 3 As soon as you enter XRD, the 500 will turn on OUT1 for 50 ms.

Step 4 You must have 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

21 is placed on the two lowest positions. The final setting of the sequence value becomes 21.

Step 5 The Model 500 now checks to see if there is another strobe output. Since we do not have another strobe output, the Model 500 controller executes Sequence #21 which moves the motor 25,000 steps.

XRP Programming	Sequence Run With Pause			VALID Software Version A
SYNTAX <a>XRPn	UNITS n = sequence #	RANGE 1 - 100	DEFAULT None	ATTRIBUTES Buffered Savable in Sequence
EXECUTION TIME		SEE ALSO C, XD, XE, XR, XT		

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 Model 500 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.

Example

<u>Command</u>	<u>Description</u>
> 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	Model 500 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.

XS Status	Sequence Execution Status			VALID Software Version A
SYNTAX aXS	UNITS None	RANGE None	DEFAULT None	ATTRIBUTES Immediate Savable in Sequence
EXECUTION TIME		SEE ALSO XTR		
RESPONSE TO aXS IS *SEQUENCE_n_COMMAND_n or *SEQUENCE_EXECUTION_NOT_IN_PROGRESS				

Description 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 message indicating that is transmitted. This command is useful to determine the progress of program execution.

Example

<u>Command</u>	<u>Description</u>
> 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
D20000	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 Set-Up	Sequence Step Mode			VALID Software Version A
SYNTAX aXSTn	UNITS n = mode	RANGE 0 or 1	DEFAULT 0	ATTRIBUTES Buffered Savable in Sequence
EXECUTION TIME		SEE ALSO #, XR, XTR		
RESPONSE TO aXST IS *n_STEP_MODE_ACTIVE or *n_STEP_MODE_INACTIVE				

Description

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 the 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 the Trace (XTR) mode, the controller will 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 the sequence execution.

Example

<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
> 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 m 10,000 steps
> #	Execute the 5th command
SEQUENCE_001_COMMAND_XT	Displays the last command executed

XT Programming	Sequence Termination			VALID Software Version A
SYNTAX <a>XT	UNITS None	RANGE None	DEFAULT None	ATTRIBUTES Buffered Savable in Sequence
EXECUTION TIME		SEE ALSO XD, XE, XR		

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.

Example

<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 Set-Up	Set Trace Mode			VALID Software Version A
SYNTAX aXTRn	UNITS n = mode	RANGE 0 or 1	DEFAULT 0	ATTRIBUTES Buffered Savable in Sequence
EXECUTION TIME		SEE ALSO XR, XST		
RESPONSE TO aXTR IS *n_TRACE_MODE_ACTIVE or *n_TRACE_MODE_INACTIVE				

Description This command transmits the command that is being executed from the Model 500 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 also display the iteration count.

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

Example

<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
> 1XTR1	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
```

XU Set-Up	Upload Sequence			VALID Software Version A
SYNTAX aXU	UNITS n = sequence #	RANGE 1 - 100	DEFAULT None	ATTRIBUTES Buffered Savable in Sequence
EXECUTION TIME		SEE ALSO XD, XE, XT		
RESPONSE TO aXU IS *(sequence contents)				

Description This command uploads the identified 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, *_ is transmitted to the host.

Example

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

XWHEN Set-Up	Set When Sequence			VALID Software Version A
SYNTAX aXWHENn	UNITS n = sequence #	RANGE 0 - 100	DEFAULT 0	ATTRIBUTES Buffered Savable in Sequence
EXECUTION TIME		SEE ALSO WHEN		
RESPONSE TO aXWHEN IS *n				

Description This command selects the sequence that will be executed upon a **WHEN** condition evaluating true. A selection of 0 causes no sequences to be executed. The **WHEN** condition continually examines a set of conditions, and if the condition evaluates true, the **XWHEN** sequence executes.

Example

<u>Command</u>	<u>Description</u>
> XWHEN5	Execute sequence 5 when the WHEN condition is true

Y Programming	Stop Loop			VALID Software Version A
SYNTAX <a>Y	UNITS None	RANGE None	DEFAULT None	ATTRIBUTES Immediate Not Saved
EXECUTION TIME		SEE ALSO L, LRD, N		

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 Model 500 reaches the last command of the current loop. At that time, the Model 500 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.

Example	<u>Command</u>	<u>Description</u>
	> 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

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 Programming	Reset			VALID Software Version A
SYNTAX <a>Z	UNITS None	RANGE None	DEFAULT None	ATTRIBUTES Immediate Not Saved
EXECUTION TIME		SEE ALSO K, S		

Description The Reset (z) command is equivalent to cycling AC power to the Model 500. 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 Model 500 is busy for 3,500 ms and ignores all commands.**

This command sets all position counters to zero.

Index

- 500 INDEXER STATUS 79
- ABSOLUTE ENCODERS 23, 28, 33, 35, 77
 ABSOLUTE POSITION 17, 28, 74, 75, 77
 ABSOLUTE POSITION MODE 61
 COUNTER 91, 97
 ABS (VARIABLE OPERATND
 SUBSTITUTE) 109
 SET TO ZERO 77
 POSITION REPORT 75
 ACCELERATION 6, 7, 36
 JOG 51
 NON-LINEAR 84
 ACTIVE INPUT LEVEL 48
 AL-C 23, 35
 ALTERNATE MODE 93
 ALTERNATE MODE STOP 93
 AR-C 23, 35
- BACKLASH 9
 BACKSPACE 42
 BACKUP TO HOME SWITCH 38, 68
 BOUNCY SWITCHES 100
 BUFFER STATUS 7, 10
 BUFFERED COMMANDS 3
- C DRIVE 63
 CARRIAGE RETURN 12
 CHANGE SUMMARY I
 CHECKSUM 119
 COMMAND BUFFER 54, 87, 130
 SAVE ON LIMIT 93
 SAVE ON STOP 94, 98
 COMMAND TYPE 2
 COMMENT FIELD 5
 CONFIGURE ENCODER RESOLUTION 23
 CONTINUE 10, 54, 76, 107
 CONTINUE SEQUENCE SCAN AFTER STOP
 71
 CONTINUOUS MODE 60
 CONTINUOUS SCAN MODE 95, 122
 CORRECTION VELOCITY 64
 COUNT DEVIATION 25
- DAISY-CHAIN 94
 DATA INPUTS 19, 29, 46, 58, 103, 110, 124
 DATA SIGN INPUT 47
 DATA VALID INPUT 47
 DE-ENERGIZE 67, 68, 97
 DEADBAND WINDOW 21
 DEBOUNCE TIME 18, 28, 57, 100, 103
 FOR ALL INPUTS 100
 DECELERATION 6, 7, 36, 98
 JOG 51
 LIMIT 55
 DECREASED FOLLOWING VALUE INPUT 47
 DEFINE ACTIVE STATE OF HOME SWITCH 69
 DFS 13
 DIRECTION 43
 DIRECTION INPUT 45
 DISABLE INPUTS 15
 DISABLE OUTPUTS 15
 DISABLE RS-232C COMMUNICATION 24
 DISPLAY
 ACTUAL POSITION 17
 ACTUAL VELOCITY 21
 FLAGS FOR INDEXER STATUS 14
 FLAGS FOR SERVO PARAMETERS 13
 PARAMETERS 18
 POSITION 17
 POSITION ERROR 17
 SETPOINT POSITION 17
 VELOCITY SETPOINT 21
 DISTANCE 13, 36
 DISTANCE POINT 16
 DRIVE FAULT POLARITY 97
 DRIVE SHUTDOWN INPUT 45
 DWELL 100
- ELSE 23
 EMERGENCY STOP 54
 ENABLE COMMUNICATIONS INTERFACE 22
 ENCODER
 FOLLOWING MODE 26
 FUNCTION REPORT 30
 INPUTS 33
 POSITION 77
 PULSES 30
 RESOLUTION 23, 27, 31
 SELECT 33
 STEP MODE 31, 38
 Z CHANNEL INPUT 69
 END LOOP 130
 END OF IF 65
 END OF LOOP 64
 END OF WHILE 66
 END POSITION PROFILE 65
 END-OF-LOOP 54
 END-OF-TRAVEL LIMIT INPUT (SEE *LIMITS*,
END-OF-TRAVEL)
 ENERGIZE 67, 68, 97
 ERASE A SEQUENCE 120
 ERROR CODES 121
 ERROR CONDITIONS 86
 ERROR MESSAGES 86
 ERRORS REPORT 86
 EVALUATION COMMANDS/STATEMENTS 3,
 23, 44, 65, 66, 82, 107, 108, 117, 118
- FACTORY DEFAULT SETTINGS 83
 FAULT SEQUENCE 121
 FEP (FOLLOWING ENCODER POSITION) 109
 FINAL HOMING DIRECTION 70
 FOL VALUE 33
 FOLLOWER COUNTER TO ZERO 75
 FOLLOWER POSITION REPORT 74
 FOLLOWING
 ADJUSTMENT 25
 BASE 25
 ENCODER ABSOLUTE POINT 28
 ENCODER POINT 27
 ENCODER POSITION (FEP) 109
 ERROR 108
 INCREMENT 26
 LEARN COUNT 25
 LEARN MODE 25, 34
 MIMIC MODE 30
 MODE 33
 PERCENT 26
 RATIO 26-27, 96
 SELF CORRECTION MODE 34
 SYNCHRONIZED ACCEL. 33
 SYNCHRONIZATION COUNT 25
 SYNCHRONIZATION RATE 24
 TIME 101
 VALUE 34
 FRONT PANEL PUSHBUTTONS 11, 96
 FUNCTION SETUP REPORT 68, 92
- GO 36
 GO DEFINED 37
 GO HOME 38, 69
 ACCELERATION 39
 DECELERATION 39
 FINAL DIRECTION 70
 FINAL VELOCITY 40
 HOME SWITCH
 ACTIVE STATE DEFINITION 69
 STATUS 82
 VELOCITY 38, 40
 GO HOME INPUT 47
 GO HOME STATUS REPORT 82
 GO INPUT 45
 GOSUB SEQUENCE 41
 GOTO SEQUENCE 42, 122
- HALT 43
 HANDSHAKING 102
 HEXADECIMAL POSITION REPORT 115
 HIGH-LEVEL PROGRAMMING COMMANDS 2
 HOME LIMIT (SEE *LIMITS*, *HOME* BELOW)
 HOME REFERENCE 69
 HOME REGION 69
- IF 44
- IMMEDIATE COMMANDS 3
 IMMEDIATE OUTPUT 49
 IMMEDIATE PAUSE 54
 IMMEDIATE VELOCITY 50
 INCREASE FOLLOWING VALUE INPUT 47
 INCREMENTAL ENCODERS 33, 35, 69, 77
 INCREMENTAL POSITION MODE 61, 74
 INDEXER STATUS 14
 INPUTS
 DISABLE 15
 LIMITS (SEE *LIMITS* BELOW)
 PROGRAMMABLE (SEE
 PROGRAMMABLE INPUTS BELOW)
 SET ACTIVE LEVEL 48
 SET DEBOUNCE 100
 STATUS 50
 INTERACTIVE MODE 94
- JOG
 ACCELERATION 51
 CCW INPUT 46
 CW INPUT 46
 DECELERATION 51
 ENABLE 70
 HIGH VELOCITY 52
 LOW VELOCITY 53
 SPEED SELECT INPUT 46
- KILL 54
 KILL INPUT 45
 KILL SEQUENCE 121
- LED DISPLAY 56
 LEDS 66
 LIMITS, END-OF-TRAVEL 36, 38, 55, 89, 93,
 101
 DEBOUNCE CHANGE 100
 DECELERATION 55
 DISABLE 55
 SET ACTIVE STATE 68
 SWITCH STATUS 80
 LIMITS, HOME 36, 38, 55, 69-70, 89, 91, 101
 STATUS 80
 LINE FEED 56
 LOOP 54, 64, 81, 128, 130
 LOOP COUNT 57
 LOW-RESOLUTION DRIVES 113
- MATHEMATICAL OPERATIONS 108
 MAXIMUM CORRECTION VELOCITY 64
 MAXIMUM VELOCITY 63
 MEMORY AVAILABLE 118
 MEMORY LOCK INPUT 46
 MESSAGE MODE 96
 MNEMONIC CODE 2
 MODE CONTINUOUS 60
 MODE NORMAL 60
 MODE POSITION ABSOLUTE 61
 MODE POSITION INCREMENTAL 61
 MODE POSITION PROFILE 62, 65
 MOTOR POSITION OPERAND SUBSTITUTE
 (POS) IN VARIABLE 109
 MOTOR PULSES 30
 MOTOR RESOLUTION 63, 101
 MOTOR STEP MODE 31
 MOVE DEFINITION 37
 MOVE TIME REPORT 101
- NESTING 41
 NIF 65
 NONVOLATILE MEMORY CHECKSUM 119
 NORMAL MODE 60
- OFF 67
 OUTPUTS 66, 72
 ACTIVE LEVEL 73
 DISABLE 15
 FUNCTIONS 72
 LEVEL 73
 ON POSITION 74
 PATTERN 66
 PROGRAMMABLE 49, 66, 73

- SQUARE WAVE 76
- STEP FREQUENCY 33
- PARALLEL I/O READS
 - DISTANCE 18
 - FOLLOWING VALUE 28
 - LOOP COUNT 57
 - SEQUENCE VALUE 124
 - TIME DELAY VALUE 103
 - VARIABLE VALUE 109
 - VELOCITY VALUE 111
- PAUSE 10, 76, 81
 - DURING SEQUENCE EXECUTION 126
- PAUSE AND WAIT FOR CONTINUE 107
- PAUSE/CONTINUE INPUT 45
- PK130 63
- PK2 63
- PK3 63
- PLC 18, 28, 47, 57, 91, 99, 103, 109, 111, 124
- POLARITY 97
- POSITION
 - ENCODER 77
 - ABS 109
 - FOLLOWING (FEP) 109
 - ERROR 8, 32
 - CONFIGURATION 11
 - DISPLAY 17
 - MAINTENANCE 8, 9, 21, 31, 32
 - MOTOR — VARIABLE OPERAND
 - SUBSTITUTE (POS) 109
 - POWER-UP 74
 - PROFILE MODE 27, 28
 - PROFILING MODE 62, 65
 - REPORT
 - ABSOLUTE 75
 - HEXADECIMAL 115
 - SIGNED BINARY 114
 - SIGNED HEXADECIMAL 116
 - SETPOINT 17
 - ZERO 61
 - ZERO INPUT 47
- POWER-UP
 - ACKNOWLEDGE PROG. INPUTS 70
 - POSITION 74
 - SEQUENCE (#100) 70, 123
- PRE-DEFINED SEQUENCES 37
- PRIMARY AXIS 33
- PRIMARY ENCODER 33, 35
- PROGRAMMABLE INPUTS
 - ASSIGN FUNCTIONS 45
 - ACKNOWLEDGE ON POWER UP 70
 - DISABLE 15
- PROGRAMMABLE OUTPUT BITS 49, 66
- PROGRAMMABLE OUTPUTS 73
- PROGRAMMING COMMANDS 2
- PROMPT (>) 94
- PROMPT (?) 96
- PROPORTIONAL GAIN 8, 11
 - MAXIMUM 9, 12
- PULSE FOLLOWING 35
- PULSE WIDTH 63
- PUSHBUTTON CONFIGURATION 11
- QUOTE 78
- RATIO CORRECTION 26
- RATIO SELECT 96
- REFERENCE EDGE OF HOME SWITCH 70
- REGISTRATION DEBOUNCE 100
- REGISTRATION INPUT 46, 81
- REGISTRATION MOVES 81
- REMOTE STOP INPUT 95
- REPEAT 82, 128
- RESET 130
- RESET INPUT 46
- RESOLUTION 63
- RESUME EXECUTION 95
- RESUME FUNCTION 10
- RETURN TO FACTORY SETTINGS 83
- REVISION LEVEL REPORT 87
- RS-232C COMMUNICATION
 - ECHO CONTROL 92
 - ENABLE 22
 - TRANSMIT VARIABLE AND STRING 106
 - SEQUENCE UPLOAD 129
- RUN A SEQUENCE 123
- SCAN TIME 18, 28, 57
- SCAN TIME DELAY 91
- SELF-CORRECTION 34
- SEQUENCE
 - BUFFER 127
 - CHECKSUM REPORT 119
 - DEFINITION 119
 - DIRECTORY 120
 - ERASE 120
 - EXECUTION STATUS RPT. 85, 126
 - INTERRUPTED RUN MODE 122
 - MEMORY AVAILABLE REPORT 118
 - RUN 123
 - RUN WITH PAUSE 126
 - SCAN MODE 71
 - SELECT 96
 - SELECT INPUTS 45, 95, 122
 - STEP MODE 127
 - TERMINATION 128
 - UPLOAD 129
- SERVO PARAMETERS 13
- SET VARIABLE INTERACTIVELY 86
- SET-UP COMMANDS 2
- SHUTDOWN 67, 68, 81, 97
- SIGNED BINARY POSITION REPORT 114
- SIGNED HEXADECIMAL POSIT. REQUEST 116
- SINGLE STEP MODE 5
- SMOOTHNESS 101
- SOFTWARE END-OF-TRAVEL LIMITS 89
- SOFTWARE LIMIT DISABLE 90
- SQUARE WAVE OUTPUT 76, 77
- START/STOP VELOCITY 113
- STATUS COMMANDS 2, 3
- STEP SEQUENCE 5
- STOP 87, 98
- STOP INPUT 45
- STOP LOOP 54, 130
- STOP ON STALL 32
- STRING 106
- STROBE OUTPUTS 19, 29, 58, 98, 103, 110, 124
 - DELAY TIME 98, 103
- SYNCHRONIZATION 26, 76, 102
 - ACCELERATION 24, 25, 33
 - INPUT 46
- SYNTAX 3
- SYSTEM RESPONSE 3
- TERMINATE LOOP INPUT 46
- TEST 101
- THUMBWHEELS 18, 99, 103
 - INSTALLATION 106
 - OPERATION MODE 106
 - TM8 THUMBWHEEL MODULE 106
- TIME DELAY 100
- TM8 THUMBWHEEL MODULE 106
- TRACE MODE 128
- TRACKING MODE 35
- TRANSMIT VARIABLE AND STRING 106
- TRIGGER INPUTS 45, 81, 102
 - STATUS 105
- TROUBLESHOOTING 86
- UNTIL 107
- UPLOAD SEQUENCE 129
- USER FAULT INPUT 47
- USER FLAG 88
- VARIABLES 86, 106, 108
 - OPERAND SUBSTITUTES 109
- VELOCITY 36, 50, 52, 53, 108
 - CORRECTION 64
 - DISPLAY 21
 - MAXIMUM 63, 108
 - MINIMUM 113
 - PROFILING MODE 60, 78, 79, 84
 - SETPOINT 21
 - START/STOP 113
- WAIT FOR CONTINUE 10
- WAIT FOR TRIGGER 102
- WHEN 117, 129
- WHEN SEQUENCE 129
- WHILE 66, 118, 128
- XWHEN 117, 129
- Z CHANNEL
 - INPUT ENABLE 69
 - LOOKING FOR DURING HOMING 38, 71
 - ZERO POSITION 61, 91

