

 **Helpful Hint:**

Variable assignments can be made in sequences or in Immediate Terminal mode. If a variable with 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.

Assigning Variables to Constants

Step ①

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

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.

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 ZX in incremental mode
> MN	Enters the normal mode
> FSI0	Places a ZXF in the indexer mode. <i>Required only with Following option.</i>
> XE1	Erases sequence #1
> XD1	Defines sequence #1
> A10	Sets the Acceleration
> AD10	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

Assign numbers to the variables **VAR1** and **VAR2**

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

Step ③

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

<u>Command</u>	<u>Description</u>
> 1V	*V05.00000
> 1D	*D+000025000

Step ④

Now change the values of the variables as follows.

<u>Command</u>	<u>Description</u>
> VAR1=75000	The distance parameter is 75,000
> VAR2=10	The velocity parameter is assigned 10

Step ⑤

Repeat steps ③ and ④.

Entering Variables Via RS-232C

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

Step ①

Define the following sequence:

 **Helpful Hint:**

To enter the variable, you must precede the number with an exclamation point (!).

<u>Command</u>	<u>Description</u>
> 1XE1	Erases sequence #1
> 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	Sequence stops and waits for a # to be entered into variable 5
> 1CR	Inserts a carriage return
> 1LF	Inserts a line feed
> L (VAR5)	The loop count is loaded with the number of parts to make
> D25000	Distance is 25000 steps
> G	Initiates motion
> N	Ends the loop

> 1"FINISHED_WITH_PARTS_RUN	Indicates that the run is finished
> 1CR	Inserts a carriage return
> 1LF	Inserts a line feed
> XT	Ends sequence definition

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.

Step ② Turn on 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 ZX has the ability to perform simple math functions with its variables (add, subtract, multiply, and divide). When performing these math functions, the ZX *truncates* the decimal after five places to the right. The following sequence of steps illustrates the math capabilities of the ZX.

Step ①

Addition:

<u>Command</u>	<u>Response</u>
> VAR1=5	
> VAR23=1000.565	
> VAR11=VAR1+VAR23	
> VAR23=10+VAR23	
> VAR1=VAR1+5	
> 1VAR1	*+000000000000010.00000
> 1VAR11	*+000000000001005.56500
> 1VAR23	*+00000000001010.56500

Step ②

Subtraction:

<u>Command</u>	<u>Response</u>
> VAR3=10	
> VAR20=15.5	
> VAR3=VAR3-VAR20	
> VAR19=VAR20-VAR3	
> 1VAR3	*-00000000000005.50000
> 1VAR20	*+00000000000015.50000
> 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	*+000000000000155.00000
> 1VAR20	*+00000000000015.50000
> 1VAR19	*+000000000001534.50000

Step ④

Division:

<u>Command</u>	<u>Response</u>
----------------	-----------------

```

> VAR3=10
> VAR20=15.5
> VAR3=VAR3/VAR20
> VAR30=75
> VAR19=VAR30/VAR3
> 1VAR3 *+0000000000000000.64516
> 1VAR20 *+0000000000000015.50000
> 1VAR30 *+0000000000000075.00000
> 1VAR19 *+000000000000116.25023

```

Complex Branching and Looping

The ZX 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 ZX 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.

Helpful Hint:
The motor moves a total of 10,000 steps

Command	Description
> PS	Pauses 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

Helpful Hint:
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 it to stop [Stop (s) or Kill (κ)].

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

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 ZX.

- 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 *ZX Indexer/Drive 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 (**FL00111X0**) and modify it within sequences to mark where the program has gone or to indicate any special state so a conditional statement can be made.

```

> WHILE(FL00111XXXX)
> D100
> G

```

Input State > **NWHILE**
 An example of this statement's use (**IN1111100010**) 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
```

Conditional Looping The ZX supports two conditional looping structures—**REPEAT/UNTIL** and **WHILE**.

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

Step ①

<u>Command</u>	<u>Description</u>
> 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 1110 input condition occurs or VAR5 is > 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	Ends sequence definition

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. When using a **REPEAT/UNTIL** loop, all of the commands between **REPEAT** and **UNTIL** are executed at least once before the loop reaches the evaluation statement in the **UNTIL** line. This is not true with the **WHILE** loop.

WHILE

With the **WHILE** command, all commands between **WHILE** and **NWHILE** are repeated **WHILE** condition evaluates true. The evaluation of the condition is done at the end of any current move in progress.

Step ①

<u>Command</u>	<u>Description</u>
> 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	Last command in the WHILE loop
> 1"DONE_LOOPING	Indicates that the loop is done

```

> 1CR           Inserts a carriage return
> 1LF           Inserts a line feed
> XT            Ends sequence definition

```

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 the **VAR5** counter to 10. If the input pattern is not **XXX111Ø**, the loop will not be run.

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 Flag

The **IF** command can check for error conditions. If an error exists, a conditional command may 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 [ZX Indexer/Drive Software Reference Guide](#) for more information on the **ER** evaluation with the **IF** command.

Command	Description
> XE1Ø	Erases sequence #10
> XD1Ø	Defines sequence #10—when a fault occurs, sequence #10 will run—the sequence is defined with XFK1Ø (below)
IF (ER1)	If hardware CCW limit switch is reached, perform following commands:
1 "CCW_LIMIT_HIT	Display the error message
NIF	Ends IF statement
IF (ERX1)	If hardware CW limit switch is reached, perform 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 Flag

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.

 **Helpful Hint:**
Refer to the [ZX Indexer/Drive Software Reference Guide](#) for a description of **SFL**.

Command	Description
> PS	Waits for the indexer to receive a Continue (C) command before executing the next command
> SFL1Ø1Ø	Sets user flag bits 7 & 5 and clears bits 6 & 4, remaining bits not altered
•	
•	
•	
> IF (FL1Ø1Ø)	If user flag bits 5 & 7 are set, and bits 6 & 4 are clear, perform following commands
> A1Ø	Sets acceleration to 10 rps ²
> V5	Sets velocity to 5 rps
> D25ØØØ	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**, **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 ***ZX Indexer/Drive Software Reference Guide***.

<u>Command</u>	<u>Description</u>
> 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.
1 "DONE	Ends message saying done
NIF	Ends IF statement
> 1XT	Ends sequence definition

Use **DIN** or the inputs 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. The limitation of how many comparisons can be made is limited to a command line length of 80 characters.

Command	Description
> XE8	Erases sequence 8
> XD8	Defines sequence 8
VAR5=15	Variable 5 = 15
IF (VAR5>10_AND_VAR4=20)	If Variable 5 is greater 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 sequence definition

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 ZX will process 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
- FP**—Sets Distance Points from a following encoder with Motion Profiling mode (ZXF only, refer to *Chapter 5 Following*)

While the ZX is in Motion Profiling mode, you can execute any command while a move is being made. When the ZX 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 ZX then waits at the trigger command (**TR**) until the condition is true, at which point it changes the velocity. When the ZX encounters the **NG** command, it will not receive any additional commands until the 50,000-step move is completed. If the Trigger condition is not met during the 50,000 step move, the move is completed and the program waits on the trigger condition before finishing the sequence.

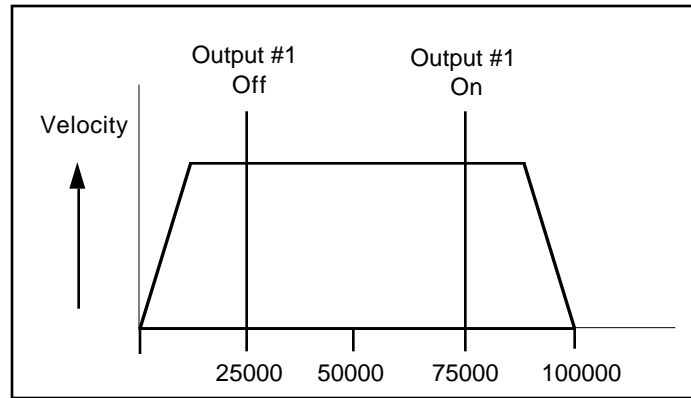
Changes Based on Distance

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.

Helpful Hint:
The following sequence example executes the profile shown in the figure below.

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



In Incremental mode, commands are processed until the **DP** command is reached. The **ZX** pauses at the **DP** command until the motor moves 25,000 steps. The **ZX** 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 **ZX** pauses until the motor moves 50,000 steps. The **ZX** then turns output #1 off. In this example, if the **ZX** 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 ZX to Continuous mode
MPP	Sets the ZX 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 **XR1**. 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 **ZX** is still in Motion Profiling mode. Enter the **NG** command to exit the mode.

Stopping Motion (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>
> IN1A	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>
> XD1	Begins the definition of sequence #1
V5	Sets velocity to 5 rps
A100	Sets acceleration to 100 rps ²
MC	Sets the ZX to Continuous mode
MPP	Sets the ZX 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
> XT	Ends sequence #1 definition

Step ③ Issue the Display Parameters (**DR**) command. The ZX 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 ZX from stopping without finishing the sequence that it is currently executing, a software switch has been provided that will cause the ZX 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 ZX 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 (**SSJ1**) and the **STOP** command, you must enable **OSI1** to prevent the Sequence Scan mode from being disabled after a stop. Under normal conditions, Sequence Scan mode is aborted when the ZX encounters a **STOP** command. If the **STOP** command is issued from a stop input or from within a sequence, Sequence Scan mode will be aborted. Usually, you will not want to abort the mode. The **SSH1** command will allow the ZX to complete the execution of the current sequence from the point at which the **STOP** was issued. The following example illustrates such a sequence.

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

<u>Command</u>	<u>Description</u>
> IN1A	Configures input #1 as a trigger input
> IN2B	Configures input #2 as a sequence select input
> OUT1A	Enables output #1 as a programmable output

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

Enable the Save Buffer on Stop with the **SSH1** command.

Enable the Preserve Sequence Scan mode with the **OSI1** command.

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

<u>Command</u>	<u>Description</u>
> XD1	Begins the definition of sequence #1
V5	Sets velocity to 5 rps
A100	Sets acceleration to 100 rps ²
MC	Sets the ZX to Continuous mode
MPP	Sets the ZX 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
O1	Turn on output #1
> XT	Ends sequence #1 definition

Step ④ Execute the sequence by activating input #2. Stop the move at any time by activating input

#1. Output #1 will be activated after the stop.

Step ⑤

The ZX 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

Motion Profiling mode provides flexibility in the complexity of the ZX control 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 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 ZX 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
1DP200000	Sets distance point to 200,000
V5	Changes velocity to 2 rps
BCPP45	Changes gain to 45% of maximum
NG	Exits ZX 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 position counter to 0
1MC	Sets the ZX to Continuous mode
1MPP	Sets ZX 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 ZX's position is > 400,000
STOP	Halts command processing
NG	Exits ZX 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 ZX

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

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

Programmable Inputs and Outputs

The ZX has a flexible scheme for defining I/O in a way that is suitable for most applications. There are 7 programmable inputs (**I1** - **I7** 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 ZX. This section explains some of the functions that the inputs and outputs can perform and how to use thumbwheels for an interface with the ZX.

Output Functions

You can turn the programmable outputs (**O1** - **O4**) on and off with the Output (**O**) and Immediate Output (**IO**) commands. Outputs **O1** - **O4** 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 *ZX Indexer/Drive Software Reference Guide* for descriptions of the available functions.

Helpful Hint:
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 of 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

Helpful Hint:
To see how the outputs are currently defined, type **1OUT**.

Command	Description
> PS	Pauses command execution until the indexer receives a Continue (C) command
> MN	Sets unit to Normal mode
> A10	Sets acceleration to 10 rps ²
> V5	Sets velocity to 5 rps
> D25000	Sets distance to 25,000 steps
> OUT1A	Sets O1 as a programmable output
> OUT2A	Sets O2 as a programmable output
> OUT3B	Sets O3 as a Moving/Not Moving output
> o10	Turns O1 on and O2 off
> G	Executes the move
> O01	Turn O1 off and O1 on
> C	Initiates command execution to resume

O1 and **O2** are programmable outputs and **O3** as a Moving/Not Moving output. Before the motor moves 25,000 steps, **O1** turns on and **O2** turns off. These outputs remain in this state until the move is completed, then **O1** turns off and **O2** turns on. While the motor is moving, **O3** remains on.

Pulse Output Half Axis

The ZX can be programmed a second half axis. The output half axis can control distance and constant speed for another drive/motor system. A second output can control the distance. The following commands set up a second axis and make it move.

Command	Description
> OUT1U	Configures the output to be a pulse output
> OUT2A	The output will be used to control direction
> PUL1000,2	The output will send 1000 pulses at a rate of 2 ms
> PUL	This command 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	N:	Data
B:	Sequence Select	O:	No Function assigned
C:	Kill	P:	Memory Lock
D:	Stop	Q:	Registration input (I7 Only)
E:	Command Enable	R:	Reset
F:	Pause/Continue	S:	Go Home
G:	Go	T:	Position Zero
H:	Direction	U:	User Fault
I:	Synchronization	V:	Data Valid
J:	Jog+ (CW)	W:	Data Sign
K:	Jog- (CCW)	X:	Increase Following ratio
L:	Jog Speed Select	Y:	Decrease Following ratio
M:	Not Used	Z:	No Function assigned

To designate each input pin to a particular function, use the Set Input Functions (**IN**) command. To current input definitions, type **1IN**. To see the inputs' states, use **1IS**. 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 **I7** can be assigned as a registration input.)

Switches

Triggers

This section covers ZX triggers and sequence scanning with inputs.

You can use the Trigger (**TR**) command to pause a sequence of buffered commands until one or more inputs come to a preferred state. **I1 - I7** are set at the factory (default setting) to function as trigger inputs.

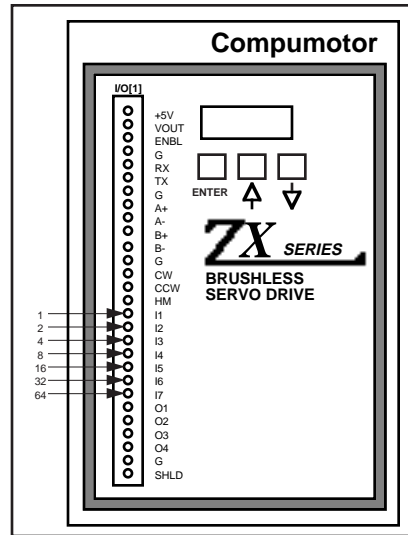
<u>Command</u>	<u>Description</u>
> IN1D	Sets I1 as Stop(S) input
> IN2A	Sets I2 as trigger input 1
> IN3A	Sets I3 as trigger input 2
> IN4A	Sets I4 as trigger input 3
> MC	Sets the unit to Continuous mode
> MPP	Enters the Motion Profiling mode
> A15	Sets acceleration to 15 rps ²
> AD15	Sets acceleration to 15 rps ²
> TR1	Waits for trigger input 1 (I2) to be on
> G	Executes a go (Go) command
> L	Loops infinitely
> V5	Sets velocity to 5 rps
> TRX01	Waits for trigger input 2 (I3) to be off and trigger input 3 (I4) to be on
> V1	Sets velocity to 1 rps
> TRX10	Waits for trigger input 2 (I3) to be on and trigger input 3 (I4) to be off
> N	Ends the loop

This program configures **I1** as a stop input and **I2** as a trigger (start) input. **I3** and **I4** are programmable trigger inputs. If you activate the start input (**I2**), the motor will move. Since the ZX 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 toggles continuously between 1 - 5 rps as the trigger statements come true and will not stop until the stop input is activated. If you activate **I1** during the operation of the ZX, the indexer immediately stops the operation and clears the command buffer (assuming that **SSH** is set to \emptyset).

Sequence Select & Sequence Scan

Inputs can be defined as sequence select inputs. This allows you to execute sequences defined via RS-232C by activating the sequence select inputs. *Binary weighting is fixed on the ZX. Input 3 (I3) will always select Sequence #4.*

Helpful Hint:
This figure shows the binary weights of the ZX's inputs when all 7 inputs are configured as sequence select inputs.



Dedicated Binary Weight of ZX Inputs

One possible input configuration and binary weighting is given below.

Input	Function	Binary Weighting
I1	Sequence Select	1
I2	Sequence Select	2
I3	Sequence Select	4
I4	Sequence Select	8
I5	Sequence Select	16
I6	Trigger	—
I7	Stop	—

Input Configuration/Binary Weighting Example

If the inputs are configured as in Table 4-4, sequence #6 will be executed by activating inputs #2 and #3. Sequence #19 will be executed by activating inputs #1, #2, and #5. If input I1 is set-up as a trigger input, then only even numbered sequences can be used.

To execute sequences, the ZX must be placed in Sequence Scan mode. In this mode, the ZX will continuously scan the input lines and execute the sequence selected by the active sequence select lines. The **SSJ** command is used to enable/disable the sequence scan mode. When **SSJ1** is entered, the Sequence Scan mode is enabled. To disable the mode, enter **SSJ0**.

Once enabled (**SSJ1**), the ZX will run the sequence number that the active sequence select inputs and their respective binary weightings represent. After executing and completing the selected sequence, the ZX 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 ZX to immediately executed 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.

Helpful Hint:
The ZX executes these commands on power up—the ZX can read up to 99 sequences from the sequence-select inputs.

Command	Definition
> XE100	Erases sequence #100
> XD100	Defines sequence #100
SSJ1	Executes sequences via PLC input
SN20	Sets scan time to 20 msec
XQ1	Sets ZX 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
OUT1C     Sets Output 1 as sequence-in-progress output
LD3       Disables the limits (not necessary if limits are connected)
> XT      Ends the sequence definition

```

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.

Switch #	Input
Switch #1	I1
Switch #2	I2
Switch #4	I3
Switch #8	I4
Switch #16	I5
Switch #32	I6
Switch #64	I7
GND	

Applying Switches to Inputs

Step ⑥

To execute sequences, cycle power to the ZX. The system will execute sequence #100.

Step ⑦

You can now execute sequences by closing the corresponding switches.

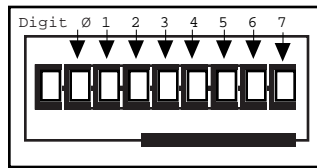
- Close switch 1 to execute sequence #1
- Close switch 2 to execute sequence #2
- Close switches 1 & 2 to execute sequence #3
- Close switches 1, 2, 6, and 7 to execute sequence #99

Thumbwheel Interface

With the ZX, you can use up to 16 digits of thumbwheels. The ZX uses 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 following section assumes that you are using Compumotor's TM8 module with the ZX.* The ZX commands and format that allow for thumbwheel data entry are:

Command	Description
DRDxyz	Read distance via thumbwheels
VRDxyz	Read velocity via thumbwheels
LRDxyz	Read loop count via thumbwheels
TRDxyz	Read time delay via thumbwheels
VARDn,xyz	Read variables via thumbwheels
XRDxy	Read sequence count via thumbwheels
FRDxyz	Read following ratio via thumbwheels

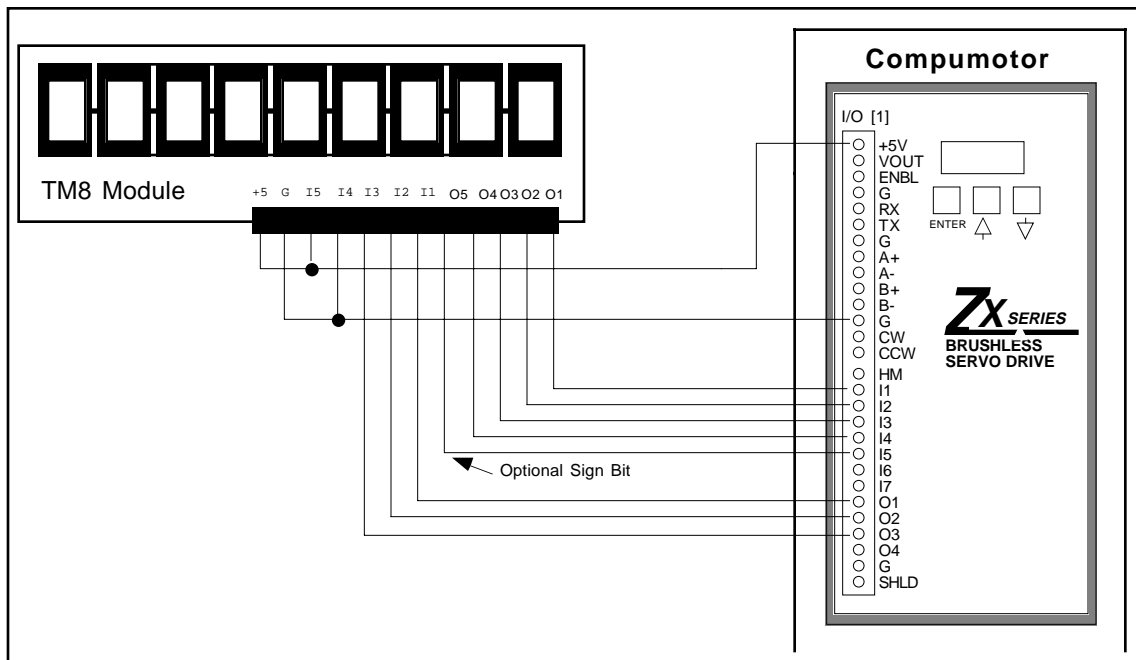
Variables **x** and **y** may range from 0 to 7. They represent the thumbwheel digits to be requested. The left-most digit of the TM8 Module is $x=0$ and the right-most is $y=7$. The other digits are sequentially represented between 0 and 7.



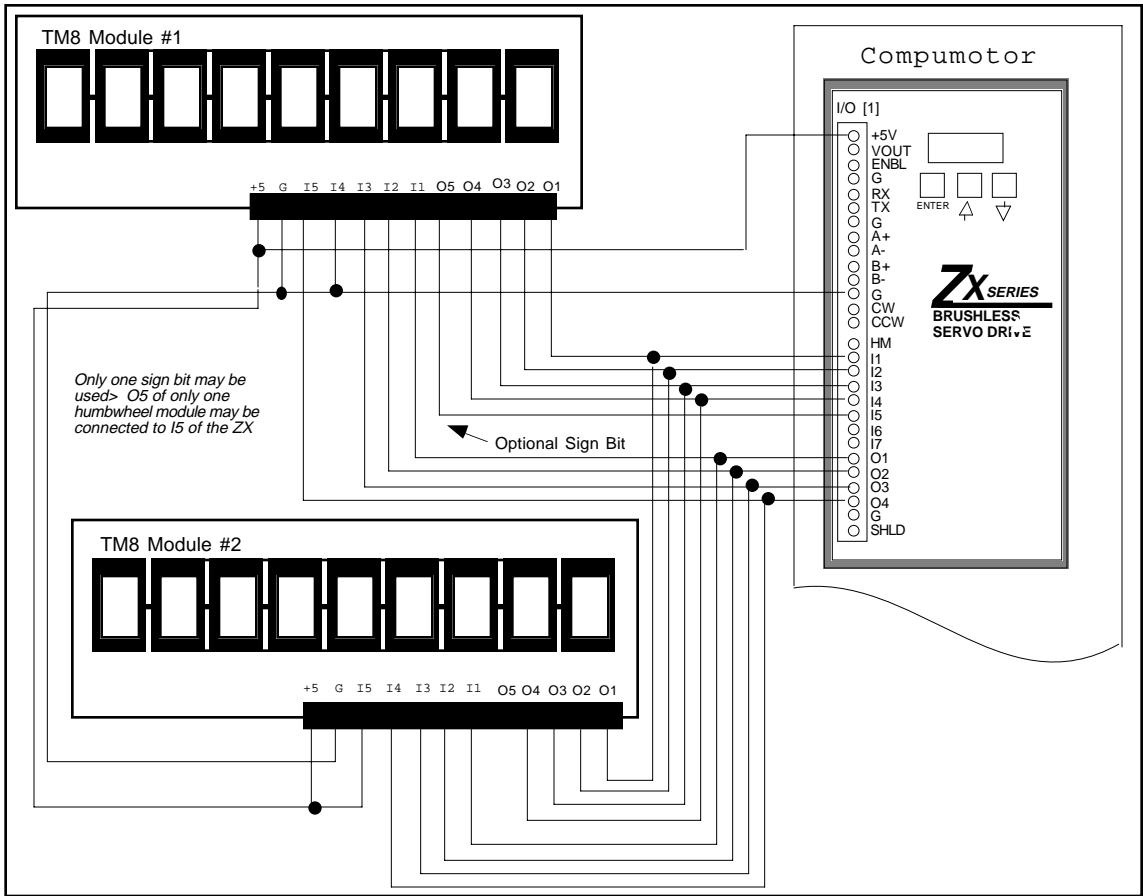
- To Request One Digit $x = y =$ the desired digit number
- To Request All Digits $x = 0$ and $y = 7$ (or do not use the *x, y, and z* fields)
- To Request Several Digits $0 < x < y < 7$

The **z** field scales the thumbwheel read value by 10^z . When reading digits from the thumbwheel, the least significant digits will be filled first. The **z** field allows you to position the decimal point in these commands where needed. *All of the fields (x,y, and z) must be used or none of the fields must be used.* Refer to the ZX Indexer/Drive Software Reference Guide for more information. To use Compumotor's TM8 Module, follow the procedures below.

Step 1 If you use one TM8 Module, wire your module to the ZX as shown in the figure below. If you use two TM8 Modules (the maximum allowed per **each** ZX).



Wiring 1 Thumbwheel Unit (TM8) to the ZX



Wiring 2 Thumbwheel Units (TM8) to the ZX

Step ②

Configure your ZX as follows:

<u>Command</u>	<u>Description</u>
> OUT1J	O1 configured as a strobe
> OUT2J	O2 configured as a strobe
> OUT3J	O3 configured as a strobe
> IN1N	I1 configured as a data input
> IN2N	I2 configured as a data input
> IN3N	I3 configured as a data input
> IN4N	I4 configured as a data input
> IN5W	I5 configured as a sign input (optional)
> INL0	Inputs configured active low
> STR50	Data strobe time of 50 ms per digit read. If one TM8 Module is used, you should now be ready to read in thumbwheel data. If two TM8 Modules are used, enter the additional set-up commands. <i>The minimum strobe time recommended for the TM8 module is 10 ms.</i>
> OUT4A	O4 configured as a programmable output
> OUTL1	Outputs set active high
> O1	Set output 4 high—this enables TM8 Module #1

Step ③

Set the thumbwheel digits on your TM8 Module to **+12345678**. If two TM8 Modules are used, set the second module to **-87654321**. To verify that you have wired your TM8 Module(s) correctly and configured your ZX 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 the response shown is not received, retry step 1.

If you are using two TM8 Modules, enter the following commands:

> O0	O4 becomes 0V—this disables Module #1 and enables Module #2.
> DRD	Request distance data from all 8 thumbwheel digits
> 1D	Displays the distance read— D+87654321 <i>The sign is positive.</i> Only one sign digit may be used when two TM8 Modules are used. If the response shown is not received, retry step 1.
> O1	Reenables the first TM8 Module.

Thumbwheel Examples

The following thumbwheel examples are to clarify the TM8 Module usage. If using two thumbwheel modules, enter the following commands:

Helpful Hint:

Note how the 10^2 field can position the decimal point where needed.

<u>Command</u>	<u>Description</u>
> DRD050	Request distance data from the first six thumbwheels
> 1D	D+123456
> DRD052	Same as above, but scaled by 10^2
> 1D	D+12345600
> VRD670	Request the velocity data from the last two TM8 Module thumbwheels
> 1V	V.0078
> VRD673	As above, but now scaled by 10^3
> 1V	Displays the velocity read— V7.8000

Helpful Hint:

The following commands show how you can use the TM8 module to enter variables.

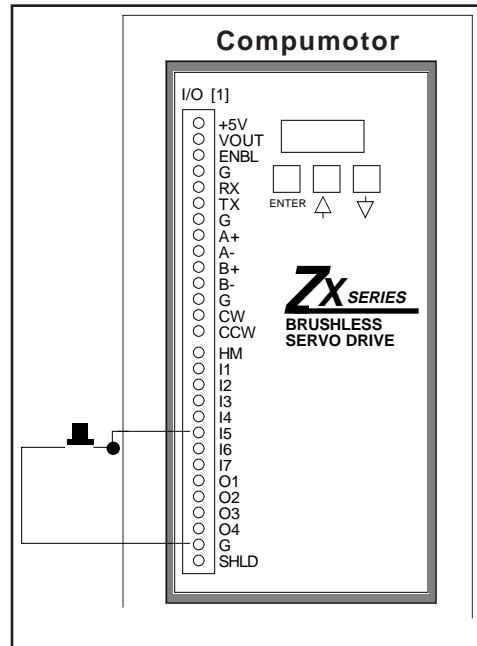
<u>Command</u>	<u>Description</u>
> 1VARD12,450	This requests data from the TM8 Module's 5th and 6th digits
> 1VAR12	VAR12=*0000000000.00056
> 1VARD12,459	As above, but scaled by 10^9
> 1VAR12	VAR12=*0000056000.00000

Selecting Sequences with the Thumbwheel Module

The following example shows how the ZX is typically used with thumbwheels. In the example, only three sequences are entered. As many as 100 sequences may be defined and up to 99 may be executed with the Compumotor Thumbwheel Module. Sequence #100 is automatically executed during power-up or reset. Refer to the Reset (Z) command in the [ZX Indexer/Drive Software Reference Guide](#).

Ensure that the thumbwheel module is properly installed. Wire input 5 with a normally open switch as shown in the figure below. The switch shown in this configuration is a sequence-start switch.

Helpful Hint:
Position Mode Tuning
Procedure



Sequence Start Configuration

Step ①

Define a power-up sequence. Set inputs I1 - I4 as data inputs and I5 as a data valid input. Enter the following program.

Command	Description
> XE100	Erases Sequence #100
> XD100	Begins definition of Sequence #100
IN1N	Sets I1 as a data input
IN2N	Sets I2 as a data input
IN3N	Sets I3 as a data input
IN4N	Sets I4 as a data input
IN5V	Sets I5 as a data valid input
INL0	Sets 0V as active level
STR10	Sets strobe time of 10 ms per digit
OUT1J	Sets O1 as a strobe output
OUT2J	Sets O2 as a strobe output
OUT3J	Sets O3 as a strobe output
L	Start a continuous loop
XRD01	Run the sequence displayed on thumbwheel digits 0 & 1
N	Ends loop
> XT	Ends definition of Sequence #100

Step ②

Define any sequences that your application may need.

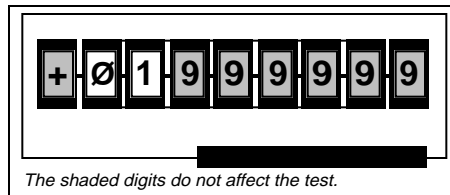
Command	Description
> XE1	Erases Sequence #1
> XD1	Begins definition of Sequence #1
MN	Sets mode to Normal
A25	Sets acceleration to 25 rps ²
AD25	Sets deceleration to 25 rps ²
V5	Sets velocity to 5 rps
D25000	Sets distance to 25000 steps
G	Executes the move (Go)
> XT	Ends definition of Sequence #1
> XE2	Erases Sequence #2
> XD2	Begins definition of Sequence #2
MN	Sets mode to Normal

A25	Sets acceleration to 25 rps ²
AD25	Sets deceleration to 25 rps ²
V5	Sets velocity to 5 rps
D10000	Sets distance to 10000 steps
G	Executes the move (Go)
> XT	Ends definition of Sequence #2
> XE99	Erases Sequence #99
> XD99	Begins definition of Sequence #99
MN	Sets mode to Normal
A25	Sets acceleration to 25 rps ²
AD25	Sets deceleration to 25 rps ²
V5	Sets velocity to 5 rps
D-35000	Sets distance to -35000 steps
G	Executes the move (Go)
> XT	Ends definition of Sequence #99

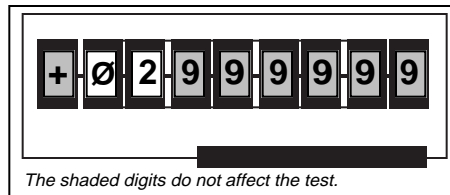
Step ③ Reset the ZX. This will prepare the power-up sequence (sequence #100) to be executed when you apply power to the ZX.

<u>Command</u>	<u>Description</u>
> z	Resets the ZX

Step ④ Set thumbwheel digits 0 and 1 to 01 and push start to move the servo motor 35,000 steps CW.

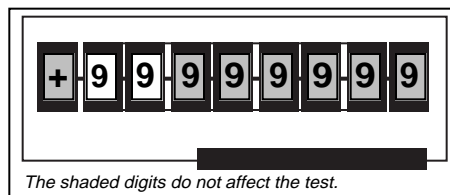


Step ⑤ Set thumbwheel digits 0 and 1 to 02 and push start to move the servo motor 10,000 steps CW.



Step ⑥ Set thumbwheel digits 0 and 1 to 99 and push start to move the servo motor 10,000 steps CCW.

Helpful Hint:
If you select an invalid or unprogrammed sequence with the thumbwheels, no motion will occur (nothing will happen).



PLC Operation

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

Interfacing with a PLC

In many applications it is desirable to interface to a PLC. The ZX 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 ZX and the motion segment of a process.

Data Read with a PLC

As in the thumbwheel case, the PLC can be used to enter data for sequence select (**XR**D), distance (**DR**D), velocity (**VR**D), loop count (**LR**D), time delay (**TR**D), and variable data (**VAR**D). Detailed explanations of these commands are given here.

To read data from the PLC, the ZX's inputs #1 - #4 must be configured as data inputs with an active low level (**INL**Ø).

If a sign digit is required, ZX input 5 should be configured as a sign input. Configure outputs #1 - #3 as data strobe outputs.

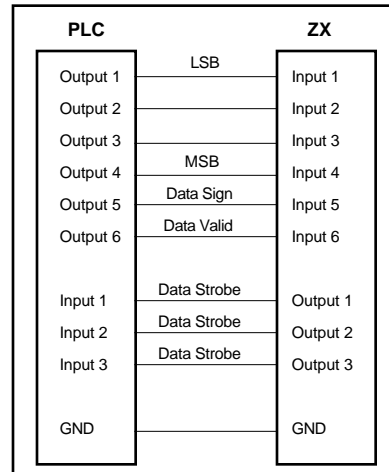
When the ZX executes a data read command, it cycles its outputs and reads BCD data. The maximum rate that the ZX will cycle through the output levels is set by the Strobe Output Delay Time (**STR**) command. The ZX synchronously cycles through the states at the **STR** time if no data valid line is used. If a data valid line is used, the ZX will maintain its current state until the data valid input is activated. This allows the PLC to control the ZX's data strobe rate.

ZX Outputs			ZX Inputs					
01	02	03	Data In (Active Low)	I1	I2	I3	I4	I5
low	low	low	MSD (Digit 1)	lsb			msb	± sign bit
high	low	low	Digit 2	"			"	"
low	high	low	Digit 3	"			"	"
high	high	low	Digit 4	"			"	"
low	low	high	Digit 5	"			"	"
high	low	high	Digit 6	"			"	"
low	high	high	Digit 7	"			"	"
high	high	high	LSD (Digit 8)	"			"	"

Output Voltage Levels

Helpful Hint:

This figure shows a possible PLC-to-ZX connection that would allow the ZX to input data from the PLC. In this configuration, a data valid line (16) controls the ZX's strobe rate and a data sign line is used.



PLC/ZX Connection

If the ZX runs a Read Distance Via Parallel I/O (**DR**D) command, the following events must occur to transfer distance data from the PLC to the ZX.

Helpful Hint:

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

- ① The ZX executes a **DR**D command and places its outputs #1 - #3 at ØV.
- ② The PLC places and holds a BCD digit at the ZX's inputs #1 - #4. (This value will be the most significant distance digit). The PLC places a sign value at ZX input #5. (**This sign bit must be the same for each digit read.**)
- ③ When the data is valid, the PLC should activate the data valid line. The ZX will read the digit and sign values.
- ④ The PLC should deactivate the data valid line.
- ⑤ After reading a data valid, the ZX will place its output #1 at high voltage (5 - 24V) and outputs

#2 and #3 at ØV.

- ⑥ The PLC must place and hold its second BCD digit at the ZX's inputs #1 - #4. The PLC should place the same sign value as that given for input #5.
- ⑦ When the data is valid, the PLC must activate the data valid line. The ZX will read this second BCD digit as the second significant distance digit.
- ⑧ The PLC must deactivate the data valid line.

Helpful Hint:
The **XRD** command does not use the z scaling field.

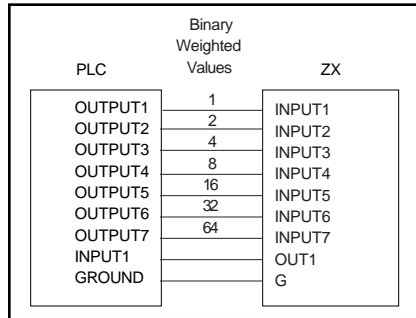
If data scaling is desired, refer to the *Data* section earlier in this chapter. The additional x, y, and z fields may be used. The x field must equal Ø, the y field must equal 7, and z must be greater than or equal to Ø and less than 10. For example, a **DRDØ71** command will cycle the outputs through the sequence and read the data one digit at a time for eight digits. The data however will be multiplied by 10¹ when it is stored in the data's register.

Sequence Select With a PLC

The PLC can execute sequences through two different methods. First, the sequences may be selected by using the inputs as sequence select inputs. In this case, the input lines are binary weighted. The ZX must be in Sequence Scan mode (**SSJ1**) and may either operate in Interrupted mode (**XQ1**) or Continuous Scan mode (**XQØ**).

The PLC activates the lines that will execute the desired sequence.

Helpful Hint:
It may be desirable to have the ZX indicate to the PLC when it has completed a sequence or to indicate to the PLC when it should select another sequence. A programmable output can be used for this handshake to the PLC.



PLC Connection

The PLC may also use the **XRD** command to select and run ZX sequences. Only ZX output #1 should be configured as a strobe output. Inputs #1 - #4 must be configured as data inputs (**INLØ**). A data valid line may also be used.

If the ZX executes the **XRD** command when its I/O is configured as described above, it will set its output #1 to a voltage low for the time defined by the **STR** command or until the data valid line is active. It will then read in the most significant BCD digit of a sequence run command. The ZX will then set the voltage level high for output #7 (again, for the appropriate period) and read in the least significant digit of sequence run data. The ZX will then execute the sequence defined by the PLC if it exits.

ZX Output	ZX Inputs			
Ø1	I1	I2	I3	I4
Low	LSB	MS BCD Digit	MSB	
High	LSB	LS BCD Digit	MSB	

Strobe and Input Sequence

Miscellaneous PLC Control

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

Front Panel Operation

You can use the ZX's front panel pushbuttons and display to execute sequences. Refer to the *Alphanumeric Display and Pushbuttons* section for information on how to use this feature.

Remote Panel Operation

There is an optional remote panel that can be used by the ZX and ZX-F to input data, display messages and prompt the operator for appropriate actions. The RP240 is a panel that communicates over RS-232C with the ZX. It has a 2 line 40 character display, a numeric keypad, and programmable function keys. Contact your distributor or Compumotor for more information on the RP240.

Host Computer Operation

You may use a host computer to execute motion programs via the RS-232C serial interface. The host computer can run interactively from a basic or C program. In this case, the high-level program controls the ZX and acts as an interface to the user (note the following program example).

```
1 ' ZX.BAS PROGRAM
2 '
3 ' *****
5 ' * This program controls the RS232 Communication line to execute 2 different moves *
6 ' * using the ZX *
*****
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 ZX
122 PRINT #1,Z$;
124 Q$ = INPUT$(8,1)
900 ' *****
902 ' * Lines 1000-1060 sends a move down to the first ZX. Computer waits for the Line *
903 ' * Feed from the ZX indicating that the motor has finished its move. Computer *
904 ' * will not command second ZX to move until the first move is completed. *
*****
1000 MOVE1$ = "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$(22,1) ' Read echoes from ZX.
1020 LF$ = INPUT$(1,1) ' Wait for line feed from ZX
1040 IF LF$ <> 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 ' *****
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 ZX also prints a line feed *
1906 ' * after finishing the second move. *
1907 ' * As soon as the computer receives the line feed from ZX, the program will go back*
1908 ' * to the first move. *
*****
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 ZX 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.

> 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	Send string to host
> 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

ZX Application Examples

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

Application Development Approach

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:

Examples

- ① Identify and define the types of motion required in your application.
 - 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

Once you have identified the motion, determine the gains that will be required for the ZX to perform the moves with the proper performance. To do this, enter the speed, acceleration, and decelerations required for the move and determine the gains necessary for the required performance. To tune the drive, refer to the *Servo Tuning* section in this chapter. After determining the gains, save them with the **SV** command.

Examples

- 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.
 - Input states causing motion to occur; triggers
 - Sequence execution
 - RP240 Remote Panel
 - Thumbwheel inputs
 - Power up sequence
 - PLC
 - Variable conditions
 - Time delays

Examples

- ③ Determine what configuration commands are necessary to set the ZX in the proper state for execution of the motion programs.
 - 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 determine what set-up commands are required, place them in the power-up sequence, sequence #100. In some applications, the ZX 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, cycle power, issue a Reset (**Z**) command or simply run sequence #100 with an **XR100** command to configure the ZX 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 (thumbwheels or a PLC and sequences) is selected via the inputs. 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 ZX 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.

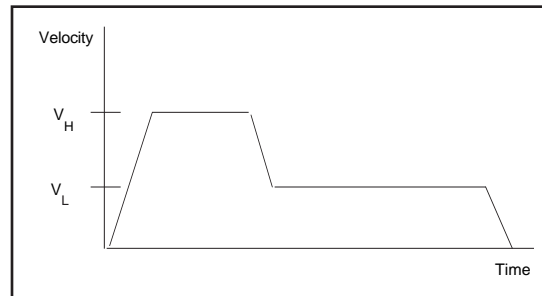
Application Example #1

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 move 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, other grind moves can be made.

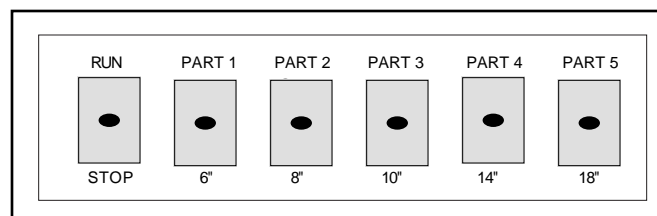
 **Helpful Hint:**
Motion profile of the grinding application



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 ZX 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.



Front Panel of User Interface

What

SSJ1 Sequence Scan mode

Configuration Commands Are Required?	<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 ZX stops on
	<input type="checkbox"/> GHV	Select homing velocity
	<input type="checkbox"/> GHF	Final homing approach velocity

Execution Part selection switches select sequences that contain the move instructions for specific parts. 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 stop. 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 command is 5000 steps/rev (**CMR5000**). This value is stored in battery backed RAM and need not be in the program. 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	<u>Command</u>	<u>Description</u>
	XD100	Begins definition of sequence #100
	MPI	Incremental move mode
	MN	Normal preset mode
	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 inputs 1-5 as sequence select inputs
	IN2B	
	IN3B	
	IN4B	
	IN5B	
	IN6A	Sets up input 6 as a trigger
	SSJ1	Place the ZX 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	<u>Command</u>	<u>Description</u>
	XD1	Begins definition of sequence #1
	V4	Defines the fast velocity
	D240000	Total move distance
	REPEAT	Repeat loop to make parts continuously
	MPP	Enters 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 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 (INXXXXXXXXX0)	Continue the repeat loop until the run/stop is put in the stop location
	TR1	Wait until the run/stop switch is 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	<u>Command</u>	<u>Description</u>
	XD2	Begin definition of sequence #2
	V4	Defines the fast velocity
	D240000	Total move distance
	REPEAT	Repeat loop to make parts continuously
	MPP	Enters Profiling mode—velocity can be changed on the fly
	G	Initiates motion
	DP160000	After 160,000 steps decelerate to the grind velocity
	V . 5	Grind velocity
	NG	Ends 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 (INXXXXXXXXX0)	Continue the repeat loop until the run/stop switch is put in the stop location
	TR1	Wait until the run/stop switch is in the run location then finish this sequence and scan the inputs to select another part sequence
	XT	Ends Sequence definition

Sequence #4	<u>Command</u>	<u>Description</u>
	XD4	Begin definition of sequence #4
	V4	Defines the fast velocity
	D240000	Total move distance
	REPEAT	Repeat loop to make parts continuously
	MPP	Enters Profiling mode—velocity can be changed on the fly

	G	Initiates motion
	DP140000	After 140,000 steps decelerate to the grind velocity
	V.5	Grind velocity
	NG	Ends 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 (INXXXXXXXXX0)	Continue the repeat loop until the run/stop switch is put in the stop location
	TR1	Wait until the run/stop switch is in the run location then finish this sequence and scan the inputs to select another part sequence
	XT	Ends Sequence definition
Sequence #8	XD16	Begin definition of sequence #16
	V4	Defines the fast velocity
	D240000	Total move distance
	REPEAT	Repeat loop to make parts continuously
	MPP	Enters Profiling mode—velocity can be changed on the fly
	G	Initiates motion
	DP100000	After 100,000 steps decelerate to the grind velocity
	V.5	Grind velocity
	NG	Ends 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 (INXXXXXXXXX0)	Continue the repeat loop until the run/stop switch is put in the stop location
	TR1	Wait until the run/stop switch is in the run location then finish this sequence and scan the inputs to select another part sequence
	XT	Ends Sequence definition
Sequence #16	XD2	Begin definition of sequence #2
	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
	DP80000	After 80,000 steps decelerate to the grind velocity
	V.5	Grind velocity
	NG	Ends 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 (INXXXXXXXXX0)	Continue the repeat loop until the run/stop switch is put in the stop location
	TR1	Wait until the run/stop switch is in the run location then finish this sequence and scan the inputs to select another part sequence
	XT	Ends Sequence definition

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

Application Example #2

If example #1 is modified slightly, the same motion can be accomplished using a different user interface. In 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.

Types of Motion Required

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 ZX 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 ZX stops on
- GHV Select homing velocity
- GHF Final homing approach velocity

Execution

The process runs identically to the previous process, 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 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 ZX output.

Required Sequences

The resolution set by the Configure Motor Resolution command is 5000 steps/rev (CMR5000).

Sequence #100	Command	Description
	XD100	Begins definition of sequence #100
	MPI	Incremental move mode
	MN	Normal preset mode
	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 inputs 1 - 4 as data inputs
	IN2N	
	IN3N	
	IN4N	
	IN5A	Sets up input 5 as a trigger input
	OUT1J	Sets Outputs 1-3 as strobe outputs
	OUT2J	
	OUT3J	
	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
	VARD1	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 (INXXXXXX0)	Continue the repeat loop until the run/stop switch is put 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 pauses at the trigger command until Run mode is selected. The thumbwheels will be read again so a new grind move can be run. This is more flexible and allows many different grind moves to be executed.

Application Example #3

This application is a process of which the ZX is controlling one axis. A PLC controls the process. In this process the ZX 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 ZX 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 ZX 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 ZX should move to this position. The PLC controls motion. Three different types of material will require different high and low velocities.

Type of Motion Required

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 ZX based on the distance it traveled during

the continuous move.

What Will Cause Motion and When?

The PLC will control the entire process. The ZX 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 ZX will make a preset move back to the start and the PLC will select one of the sequences again.

What Configuration Commands Are Required?

- SSJ1** Sequence Scan mode
- XQ1** Interrupted Sequence Scan mode
- INnA** Trigger input, n is the input number
- INnB** Sequence select input
- OUTnC** Sequence in progress output
- OSB** Back up to home switch
- OSG** Select final home approach direction
- OSH** Select edge of home switch which ZX stops on
- GHV** Select homing velocity
- GHF** Final homing approach velocity
- MC** Sets the ZX in the Continuous mode
- MN** Sets the ZX in the Preset mode

Execution

The ZX will be in Sequence Scan mode. The PLC selects one of the three sequences. High speed or low speed is controlled with one line. One input will also be configured as a stop input. It will set the velocity level input to **1** for high speed and **0** for low speed. When a stop is issued, the PLC will wait for the ZX 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 run until the ZX 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

<u>Command</u>	<u>Description</u>
xdl00	Begins definition of sequence #100
MPI	Incremental move mode
MC	Sets ZX to Continuous mode
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 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 ZX in Continuous Scan mode
A100	Set the acceleration to 100 rps ²
xt	Ends definition of sequence #100

Sequence #1

<u>Command</u>	<u>Description</u>
xdl1	Begins definition of sequence #1
PZ	Zeroes (clears) the position counter
MC	Sets the ZX 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 (INXXXXX1)	Checks for high or low velocity
V3	Sets the velocity to the low velocity
ELSE	If not low velocity then
V18.5	Sets the velocity high
NIF	End of IF statement
UNTIL (INXXXXXXXX1)	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 ZX in the preset mode
G	Returns to the starting point
xt	Ends definition of sequence #1

Sequence #2

xdl1	Begins definition of sequence #1
PZ	Zeroes (clears) the position counter
MC	Sets the ZX in continuous mode
V12	Sets the high velocity to 12 rps
H+	Sets the direction to CW
MPP	Enables MPP mode
G	Begins motion
REPEAT	Loops until stopped
IF (INXXXXX1)	Checks for high or low velocity
V1	Sets the velocity to the low velocity
ELSE	If not low velocity then
V12	Sets the velocity high
NIF	End of IF statement
UNTIL (INXXXXXXXX1)	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

Sequence #3

H	Sets the direction CCW
MN	Sets the ZX in the preset mode
G	Returns to the starting point
XT	Ends definition of sequence #1
XD1	Begins definition of sequence #1
PZ	Zeroes (clears) the position counter
MC	Sets the ZX in continuous mode
V9	Sets the high velocity to 9 rps
H+	Sets the direction to CW
MPP	Enables MPP mode
G	Begins motion
REPEAT	Loops until stopped
IF (INXXXXXXXX1)	Checks for high or low velocity
V.5	Sets the velocity to the low velocity
ELSE	If not low velocity then
V9	Sets the velocity high
NIF	End of IF statement
UNTIL (INXXXXXXXX1)	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 ZX in the preset mode
G	Returns to the starting point
XT	Ends definition of sequence #1