

High-Level Programming Tools

The Model 500's X-language includes some commands that are common in most high-level programming languages (Pascal, Fortran or BASIC). In addition to these commands, 30 variables VAR1-VAR30 are provided for performing mathematical functions and boolean comparisons. You can access some system variables — POS (commanded position or setpoint), ABS (actual encoder position), and FEP (following encoder position). This section will introduce and explain how to use these structures with the 500.

Branching commands evaluate conditions statements to make branching decisions. If the condition is true, one set of commands is processed. If the condition is false, another set of commands will be executed. The structures that evaluate conditions are listed below.

IF (condition true)—execute these commands
 ELSE—execute these commands
 NIF

WHILE (condition true)—execute these commands
 NWHILE

REPEAT—Execute these commands
 UNTIL (condition true)

The condition statements that are evaluated can be very complex. The condition statements support all of the following decisions:

Variables

The 500 has up to 30 variables that can be used to perform multiplication, division, addition, and subtraction. You can assign these variables to various motion parameters. These parameters and the syntax of assigning a variable to them are listed below.

Command	Description
D (VAR1)	Loads the distance with the value of variable 1
V (VAR4)	Loads the velocity with the value of variable 4
A (VAR5)	Loads the acceleration with the value of variable 5
AD (VAR7)	Loads the deceleration with the value of variable 7
FP (VAR9)	Loads the following port with the value of variable 9
DP (VAR1)	Loads the distance point with the value of variable 1
L (VAR30)	Loads the loop count with the value of variable 30
XR (VAR21)	Executes the sequence number held in variable 21
T (VAR3)	Loads the time delay with the value of variable 3
FOL (VAR29)	Loads the following ratio with the value of variable 29

Variable assignments can be made in sequences or in the *immediate mode* (see the **Model 500 Software Reference Guide**). If a variable that has a fractional portion is assigned to a parameter that requires a whole number (such as distance), the variable is rounded to the nearest whole number and assigned to the parameter. To illustrate the use of the variables and the math functions they can perform, follow the steps below.

In addition, you can assign to the variables the value of the position counter (POS), encoder counter (ABS), or the following encoder counter. For instance, VAR1=POS is equivalent to leading VAR1 with the value reported by the 1PR status report. ABS is equivalent to the value of the 1PX status report.

Assigning Variables to Constants

You can set parameters as variables in a sequence (D(VAR)). You can define these variables by assigning a constant value. When the sequence is run, this value is assigned to the corresponding parameter in the sequence.

STEP ① The sequence below executes a move and travels the distance provided in variable #1 and the velocity provided in variable #2.

<u>Command</u>	<u>Description</u>
> LD3	Disables limits (<i>Not needed if limits are installed</i>)
> MPI	Places the 500 in incremental mode
> MN	Enters the normal mode
> FSIØ	Places a 500F in the indexer mode. Not needed if you do not have the Following option.
> XE1	Erases sequence #1
> XD1	Defines sequence #1
> A1Ø	Sets the Acceleration
> AD1Ø	Sets the Deceleration
> V(VAR2)	Assigns variable #2 to the velocity term
> D(VAR1)	Assigns variable #1 to the distance term
> G	Initiates motion
> XT	Ends the sequence definition

STEP ② You will now assign numbers to the variables VAR1 and VAR2

<u>Command</u>	<u>Description</u>
> VAR1=25ØØØ	The distance parameter is assigned 25000
> VAR2=5	The velocity parameter is assigned 5

STEP ③ Execute sequence #1 with XR1. The motor will move 25000 steps at 5 rps. Verify that the distance (D) and the velocity(v) have been assigned these values.

<u>Command</u>	<u>Response</u>
> 1V	*VØ5.ØØØØØ
> 1D	*D+ØØØØ25ØØØ

STEP ④ Now change the values of the variables as follows.

<u>Command</u>	<u>Description</u>
> VAR1=75ØØØ	The distance parameter is 75,000
> VAR2=1Ø	The velocity parameter is assigned 10

STEP ⑤ Repeat steps ③ and ④.

Entering Variables Via RS-232C

When using variables in sequences, you can use the RSIN command to interactively prompt the user to enter a variable number.

STEP ① Define the following sequence:

<u>Command</u>	<u>Description</u>
> 1XE1	
> 1XD1	Defines sequence #1
1"ENTER_THE_NUMBER_OF_PARTS	Prompts you for the # of parts to make
1CR	Inserts a carriage return
1LF	Inserts a line feed
VAR5=RSIN	The sequence stops here waiting for you to enter a #
1CR	Inserts a carriage return
1LF	Inserts a line feed
L(VAR5)	The loop count is loaded with the number of parts to make
D25ØØØ	Distance is 25000 steps
G	Initiates motion
N	Ends the loop
1"FINISHED_WITH_PARTS_RUN	Indicates that the run is finished
1CR	
1LF	
> XT	

The **RSIN** command prompts you to enter a variable via RS-232C that will be used in a sequence. When a variable is assigned to **RSIN**, the execution of the sequence is stopped until you enter the variable. To enter the variable, you must precede the number with an exclamation point (!).

STEP ② Turn on the Trace mode and run the sequence.

```
> 1XTR1
> XR1
```

STEP ③ The program will stop at the **RSIN** command and wait for you to be enter a variable number. Enter the variable number.

```
> !10
```

The sequence executes a 25,000-step move 10 times making 10 parts.

Math Operations In addition to assigning constants to variables, two system parameters can be assigned to a variable.

- **POS**—Commanded Position
- **FEP**—Following Encoder Position

These parameters are assigned as if they were constants. For example:

<u>Command</u>	<u>Response</u>
> VAR1=POS	Assigns the current value of the command position to variable #1

Performing Math Operations with Variables

The 500 has the ability to perform simple math functions with its variables (add, subtract, multiply, and divide). The following sequence of steps illustrates the math capabilities of the 500.

STEP ① **Addition:**

<u>Command</u>	<u>Response</u>
> VAR1=5	
> VAR23=1000.565	
> VAR11=VAR1+VAR23	
> VAR23=VAR10+VAR23	
> VAR1=VAR1+5	
> 1VAR1	*+0000000000000010.00000
> 1VAR11	*+0000000000001005.56500
> 1VAR23	*+001010.56500

STEP ② **Subtraction:**

<u>Command</u>	<u>Response</u>
> VAR3=10	
> VAR20=15.5	
> VAR3=VAR3-VAR20	
> VAR19=VAR20-VAR3	
> 1VAR3	*-000000000000005.50000
> 1VAR20	*+000000000000015.50000
> 1VAR19	*00000000000021.00000

STEP ③

Multiplication:

<u>Command</u>	<u>Response</u>
> VAR3=10	
> VAR20=15.5	
> VAR3=VAR3*VAR20	
> VAR19=99	
> VAR19=VAR20*VAR19	
> 1VAR3	*+00000000000000155.00000
> 1VAR20	*+0000000000000015.50000
> 1VAR19	*+000000000000001534.50000

STEP ④

Division:

<u>Command</u>	<u>Response</u>
> VAR3=10	
> VAR20=15.5	
> VAR3=VAR3/VAR20	
> VAR30=75	
> VAR19=VAR30/VAR3	
> 1VAR3	*+000000000000000.64516
> 1VAR20	*+0000000000000015.50000
> 1VAR30	*+0000000000000075.00000
> 1VAR19	*+00000000000000116.25023

Complex Branching and Looping

The 500 supports the high-level language structures for branching and looping. Each conditional branch or loop evaluates a condition statement. Depending on whether this condition statement evaluates true or not determines where the 500 will branch to. The unconditional branching and looping statements have been introduced already. These are the GOTO (Branch), GOSUB (Branch & Return) and L (Loop) command. The L command is explained further here.

Unconditional Looping

The loop command is an unconditional looping command You may use the Loop (L) command to repeat a series of commands. You can nest Loop commands up to 16 levels deep.

<u>Command</u>	<u>Description</u>
> PS	Pauses command execution until the indexer receives a Continue (c) command
> MPI	Sets unit to Incremental mode
> A50	Sets acceleration to 50 rps ²
> V5	Sets velocity to 5 rps
> L5	Loops 5 times
> D2000	Sets distance to 2,000 steps
> G	Executes the move (Go)
> T2	Delays 2 seconds after the move
> N	Ends loop
> C	Initiates command execution to resume

The motor moves a total of 10,000 steps

The example below shows how you can nest a loop inside a loop. In this example, the motor makes two moves and returns a line feed. The unit repeats these procedures until you instruct the it to stop.

Description	Command
Pauses command execution	> PS
Loops indefinitely	> L
Sends a line feed	> 1LF
Loops twice	> L2
Executes 2,000-step move	> G
Waits 0.5 seconds	> T.5
Ends loop	> N
Ends loop	> N
Continues command execution	> C

This command execution continues until you issue the Stop (s) or Kill (κ) commands.

Unconditional Branching The unconditional branching commands GOTO and GOSUB were explained earlier in the sequence section.

Conditional Statements You can use the following types of conditional statements with the 500.

- Error Flags
- User Flags
- Input State
- Boolean Comparisons

ERROR FLAGS

The error flag (ERXXXXXX) is useful if you want to trap different error conditions and create different sequences to respond to them. The ***Model 500 Software Reference Guide*** explains the errors that can be trapped in the evaluation command description. An example of the statement's use is provided below.

```
> IF(ER110XXXX)
> GOSUB2
> NIF
```

USER FLAGS

You can set the user flag (FL000111X0) and modify it within sequences to mark where the program has gone or to indicate any special state so that a conditional statement can be made. An example of the statement's use is provided below.

```
> WHILE(FL0011XXXX)
> D100
> G
> NWHILE
```

INPUT STATE

An example of this statement's use (IN111100010) is provided below.

```
> REPEAT
> GOSUB4
> UNTIL(IN001XX11)
```

Variable comparisons

- VARn>VARm
- VARn<VARm
- VARn=VARm

```
IF(VAR1>VAR2)
WHILE(VAR3=10)
GOSUB2
NWHILE
NIF
```

**BOOLEAN
COMPARISONS**

An example of the statement's use is provided below.

```
WHILE (VAR3>10_AND_IN110011)
GOSUB8
NWHILE
```

Condition statements are explained in the ***Model 500 Software Reference Guide***.

**Conditional
Looping**

The 500 supports two conditional looping structures—REPEAT/UNTIL and WHILE.

**REPEAT/
UNTIL**

With the REPEAT command, all commands are repeated between REPEAT and UNTIL. The 500 stops executing the commands when the UNTIL condition is true.

STEP ①

Command	Description
> VAR5=0	Initializes variable 5 to 0
> 1XE10	Erases sequence #10
> 1XD10	Defines sequence #10
REPEAT	Begins the REPEAT loop
A50	Acceleration is 50 rps ²
AD50	Deceleration is 50 rps ²
V5	Sets velocity to 5 rps
D25000	Distance is 25,000 steps
G	Executes the move (Go)
VAR5=VAR5+1	Variable 5 counts up from 0
UNTIL (INXXX1110_OR_VAR5>10)	When the 1110 input condition occurs or VAR5 is greater than 10, the loop will stop.
1"DONE_LOOPING	Quote command indicates that the loop is finished
1CR	Inserts a carriage return
1LF	Inserts a line feed
> XT	

STEP ②

Use the Trace mode to display the commands as they are run.

```
> 1XTR1
> XR10
```

The loop can be exited either by using the DIN command to satisfy the input state or letting the VAR5 counter count to 10.

WHILE

With the WHILE command, all commands are repeated between WHILE and NWHILE until the WHILE condition is true.

STEP ①

Command	Description
> VAR5=0	Initializes variable 5 to 0
> 1XE10	Erases sequence #10
> 1XD10	Defines sequence #10
WHILE (INXXX1110_OR_VAR5<10)	While the input pattern is equal to XXX1110 or variable 5 is less than 10, repeat the loop
A50	Acceleration is 50 rps ²
AD50	Deceleration is 50 rps ²
V5	Velocity is 5 rps
D25000	Distance is 25000 steps
G	Executes the move (Go)
VAR5=VAR5+1	Variable 5 counts up from 0
NWHILE	
1"DONE_LOOPING	Indicates that the loop is done
1CR	Inserts a carriage return
1LF	Inserts a line feed
> XT	

STEP ② Use the Trace mode to display commands as they are run.

```
> 1XTR1
> XR1Ø
```

You can exit the loop with the **DIN** command (cause the input state to not match the **IN** command). You can also exit the loop by setting **VAR5** counter count to 10. If the input pattern is not **XXX111Ø**, the loop will not be executed.

Conditional Branching

You can use the **IF** statement for conditional branching. All commands between **IF** and **ELSE** are executed if the condition is true. If the condition is false, the commands between **ELSE** and **NIF** are executed. If the **ELSE** is not needed, it may be omitted. The commands between **IF** and **NIF** are executed if the condition is true. Examples of these statements are provided.

- Error Flag
- User Flag
- Input State

Error Flags

The **IF** command checks for error conditions. If an error exists, a conditional command will be executed. This command is useful if you wish to trap different error conditions (Drive Disabled, User Fault Input Activated, Excessive Position Error, etc). Refer to the **Model 500 Software Reference Guide**.

<u>Command</u>	<u>Description</u>
> XE1Ø	Erases sequence #10
> XD1Ø	Defines Sequence #10—when a fault occurs, sequence #10 will execute—the sequence is defined with the Set Fault or Kill Sequence (XFK1Ø) command
IF(ER1)	If hardware CCW limit switch is reached, perform the following commands:
1"CCW_LIMIT_HIT	Display the error message
NIF	Ends IF statement
IF(ERX1)	If hardware CW limit switch is reached, perform the following commands:
1"CW_LIMIT_HIT	Display the error message
NIF	Ends IF statement
> XT	Ends Sequence loop
> XFK1Ø	Sets Sequence #10 as the Fault sequence

User Flags

This command uses the pattern set by the User Flag (**SFL**) command to run conditional commands. This command is useful if you want to make a decision based on previous sequence executions that will set or clear the user flag bits. For example, if an application has several sequences, you can assign different bit patterns with the **SFL** command at the end of each sequence. If you select these sequences from the host computer, you may wish to make different moves depending on the sequence you ran. Refer to the ***Model 500 Software Reference Guide*** for a detailed description of the **SFL** command.

Command	Description
> PS	Waits for the indexer to receive a Continue (C) command before executing the next command
> SFL1010	Sets user flag bits 7 and 5 and clears bits 6 and 4, the remaining bits are not altered
> IF(FL1010)	If user flag bits 5 and 7 are set, and bits 6 and 4 are clear, perform the following commands
> 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)
> NIF	Ends IF statement
> C	Continues execution

If the **FL** pattern matches the **SFL** setting, the motor moves 25,000 steps. You can change the **SFL** pattern at different points in sequences to map a path for sequence execution.

Input State

This command compares the input pattern (**CW**, **CCW**, and **HM** and **I1 - I7**) to execute the conditional commands. This command is useful for branching and performing conditional moves using the programmable inputs. For a detailed description of this command, refer to the ***Model 500 Software Reference Guide***.

Command	Description
> 1XE5	Erases sequence #5
> 1XD5	Defines sequence #5
IF(INXXX10)	If I1 is active and I2 is not active, issue the following commands:
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)
NIF	Ends IF statement
IF(INXXX01)	If I1 is not active (open) and I2 is active (closed), issue the following commands:
A10	Sets acceleration to 10 rps ²
V5	Sets velocity to 5 rps
D-5000	Sets distance to 5,000 steps in the opposite direction
G	Executes the move (Go)
NIF	Ends IF statement
IF(INXXX1)	If I1 is active, do the following command.
L"DONE	Ends message saying done
NIF	Ends IF statement
> 1XT	Ends sequence definition

Use the **DIN** command or the inputs themselves to execute the different trigger input states. You can use the Trace mode to see what commands are executed.

Branching Using Variables and Boolean Logic

You can use the **IF** statement to branch based on variable values. Multiple comparisons can be made in one condition statement using the boolean **OR** and **AND** functions.

Command	Description
> XE8	Erases sequence 8
> XD8	Defines sequence 8
VAR5=15	Variable 5 = 10
IF (VAR5>10_AND_VAR4=20)	If Variable 5 is less than 10 and variable 4 = 20 then perform the commands until the ELSE
A100	Sets acceleration to 100 rps ²
AD100	Sets deceleration to 100 rps ²
V5	Sets velocity to 5 rps
D25000	Sets distance to 25,000 steps
G	Executes the move (Go)
VAR5=VAR5-1	Variable 5 decrements one
ELSE	Ends IF statement
A100	Sets acceleration to 100 rps ²
AD100	Sets deceleration to 100 rps ²
V5	Sets velocity to 5 rps
D-5000	Sets distance to 5,000 steps in the opposite direction
G	Executes the move (Go)
NIF	Ends the IF
> XT	Ends definition of sequence 8

Motion Profiling Mode—On-the-Fly Changes

The Motion Profiling mode allows you to execute buffered commands while a move is being made (on-the-fly). When you enter this mode, the 500 will enter all the commands that you enter immediately. You can enter and exit this mode from within a sequence. This mode allows you to change velocity on-the-fly based on distance, turn on outputs based on distance, perform math and other commands while in motion. The following commands are used with Motion Profiling mode.

- MPP—Enter Motion Profiling Mode
- NG—Exits Motion Profiling Mode
- DP—Sets Distance Points within Motion Profiling Mode

While the 500 is in Motion Profiling mode, you can execute any command while a move is being made. When the 500 encounters an **MPP** command in a sequence, all subsequent commands will be executed until the **NG** command is encountered. An example of the **MPP** command is provided below.

```
XD1 D50000 V1 MPP G O1 TR1X1 V4 NG XT
```

In this example, a 50,000-step move is made. The initial velocity is 1 rps. Motion begins with the **G** command. Output 1 is turned on with the **O1** command. The 500 then runs the trigger command (**TR**) until the condition is true, at which point it changes the velocity. When the 500 encounters the **NG** command, it will not receive any additional commands until the 50,000-step move is completed.

Changes Based on Distance

Changes are made based on distance using the distance point (**DP**) command. This command causes a delay in the processing of the commands until the motor reaches the specified distance point. Processing will continue once the distance point is reached. In this way, velocity changes and the activation of outputs can be done based on distance.

The distance point is interpreted differently for Absolute mode versus Incremental mode. To change velocity on-the-fly based on distance, the Motion Profiling Mode (**MPP**) command must be used with the Distance Point (**DP**) command. The following sequence example executes the profile shown in Figure 4-11.

```
1XD1 PZ D100000 V2 MPP G DP25000 O1 DP50000 O0 NG XT
```

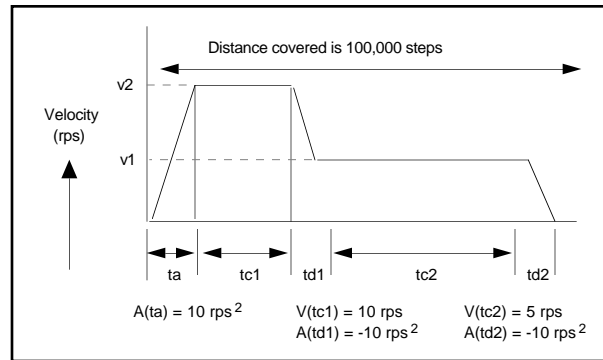


Figure 4-11. Motion Profiling Mode Example

In Incremental mode, commands are processed until the DP command is reached. The 500 pauses at the DP command until the motor moves 25,000 steps. The 500 then turns on output #1. The 25,000 steps are counted from the point at which the command is encountered. When DP50000 is reached, the 500 pauses until the motor moves 50,000 steps. The 500 then turns output #1 off. In this example, if the 500 is in Incremental mode, the output will be turned on at 25,000 steps and turned off after the motor has moved a total of 75,000 steps from the beginning of the first move.

In absolute mode, the value specified with DP is interpreted as an absolute position relative to the zero point location. In this example, the output would be turned on at 25,000 steps and turned off after the motor had passed the 50,000 step point.

Stopping Motion with a Stop Input

A common use of the Motion Profiling mode is to perform a continuous move and stop the move from the inputs. This can be done in two ways. You can define an input as a stop input. Motion can be stopped by activating the stop input.

The other method is to use the STOP command and place it within a sequence. The following step-by-step example illustrates these two methods.

STEP ①

Define the input as a stop input.

Command	Description
> IN1D	Defines input #1 as a stop input

STEP ②

Create the sequence with a continuous move.

Command	Description
> XD1	Begins definition of sequence #1
V5	Sets velocity to 5 rps
A100	Sets acceleration to 100 rps ²
MC	Sets the 500 to Continuous mode
MPP	Sets the 500 to Motion Mode Profiling
G	Executes the move (Go)
NG	Exits Motion Profiling mode
> XT	Ends sequence #1 definition

STEP ③ Execute sequence #1 with an `XR1` command. The motor will move at 5 rps and will not stop until you activate the stop input (input #1). Activate the stop input.

The motor will stop at a controlled deceleration. In this case, the buffer will be dumped and the sequences will not be finished. The `STOP` command caused program control to exit the sequence. The `G` command was the last command that was run.

STEP ④ Issue the Display Parameters (`DR`) command. The 500 is still in Motion Profiling mode. Enter the `NG` command to exit the mode.

Stopping Motion with the STOP Command

The following step-by-step example illustrates the method of stopping a continuous move with the `STOP` command.

STEP ① Configure input #1 as a trigger input.

<u>Command</u>	<u>Description</u>
> <code>IN1A</code>	Configures input #1 as a trigger input

STEP ② Create a sequence with a `STOP` command after the trigger.

<u>Command</u>	<u>Description</u>
> <code>XD1</code>	Begins the definition of sequence #1
<code>V5</code>	Sets velocity to 5 rps
<code>A100</code>	Sets acceleration to 100 rps ²
<code>MC</code>	Sets the 500 to Continuous mode
<code>MPP</code>	Sets the 500 to Motion Mode Profiling
<code>G</code>	Executes the move (Go)
<code>TR1</code>	Activates trigger #1
<code>STOP</code>	Stops motion when the trigger condition is met
<code>NG</code>	Exits Motion Profiling mode
> <code>XT</code>	Ends sequence #1 definition

STEP ③ Issue the Display Parameters (`DR`) command. The 500 is still in Motion Profiling mode.

In both of the examples, an `NG` command was required after motion was stopped. When a `STOP` command is issued, the command buffer is emptied. Therefore, the commands that have not been executed in the sequence at the time the stop occurs will not be executed. In both examples, the `NG` command is not executed because motion was stopped. In fact, `NG` can never be executed under these conditions. *The important point is that if the `STOP` command is used to stop continuous motion, the `NG` must be issued either at the beginning of the next sequence or directly via the RS-232C interface.*

To prevent the 500 from stopping without finishing the sequence that it is currently executing, a software switch has been provided that will cause the 500 to continue executing the sequence it was running when the `STOP` was issued. By entering the Clear/Save the Command Buffer on Stop (`SSH1`) command, the 500 will stop motion when it encounters a `STOP` and continue processing the commands in the sequence. *Enter `SSH1` and repeat the previous example. Notice how the Motion Profiling mode is exited.*

Sequence Scan Mode and the Stop Command

In applications that use the Sequence Scan mode and the **STOP** command, you must use the (**SSH**) command to prevent the Sequence Scan mode from being disabled. Under normal conditions, the Sequence Scan mode is aborted when the 500 encounters a **STOP** command. If the **STOP** command is issued from a stop input or from within a sequence, the Sequence Scan mode will be aborted. Usually, you will not want to abort the mode. The **SSH1** command will allow the 500 to complete the execution of the current sequence from the point at which the **STOP** was issued. You must re-enable the Sequence Scan mode by placing the **SSJ1** command in the sequence after the point at which the stop will occur. The following example illustrates such a sequence.

STEP ① Configure input #1 as a trigger input and sequence #2 as a sequence select input.

Command	Description
> IN1A	Configures input #1 as a trigger input
> IN2B	Configures input #2 as a sequence select input

STEP ② Enable the Sequence Scan mode with the **SSJ1** command.

STEP ③ Create a sequence with a **STOP** command after the trigger.

Command	Description
> XD1	Begins the definition of sequence #1
V5	Sets velocity to 5 rps
A100	Sets acceleration to 100 rps ²
MC	Sets the 500 to Continuous mode
MPP	Sets the 500 to Motion Mode Profiling
G	Executes the move (Go)
TR1	Activates trigger #1
STOP	Stops motion when the trigger condition is met
NG	Exits Motion Profiling mode
SSJ1	Re-enable Sequence Scan mode
> XT	Ends sequence #1 definition

STEP ④ Execute the sequence by activating input #21. Stop the move at any time by activating input #1.

STEP ⑤ The 500 is still in Sequence Scan mode and the move can be repeated by activating input #2 again (and stopped with input #1).

Other Uses of the Motion Profiling Mode

The Motion Profiling mode allows a great amount of flexibility in the complexity of the control of the 500 during motion. You can change gains on-the-fly based on distance or following error. You can turn on outputs, change velocity, and perform math functions. The primary application concern to consider during sequence execution is the amount of time required to perform the commands. In some cases, the execution of commands may depend on the motion. The following examples show additional uses of the Motion Profiling mode.

CHANGING GAINS ON-THE-FLY	<u>Command</u>	<u>Description</u>
	> 1XD1	Begins the definition of sequence #1
	1D400000	Sets distance to 400,000 steps
	1V5	Sets velocity to 5 rps
	1MPP	Sets 500 to Motion Profiling mode
	1G	Executes the move (Go)
	1DP150000	Sets distance point to 150,000
	1V2	Changes velocity to 2 rps
	1BCPP20	Changes gain to 20% of maximum
	1DP250000	Sets distance point to 250,000
	V5	Changes velocity to 2 rps
	BCPP45	Changes gain to 45% of maximum
	NG	Exits 500 from Motion Profiling mode
> XT	Ends the definition of sequence #1	
TURNING ON INPUTS, USING TIME DELAYS, AND MATH	<u>Command</u>	<u>Description</u>
	> 1XD1	Begins the definition of sequence #1
	1PZ	Sets axis #1 to Ø
	1MC	Sets the 500 to Continuous mode
	1MPP	Sets 500 to Motion Profiling mode
	1G	Executes the move (Go)
	1T.5	Sets a 0.5 second delay
	1O110	Turns outputs #1 and #2 on, #3 off
	1T.5	Sets a 0.5 second delay
	1O001	Turns outputs #1 and #2 off, #3 on
	REPEAT	Starts repeat loop
	VAR1=VAR1+1	Increases variable #1 by 1
	T.1	Sets a 0.1 second delay
	UNTIL (POS>400000)	Continues looping until the 500's position is > 400,000
	STOP	Halts command processing
	NG	Exits 500 from Motion Profiling mode
	> XT	Ends the definition of sequence #1

Triggers, input states (INXX111) time delays, and the distance points are the commands that will allow you to control when and where procedures occur during motion in your program. The Motion Profiling mode offers you the flexibility to accomplish a variety of different application needs.

Interfacing to the 500

This section discusses the various interfaces that you may use with the 500.

- Operation from Programmable Inputs and Outputs
 - Switches
 - Thumbwheels
- PLC Operation
- Front Panel Operation
- Remote Panel (RP240) Operation
- Host Computer Operation

Programmable Inputs and Outputs

The 500 has a very flexible input and output scheme for defining I/O in a way that is suitable for almost any application. There are 13 programmable inputs (IN1 - IN13 on the front panel). The other three inputs are dedicated for limits. There are also 4 programmable outputs. Using the inputs in combination with the outputs you can use up to 16 digits of thumbwheels with the 500. This section explains some of the functions that the inputs and outputs can perform and explains how to use thumbwheels for an interface with the 500.

Output Functions

You can turn the programmable outputs (**OUT1 - OUT8**) on and off with the Output (O) and Immediate Output (IO) commands. Outputs **OUT1 - OUT8** are factory set as programmable outputs. However, you can configure all of the outputs to perform different functions (Moving/Not Moving, Amp Off, Strobe, etc.) with the Configure Output (OUT) command. Refer to the OUT command in the **Model 500 Software Reference Guide** for descriptions of the available functions. You can use these outputs to turn on and off other devices (i.e., lights, switches, etc.). The output functions have unique letter assignments.

A:	Programmable Output
B:	Moving/Not Moving
C:	Sequence in Progress
D:	At Soft or Hard Limits
E:	At Position Zero
F:	Fault Indicator
G:	Not Used
H:	Amp Off
J:	Strobe Out
K:	Invalid Command Error
L:	Position Error Fault
M:	Not Used
N:	CW Software Limit Reached
P:	CCW Software Limit Reached
R:	CW Hardware Limit Reached
S:	CCW Hardware Limit Reached
T:	Output Based on Position
U:	Programmable Pulse output

<u>Command</u>	<u>Description</u>
> PS	Pauses command execution until the indexer receives a Continue (C) command
> MN	Sets unit to Normal mode
> A1Ø	Sets acceleration to 10 rps ²
> V5	Sets velocity to 5 rps
> D25ØØØ	Sets distance to 25,000 steps
> OUT1A	Sets OUT1 as a programmable output
> OUT2A	Sets OUT2 as a programmable output
> OUT3B	Sets OUT3 as a Moving/Not Moving output
> o1Ø	Turns OUT1 on and OUT2 off
> G	Executes the move
> Ø1	Turn OUT1 off and OUT1 on
> C	Initiates command execution to resume

This example defines **OUT1** and **OUT2** as programmable outputs and **OUT3** as a Moving/Not Moving output. Before the motor moves 25,000 steps, **OUT1** is turned on and **OUT2** is turned off. These outputs will remain in this state until the move is completed, then **OUT1** will turn off and **OUT2** will be turned on. While the motor is moving, **OUT3** remains on.

Pulse Output-Half Axis

A second half-axis can be programmed using the 500. This output can control the distance and a constant speed for another drive/motor system. A second output can be used to control the distance. The following commands will set up a second axis and make it move.

<u>Command</u>	<u>Description</u>
> OUT1U	Configures the output to be a pulse output
> OUT2A	The output will be used to control direction
> PU1ØØØ,2	The output will send 1000 pulses at a rate of 2 ms
> PUL	This commands begins the pulse train out of output 1

Input Functions The inputs can individually be programmed to perform any of the following functions. Each function has an assigned letter:

A: Trigger
 B: Sequence Select
 C: Kill
 D: Stop
 E: Command Enable
 F: Pause/Continue
 G: Go
 H: Direction
 I: Synchronization
 J: Jog+ (CW)
 K: Jog- (CCW)
 L: Jog Speed Select
 M: Not Used
 N: Data
 O: No Function assigned
 P: Memory Lock
 Q: Registration input (IN10 - IN13 Only)
 R: Reset
 S: Go Home
 T: Position Zero
 U: User Fault
 V: Data Valid
 W: Data Sign
 X: Increase Following ratio
 Y: Decrease Following ratio
 Z: No Function assigned

To designate each input pin to a particular function, use the Set Input Functions (IN) command. To see what the inputs are currently defined as, type 1IN. To see the inputs' states, use the 1IS command. Enter the following commands.

```
> 1IN
```

Change input 1 to be a stop input by entering:

```
> IN1D
```

Check that it was assigned properly by again entering:

```
> 1IN1
```

With this method, you can assign all the inputs to any of the functions listed above. (Except the registration input function. Only inputs IN10 - IN13 can be assigned as registration inputs.)

Switches

This section contains information on 500 triggers and sequence scanning with inputs.

Triggers

You can use the Trigger Pause (TR) command to pause a sequence of buffered commands until one or more inputs come to a preferred state. Inputs IN10 - IN13 are set at the factory (default setting) to function as trigger inputs.

Command	Description
> IN1D	Sets IN1 as Stop(S) input
> IN2A	Sets IN2 as trigger input 1.
> IN3A	Sets IN3 as trigger input 2
> IN4A	Sets IN4 as trigger input 3
> MC	Sets the unit to Continuous mode
> MPP	Enters the Motion Profiling mode
> A15	Sets acceleration to 15 rps ²
> AD15	Sets acceleration to 15 rps ²
> TR1	Waits for trigger input 1 (IN2) to be on
> G	Executes a go (Go) command
> L	Loops infinitely
V5	Sets velocity to 5 rps
TRX01	Waits for trigger input 2 (IN3) to be off and trigger input 3 (IN4) to be on
V1	Sets velocity to 5 rps
TRX10	Waits for trigger input 2 (IN3) to be on and trigger input 3 (IN4) to be off
> N	Ends the loop

This example program configures IN1 as a stop input and IN2 as a trigger (Start) input. IN3 and IN4 are programmable trigger inputs. If you activate the Start input (IN2), the motor will begin moving. Because the 500 is in Motion Profiling mode, it will execute the loop and subsequent commands. As it reaches each trigger statement, it waits for that state to become true and changes the velocity to the velocity following the command. The loop is infinite so it will continuously toggle between 1 and 5 rps as the trigger statements come true and will not stop until the stop input is activated. If you activate IN1 during the operation of the 500, the indexer immediately stops the operation and clears the command buffer.

Sequence Select & Sequence Scanning

Inputs can be defined as sequence select inputs. This allows you to execute sequences defined via RS-232C by activating the sequence select inputs. Sequence select inputs are assigned BCD weights. The lowest input number assigned as a sequence select input will have the least significant value. Figure 4-12 shows the BCD weights of the 500's inputs when inputs IN1—IN8 are configured as sequence select inputs.

The inputs are weighted with the least weight on the smallest numbered input, as shown in Figure 4-12. If we had selected IN6, IN9, IN10 & IN13 instead of IN5, IN6, IN7 & IN8, then the weights would be as follows:

- IN6 = 10
- IN9 = 20
- IN10 = 40
- IN13 = 80

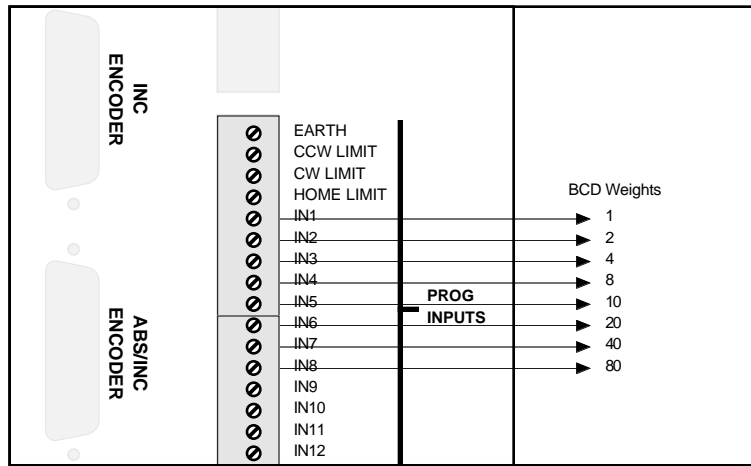


Figure 4-12. Dedicated BCD Weight of 500 Inputs

Table 4-1 shows one possible input configuration and BCD weighting.

Input	Function	Binary Weighting
IN1	Sequence Select	1
IN2	Sequence Select	2
IN3	Sequence Select	4
IN4	Sequence Select	8
IN6	Sequence Select	10
IN9	Sequence Select	20
IN10	Sequence Select	40
IN13	Sequence Select	80

Table 4-1. Input Configuration/BCD Weighting Example

If the inputs are configured as in Table 4-1, sequence #6 will be executed by activating IN2 and IN3. Sequence #29 will be executed by activating inputs IN1, IN4, and IN9.

To execute sequences using the sequence select lines, place the 500 in Sequence Scan mode with the `SSJ1` command. In this mode, the 500 will continuously scan the input lines and execute the sequence selected by the active sequence select lines. To disable the sequence scan mode, enter `SSJ0`.

Once enabled (`SSJ1`), the 500 will run the sequence number that the active sequence select inputs and their respective BCD weights represent. After executing and completing the selected sequence, the 500 will scan the inputs again and run the selected sequences. If a sequence is selected that has not been defined via RS-232C, no sequence will be executed.

If it is not desirable for the 500 to immediately execute another sequence after running the currently selected sequences the Sequence Interrupted Run Mode (`XQ1`) can be enabled. In this mode, after executing a sequence, all sequence select lines must be placed in an inactive state before a new sequence can be selected. The active state of the inputs is determined by the `INL` command.

The Scan (`SN`) command determines how long the sequence select input must be maintained before the indexer executes the program. This delay is referred to as **debounce time**. The following examples demonstrate how to use the Scan mode.

STEP ①

Define a power-up sequence.

Command	Definition
> <code>XE100</code>	Erases sequence #100
> <code>XD100</code>	Defines sequence #100
<code>SSJ1</code>	Executes sequences via PLC input
<code>SN5</code>	Sets scan time to 5 msec
<code>XQ1</code>	Sets 500 to interrupted run mode
<code>A10</code>	Sets acceleration to 10 rps ²
<code>AD10</code>	Sets deceleration to 10 rps ²
<code>V2</code>	Sets velocity to 2 rps
<code>IN1B</code>	Sets Input 1 as a sequence-select input
<code>IN2B</code>	Sets Input 2 as a sequence-select input
<code>IN3B</code>	Sets Input 3 as a sequence-select input
<code>IN4B</code>	Sets Input 4 as a sequence-select input
<code>IN5B</code>	Sets Input 5 as a sequence-select input
<code>IN6B</code>	Sets Input 6 as a sequence-select input
<code>IN7B</code>	Sets Input 7 as a sequence-select input
<code>IN8B</code>	Sets Input 8 as a sequence-select input
<code>OUT1C</code>	Sets Output 1 as sequence-in-progress output
<code>LD3</code>	Disables the limits (if they are not connected)
> <code>XT</code>	Ends the sequence definition

Every time you power up the 500, it executes these commands and enables the 500 to read up to 99 sequences from the sequence-select inputs.

STEP ② Define any sequences that your application may require.

<u>Command</u>	<u>Description</u>
> XE1	Erases Sequence #1
> XD1	Defines Sequence #1
D2000	Sets distance to 2,000 steps
G	Executes the move (Go)
> XT	Ends Sequence #1 definition

<u>Command</u>	<u>Description</u>
> XE2	Erases Sequence #2
> XD2	Defines Sequence #2
D4000	Sets distance to 4,000 steps
G	Executes the move (Go)
> XT	Ends Sequence #2 definition

<u>Command</u>	<u>Description</u>
> XE3	Erases Sequence #3
> XD3	Defines Sequence #3
D8000	Sets distance to 8,000 steps
G	Executes the move (Go)
> XT	Ends Sequence #3 definition

<u>Command</u>	<u>Description</u>
> XE99	Erases Sequence #99
> XD99	Defines Sequence #99
D-14000	Sets distance to -14,000 steps
G	Executes the move (Go)
> XT	Ends Sequence #99 definition

STEP ③ Verify that your programs were stored properly by uploading each entered sequence (**XU**). If you receive responses that differ from what you programmed, re-enter those sequences.

STEP ④ Run each program from the RS-232C interface with the Run Sequence (**XR**) command. Make sure that the motor moves the distance that you specify.

STEP ⑤ Apply switches to the inputs that are normally-open (see Table 4-2).

Switch #	Input
Switch #1	IN1
Switch #2	IN2
Switch #4	IN3
Switch #8	IN4
Switch #10	IN5
Switch #20	IN6
Switch #40	IN7
Switch #80	IN8
GND	

Table 4-2. Applying Switches to Inputs

STEP ⑥ To execute sequences cycle power to the 500. The system will execute sequence #100.

STEP ⑦ You can now execute sequences by closing the corresponding switches.

- Close switch 1 to execute sequence #1
- Close switch 2 to execute sequence #2
- Close switches 1 & 2 to execute sequence #3
- Close switches 1, 8, 10, and 80 to execute sequence #99

Stopping Motion while in Seq. Scan Mode

The sequence scan mode can be activated by placing the `SSJ1` command into sequence #100 (power-up sequence). If you want to use a stop input or a `STOP` command in your sequences, then one of three methods of operation may be desired.

One method is to use the `STOP` command to stop a move and exit the sequence scan mode. In this method, an `SSJ1` command must be issued to re-enter the sequence scan mode. The current motion in the sequence being executed will be stopped and no more sequences will be executed, and the remaining commands in the sequence (after the motion) will not be executed.

A second method is to have the `STOP` command stop motion of the current move being executed and to complete execution of the remaining commands in that sequence. In this method, the sequence is finished although the move is stopped. This mode is enabled with the `SSH1` command. In this mode, sequence scanning will also stop, and must be re-enabled by issuing the `SSJ1`.

A third method is to stop motion, complete the current sequence, and to stay in the sequence scanning mode. In this method, the current move in the sequence is stopped, all subsequent commands in that sequence are executed, and when the sequence is complete the 500 again scans the inputs to execute the next valid sequence. This mode is enabled by issuing the `SSH1` and `OSI1` commands.

Thumbwheel Interface

The Model 500 allows two methods for thumbwheel use. One method allows you to wire your own thumbwheels. The other uses Compumotor's TM8 thumbwheel module.

With the 500, you can use up to 40 digits of thumbwheels. The TM8 requires a multiplexed BCD input scheme to read thumbwheel data. Therefore, a decode circuit must be used for thumbwheels. Compumotor recommends that you purchase Compumotor's TM8 module if you desire to use a thumbwheel interface. The TM8 contains the decode logic; therefore, only wiring is needed.

The 500 commands and format that allow for thumbwheel data entry are:

<u>Command</u>	<u>Description</u>
<code>DRD</code>	Read distance via thumbwheels
<code>VRD</code>	Read velocity via thumbwheels
<code>LRD</code>	Read loop count via thumbwheels
<code>TRD</code>	Read time delay via thumbwheels
<code>VARDn</code>	Read variables via thumbwheels
<code>XRD</code>	Read sequence count via thumbwheels
<code>FRD</code>	Read following ratio via thumbwheels

Using the TM8 Module

To use Compumotor's TM8 Module, follow the procedures below.

- STEP ①** To select the TM8 module, first type the `TW1` command to enable the *TM8 Mode*.
- STEP ②** Wire your TM8 module to the 500 as shown in Figure 4-13. If you are using more than one TM8 Module (the maximum allowed is six for each 500), wire the modules as in Figure 4-14.

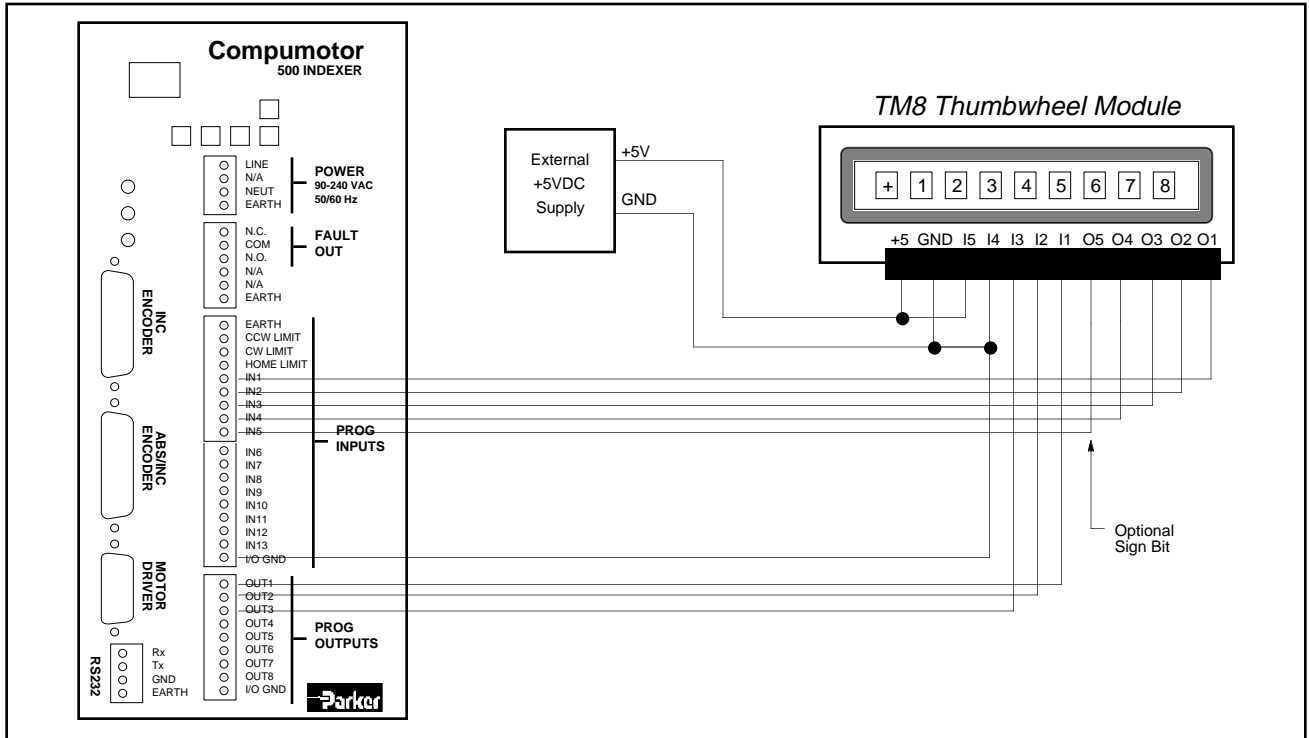


Figure 4-13. Wiring One TM8 Thumbwheel Module to the 500

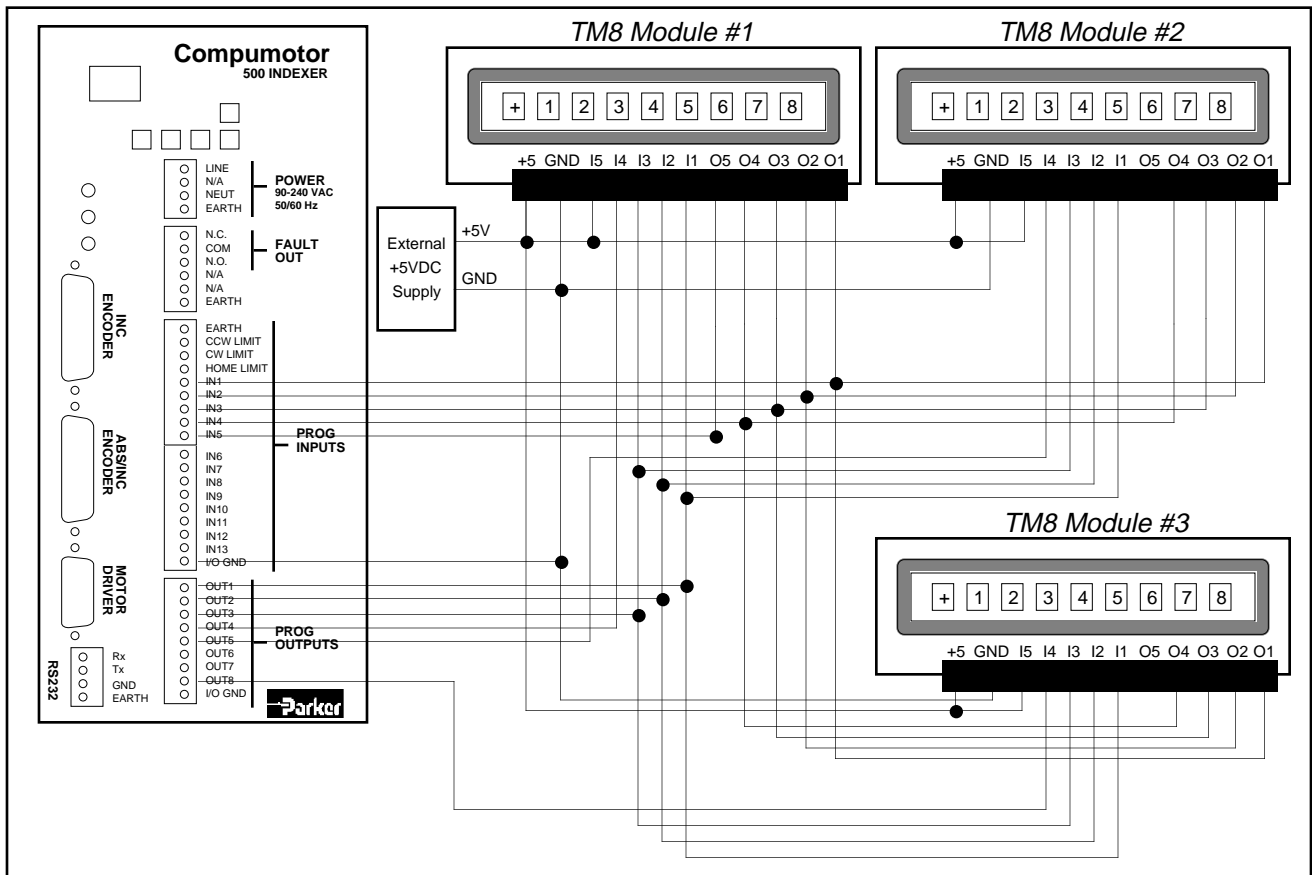


Figure 4-14. Wiring Multiple TM8 Thumbwheel Modules to the 500

STEP ③ Configure your 500 as follows:

<u>Command</u>	<u>Description</u>
> OUT1J	OUT1 configured as a strobe
> OUT2J	OUT2 configured as a strobe
> OUT3J	OUT3 configured as a strobe
> IN1N	IN1 configured as a data input
> IN2N	IN2 configured as a data input
> IN3N	IN3 configured as a data input
> IN4N	IN4 configured as a data input
> IN5W	IN5 configured as a sign input (optional)
> INLØ	Inputs configured active low
> STR1Ø	Data strobe time of 10 ms per digit read. If using one TM8 Module, you should now be ready to read in thumbwheel data. If using two TM8 Modules, enter the additional set-up commands. The minimum strobe time recommended for the TM8 module is 10 ms.
> OUT4A	OUT4 configured as a programmable output
> OUT5A	OUT5 configured as a programmable output
> OUT8A	OUT8 configured as a programmable output
> OUTL1	Outputs set active high
> OØ1XX1	Set output 4 low—enables TM8 unit #1 if you are using multiple TM8s

STEP ④ Set the thumbwheel digits on your TM8 module to **+12345678** (if using multiple TM8 modules, set the modules to some other value). To verify that you have wired your TM8 module(s) correctly and configured your 500 I/O properly, enter the following commands:

<u>Command</u>	<u>Description</u>
> DRD	Request distance data from all 8 thumbwheel digits
> 1D	Displays the distance read—D+12345678. If you do not receive the response shown, return to step 1 and retry.

If you are using multiple TM8 modules, enter the following commands:

> OØ1XX1	Enables TM8 module #1 and disables modules #2 and #3
> DRD	Request distance data from all 8 thumbwheel digits
> 1D	Displays the distance read—D+12345678 <i>The sign is positive.</i> This is because only one sign digit may be used when two TM8 modules are used. If you do not receive the response shown, return to step 1 and retry.
> O11XX1	Disables all TM8 modules if using multiple modules

Thumbwheel Example The following thumbwheel example clarifies how to use the TM8 module. If using multiple modules, enter the following commands:

<u>Command</u>	<u>Description</u>
> OØ1XX1	Enables TM8 module #1 and disables modules #2 and #3
> VARD1	Reads the thumbwheels into Variable 1
> O1ØXX1	Enables TM8 module #2 and disables modules #1 and #3
> VARD2	Reads the thumbwheels into Variable 2
> O11XXØ	Enables TM8 module #3 and disables modules #1 and #2
> VARD3	Reads the thumbwheels into Variable 3

14 on the TM8 module acts as an enable input that is active low. 15 on the TM8 acts as an enable input that is active high. By using the first three Model 500 outputs (OUT1 - OUT3) as *strobe* outputs for decoding the thumbwheel digits, the remaining outputs can be used for selecting multiple thumbwheels. In Figure 4-14, the active low input (14 on the TM8) functions as the *module select input*. One output line is dedicated to select a particular module. The three outputs that decode the digits are common to all modules used.

Using your own Thumbwheel Module

As an alternative to Compumotor's TM8 Module, you can use your own thumbwheels. To use your own thumbwheels, you must use 8 inputs for thumbwheel operation. Use the following steps to set up and read the thumbwheel interface. Refer to the **Model 500 Software Reference Guide** for detailed descriptions of the commands used below.

- STEP ①** Enable the PLC Mode with the `TW0` command. The PLC mode allows you to use the 500 with a PLC or your own thumbwheels. The 500 provides certain commands that aid you in using a PLC. This will be covered later in the section entitled *PLC Operation*.
- STEP ②** Wire your thumbwheels according to the schematic in Figure 4-15.

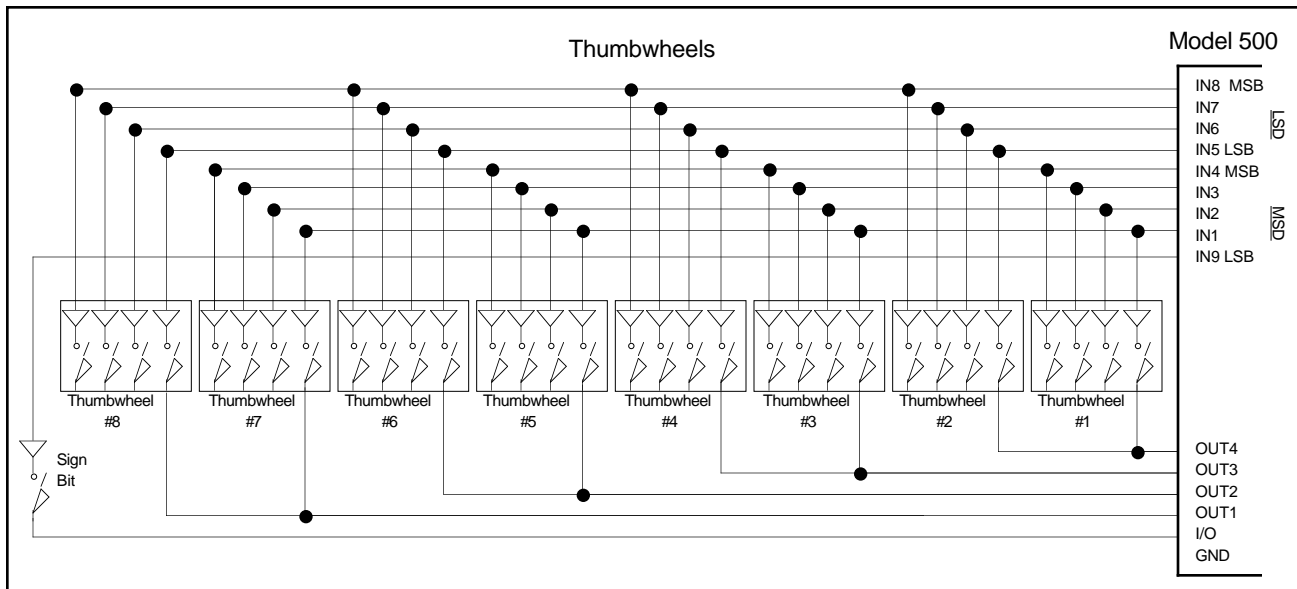


Figure 4-15. Thumbwheel Connections to the Model 500

- STEP ③** Set up the inputs and outputs for operation with thumbwheels. The data valid input will be an input which the operator holds active to let the Model 500 read the thumbwheels. This input is not necessary; however, it is often used when interfacing with PLCs.

Command	Description
> <code>OUT1J</code>	Sets Output #1 (OUT1) as a programmable output
> <code>OUT2J</code>	Sets Output #2 (OUT2) as a programmable output
> <code>OUT3J</code>	Sets Output #3 (OUT3) as a programmable output
> <code>OUT4J</code>	Sets Output #4 (OUT4) as a programmable output
> <code>IN1N</code>	Sets Input #1 (IN1) as a data input
> <code>IN2N</code>	Sets Input #2 (IN2) as a data input
> <code>IN3N</code>	Sets Input #3 (IN3) as a data input
> <code>IN4N</code>	Sets Input #4 (IN4) as a data input
> <code>IN5N</code>	Sets Input #5 (IN5) as a data input
> <code>IN6N</code>	Sets Input #6 (IN6) as a data input
> <code>IN7N</code>	Sets Input #7 (IN7) as a data input
> <code>IN8N</code>	Sets Input #8 (IN8) as a data input
> <code>IN9W</code>	Sets Input #9 (IN9) as a data sign input
> <code>STR12</code>	Sets the strobe time to 12 milliseconds

STEP ④ Each output strobes in two digits at a time. This is how the thumbwheels are read. The sign bit is optional. Set the thumbwheels to +12345678 and type in the following commands:

Command	Description
> DRD	Reads the thumbwheels into the distance parameter
> 1D	Reports the distance
> VARD1	Reads the thumbwheels into Variable 1
> 1VAR1	Reports Variable 1

Selecting Sequences with the Thumbwheel Module

The following example shows how the Model 500 can be used to execute sequences with remote thumbwheels. In the example, only 3 sequences are entered. As many as 100 sequences may be defined and up to 99 may be executed using remote thumbwheels. Sequence 100 is automatically executed during power up or reset. Refer to the Reset (z) command in the **Model 500 Software Reference Guide**. Install your thumbwheels as shown in Figure 4-16.

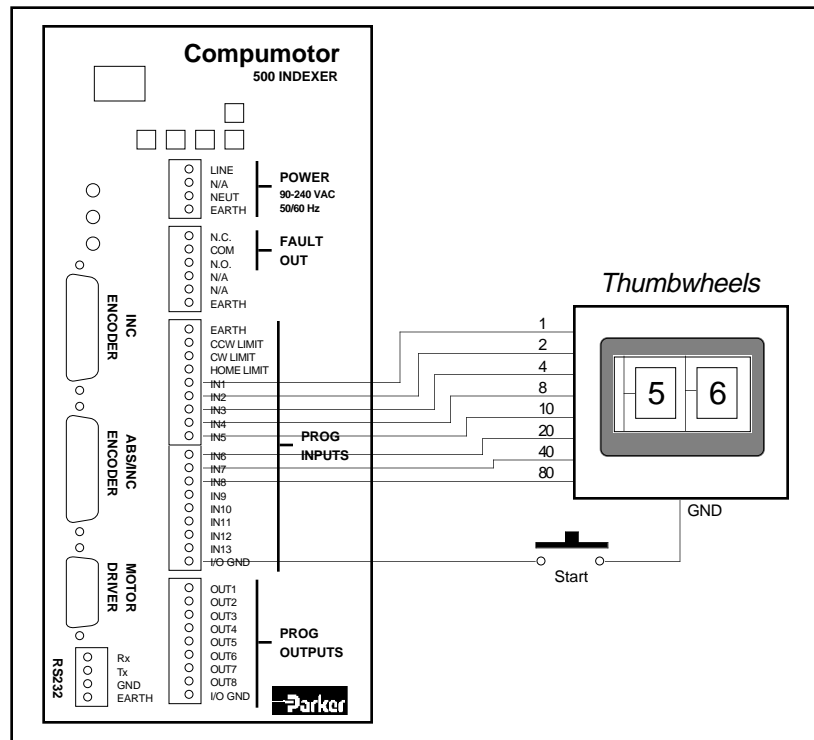


Figure 4-16. Model 500 Thumbwheel Installation

The inputs are defines as *sequence select* inputs rather than *data* inputs. The 500 is used in the sequence scan mode (set up with the SSJ1 command). When the start button is pushed, the input lines are opened and closed according to the thumbwheel lines.

STEP ① Define a power-up sequence. Set up inputs **IN1 - IN8** as sequence-select inputs using the Configure Input (**IN**) commands. The **IN1** command is the least significant bit of the 1s digit. The **IN8** command is the most significant digit of the 10s digit. The BCD values of inputs **IN1 - IN8** are shown in Table 4-3.

Command	Description
> XE100	Erases sequence #100
> XD100	Defines sequence #100
IN1B	Sets up Input #1 as the sequence input
IN2B	Sets up Input #2 as the sequence input
IN3B	Sets up Input #3 as the sequence input
IN4B	Sets up Input #4 as the sequence input
IN5B	Sets up Input #5 as the sequence input
IN6B	Sets up Input #6 as the sequence input
IN7B	Sets up Input #7 as the sequence input
IN8B	Sets up Input 8 as the sequence input
INL0	Sets active input level to active low (ON = 0V applied to the input)
SSJ1	Set Model 500 to run programs from remote interface
SN10	Set scan time to 10 msec
XQ1	Enable sequence hold
> XT	Ends sequence definition

Inputs	IN8	IN7	IN6	IN5	IN4	IN3	IN2	IN1
BCD Value	80	40	20	10	8	4	2	1
	10's Digit				1's Digit			

Table 4-3. Input BCD Values

STEP ② Define any sequences that your application may need.

Command	Description
> XE1	Erases sequence #1
> XD1	Starts sequence #1 definition
MN	Sets mode to normal
A25	Sets acceleration to 25 rps
AD25	Sets deceleration to 25 rps
V5	Sets velocity to 5 rps
D25000	Sets distance to 25,000 steps
G	Executes the move (Go)
> XT	Ends sequence definition

Command	Description
> XE5	Erases sequence #5
> XD5	Starts sequence #5 definition
MN	Sets mode to normal
A25	Sets acceleration to 25 rps ²
AD25	Sets deceleration to 25 rps
V5	Sets velocity to 5 rps
D10000	Sets distance to 10,000 steps
G	Executes the move (Go)
> XT	Ends sequence definition

Command	Description
> XE99	Erases sequence #99
> XD99	Starts sequence #99 definition
MN	Sets mode to normal
A25	Sets acceleration to 25 rps ²
AD25	Sets deceleration to 25 rps
V5	Sets velocity to 5 rps
D-35000	Sets distance to -35,000 steps
G	Executes the move (Go)
> XT	Ends sequence definition

STEP ③ Connect the thumbwheels to the **IN1 - IN8** inputs as shown in Figure 4-16.

STEP ④ Reset the Model 500 Indexer. This will prepare the power up (Sequence #100) to be executed when you apply power to the indexer.

<u>Command</u>	<u>Description</u>
> z	Resets the Model 500

STEP ⑤ Set your thumbwheel to 1 and push start to move the motor 25,000 steps in the positive direction.

STEP ⑥ Set your thumbwheel to 5 and push start to move the motor 10,000 steps in the positive direction.

STEP ⑦ Set your thumbwheel to 99 and push start to move the motor 35,000 steps in the negative direction.

If you select (with the thumbwheel) an invalid or unprogrammed sequence, nothing will happen.

PLC Operation

This section explains and provides examples of how to use a PLC with the 500 Indexer/Drive.

Interfacing with a PLC

In many applications it is desirable to interface to a PLC. The 500 performs the motion segment of a more involved process controlled by a PLC. In these applications, the PLC will execute sequences, load data, manipulate inputs and perform other specific input functions to control 500 and the motion segment of a process.

The PLC is selected with the **TW0** command. It requires that eight of the inputs are used for data entry. The data is always provided to the 500 from the PLC in a BCD format. The PLC can be used to select sequences in conjunction with the sequence scan mode (**SSJ1**), or, as with thumbwheels, the PLC can be used to enter data using the data read commands (**XRD**, **DRD**, **VRD**, **LRD**, **TRD**, or **VARD**).

Data Read with a PLC

As in the thumbwheel case described earlier, the PLC can be used to enter data for sequence select (**XRD**), distance (**DRD**), velocity (**VRD**), loop count (**LRD**), time delay (**TRD**), and variable data (**VARD**). Detailed explanations of these commands are provided in the **Model 500 Software Reference Guide**.

When using data read commands, the data is read in the same manner as when using your own thumbwheels. Each strobe output selects two BCD digits of data. Multiple strobe outputs select multiple digit pairs.

The timing for entering data can be controlled in two ways: (1) setting the strobe time with the **STR** command, and (2), defining a Model 500 input as a *data valid* input.

**CONTROL
TIMING —
STROBE
DURATION**

The first method of controlling timing for entering data is to use the **STR** command to set the length of time for which the strobe outputs stay active. Using this method, the strobe time would be set higher than the PLC's scan time. The PLC interprets the strobe and places data on the data lines. The 500 waits for the duration of the strobe time (**STR** setting), then reads the data and issues the next strobe. This is done internally in the 500 until it has issued all strobes that are defined. Of outputs #1 - #4 are defined as strobes, the 500 issues the strobes in the order of **OUT1**, **OUT2**, **OUT3**, and then **OUT4**.

To read data from the PLC, the 500's inputs #1 - #8 (**IN1 - IN8**) must be configured as data inputs with an active low level (**INLØ**).

If a sign digit is required, 500 input #9 should be configured as a sign input. Configure outputs #1 - #4 as data strobe outputs.

When the 500 executes a data read command, it will cycle its outputs and read BCD data as shown in Table 4-4.

Model 500 Outputs				Model 500 Inputs							
OUT1	OUT2	OUT3	OUT4	IN1	IN2	IN3	IN4	IN5	IN6	IN7	IN8
low	high	high	high	Digit 8 LSB	Digit 8	Digit 8	Digit 8 MSB	Digit 7 LSB	Digit 7	Digit 7	Digit 7 MSB
high	low	high	high	Digit 6 LSB	Digit 6	Digit 6	Digit 6 MSB	Digit 5 LSB	Digit 5	Digit 5	Digit 5 MSB
high	high	low	high	Digit 4 LSB	Digit 4	Digit 4	Digit 4 MSB	Digit 3 LSB	Digit 3	Digit 3	Digit 3 MSB
high	high	high	low	Digit 2 LSB	Digit 2	Digit 2	Digit 2 MSB	Digit 1 LSB	Digit 1	Digit 1	Digit 1 MSB

Table 4-4. Output Voltage Levels

Figure 4-17 shows a possible PLC-to-500 connection that would allow the 500 to input data from the PLC.

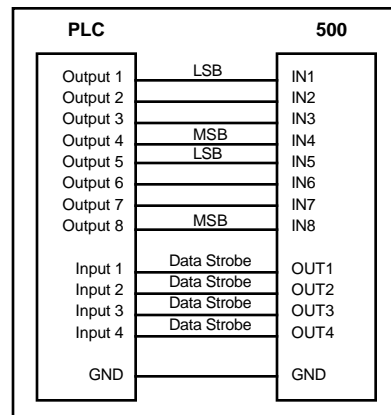


Figure 4-17. PLC-to-500 Connection

If the 500 executes a Read Distance Via Parallel I/O (DRD) command, the following events would have to occur to transfer distance data from the PLC to the 500.

- ① The 500 executes a DRD command and turns on OUT1.
- ② The PLC places and holds 2 BCD digits (digits 7 & 8) at the 500's IN1 - IN8. (This value will be the most significant distance digit). The PLC places a sign value at the 500's IN9. (***This sign bit must be the same for each digit read.***)
- ③ After the strobe time has elapsed, the 500 will read the digits and sign values.
- ④ The 500 will place its OUT1 at high voltage (5 - 24V) and will turn on OUT2.
- ⑤ The PLC must place and hold its next set of BCD digits (digits 5 & 6) at the 500's IN1 - IN8. The PLC should place the same sign value as that given for IN9.
- ⑥ After the strobe time has elapsed, the 500 will read this second set of BCD digits as the third and fourth significant distance digits.

This process continues (see Table 4-4) until the 500 reads the eighth digit (LSD). At this point, the 500 enters the eight digits read into its distance register and proceeds with the execution of subsequent commands.

**CONTROL
TIMING —
DATA VALID
INPUT**

The second method of controlling timing is to define a 500 input as a *data valid* input using the IN command. In this case, the 500 will issue a strobe, then the PLC will place the data onto the data lines and toggle the data valid input to the 500. The 500 will not read the data until the data valid is issued. The 500 will then read the data and issue the next strobe.

For example, input #10 (IN10) could be defined as a data valid input (input function V) with the IN10V command. After each strobe from the 500, the PLC places two digits on the data lines, then issues data valid.

Sequences can be selected and executed using the XRD command. The next section explains an alternative to using the data entry format.

**Sequence
Select With a
PLC**

As described in the previous sections, you can execute sequences by reading the sequence to be executed from data inputs. As an alternative, you could also execute sequences using a PLC by defining the inputs as *sequence select* inputs.

Again, the inputs are BCD weighted. As explained earlier in the *Sequence Select & Sequence Scanning* section, you must use the sequence scan (SSJ1) mode. Different modes of sequence scanning are accomplished using the XQ, SSH, and OSI commands (refer to the **Model 500 Software Reference Guide**).

The 500 inputs are *not sinking inputs* (refer to Chapter 6, *Hardware Reference*).

The PLC activates the appropriate inputs to execute the desired sequence. At the end of each sequence, you can program the 500 to turn on an output (using the O command) to indicate to the PLC to select the next sequence.

The following are step-by-step procedures to run sequences from your PLC. First, you need to enter the programs into the Model 500. You will need a terminal or a computer with RS-232C communication capability. You need to define the sequences before you can execute them with your PLC's BCD outputs. The Model 500 indexer automatically saves these sequences (battery backed-up system).

STEP ① Define a power-up sequence.

<u>Command</u>	<u>Definition</u>
> XE100	Erases sequence #100
> XD100	Defines sequence #100
SSJ1	Executes sequences via PLC input
SN20	Sets scan time to 20 msec
XQ1	Sets Model 500 to interrupted run mode
A10	Sets acceleration to 10 rps ²
AD10	Sets deceleration to 10 rps ²
V2	Sets velocity to 2 rps
IN1B	Sets Input 1 as a sequence-select input
IN2B	Sets Input 2 as a sequence-select input
IN3B	Sets Input 3 as a sequence-select input
IN4B	Sets Input 4 as a sequence-select input
IN5B	Sets Input 5 as a sequence-select input
IN6B	Sets Input 6 as a sequence-select input
IN7B	Sets Input 7 as a sequence-select input
IN8B	Sets Input 8 as a sequence-select input
OUT1C	Sets Output 1 as sequence-in-progress output
LD3	Disables the limits (if they are not connected)
> XT	Ends the sequence definition

Every time you power up the Model 500, it executes these commands and enables the Model 500 to read up to 99 sequences from the sequence-select inputs.

STEP ② Define any sequences that your application may require.

<u>Command</u>	<u>Description</u>
> XE1	Erases Sequence #1
> XD1	Defines Sequence #1
D2000	Sets distance to 2,000 steps
G	Executes the move (Go)
> XT	Ends Sequence #1 definition

<u>Command</u>	<u>Description</u>
> XE2	Erases Sequence #2
> XD2	Defines Sequence #2
D4000	Sets distance to 4,000 steps
G	Executes the move (Go)
> XT	Ends Sequence #2 definition

<u>Command</u>	<u>Description</u>
> XE3	Erases Sequence #3
> XD3	Defines Sequence #3
D8000	Sets distance to 8,000 steps
G	Executes the move (Go)
> XT	Ends Sequence #3 definition

<u>Command</u>	<u>Description</u>
> XE99	Erases Sequence #99
> XD99	Defines Sequence #99
D-14000	Sets distance to -14,000 steps
G	Executes the move (Go)
> XT	Ends Sequence #99 definition

STEP ③ Verify that your programs were stored properly by uploading each entered sequence (XU). If you receive responses that differ from what you programmed, re-enter those sequences.

STEP ④ Run each program from the RS-232C interface with the Run Sequence (XR) command. Make sure that the motor moves the distance that you specify.

STEP ⑤ Assuming your PLC accepts open-collector outputs connect the inputs and outputs as shown in Figure 4-18. If not, you will need to add pull-up resistors to the outputs.

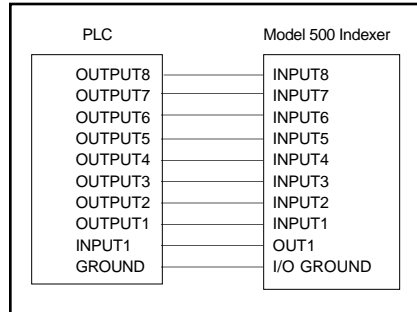


Figure 4-18. PLC Connection

STEP ⑥ Refer to the user guide that accompanied your PLC unit to turn on the proper combination of outputs to execute one of the four sequences programmed and stored in the Model 500 indexer. You should program the PLC to search for the sequence completion output from the 500 before the 500 begins searching for the next sequence to execute; this type of interaction is performed through a *handshake* between the 500 and the PLC — the signal transmitted indicates that another sequence is ready to be executed.

- Turn on Output 1 (only) to execute sequence #1.
- Turn on Output 2 (only) to execute sequence #2.
- Turn on Outputs 1 and 2 (only) to execute sequence #3.
- Turn on Outputs 1, 4, 5, and 8 (only) to execute sequence #99.

STEP ⑦ Cycle down power to the Model 500. The system will execute sequence 100.

STEP ⑧ Turn on the appropriate sequence-select input [set for the least Scan Time (SN)] to execute the proper sequences. Since the sequence hold feature (refer to the XQ1 command) was enabled during the power-up sequence, your PLC program must turn off all of the sequence-select inputs before you can select another sequence.

Miscellaneous Control by a PLC

You can use a PLC to control the activation of the inputs for all of the input functions that the 500 supports (see the *Programmable Input* section). For example, you can use the PLC to stop, kill, go, go home, or reset the 500.

Front Panel Operation

The default function of the front panel pushbuttons is to display errors and to change baud rate, device address, and the gain for position maintenance.

Procedures for changing the baud rate and device address and checking the revision level are provided in Chapter 2, *Getting Started*. To change the proportional gain, use the following procedure:

STEP ① Display the proportional gain by pressing the 1s and ENTER pushbuttons simultaneously (see Figure 4-19).

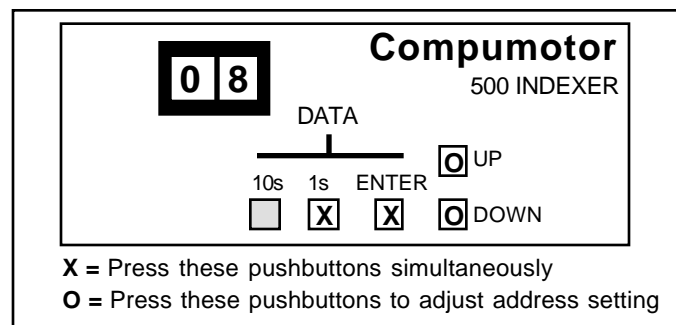


Figure 4-19. Changing Proportional Gain Via Front Panel Pushbuttons

- STEP ② While pressing the **1s** and **ENTER** pushbuttons simultaneously, you can use either the **UP** or the **DOWN** pushbuttons to increase or decrease the gain.

Executing Sequences via Pushbuttons

You can also use the pushbuttons to execute sequences. To use this feature you must issue the following commands:

Command	Description
> SSO1	Enables pushbuttons to function as sequence-select buttons
> CPB1	Enables mode for executing sequences via front panel pushbuttons

Use the following procedure to select a sequence number:

- STEP ① Press and hold down the appropriate button (**10s** or **1s**) for the respective digit in the sequence number. In this mode of operation, the **10s** button represents the 10s digit of the sequence, and the **1s** button represents the 1s digit of the sequence.
- STEP ② While pressing the **10s** or **1s** button, press the **UP** or **DOWN** button to select the proper digit.
- STEP ③ Once you have the proper sequence number displayed, press the **ENTER** button to execute the sequence.

Summary of Pushbutton Features

Table 4-5 lists the functions of the pushbuttons. The **X** symbol means push the pushbutton; the **∅** symbol means do not push the pushbutton.

10's	1's	ENTER	UP	DOWN	Function
∅	∅	∅	∅	∅	Display Errors
X	∅	∅	∅	∅	Display Sequence* or Following ratio
X	∅	∅	X	∅	Increase Sequence* or Following
X	∅	∅	∅	X	Decrease Sequence* or Following
∅	X	∅	∅	∅	Display Sequence* or Following
∅	X	∅	X	∅	Increase Sequence* or Following
X	X	∅	∅	X	Decrease Sequence* or Following
∅	∅	X	∅	∅	Execute Sequence* or Enter Following
∅	∅	X	X	∅	Reserved
∅	∅	X	∅	X	Reserved
X	X	∅	∅	∅	Display RS-232 Baud Rate
X	X	∅	X	∅	Increase RS-232 Baud Rate
X	X	∅	∅	X	Decrease RS-232 Baud Rate
X	∅	X	∅	∅	Display RS-232 Device Address
X	∅	X	X	∅	Increase RS-232 Device Address
X	∅	X	∅	X	Decrease RS-232 Device Address
∅	X	X	∅	∅	Display Proportional Gain
∅	X	X	X	∅	Increase Proportional Gain
∅	X	X	∅	X	Decrease Proportional Gain
X	X	X	∅	∅	Display Software Rev.Level
X	X	X	X	∅	Reserved
∅	∅	∅	X	X	Reset
X	X	X	X	∅	Reserved

* Function applicable only if **SSO1** and **CPB1** commands have been entered for executing sequences via front panel pushbuttons

Table 4-5. Model 500 Pushbutton Features

Remote Panel Operation

The 500 is compatible with the Compumotor RP240 remote front panel. The RP240 has a 2-line 40-character display, a numeric keypad, and programmable function keys. The RP240 communicates with the 500 over RS-232C and allows you to input data, display messages, and prompt the user for appropriate actions or data input. For more information on the RP240, refer to the ***Model RP240 User Guide***, or contact your local distributor or Compumotor.

Host Computer Operation

Another choice for a user interface is to use a host computer and execute a motion program using the RS-232C serial interface. Two types of operation can be performed when using a host computer. The host can run interactively from a Basic or C program. In this case, the high-level program controls the 500 and acts as an interface to the user. The following program is an example of this type of operation.

```

1 '                               500.BAS PROGRAM
2 '
3 ' *****
4 ' *
5 ' * This program controls the RS232 Communication line to execute 2 different moves
6 ' * using the 500
7 ' *
8 ' *****
15 OPEN "COM1:9600,N,8,1,RS,CS,DS,CD" AS #1 ' Open Communication port
20 V$ = "": Q$ = "": ECHO$ = "": LF$ = "": ' Initialize variables
90 CLS
100 LOCATE 12,15
105 PRINT " PRESS ANY KEY TO START THE PROGRAM "
107 V$ = INKEY$: IF LEN(V$) = 0 THEN 100 ' Wait for input from user
120 Z$ = "K K LD3" ' Reset and disable limits on the 500
122 PRINT #1,Z$;
124 Q$ = INPUT$(8,1)
900 ' *****
901 ' *
902 ' * Lines 1000-1060 sends a move down to the first 500. Computer waits for the Line
903 ' * Feed from the 500 indicating that the motor has finished its move. Computer
904 ' * will not command second 500 to move until the first move is completed.
905 ' *
906 ' *****
1000 MOVE1$ = "MN A1 V2 D40000 G 1LF " ' Define move 1
1005 CLS
1007 LOCATE 12,15: PRINT " DOING MOVE 1 "
1010 PRINT #1,MOVE1$ ' Perform move 1.
1015 ECHO$ = INPUT$(23,1) ' Read echoes from 500.
1020 LF$ = INPUT$(1,1) ' Wait for line feed from 500
1040 IF LF$ <> CHR$(10) GOTO 1020 ' indicating end of move.
1045 CLS
1047 LOCATE 12,15
1050 PRINT "MOVE 1 DONE" ' Let user know that move 1 is done
1060 LOCATE 15,15: PRINT " PRESS ANY KEY TO GO ON TO SECOND MOVE "
1070 V$ = INKEY$: IF LEN(V$) = 0 THEN 1060
1900 '
1901 ' *
1902 ' * After axis one is done, we request that you hit any key to go on to second move.
1903 ' * In real application, we would expect you to go ahead with the process and work on
1904 ' * on the part before going on to next move. (i.e., Activate a punch)
1905 ' * Now that first move is finished go on to move 2. The 500 also prints a line feed
1906 ' * after finishing the second move.
1907 ' * As soon as the computer receives the line feed from 500, the program will go back
1908 ' * to the first move.
1909 ' *
1910 ' *****
2000 MOVE2$ = "H G 1LF "
2005 CLS
2007 LOCATE 12,15: PRINT " DOING MOVE 2 "
2010 PRINT #1,MOVE2$
2015 ECHO$ = INPUT$(8,1)
2020 LF$ = INPUT$(1,1)
2040 IF LF$ <> CHR$(10) GOTO 2020
2045 CLS
2047 LOCATE 12,15
2050 PRINT "MOVE 2 DONE "
2060 FOR I = 1 TO 1000: NEXT I
2070 GOTO 20 ' Go back to beginning of program.

```

The second method is for the 500 to run a stored program and prompt the user interactively as a part of the program. To do this, you must use the RS-232C Input (RSIN) and Quote (") commands. The Quote command sends messages over the RS-232C link to the host. The message may be status data or a request for information. If it is a request, RSIN allows you to enter data into any of the general-purpose variables. The motion program may use these variables. The following program is an example of a stored sequence that interactively requests the number of parts and the size.

Command	Description
> 1XE1	Erase sequence #1
> 1XD1	Define sequence #1
1"Enter_the_number_of_parts_to_be_made"	Send string to host
VAR1=RSIN	Set variable #1 to the value input by the user
1"Enter_the_size_of_the_parts"	
VAR2=RSIN	Set variable #2 to the value input by the user
VAR2=VAR2*25000	Scale variable #2 by 25000 for user units
D(VAR2)	Set distance (part size) to variable #2
L(VAR1)	Set loop count (part count) to variable #1
G	Execute move
N	Continue loop
> XT	Terminate sequence

The user inputs data by typing an exclamation mark (!) followed by the data.

Multi-axis Control (Daisy-chaining)

You may daisy-chain up to 16 Model 500s to one serial port on a terminal. Figure 4-20 shows a three-indexer daisy-chain configuration from one controlling terminal or computer.

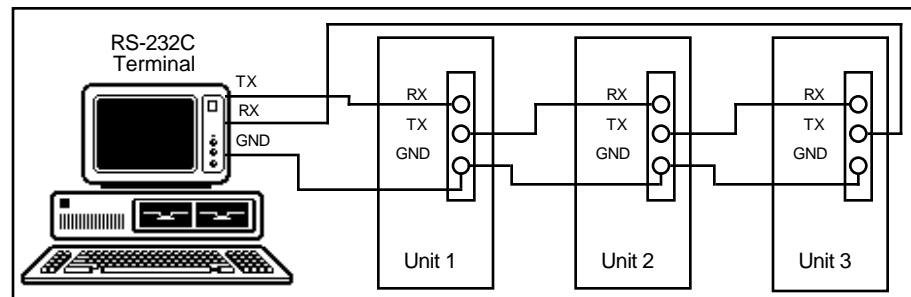


Figure 4-20. RS-232C Daisy-Chain Configuration

If you are daisy-chaining 500s, you should establish different addresses for each unit so you can distinguish them when you are programming. Refer to Chapter 2, *Getting Started*, for instructions on displaying and changing the 500's device address.

Commands prefixed with a device address affect only the unit specified. Commands without a device address affect all units on the daisy chain. For instance, entering the 1G command instructs only unit #1 to go (move), but entering the G command instructs all units on the daisy-chain to go.

Any command that causes the drive to transmit information from the RS-232C port (such as a status or report command), must be prefixed with a device address. This prevents daisy-chained units from all transmitting at the same time.

No 500 executes a device-specific command unless the unit number specified with the command matches the 500's unit number. Device-specific commands include both buffered and immediate commands.

You must use status request commands in an orderly fashion. Commands should only be issued when the host is ready to read the response. You should not send new commands until you receive a response from the previous status request command. In particular, you should not issue an immediate-status command until the host receives a buffered command status response. If this is not followed, the command responses will be intertwined, rendering the data useless.

If you enable the Interactive mode (`SSI1`), only the 500 that is set to address 1 will respond with a prompt (`>`). This prevents all the 500s from sending out `>` in a daisy-chain. Typically, you should disable the Interactive mode when you use a host computer with the 500.

Sample Application and Commands

The following move example is for three indexers on an RS-232C daisy-chain.

Command	Description
<code>> MN</code>	Sets unit to Preset mode
<code>> A5</code>	Sets acceleration to 5 rps ² for all three controllers
<code>> V10</code>	Sets velocity to 5 rps for all three controllers
<code>> LD3</code>	Disables limits (if they are not connected)
<code>> 1D25000</code>	Sets Axis 1 distance to 25,000 steps
<code>> 2D50000</code>	Sets Axis 2 distance to 50,000 steps
<code>> 3D100000</code>	Sets Axis 3 distance to 100,000 steps
<code>> G</code>	Moves all axes.

Unit 1 moves 25,000 steps, unit 2 moves 50,000 steps, and unit 3 moves 100,000 steps. All three units use the same acceleration and velocity rates. All three units begin movement at about the same time.

Communicating with Other Compumotor Indexers

Because of their common programming language, Compumotor Model 500s, SXs, and ZXs can communicate with each other over a daisy-chain using the `TX` command. The `TX` command allows one unit to send a command to another unit with the value in a variable appended to the command. The **Model 500 Software Reference Guide** provides a detailed description of the `TX` command.

For example, if a 500 and an SX are daisy-chained, only one set of thumbwheels is needed to enter data into both units. One 500 can serve as the *master*, reading data in via the thumbwheels. The data is read into a variable. This data can then be sent the SX as a distance command. The commands for this example are as follows:

Command	Description
<code>> 1VARD1</code>	Unit 1 (one of the 500s) reads the thumbwheels: <code>VAR1=125.00000</code>
<code>> 1VAR1=VAR1*25000</code>	Convert to motor pulses (25000 pulses/inch)
<code>> TX1,0,0,2D</code>	<code>2D3125000</code> is transmitted to unit 2 (the SX)

Multi-Axis Interface Program Example

The following program is very similar to `Model 500.BAS`, except this program controls two Model 500s on a daisy-chain. This program assumes the device address of two Model 500s to be #1 and #2 respectively. The program does the following:

- ① Executes the first move upon user input
- ② Waits for a line feed from the Model 500 indexer, which indicates the end of the move.
- ③ Executes the second move upon user input.
- ④ Waits for a line feed from the Model 500 indexer, which indicates the end of the move. It then begins the process again.

```

1 '                               500-2.BAS PROGRAM
2 '
3 ' *****
4 ' *
5 ' * This program controls the RS232 Communication line to execute 2 different moves using *
6 ' * two Model 500 drives.
7 ' *
*****
15 OPEN "COM2:9600,N,8,1,RS,CS,DS,CD" AS #1 ' Open Communication port
20 V$ = "": Q$ = "": ECHO$ = "": LF$ = "": ' Initialize variables
90 CLS
100 LOCATE 12,15
105 PRINT " PRESS ANY KEY TO START THE PROGRAM "
107 V$ = INKEY$: IF LEN(V$) = 0 THEN 100 ' Wait for input from user
120 Z$ = "K LD3" ' Reset the Model 500 indexer
122 PRINT #1,Z$;
124 Q$ = INPUT$(8,1)
900 ' *****
901 ' *
902 ' * Line 1000-1060 sends a move down to the first Model 500. Computer
903 ' * waits for the Line Feed from the Model 500 indicating that the motor
904 ' * has finished its move. Computer will not command second Model 500
905 ' * until axis 1 has completed its move.
906 ' *
*****
1000 MOVE1$ = "1A1 1V2 1D40000 1G 1LF " ' Define move for Axis 1
1005 CLS
1007 LOCATE 12,15: PRINT " MOVING AXIS 1 "
1010 PRINT #1,MOVE1$ ' Move axis 1.
1015 ECHO$ = INPUT$(23,1) ' Read echoes from Model 500
1020 LF$ = INPUT$(1,1) ' Wait for line feed from Model 500
1040 IF LF$ <> CHR$(10) GOTO 1020 ' indicating end of move.
1045 CLS
1047 LOCATE 12,15
1050 PRINT "AXIS 1 FINISHED ITS MOVE " ' Let user know axis 1 done
1060 LOCATE 15,15: PRINT " PRESS ANY KEY TO MOVE SECOND AXIS "
1070 V$ = INKEY$: IF LEN(V$) = 0 THEN 1060
1900 '
*****
1901 ' *
1902 ' * After axis one is done, we request that you press any key to go on
1903 ' * to second move. In real application, we would expect you to
1904 ' * go ahead with the process and work on the part before going on to
1905 ' * next move. (i.e., Activate a punch)
1906 ' *
1907 ' * Now that first axis finished its move, we go on to move axis 2.
1908 ' * Second Model 500 also prints a line feed after finishing the move.
1909 ' * As soon as computer receives the line feed from Model 500, program will
1910 ' * go back to the first move.
1911 ' *
*****
2000 MOVE2$ = "2A1 2V2 2D10000 2G 2LF "
2005 CLS
2007 LOCATE 12,15: PRINT " AXIS 2 MOVING "
2010 PRINT #1,MOVE2$
2015 ECHO$ = INPUT$(23,1)
2020 LF$ = INPUT$(1,1)
2040 IF LF$ <> CHR$(10) GOTO 2020
2045 CLS
2047 LOCATE 12,15
2050 PRINT "AXIS 2 FINISHED ITS MOVE "
2060 FOR I = 1 TO 1000: NEXT I
2070 GOTO 20 ' Go back to beginning of program.

```

Model 500 Application Examples

Application Development Approach

This section provides application examples which illustrate the features and capabilities discussed in this chapter.

This section will help you to develop your application program. These steps provide a guideline for identifying and organizing your program's needs. Examples of actual programs demonstrate how the different programming concepts are used together. Use the following steps:

- ① Identify and define the types of motion required in your application. Examples are as follows:
 - Moves to a home switch; go home moves
 - Jogging the motor
 - Preset moves of triangular or trapezoidal velocity profiles
 - Registration moves
 - Moves with changing velocities not based on distance
 - Moves of changing velocities based on distance
 - Motion based on math calculations
 - Motion with outputs turning on at distance points
- ② Determine what will cause the motion to occur and when it will cause motion to occur. Choose the interface(s) for execution of the motion programs. Examples are as follows:
 - Input states causing motion to occur; triggers
 - Sequence execution
 - Thumbwheel inputs
 - Power up sequence
 - PLC
 - Variable conditions
 - Time delays
- ③ Determine what configuration commands are necessary to set the 500 in the proper state for execution of the motion programs. Examples are as follows:
 - Sequence scan mode—SSJ1, XQ1, SSH1
 - Input/Output configuration—IN1A, IN4D, OUT3J
 - Motion parameters—A, AD, V, D
 - Profile commands—MC, MN, MPI, MPA, MPP
 - Homing parameters—GHA, GHV, GHF, GHAD, OSB, OSG, OSH
 - Other special functions of the SS and FS commands

Once you have determined what set-up commands are required, you will typically place them in the power-up sequence, sequence #100. However, this may not always be the case. In some applications, the 500 may be required to be in different modes for different parts of the application. For example, some moves may be Continuous mode moves and some may be Preset mode moves. In this case, you can place the appropriate set-up commands in the sequence in which that particular motion is required. After entering the power-up sequence, you can cycle power, issue a Reset (Z) command, or simply run sequence #100 with an XR100 command to configure the 500 according to the set-up commands in the power-up sequence.

The manner in which the remaining sequences are programmed will be determined largely by the type of interface you choose. There are three typical types of applications that require different methods of executing motion through sequences.

Method ①

The interface may be discrete inputs which select or control sequences. In this case a program of several sequences will have the program flow controlled by the inputs or there may be a small number of sequences selected by discrete inputs as sequence select lines.

Method ② The interface is thumbwheels or a PLC and sequences are selected via the inputs. In this case the different motion requirements are usually in each sequence.

Method ③ The third typical application has data or move parameter information entered via the inputs. In this case, there are usually fewer sequences but they require some information from the data entered via the 500 inputs.

These types of applications are not all-inclusive, but they do offer ideas in how you might set up your interface to execute the required motion.

Application Example #1

In this application, parts are formed by a grinder. The part that is being ground moves at a high speed toward the grinder. Just before it hits the grinder, it slows to a lower speed and the grinding operation begins. Because there are different parts of different lengths, different distances are required for the grinding part of the operation. The distance from the point where the part is loaded to the point where the grinding of the part is finished is a fixed distance. What varies is the point at which the grinding begins. Only five different parts are made. The distance from the point of loading the part to the end of the grind is 24 inches. The motor is on a leadscrew with a pitch of 2. The operator interface must be very simple and include part-selection and run/stop switches.

Types of Motion Required

In this application, a preset move must be made. The first part of the distance must be moved at a high speed and the second part at a low speed. The point at which the move changes from high speed to low speed varies with each part type. A homing move positions the machine at the location where parts are loaded. From this location, the other grind moves can be made. Figure 4-21 is a motion profile of the grinding application.

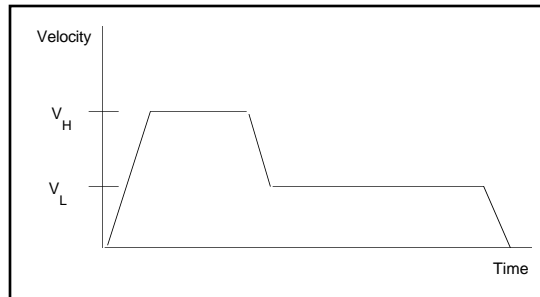
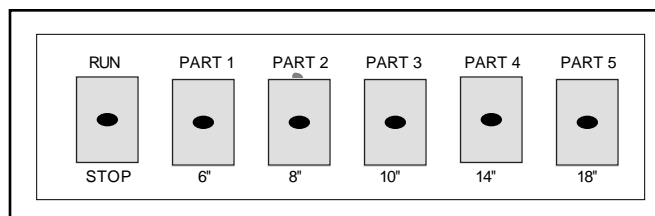


Figure 4-21. Grinding Application Motion Profile

What Will Cause Motion and When?

It is necessary to have the machine move to its part loading location when it starts so that it is ready to load parts and make grind moves. A home switch will be placed at the loading point to allow the 500 to position itself at the home location. A toggle switch allows the user to select either the Run mode or Stop mode. When the switch is in Run mode, it will continuously make parts of the type selected until the run/stop switch is placed in the Stop mode. The parts are selected by turning on the switch labeled for the particular part. The interface will be 6 switches. One switch will toggle between the run or stop mode. The user can select the part with the other 5 switches. The part-selection switches are on/off switches as opposed to momentary contact switches. Figure 4-22 shows the switch configuration.



What Configuration Commands Are Required?

Figure 4-22. Front Panel of User Interface

<input type="checkbox"/>	SSJ1	Sequence Scan mode
<input type="checkbox"/>	INnA	Trigger input, n is the input number
<input type="checkbox"/>	INnB	Sequence select input
<input type="checkbox"/>	OSB	Back up to home switch
<input type="checkbox"/>	OSG	Select final home approach direction
<input type="checkbox"/>	OSH	Select edge of home switch which 500 stops on
<input type="checkbox"/>	GHV	Select homing velocity
<input type="checkbox"/>	GHF	Final homing approach velocity

Implementation The part selection switches select a sequence that contains the necessary move for that particular part. The motion will not begin until the run/stop switch is in the run position. Parts will continue to be made until the run/stop switch is set to the stop position. The user can select Stop mode at any time during the grind move. The grind move that was executed at the time the Stop mode was selected will be completed. While in Stop mode, a new part type can be selected. The machine will operate this way as long as it is powered up. Upon power up, the machine will move to the home location to begin operation.

Required Sequences

The resolution set by the Configure Motor Resolution (CMR) command is 5000 steps/rev. With a pitch of 2, the total move distance is 240000 steps. The part sizes are 6, 8, 10, 14, and 18 inches. This makes grind distances of 60000, 80000, 100000, 140000, and 180000 steps.

SEQUENCE #100

Command	Description
> XD100	Begins definition of sequence #100
MPI	Incremental move mode
MN	Normal preset mode
CMR5000	Sets resolution to 5000 steps/rev
GHV2	Initial go home velocity if 2 rps
GHF.3	Final go home velocity of 0.3 rps
OSB1	Back up to home switch
OSH1	Stop on the CCW edge of the switch
OSG0	Final approach direction is CW
GH	Begin go home move
IN1B	Sets up input #1 as sequence select input
IN2B	Sets up input #2 as sequence select input
IN3B	Sets up input #3 as sequence select input
IN4B	Sets up input #4 as sequence select input
IN5B	Sets up input #5 as sequence select input
IN6A	Sets up input #6 as a trigger
SSJ1	Place the 500 in continuous scan mode
A100	Set the acceleration to 100 rps ²
TR1	Wait until run switch is enabled
> XT	Ends definition of sequence #100

SEQUENCE #1

Command	Description
> XD1	Begins definition of sequence #1
V4	Defines the fast velocity
D240000	Total move distance
REPEAT	Repeat loop to make parts continuously
MPP	Enters the profiling mode so velocity can be changed on the fly
G	Initiates motion
DP180000	After 180,000 steps decelerate to the grind velocity
V.5	Grind velocity
NG	Ends profiling mode (no commands executed until move is finished)
V4	Return for a new part at a high speed
H	Change direction
G	Return to the get a new part
UNTIL(XXXXXXXX0)	Continue the repeat loop until the run/stop switch has been placed in the stop location
TR1	Wait until the run/stop switch is placed in the run location then finish this sequence and scan the inputs to select another part sequence
> XT	Ends definition of sequence #1

```

SEQUENCE #2  > XD2
               V4
               D240000
               REPEAT
               MPP
               G
               DP160000
               V.5
               NG
               V4
               H
               G
               UNTIL(XXXXXXXX0)
               TR1
> XT

SEQUENCE #4  > XD4
               V4
               D240000
               REPEAT
               MPP
               G
               DP140000
               V.5
               NG
               V4
               H
               G
               UNTIL(XXXXXXXX0)
               TR1
> XT

SEQUENCE #8  > XD8
               V4
               D240000
               REPEAT
               MPP
               G
               DP100000
               V.5
               NG
               V4
               H
               G
               UNTIL(XXXXXXXX0)
               TR1
> XT

SEQUENCE #16 > XD16
               V4
               D240000
               REPEAT
               MPP
               G
               DP80000
               V.5
               NG
               V4
               H
               G
               UNTIL(XXXXXXXX0)
               TR1
> XT

```

Since the sequence select inputs are binary weighted, only one switch is needed to select each sequence above. That is why sequences 1, 2, 4, 8, and 16 were chosen. This method is useful only if there are a small number of sequences (7 or less).

Application Example #2

If we modify example #1 slightly, the same motion can be accomplished using a different user interface. In this example #2, we will have the same motion requirements, but 30 different parts will be made. The same run/stop criteria from example #1 are in effect; however, thumbwheels will be used to enter the grind distance data. The data will be entered as the length of the part. The distance of the grind move is the length of the part. After the distance is entered on the thumbwheels, the grind move will run until the stop switch is enabled. A homing move on power up is still required to place the machine in the parts loading position.

Required Motion

The required motion will be the same as example #1.

What Will Cause Motion and When?

Like example #1, a run/stop switch will be used to start and finish the grind moves. A homing move will be required to start the process. This will be done the same way as example 1. The point at which the velocity must slow to a lower velocity will be determined by thumbwheels and some math calculations. The user need only enter the length of the part in inches on the thumbwheels and then select the run mode. The 500 will then perform the required grind move.

What Configuration Commands Are Required?

- VARDn Reads the input data into variable n
- INnA Trigger input, n is the input number
- INnN Data input
- OUTnJ Sets the outputs as strobe outputs
- STR Strobe time for reading the data inputs
- OSB Back up to home switch
- OSG Select final home approach direction
- OSH Select edge of home switch which 500 stops on
- GHV Select homing velocity
- GHF Final homing approach velocity

Implementation

The process will run identically to the process above with the exception of selecting the grind move. In this case the grind move is determined by reading data in via the thumbwheels then beginning the move when the run mode is enabled. When the stop mode is enabled new grind move data can be entered on the thumbwheels. One bank of 8 thumbwheel switches is used so the enable inputs on the TM8 module are set to the enable state rather than to a 500 output.

Required Sequences

The resolution set by the Configure Motor Resolution (CMR) command is 5000 steps/rev.

SEQUENCE #100

Command	Description
> XD100	Begins definition of sequence #100
MPI	Incremental move mode
MN	Normal preset mode
CMR5000	Sets resolution to 5000 steps/rev
GHV2	Initial go home velocity if 2 rps
GHF.3	Final go home velocity of 0.3 rps
OSB1	Back up to home switch
OSH1	Stop on the CCW edge of the switch
OSG0	Final approach direction is CW
GH	Begin go home move
IN1N	Sets up input #1 as data input
IN2N	Sets up input #2 as data input
IN3N	Sets up input #3 as data input
IN4N	Sets up input #4 as data input
IN5A	Sets up input 5 as a trigger input
OUT1J	Sets up Output #1 as strobe output
OUT2J	Sets up Output #2 as strobe output
OUT3J	Sets up Output #3 as strobe output
STR10	Strobe time is 10 milliseconds
A100	Sets acceleration to 100 rps ²
XR1	Runs sequence #1
> XT	Ends definition of sequence #100

SEQUENCE #1	Command	Description
	> XD1	Begins definition of sequence #1
	L	Starts an infinite loop
	V4	Defines the fast velocity
	D240000	Total move distance
	VAR1	Reads the thumbwheels for variable 1
	VAR1=VAR1*10000	Convert part length inches to steps
	VAR1=240000-VAR1	Determine distance point DP where velocity changes to a lower speed
	REPEAT	REPEAT loop to repeatedly make parts
	MPP	Enters the profiling mode so velocity can be changed on the fly
	G	Initiates motion
	DP (VAR1)	Loads the distance point determined in variable 1
	V.5	Grind velocity
	NG	Ends the profiling mode. No commands will execute until the move is finished
	V4	Return for a new part at a high speed
	H	Change direction
	G	Return to the get a new part
	UNTIL (INXXXXXXXX0)	Continue the repeat loop until the run/stop switch has been placed in the stop location
	TR1	Wait until the run/stop switch is set to run then finish this sequence and scan the inputs to select another part sequence
	N	Ends the infinite loop
	> XT	Ends definition of sequence #1

This is the only sequence required. It is an infinite loop where parts are continuously run until the Stop mode is enabled. When this occurs, the program is waiting at the trigger command. It will remain there until the Run mode is selected. The thumbwheels will be read again so a new grind move can be run. This is more flexible and allows a many different grind moves to be executed.

Application Example #3

This application is a process of which the 500 is controlling one axis. A PLC controls the process. In this process the 500 is feeding material at two different speeds. It will either be feeding the material at a high speed or a low speed. The PLC will signal the 500 when to stop the material. The material must be able to switch between the high speed and low speed without stopping. When the PLC stops the material feed, the material is cut and the 500 must return to the point it started to begin the process again. The point at which the material feed begins is the home position for starting motion. On power up, the 500 should move to this position. The PLC controls motion. Three different types of material will require different high and low velocities.

Required Motion

This application will require continuous moves. It will also require a preset move to return to the starting location. On power up a homing move will be made. The PLC controls continuous moves. The moves will always start with the high velocity and will then be toggled between high and low by the PLC. The PLC will stop the operation when it is complete. Different move velocities will be required for the different material. The distance of the preset return move will be determined by the 500 based on the distance it traveled during the continuous move.

What Will Cause Motion and When?

The PLC will control the entire process. The 500 will home itself after the initial power up. The continuous move will then begin by the PLC selecting one of three materials. Each material will have its own high and low velocities. The continuous move will continue until the PLC issues a stop. Then the 500 will make a preset move back to the start and the PLC will select one of the sequences again.

What Configuration Commands Are Required?	<input type="checkbox"/>	SSJ1	Sequence Scan mode
	<input type="checkbox"/>	XQ1	Interrupted Sequence Scan mode
	<input type="checkbox"/>	INnA	Trigger input, n is the input number
	<input type="checkbox"/>	INnB	Sequence select input
	<input type="checkbox"/>	OUTnC	Sequence in progress output
	<input type="checkbox"/>	OSB	Back up to home switch
	<input type="checkbox"/>	OSG	Select final home approach direction
	<input type="checkbox"/>	OSH	Select edge of home switch which 500 stops on
	<input type="checkbox"/>	GHV	Select homing velocity
	<input type="checkbox"/>	GHF	Final homing approach velocity
	<input type="checkbox"/>	MC	Sets the 500 in the Continuous mode
	<input type="checkbox"/>	MN	Sets the 500 in the Preset mode

Implementation The 500 will be in Sequence Scan mode. The PLC will select 1 of the 3 sequences. It will then control the high speed or low speed with one line. One input will also be configured as a stop input. It will set the velocity level input to a 1 for high speed and a 0 for low speed. When a stop is issued, the PLC will wait for the 500 to signal that it is back to the start position and then select the next sequence with the sequence select inputs. An output is set up to turn on when a sequence is in progress. When it is off, the PLC can select a new sequence. The new sequence will not be run until the 500 has made the preset move back to the start location. Returning to the start location ends the sequence and allows the next sequence to be scanned.

SEQUENCE #100	Command	Description
>	XD100	Begins definition of sequence #100
	MPI	Incremental move mode
	MC	Sets 500 to Continuous mode
	CMR5000	Sets resolution to 5000 steps/rev
	GHV2	Initial go home velocity of 2 rps
	GHF.3	Final go home velocity of 0.3 rps
	OSB1	Back up to home switch
	OSH1	Stop on the CCW edge of the switch
	OSG0	Final approach direction is CW
	GH	Begin go home move
	IN1B	Sets up inputs 1 and 2 as sequence select inputs
	IN2B	
	IN3A	High/Low velocity input (trigger input)
	IN4A	Stop input
	OUT1C	Sets the output of a sequence-in-progress output
	XQ1	Enables the interrupted run mode
	SSJ1	Place the 500 in Continuous Scan mode
	A100	Set the acceleration to 100 rps ²
>	XT	Ends definition of sequence #100
SEQUENCE #1	Command	Description
>	XD1	Begins definition of sequence #1
	PZ	Zeroes (clears) the position counter
	MC	Sets the 500 in continuous mode
	V18.5	Sets the high velocity to 18.5 rps
	H+	Sets the direction to CW
	MPP	Enables MPP mode
	G	Begins motion
	REPEAT	Loops until stopped
	IF (INXXXXXX1)	Checks for high or low velocity
	V3	Sets the velocity to the high velocity
	ELSE	If not low velocity then
	V18.5	Sets the velocity high
	UNTIL (INXXXXXX1)	Checks for the stop signal
	NG	Ends the Profiling mode
	STOP	Stops the move
	VAR1=POS	Sets variable one equal to the position counter
	D (VAR1)	Loads the distance with the distance traveled in the Preset move
	H	Sets the direction CCW
	MN	Sets the 500 in the preset mode
	G	Returns to the starting point
>	XT	Ends definition of sequence #1

```
SEQUENCE #2 > XD2
              PZ
              MC
              V12
              H+
              MPP
              G
              REPEAT IF(INXXXXXXX1)
              V1
              ELSE
              V12
              UNTIL(INXXXXXXX1)
              NG
              STOP
              VAR1=POS
              D(VAR1)
              H
              MN
              G
> XT
```

```
SEQUENCE #3 > XD3
              PZ
              MC
              V9
              H+
              MPP
              G
              REPEAT
              IF(INXXXXXXX1)
              V.5
              ELSE
              V9
              UNTIL(INXXXXXXX1)
              NG
              STOP
              VAR1=POS
              D(VAR1)
              H
              MN
              G
> XT
```

