

## Chapter 4. APPLICATION DESIGN

### Chapter Objectives

The information in this chapter will enable you to:

- Recognize and understand important considerations that must be addressed before you implement your application
- Understand the capabilities of the system
- Customize the system to meet your requirements
- Use sample applications to help you develop your application

### Application Considerations

Successful application of a rotary motor system also requires careful consideration of the following important points:

- Move Profiles
- Mechanical Resonance
- Ringing or Overshoot
- Move Times (calculated vs. actual)
- Positional Accuracy and Repeatability

### *Move Profiles*

A motion profile represents the velocity of the motor during a period of time in which the motor changes position. The type of motion profile that you need depends upon the motion control requirement that you specify. The basic types of motion profiles are described below. All of the profiles discussed in this chapter can be performed with the SD/IFX system.

The IFX allows you to define your own velocity profile or use pre-programmed profiles that optimize the performance of the rotary motor.

### Triangular and Trapezoidal Profiles

For constant acceleration indexing systems, velocity, acceleration, and distance parameters are defined before the system can execute a preset move. The value of these parameters determines the type of motion profile as either triangular or trapezoidal. A triangular profile results when the velocity and acceleration are set such that the defined velocity is not attained before the motor travels half of the specified distance. This results from either a relatively low acceleration, a relatively high velocity, or both. For example, if you set the acceleration at  $1 \text{ rev/sec}^2$ , velocity at  $2 \text{ revs/sec}$ , and distance at 800 steps (2 revolutions in the 400 step/rev mode)—a triangular motion profile is the result. This occurs because the motor shaft will have reached the defined velocity of 2 rps and traveled half of the defined distance. The motion profile for this move is shown in Figure 4-1.

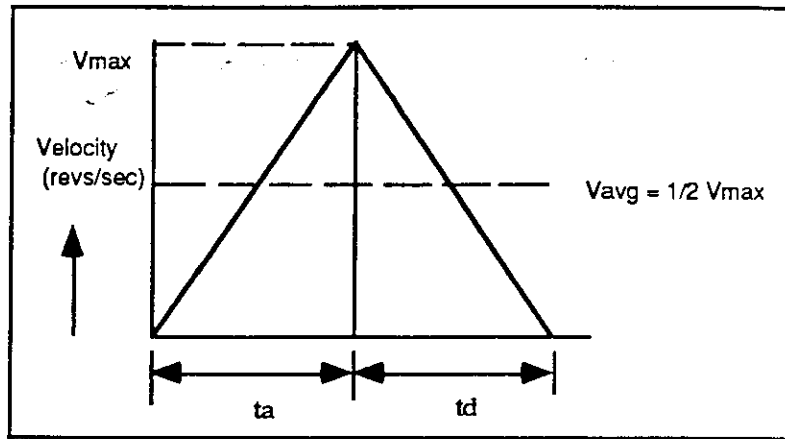


Figure 4-1. Triangular Profile

A trapezoidal move profile results when the defined velocity is attained before the motor shaft has moved half of the specified distance. A trapezoidal move may occur if you specify a low velocity with a high acceleration or a long distance. For example, if you set the acceleration at 10 revs/sec<sup>2</sup>, the velocity at 1 rev/sec, and the distance at 1,600 steps (4 revolutions in the 400 step/rev mode), the resulting motion profile will resemble the profile shown in Figure 4-2.

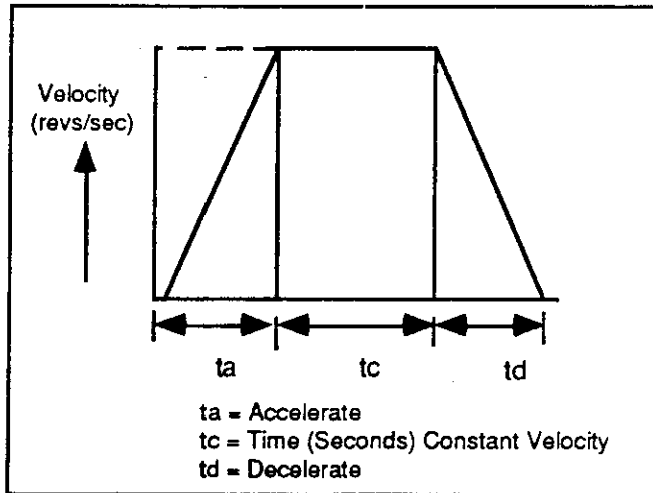


Figure 4-2. Trapezoidal Profile

## Custom Profiles

You can define a custom profile with the Rate Multiplier in Velocity Streaming Mode (**RM**) command. With this command, the IFX makes an instantaneous change to the specified velocity. Sending **RM** commands to the IFX in rapid succession provides smoother motion by virtue of S-curve acceleration (see Figure 4-3). Testing and modification may be required to establish the correct sequence of **RM** commands.

**NOTE:** To perform custom profiling with the **RM** command, you must first set the IFX to the Velocity Profiling Mode with the **Q1** command. Use the **Q0** command to exit the velocity profiling mode.

| Command       | Description                   |
|---------------|-------------------------------|
| <b>Q1</b>     | Enter Velocity Profiling mode |
| <b>RM0032</b> | Accelerate to 0.25 rps        |
| <b>RM0064</b> | Accelerate to 0.5 rps         |
| <b>RM00C9</b> | Accelerate to 1.0 rps         |
| <b>RM0064</b> | Decelerate to 0.5 rps         |
| <b>RM0032</b> | Decelerate to 0.25 rps        |
| <b>Q0</b>     | Exit velocity profiling mode  |

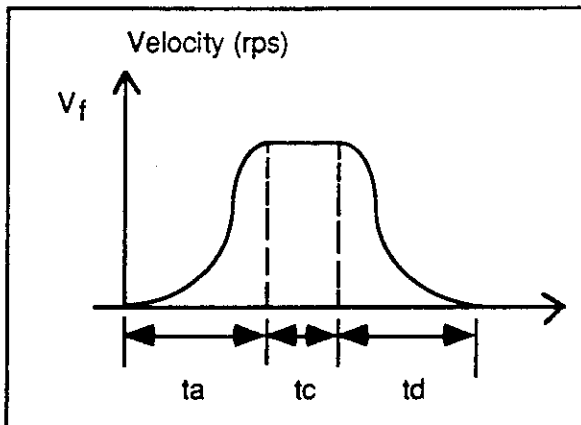


Figure 4-3. S-Curve Profile

**RM** commands can be combined with time delays in a sequence buffer to create very complex move profiles. The maximum buffer update rate is 7 msec (@9600 baud) and the buffer capacity is fixed by memory size to 2,000 bytes.

***Mechanical  
Resonance***

Resonance exists in all stepper motors. If you operate in a full-step or half-step mode, the motor may stall at speeds below 250 full-steps/sec.

One way to reduce resonance at low speeds (below 250 full-steps/sec) is to add inertia to the motor shaft. This may be accomplished by putting a drill chuck on the back shaft. *Note that this technique is applicable only to double-shaft motors with the shaft extending from both ends of the motor.* In extreme cases, you may also need a viscous damper to balance the load. One of the manufacturers of viscous dampers is listed below:

Ferrofluidics Corporation  
40 Simon Street  
Nashua, NH 03061  
(603) 883-9800

***Ringling or  
Overshoot***

The motor's springiness, along with its mass, form an underdamped resonant system that rings in response to acceleration transients (such as at the end of a move). Ringing at the end of a move prolongs settling time. The actual settling time of a system depends on the motor's stiffness, the mass of the load, and any frictional forces that may be present. By adding a little friction, you can decrease the motor's settling time.

***Move Times:  
Calculated vs  
Actual***

You can calculate the time it takes to complete a move by using the acceleration, velocity, and distance values that you define. However, you should not assume that this value is the actual move time. There is calculation delay and motor settling time that make your move longer. After you issue the Go (G) command, the IFX indexer can take up to 5 milliseconds to calculate the move before the motor starts moving. You should also expect some time for the motor to settle into position.

### **Positional Accuracy vs. Repeatability**

In rotary positioning systems, some applications require high absolute accuracy. Others require repeatability. You should clearly define and distinguish these two concepts when you address the issue of system performance.

If the positioning system is taken to a fixed place and the coordinates of that point are recorded, the only concern is how well the system repeats when you command it to go back to the same point. For many systems, what is meant by accuracy is really repeatability. Repeatability measures how accurately you can repeat moves to the same position.

Accuracy on the other hand, is the error in finding a random position. For example, suppose the job is to measure the size of an object. The size of the object is determined by moving the positioning system to a point on the object and using the move distance required to get there as the measurement value. In this situation, basic system accuracy is important. The system accuracy must be better than the tolerance on the measurement that is desired.

The accuracy of the SD/IFX system is effected by the following errors:

- Uni-directional Repeatability. The error measured by repeated moves to the same point from different distances in the same direction.
- Hysteresis. The backlash of the motor when it changes direction (caused by magnetic non-linearity)

For more information on accuracy and repeatability, consult the technical data section of The Compumotor Catalog

### **Open-Loop Accuracy**

Open-loop accuracy is dependent on the construction of the motor.

### **Closed-Loop Accuracy**

Closed-loop accuracy is determined by the resolution and accuracy of the encoder. When enabled, the IFX attempts to position the motor within the specified deadband from the encoder. Typically, this means the motor will be positioned to within one encoder step. To do this satisfactorily, the encoder must have a lower resolution than the motor. If the step size of the motor is equal to or greater than the step size of the encoder, the motor will be unable to maintain the position and may become unstable. In a system with adequate motor-to-encoder resolution, the motor is able to maintain accuracy within one encoder step.

## Modes of Operation

### Normal (Preset) Mode

A preset move is a move distance that you specify (in motor steps). You can select preset moves by putting the IFX into normal mode using the Mode Normal (**MN**) command. Preset moves allow you to position the motor in relation to the motor's previous stopped position (incremental moves) or in relation to a defined zero reference position (absolute moves). You can select incremental moves by using the Mode Position Incremental (**MPI**) command. You can select absolute moves using the Mode Position Absolute (**MPA**) command.

### Incremental Preset Mode Moves

When you are in the Incremental mode (**MPI**), a preset move moves the motor the specified distance from its starting position. For example, to move the motor 2 revolutions, you must specify a preset move with a distance of +800 steps, assuming a resolution of 400 steps/rev. Every time the indexer executes this move, the motor moves 2 revolutions from its resting position. You can specify the direction of the move in one command. You specify the direction by using the optional sign (**D+800** or **D-800**), or you can define it separately with the Set Direction (**H**) command (**H+** or **H-**). Whenever you do not specify the direction, the unit defaults to the positive (CW) direction.

### SAMPLE INCREMENTAL MODE MOVES

The moves shown below are incremental moves. The distance specified is relative to the motor's current position. This is the default (power-up) positioning mode. You can invoke this mode with the Mode Position Incremental (**MPI**) command.

| <u>Command</u> | <u>Description</u>                           |
|----------------|--|
| <b>MPI</b>     | Sets unit to Incremental Position Mode       |
| <b>A2</b>      | Sets acceleration to 2 revs/sec <sup>2</sup> |
| <b>V5</b>      | Sets velocity to 5 rps                       |
| <b>D800</b>    | Sets distance to 800 steps                   |
| <b>G</b>       | Executes the move (Go)                       |
| <b>G</b>       | Repeats the move (Go)                        |
| <b>H</b>       | Reverses direction of next move              |
| <b>G</b>       | Executes the move (Go)                       |

The motor moves two CW revolutions and stops. It then moves two more CW revolutions and stops. Then the motor changes direction and moves two CCW revolutions.

| <u>Command</u> | <u>Description</u>   |
|----------------|--|
| <b>D-800</b>   | Changes the distance to 800 steps in the opposite (CCW) direction. |
| <b>G</b>       | Executes the move (Go)   |

The motor returns to its original starting position.

| <u>Command</u> | <u>Description</u>  |
|----------------|---|
| H              | Toggles the motor direction of the next move, but maintains existing acceleration, velocity, and distance parameters. |
| G              | Executes the same move profile as the previous move, but in the opposite direction(Go)                                |

The motor moves 800 steps in the positive (CW) direction.

| <u>Command</u> | <u>Description</u>                         |
|----------------|--|
| D800           | Sets distance to 800 steps.                |
| G              | Executes 800-step move (Go)                |
| T2.5           | Waits 2.5 seconds after finishing the move |
| D1000          | Sets distance to 1,000 steps               |
| G              | Executes 1,000-step move (Go)              |

As soon as you enter the T2.5 command, the time delay starts. If you wish to load all the commands before executing them, you may use the Pause (PS) and Continue (C) commands.

| <u>Command</u> | <u>Description</u>   |
|----------------|--|
| PS             | Pauses execution until the indexer receives a Continue (C) command |
| G              | Executes the 1,000-step move (Go)                                  |
| T3             | Waits 3 seconds after the move                                     |
| G              | Moves 1,000 steps  |
| C              | Starts G T3 G commands   |

### Absolute Preset Mode Moves

A preset move in the absolute mode (MPA) moves the motor the distance that you specify (in motor steps) from the absolute zero position. You can set the absolute position to zero with the Position Zero (PZ) command or by cycling the power to the drive. The absolute zero position is initially the power-up position.

The direction of an absolute preset move depends upon the motor position at the beginning of the move and the position you command it to move to. For example, if the motor is at absolute position +800, and you instruct the motor to move to position +400, the motor will move in the negative direction (CCW) a distance of 400 steps to reach the absolute position of +400.

The IFX powers up in Incremental mode. When you issue the Mode Position Absolute (MPA) command, it sets the mode to absolute. When you issue the Mode Position incremental (MPI) command the unit switches to Incremental mode. The IFX retains the absolute position, even while the unit is in the Incremental mode. You can use the Position Report (PR) command to read the absolute position.

SAMPLE  
ABSOLUTE  
MODE MOVES

The moves shown below are absolute mode (MPA) moves. The distance specified is relative to the IFX's absolute zero position.

| <u>Command</u> | <u>Description</u>                           |
|----------------|--|
| MN             | Sets the IFX in Preset Move mode             |
| MPA            | Sets the IFX to the Absolute Position mode   |
| PZ             | Sets the current absolute position to zero   |
| A5             | Sets acceleration to 5 revs/sec <sup>2</sup> |
| V3             | Sets velocity to 3 rps                       |
| D1000          | Sets move to absolute position 1,000         |
| G              | Executes move to position 1,000              |

| <u>Command</u> | <u>Description</u>  |
|----------------|---|
| D4000          | Moves the motor to absolute position 4,000. (Since the motor was already at position 1,000, it moves 3,000 additional steps in the same direction.) |
| G              | Executes the move (Go)  |
| D0             | Moves the motor to absolute position 0. (Since the motor is at absolute position 4,000, the motor moves 4,000 steps in the opposite direction.)     |
| G              | Executes the move (Go)  |

**Continuous  
Mode**

The Continuous Mode (MC) is useful for applications that require constant movement of the load, when the motor must stop after a period of time has elapsed (rather than after a fixed distance), or when the motor must be synchronized to external events such as trigger input signals. You can manipulate the motor movement with either buffered or immediate commands. After you issue the G command, buffered commands are executed in the order in which they were programmed. Immediate commands are used to instantaneously change the motor's acceleration and velocity when the motor is already in continuous motion.

SAMPLE  
CONTINUOUS  
MODE MOVES

The following Continuous Mode (MC) commands will accelerate or decelerate the motor to a specified velocity and continue at that velocity until another command is issued or until a limit is encountered.

| <u>Command</u> | <u>Description</u>   |
|----------------|--|
| LD0            | Enables CW and CCW limits  |
| MC             | Sets all moves to the Continuous mode  |
| A10            | Sets acceleration to 10 rev/sec <sup>2</sup>   |
| V5             | Sets final velocity to 5 rps   |
| G              | Executes the move (moves at 5 rps until the IFX receives a Stop (s) or Kill (x) command) |

While the motor is moving, enter the following command to change motor velocity *on-the-fly*:

| <u>Command</u> | <u>Description</u>  |
|----------------|---|
| V4             | Changes velocity to 4 rps when the indexer receives another G command |
| G              | Changes velocity to 4 rps   |

| <u>Command</u> | <u>Description</u>  |
|----------------|---|
| A 2            | Changes acceleration to 2 revs/sec <sup>2</sup>   |
| V 2            | Changes velocity to 2 rps when the indexer receives the next G command  |
| G              | Changes velocity to 2 rps using acceleration value of 2 revs/sec <sup>2</sup> (to decelerate to the new velocity) |

| <u>Command</u> | <u>Description</u>   |
|----------------|--|
| V 0            | Decelerates the motor to a stop when the indexer receives the next G command |
| G              | Decelerates the motor to a stop  |

### **Open Loop Operation**

This section contains examples of open loop moves that you can perform with the SD/IFX system. Open-loop moves *do not* use encoders to provide position information.

#### **Sample Open Loop Moves**

| <u>Command</u> | <u>Description</u>                           |
|----------------|--|
| LD 0           | <i>Enables CW and CCW Limits</i>             |
| FSB 0          | Sets IFX to motor step mode (default)        |
| MN             | Performs a single preset move                |
| A 2            | Sets acceleration to 2 revs/sec <sup>2</sup> |
| V 5            | Sets velocity to 5 rps                       |
| D 8 0 0        | Sets distance to 800 steps                   |
| G              | Executes the move (Go)                       |

The motor moves 800 steps in the positive (CW) direction.

| <u>Command</u> | <u>Description</u>                            |
|----------------|---|
| MPA            | Sets IFX to Absolute Position mode            |
| A 2            | Sets acceleration to 2 revs/sec <sup>2</sup>  |
| V 1 0          | Sets velocity to 10 rev/sec                   |
| P Z            | Sets the current position as home             |
| D 4 0 0        | Sets move to absolute position 400            |
| G              | Executes the move (Go)                        |
| D 8 0 0        | Sets move to absolute position 800            |
| G              | Moves the motor to absolute position 800 (Go) |

The motor moves 1 CW revolution and stops. It then moves another CW revolution and stops.

### Closed Loop Operation

This section contains examples of closed-loop moves that you can perform with the SD/IFX system. Closed-loop moves use incremental encoders to provide position correction signals. Motor position may be adjusted to reach the desired position.

The IFX indexer is capable of interfacing with an Incremental encoder with quadrature (single-ended or differential) TTL Square wave outputs. The encoder may be used as a means of creating a closed-loop system or as an independent means of verifying motor position. The functions that are added to a system when an encoder is used are listed below:

- Encoder referenced positioning
- Encoder position servoing
- Motor stall detection
- Higher accuracy homing function
- Multi-Axis stop on stall

To implement the closed-loop functions, you must connect an incremental encoder to the IFX. The encoder outputs must be 3 - 5 VDC. When you use encoders with single-ended outputs, do not connect channels A-, B-, and Z- to the SD/IFX motherboard's TB5 connector.

### Selecting Encoder Resolution Values

The number of encoder steps that the SD/IFX system recognizes is equal to four times the number of encoder *lines*. For example, a 25 line encoder mounted directly on the motor will generate 100 encoder steps/rev. A minimum of three motor steps per encoder step is required for successful operation of the position maintenance function. Ratios above four motor steps per encoder step ensure stability of the position maintenance servo function. Positional resolution is determined by encoder resolution.

If you install a reducer between the motor shaft and the encoder, the number of encoder steps received by the indexer is equivalent to the number of encoder steps divided by the encoder gear ratio.

For example, using a 400 step/rev motor, a 25-line encoder, and a 10:1 reducer, the ratio of motor revolutions to encoder steps would be changed as described in Table 4-1 below (this example assumes the use of a drive in the 400 step/rev mode).

| Parameter  | 1:1 Ratio           | 10:1 Ratio           |
|--|---------------------|----------------------|
| Number of encoder steps recognized by the SD/IFX (per motor rev) | 100                 | 10                   |
| Required ratio of motor revs to encoder steps                    | 3:1                 | 3:1                  |
| Motor-to-encoder step ratio                                      | 4:1<br>(sufficient) | 40:1<br>(sufficient) |

Table 4-1. Motor-to-Encoder Ratio

**Setting  
Encoder  
Resolution**

You can use the encoder to achieve greater accuracy or stall detection. Typically, the incremental encoder improves the overall accuracy of your system.

Since there are many different encoders with different resolutions, you must specify for the SD/IFX what type of encoder you have connected to the system. Using the following example as a guide, you can specify the encoder's resolution to the indexer with the Encoder Resolution (ER) command.

| <u>Command</u> | <u>Description</u>  |
|----------------|---|
| ER100          | Tells the SD/IFX that the encoder has a resolution of 100 steps/rev after quadrature (25 lines) |

**Encoder  
Step Mode**

The indexer can perform moves in either motor steps or encoder steps. In Motor Step mode (FSB0), the distance command (D) defines moves in motor steps. In Encoder Step mode (FSB1), the distance command defines moves in encoder steps.

You must set up the indexer for the correct encoder resolution. The Encoder Resolution (ER) command is used to define the encoder resolution.

The sample move below assumes the use of an encoder with an encoder-to-motor step ratio of 4:1.

| <u>Command</u> | <u>Description</u>  |
|----------------|---|
| MN             | Sets mode to normal   |
| ER100          | Sets up encoder where 100 encoder pulses (25 lines) are produced per 1 revolution of the motor. |
| FSB1           | Sets move to encoder step mode  |
| A10            | Set acceleration to 10 revs/sec <sup>2</sup>  |
| V5             | Set velocity to 5 revs/sec  |
| D500           | Set distance to 500 encoder steps   |
| G              | Executes the move (Go)  |

The motor will turn in the CW direction until 500 encoder pulses (5 revolutions) are received.

If this move does not work, refer to Chapter 7, Maintenance and Troubleshooting.

**Position  
Maintenance**

This Position Maintenance (FSC) command enables and disables the position maintenance function.

You must enable Position Maintenance (FSC1) to activate closed loop servoing and ensure that encoder step moves are positioned to the exact encoder step commanded. Enabling position maintenance will cause the indexer to servo the motor until the correct encoder position is achieved. This occurs at the end of a move (if the final position is incorrect) or any time the indexer senses a change in position while the motor is at zero velocity. You must have an encoder connected, and the indexer set in Encoder Step mode (FSB1) in order to enable position maintenance.

*NOTE: For the position maintenance function to work successfully, the ratio of motor steps to encoder steps must be greater than 3:1 (3 motor steps to each encoder step, post quadrature).*

Position maintenance will be disabled (turned OFF) automatically if a stall is detected (refer to the FSH command in Chapter 5), or if the encoder is forced outside of the deadband window.

The sample move below assumes the use of an encoder with an encoder-to-motor step ratio of 4:1.

**Example**

| <u>Command</u> | <u>Description</u>  |
|----------------|---|
| ER100          | Sets up encoder where 100 encoder pulses (25 lines) are produced per 1 revolution of the motor. |
| FSB1           | Sets move to encoder step mode  |
| FSC1           | Enables Position Maintenance  |
| A10            | Set acceleration to 10 revs/sec <sup>2</sup>  |
| V5             | Set velocity to 5 revs/sec  |
| D500           | Set distance to 500 encoder steps   |
| G              | Executes the move (Go)  |

The motor will turn in the CW direction until 500 encoder pulses (5 revolutions) are received. The Position Maintenance function instructs the IFX to servo the motor until the correct encoder position is achieved.

If this move does not work, refer to Chapter 7, Maintenance and Troubleshooting.

**Stop-On-Stall**

You can enable the Stop-on-Stall function with the **FSD1** command. The move will terminate, without any delay, as soon as a stall is detected. This function works either in Motor Step or Encoder Step mode, but can work only if an encoder is used.

**CAUTION**

Disabling the Stop-on-Stall function with the **FSD0** command will allow the IFX to finish the move regardless of a stall detection, even if the load is jammed.

The **FSD1** command is valid only if the Enable Stall Detection (**FSH1**) command has been issued.

The Stop-on-Stall function depends on the setting for backlash (set with the **DW** command) to give optimum operation. The factory default setting for backlash is 0 motor steps. If you mount the encoder on the motor, you should leave this parameter set to zero for the most accurate response.

| <u>Command</u> | <u>Description</u>                        |
|----------------|---|
| <b>DW10</b>    | Sets backlash value to 10 steps.          |
| <b>ER100</b>   | Sets encoder resolution to 100 steps/rev. |
| <b>FSH1</b>    | Enables stall detect.                     |
| <b>FSD1</b>    | Enables stop on stall.                    |

Stall detection does not occur until the error exceeds the dead band backlash (set with the **DW** command). Consequently, if the indexer does not see an encoder pulse after moving the motor 10 steps, the IFX will detect a stall and stop the motor immediately.

### Determining Backlash

You can measure the actual backlash with the following procedure. The idea is to move in one direction, stop, and make a series of one-step moves in the opposite direction. No change in encoder position occurs while the indexer takes up the backlash. The number of motor steps counted before any encoder counts are received is the measure of the backlash. Note that this is mainly magnetic backlash (or *hysteresis*) if the encoder is mounted on the motor.

Move the motor in one direction and clear the position counters with the following commands.

| <u>Command</u> | <u>Description</u>                            |
|----------------|---|
| <b>FSBØ</b>    | Sets indexer to motor step mode               |
| <b>MN</b>      | Sets Mode normal                              |
| <b>A1Ø</b>     | Sets acceleration to 10 revs/sec <sup>2</sup> |
| <b>V1</b>      | Sets velocity to 1 rps                        |
| <b>D-1ØØØ</b>  | Sets distance to 1000 CCW steps               |
| <b>G</b>       | Executes the move (Go)                        |
| <b>PZ</b>      | Sets absolute position counter to zero        |

Now execute a series of one-step moves and report both motor and encoder position each time:

| <u>Command</u> | <u>Description</u>   |
|----------------|--|
| <b>H</b>       | Changes direction to CCW   |
| <b>D1</b>      | Sets distance to 1 step  |
| <b>PS</b>      | Pauses execution of the following commands until the indexer receives the Continue (C) command                             |
| <b>L</b>       | Causes the sequence to repeat  |
| <b>G</b>       | Executes the move (Go)   |
| <b>1LF</b>     | Sends a line feed  |
| <b>1PR</b>     | Reports absolute motor position in motor steps   |
| <b>T1.Ø</b>    | Pauses the motor for 1.0 seconds   |
| <b>1LF</b>     | Sends a line feed  |
| <b>1PX</b>     | Reports the number of encoder pulses (steps) that the encoder has received since the Position Zero (PZ) command was issued |
| <b>T1.0</b>    | Pauses the motor for 1 second  |
| <b>N</b>       | Ends the loop  |
| <b>C</b>       | Clears pause and executes the move   |

While this command string is running, you may compare the position reports from the Position Report (PR) command and the Report Absolute Encoder Position (PX) command. The 1PR report represents the number of motor steps traveled, and the 1PX report represents the number of encoder steps traveled. When the response from the 1PX command changes from 0 to 1, note the response from the 1PR command. This 1PR command report is the actual backlash of your system.

To use the IFX's stall detection function, set the Dead Band Window (DW) command equal to the value of the backlash determined above and then Enable Stall Detect (FSH1).

**Output On Stall**

You can select the Output-on-Stall function with the Turn on Output #1 on Stall Detect (**FSE**) command. This is useful for signaling other components in your system that a stall has occurred.

If you enter the **FSE1** command, the IFX programmable Output #1 goes on (current flows) when a stall is detected and remains on until a new move begins. This command is valid only if the Stall Detection has been enabled (**FSH1**), and if the IFX is set to Encoder Step Mode (**FSB1**).

| <u>Command</u> | <u>Description</u>                           |
|----------------|--|
| <b>MN</b>      | Sets the system in the preset mode           |
| <b>FSB1</b>    | Enables encoder mode                         |
| <b>DW10</b>    | Selects backlash to be 10 motor steps        |
| <b>FSH1</b>    | Enables stall detect function                |
| <b>FSD1</b>    | Stops motor if a stall is detected           |
| <b>FSE1</b>    | Turns on Output 1 if stall is detected       |
| <b>A2</b>      | Sets acceleration to 2 revs/sec <sup>2</sup> |
| <b>V.1</b>     | Sets velocity to 0.1 rps                     |
| <b>D500</b>    | Sets distance to 500 encoder steps           |
| <b>G</b>       | Execute the move (Go)                        |

While the motor is moving, you can cause a stall by holding the shaft. If you cannot manually stall the motor, turn off the external 5V power to the encoder. When the stall occurs, Output 1 is turned on and the motor stops (this signals you that the motor has stalled). The motor will come to a stop.

**Multi-Axis Stop on Stall**

On a multi-axis SD/IFX system, you may wish to have all axes stop motion if a stall is detected on any axis. You can select this function via the Kill Motion on Trigger (**FSF**) command. When selected with the **FSF1** command, a signal on the Trigger 3 input terminates the move immediately, thus functioning as a remote stop input.

Follow the following steps to set up your SD/IFX system to terminate the multi-axis moves when a stall is detected on any axis:

1. Set all IFX units to Turn on Output #1 on Stall (**FSE1**)
2. Connect Output #1 (pin 15 on connector TB4) on each IFX unit to Trigger #3 (pin 11 on connector TB4) on all the other IFX units in the daisy-chain.

**Output on Position Loss**

The Turn on Output #2 When Within Dead Band (FSG) command allows the IFX to signal other system components when the motor is within the dead band. This command is valid only if Stall Detection (FSH1) has been enabled; it will have no effect otherwise.

At the end of the move, if the motor is within the specified deadband (DB), Output #2 will be turned on.

| <u>Command</u> | <u>Description</u>   |
|----------------|--|
| MN             | Sets the system in the preset mode   |
| FSB1           | Sets indexer to encoder step mode  |
| FSC1           | Enables position maintenance   |
| DB1Ø           | Selects deadband at 10 encoder steps (Position Maintenance is activated if the motor's end-of-move position is off by more than 10 encoder steps.) |
| FSH1           | Enables stall detect function  |
| FSG1           | Turns on Output 2 if the motor's end-of-move position is within the 10 encoder step deadband range   |
| A 2            | Sets acceleration to 2 revs/sec <sup>2</sup>   |
| V 5            | Sets velocity to 5 rps   |
| D 5ØØ          | Sets distance to 500 steps   |
| G              | Execute the move (Go)  |

---

## Program Control

### Triggers

You can use the Wait for Trigger (TR) command to specify a configuration of trigger conditions to be matched before executing a sequence of buffered commands. The trigger inputs are TRIG 1, TRIG 2, and TRIG 3. The three possible conditions you can specify are as follows:

- 1 = Wait for the trigger input to be high (no current flows)
- Ø = Wait for the trigger input to be low (current flows)
- X = Ignore the trigger input

| <u>Command</u> | <u>Description</u>   |
|----------------|--|
| MN             | Sets unit to Preset mode   |
| A 2            | Sets acceleration to 2 revs/sec <sup>2</sup>   |
| V 5            | Sets velocity to 5 rps   |
| D 4ØØØ         | Sets distance to 4,000 steps   |
| TRØ1X          | Waits for Trigger Input 1 to turn on and Trigger Input 2 to turn off (ignores Trigger Input 3) |
| G              | Executes 4,000-step move (Go)  |

The move will not be executed until current flows on the TRIG 1 input and no current flows on the TRIG 2 input.

**Delays**

You can use the Time Delay (T) command to halt the operation of the indexer function for a preset time. If you are in the Continuous Mode (MC), you may use the Time (T) command to run the motor at continuous velocity for a set time, then change to a different velocity.

In the Preset mode (MN), the motor finishes the move before the indexer executes the time delay.

| <u>Command</u> | <u>Description</u>  |
|----------------|---|
| P S            | Waits for the IFX to receive a Continue (C) command before executing next command |
| G              | Moves motor 4,000 steps (defined in Triggers command example above)               |
| T 5 . 0        | Waits 5 seconds after the move ends   |
| H              | Changes motor direction   |
| G              | Moves motor 4,000 steps in the opposite direction                                 |
| C              | Cancel Pause and executes the move  |

Note that the A, V, D, and MN commands from the previous sequence of commands in the Triggers section above are automatically used in this sequence.

**Loops**

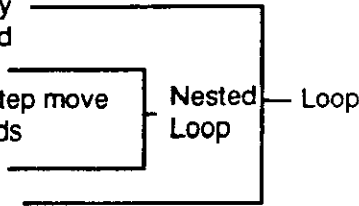
You may use the Loop (L) command to repeat certain programs. You can nest Loop commands up to 16 levels deep.

| <u>Command</u> | <u>Description</u>   |
|----------------|--|
| P S            | Pauses command execution until the indexer receives a Continue (C) command |
| M P I          | Sets mode to incremental   |
| A 5            | Sets acceleration to 5 revs/sec <sup>2</sup>                               |
| V 5            | Sets velocity to 5 rps   |
| L 5            | Performs the Loop 5 times  |
| D 2 0 0 0      | Sets distance to 2,000 steps   |
| G              | Executes the move (Go)   |
| T 2 . 0        | Delays 2 seconds after the move  |
| N              | Ends Loop  |
| C              | Initiates command execution  |

The motor moves a total of 10,000 steps (a 2,000-step move looped 5 times).

The example below shows how you can nest a small loop inside a major loop. In this example, the motor makes 2 moves and returns a line feed. The unit repeats these procedures and will continue to repeat until you instruct the unit to stop.

| <u>Command</u> | <u>Description</u>               |
|----------------|----------------------------------|
| P S            | Pauses execution until C command |
| L              | Loops indefinitely               |
| L 1 L F        | Sends a line feed                |
| L 2            | Loops Twice                      |
| G              | Execute 2,000-step move          |
| T . 5          | Waits 0.5 seconds                |
| N              | Ends Inner loop                  |
| N              | Ends Outer loop                  |
| C              | Cancel Pause and executes move   |



### Programmable Output Bits (POBs)

You can turn the programmable output bits (OUT 1 and OUT 2) on and off using the Output (o) command. You can use the outputs to signal remote controllers, turn on LEDs, sound buzzers, etc. A one (1) turns on a given output, a zero (0) turns the output off, and an X leaves the output unchanged. The outputs conduct current when they are on and do not conduct when they are off (see the o command description in Chapter 5, Software Reference).

| <u>Command</u> | <u>Description</u>   |
|----------------|--|
| MN             | Set to Mode Normal   |
| PS             | Pauses execution until indexer receives a Continue (c) command       |
| A10            | Sets acceleration to 10 revs/sec <sup>2</sup>                        |
| V5             | Sets velocity to 5 revs/sec  |
| D2000          | Sets move distance to 2,000 steps                                    |
| O01            | Sets programmable output 1 off and output 2 on                       |
| G              | Executes the move (Go)   |
| OX0            | After the move ends, leaves output 1 unchanged and sets output 2 off |
| C              | Cancels the Pause and executes the move                              |

### Move Completion Signal

When you complete a move, you may use the IFX's programming capability to signal the end of the current move. In a preset move, you may use one of the following commands:

- LF Line feed (see example #1)
- CR Carriage return (see example #2)
- O Output command (see example #3)
- " Quote command (see example #4)

#### Example #1

| <u>Command</u> | <u>Description</u>   |
|----------------|--|
| PS             | Pauses execution until indexer receives a Continue (c) command |
| A2             | Sets acceleration to 2 revs/sec <sup>2</sup>                   |
| V2             | Sets velocity to 2 rps   |
| D2000          | Sets distance to 2,000 steps                                   |
| G              | Executes the move (Go)   |
| 1LF            | Sends a line feed over the RS-232C interface                   |
| C              | Cancels the Pause and executes the move                        |

The motor moves 2,000 steps. When you complete the move, the unit issues a line feed from the IFX to the host over the RS-232C interface.

#### Example #2

| <u>Command</u> | <u>Description</u>   |
|----------------|--|
| PS             | Pauses execution until indexer receives a Continue (c) command |
| A2             | Sets acceleration to 2 revs/sec <sup>2</sup>                   |
| V2             | Sets velocity to 2 rps   |
| D2000          | Sets distance to 2,000 steps                                   |
| G              | Executes the move (Go)   |
| 1CR            | Sends a carriage return  |
| C              | Cancels the Pause and executes the move                        |

The motor moves 2,000 steps. When the IFX completes the move, it issues a carriage return to the host over the RS-232C interface.

| <b>Example #3</b> | <u>Command</u> | <u>Description</u>   |
|-------------------|----------------|--|
|                   | <b>P S</b>     | Pauses execution until indexer receives a Continue (C) command |
|                   | <b>A 2</b>     | Sets acceleration to 2 revs/sec <sup>2</sup>                   |
|                   | <b>V 2</b>     | Sets velocity to 2 rps   |
|                   | <b>D 2000</b>  | Sets distance to 2,000 steps                                   |
|                   | <b>G</b>       | Executes the move (Go)   |
|                   | <b>O 1 X</b>   | Turns on Output 1  |
|                   | <b>C</b>       | Cancel the Pause and executes the move                         |

The motor moves 2,000 steps. When the IFX completes the move, Output 1 is turned on.

| <b>Example #4</b> | <u>Command</u> | <u>Description</u>   |
|-------------------|----------------|--|
|                   | <b>P S</b>     | Pauses execution until indexer receives a Continue (C) command |
|                   | <b>A 2</b>     | Sets acceleration to 2 revs/sec <sup>2</sup>                   |
|                   | <b>V 2</b>     | Sets velocity to 2 rps   |
|                   | <b>D 2000</b>  | Sets distance to 2,000 steps                                   |
|                   | <b>G</b>       | Executes the move (Go)   |
|                   | <b>"DONE</b>   | Sends the message, DONE, to the terminal                       |
|                   | <b>C</b>       | Cancel the Pause and executes the move                         |

The motor moves 2,000 steps. When you complete the move, the IFX issues the DONE message to the host over the RS-232C interface.

## Sequences

A sequence is a series of commands. These commands are executed in their programmed order whenever the sequence is run. Immediate commands cannot be stored in a sequence, just as they cannot be stored in the command buffer. Only buffered commands may be used in a sequence.

### Sequence Programming

The IFX has 2,000 bytes of non-volatile memory (EEPROM) to store up to 7 sequences. Each sequence may have up to 256 characters (including delimiters). Note that one sequence cannot borrow any unused portion of another sequence's allocated memory.

Use the following commands to define, erase, and run sequences:

| <u>Command</u> | <u>Description</u>               |
|----------------|----------------------------------|
| <b>XD</b>      | Starts sequence definition       |
| <b>XE</b>      | Deletes sequence from EEPROM     |
| <b>XP</b>      | Sets Power-up Sequence Mode      |
| <b>XQ</b>      | Sets/resets interrupted Run mode |
| <b>XR</b>      | Runs a sequence                  |
| <b>XRP</b>     | Runs a sequence with a pause     |
| <b>XT</b>      | Ends sequence definition         |
| <b>XU</b>      | Uploads sequence                 |
| <b>XZ</b>      | Sets power-up sequence to zero   |

The commands that you enter to define a sequence are presented vertically in the examples below. This was done to help you read and understand the commands. When you are actually typing these commands into your terminal, they will be displayed horizontally.

To begin the definition of a sequence, enter the **XD** command immediately followed by sequence identifier number (1 to 7) and a delimiter (pressing the space bar or the carriage return key). The **XT** command ends the sequence definition and automatically loads the sequence into the IFX's EEPROM. All commands entered after the **XD** command and before the **XT** command are executed when the sequence is run. An example is provided below.

| <u>Command</u>   | <u>Description</u>                           |
|------------------|--|
| <b>XE 1</b>      | Erases Sequence 1                            |
| <b>XD 1</b>      | Begins definition of sequence 1              |
| <b>A 2</b>       | Sets acceleration to 2 revs/sec <sup>2</sup> |
| <b>V 1 0</b>     | Sets velocity to 10 rps                      |
| <b>D 2 0 0 0</b> | Sets distance to 2,000 steps                 |
| <b>G</b>         | Executes the move (Go)                       |
| <b>H</b>         | Reverses direction                           |
| <b>G</b>         | Executes the move (Go)                       |
| <b>XT</b>        | Ends definition of sequence 1                |
| <b>XR 1</b>      | Runs Sequence 1                              |

Once you define a sequence, it cannot be redefined until you delete it. You can delete a sequence from EEPROM by entering the **XE** command immediately followed by a sequence identifier (1 to 7) and a delimiter. You may then redefine that sequence into EEPROM. You can issue the Sequence Status Definition (**XSD**) command to verify if the last sequence definition was successful. The possible responses seen on your terminal are **\*0**, **\*1**, or **\*2**. A **\*0** means the sequence was successfully defined. A **\*1** means the sequence already exists with the number you have specified. A **\*2** means there was not enough space in the sequence buffer for that sequence.

To check the status of a sequence, issue the Sequence Status (**XSS**) command. This command must be preceded by a device address and followed immediately by the number (1 to 7) of the sequence and a delimiter. The possible responses are **\*0** (Empty), **\*1** (Bad checksum), or **\*3** (O.K.).

If you wish to check the contents of a sequence, enter the **XU** command. For example, issuing the **1XU1** command causes the IFX to send the contents of sequence number 1 to the computer terminal's screen. The 1 preceding the **XU** command is the device address which must be present since the **XU** command is a *device-specific* command. We are assuming in this example that the IFX is set up at device address 1.

**Selecting a Sequence**

After you define the sequences over the RS-232C interface, you can execute the sequences by using one of the following modes of operation:

- *Stand-alone Operation*
- *Host Computer Operation*
- *Programmable Logic Controller (PLC) Operation*

**Stand-alone Operation**

This section explains and provides examples of how to operate the SD/IFX with remote push-button jog switches and by executing sequences stored in the IFX non-volatile memory.

**Using Jog Switches**

You can use remote jog switches to manually move the motor in either the CW or CCW direction. Detailed jog switch wiring instructions are provided in Chapter 3, Installation. These two jog switches allow you to move the motor without having to issue a new software command or sequence over the RS-232C communication link. Use the jog rate pot, located on the back of the SD/IFX motherboard, to adjust the jog rate from 40 to 1,000 steps/sec. *NOTE: When you use the jog switches, the IFX is not aware of the motor's position change.*

**Using Stored Sequences**

Stored sequences to be automatically executed when you power-up the system, or executed by remote switches. You must first define the sequences into the IFX non-volatile memory. To do this, you will need a computer or PLC with RS-232C communication capabilities for programming the IFX.

**POWER-UP SEQUENCE EXECUTION**

A single, pre-defined sequence may be executed on power-up by issuing the **XP** command immediately followed by a sequence identifier number (1 to 7) and a delimiter. For example, if an **XP1** command was entered, sequence #1 would be executed the next time the IFX was powered up. Sequence #1 would continue to execute on each subsequent power-up until you issue an **XZ** command or a new **XP(0-9)** command. Once the pre-defined sequence is finished executing, the IFX returns control to the RS-232C communications port. If no sequence had been defined as Sequence 1, control would automatically go to the RS-232C communications port.

The **XP** command may be issued at any time, except during sequence definition. Once an **XP** command is entered, it is automatically saved in the IFX's non-volatile memory.

To determine which, if any, sequence will be executed on power-up, issue the Sequence Status Power-up (**XSP**) command.

**USING REMOTE  
SEQUENCE  
INPUTS**

One method of executing sequences is performed by issuing the **XP9** command. The **XP9** command will not cause any sequence to be immediately executed. Once you issue the **XP9** command, the IFX reads the sequence select inputs (Pins 12-14 on SD/IFX motherboard connector TB4) every time it is powered-up, or every time the **Z** (software reset) command is issued. The status of the sequence select inputs will be interpreted by the IFX as a sequence number (see Table 4-2). If, at the time of power up, the number represented by the sequence select inputs is a valid pre-defined sequence (numbers 1 to 7), the IFX will automatically execute that sequence. When this first sequence is finished executing, the IFX will once again read the sequence select inputs and execute the next valid sequence number present on these inputs. The IFX will continue to execute sequences in this fashion until you issue an **S** (stop) or a **K** (kill) command, or if an end-of-travel limit is encountered. If the IFX reads the number 8, or any number that has no sequence defined, it will wait until the status of the inputs changes to a valid, pre-defined sequence.

| Sequence             | SEQ 1   | SEQ 2 | SEQ 3 |
|----------------------|---|-------|-------|
| 1                    | ON  | ON    | ON    |
| 2                    | OFF   | ON    | ON    |
| 3                    | ON  | OFF   | ON    |
| 4                    | OFF   | OFF   | ON    |
| 5                    | ON  | ON    | OFF   |
| 6                    | OFF   | ON    | OFF   |
| 7                    | ON  | OFF   | OFF   |
| 8*                   | OFF   | OFF   | OFF   |
| * Non-valid sequence | OFF = open switch (not pulled to ground)<br>ON = closed switch (pulled to ground) |       |       |

Table 4-2. Sequence Select Inputs

When in the **XP9** mode, it is possible to cause the IFX to pause between sequence execution. This is done with the **XQ1** command. When the **XQ1** command is present within a sequence, the IFX will pause after the execution of the sequence and will wait for all sequence select inputs to be **OFF** (see sequence #8 in Table 4-2). The IFX will then read the status of the sequence select inputs and execute the corresponding sequence number. This interrupted run mode will continue until you issue the **XQ0** command.

You can also program the IFX to read the sequence select lines on power-up with the **XP8** command. This command causes the IFX to execute the first valid sequence number it reads on the sequence select inputs after power-up (exactly like the **XP9** command). It differs from the **XP9** command in that it returns control to the RS-232C communications port after it finishes executing the first sequence, rather than reading the sequence select lines again and executing another sequence.

**EXAMPLE** The following are step-by-step procedures demonstrating how to run sequences. Using a terminal or a computer, key in the following commands:

**STEP 1** Issue the **XP9** command.

**STEP 2** Define the following sequences:

| <u>Command</u> | <u>Description</u>                          |
|----------------|---|
| <b>XE 6</b>    | Erases Sequence 6                           |
| <b>XD 6</b>    | Defines Sequence 6                          |
| <b>MN</b>      | Sets move to normal mode                    |
| <b>A 5</b>     | Sets acceleration to 5 rev/sec <sup>2</sup> |
| <b>V 2</b>     | Set velocity to 2 rps                       |
| <b>D 800</b>   | Sets distance to 800 steps                  |
| <b>G</b>       | Executes the move (Go)                      |
| <b>XT</b>      | Ends Sequence 6 definition                  |

| <u>Command</u> | <u>Description</u>                           |
|----------------|--|
| <b>XE 7</b>    | Erases Sequence 7                            |
| <b>XD 7</b>    | Defines Sequence 7                           |
| <b>MN</b>      | Sets move to normal mode                     |
| <b>A 10</b>    | Sets acceleration to 10 rev/sec <sup>2</sup> |
| <b>V 5</b>     | Set velocity to 5 rps                        |
| <b>D -4000</b> | Sets distance to 4,000 steps (CCW)           |
| <b>G</b>       | Executes the move (Go)                       |
| <b>XT</b>      | Ends Sequence 7 definition                   |

**STEP 3** Verify that your programs were stored properly by uploading each entered sequence (**XU** preceded by a device address and followed by the number of the sequence). If you receive responses that differ from what you programmed, re-enter those sequences.

**STEP 4** Make sure all the sequence select inputs are off (not grounded).

**STEP 5** Cycle Power or enter the **Z** command. The IFX will go through the normal XP9 mode operation. In XP9 mode, the IFX scans the sequence select inputs, looking for a valid sequence according to the binary value of the three inputs (see Table 4-2).

**STEP 6** Momentarily ground the SEQ 2 input. This will execute sequence number 6. *NOTE: After the sequence has finished executing, the inputs are again scanned to execute another sequence.*

**STEP 7** Momentarily ground the SEQ 1 input. This will execute sequence number 7.

### Host Computer Operation

This section assumes you have successfully communicated with the IFX. If you have not verified that your RS-232C communications link is functioning properly, return to Chapter 2, Getting Started, and complete that check before attempting to complete this section.

An IBM-compatible software diskette providing terminal emulation is available from Parker Compumotor. To operate the software, you will need BASICA or GW BASIC programming language programs installed in your computer.

### Immediate Sequence Execution

You can execute a sequence by entering the **XR** command immediately followed by a sequence identifier number (1 to 7) and a delimiter. The sequence will be executed immediately after the delimiter (with minimal calculation delay).

You can issue the Sequence Status Run (**XSR**) command to verify if the last sequence you issued was executed successfully. The possible responses are as follows:

- $\emptyset$  Running
- Non-Zero Not running:
  - 1 In a loop
  - 2 Invalid sequence
  - 3 Erased
  - 4 Bad checksum

### Single-Axis Control

The SD/IFX system is capable of single and multiple axis applications. The principles developed for a single-axis system apply as well to multi-axis systems.

### Single-Axis Interface Program Example

If you already have BASICA or GW BASIC programming languages on your computer, you may use the following sample program designed to open a serial communication port and send and receive IFX commands. The program performs the following steps:

- Executes the first move upon user input
- Waits for a line feed from the IFX, which indicates the end of the move.
- Upon user input, executes the second move
- Waits for a line feed from the IFX, which indicates the end of the move. It then begins the process again.

This application can be looked on as moving a part out, machining the part, then bringing the part back.

```

1 '          IFX.BAS PROGRAM
2 '
3 '
.....
4 ' *
5 ' * This program controls the RS232 Communication line to execute 2
6 ' * different moves using the IFX
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$ = "Z"                                    ' Reset the IFX indexer
122 PRINT #1,Z$;
124 Q$ = INPUT$(2,1)
900 '
.....
901 ' *
902 ' * Line 1000-1060 sends a move down to the first IFX. Computer
903 ' * waits for the Line Feed from the IFX indicating that the motor
904 ' * has finished its move. Computer will not command second IFX to move
905 ' *
906 ' *
.....
1000 MOVE1$ = "1A1 1V2 1D4000 1G 1LF "          ' Define move for Axis 1
1005 CLS
1007 LOCATE 12,15: PRINT " DOING MOVE 1 "
1010 PRINT #1,MOVE1$                            ' Move axis 1.
1015 ECHO$ = INPUT$(23,1)                       ' Read echoes from IFX
1020 LF$ = INPUT$(1,1)                          ' Wait for line feed from IFX
1040 IF LF$ <> CHR$(10) GOTO 1020              ' indicating end of move.
1045 CLS
1047 LOCATE 12,15
1050 PRINT "MOVE 1 DONE"                        ' Let user know axis 1 is done
1060 LOCATE 15,15: PRINT " PRESS ANY KEY TO GO ON TO SECOND MOVE "
1070 V$ = INKEY$: IF LEN(V$) = 0 THEN 1060
1900 ' .....
1901 ' *
1902 ' * After axis one is done, we request that you press any key to go on
1903 ' * to the 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 move is finished, we go on to move #2.
1908 ' * IFX also prints a line feed after finishing the second move.
1909 ' * As soon as computer receives the line feed from IFX, program will
1910 ' * go back to the first move.
1911 ' *
1912 ' .....

```

```

2000 MOVE2$ = "A10 V5 D-10000 G H G 1LF"
2005 CLS
2007 LOCATE 12,15: PRINT " DOING MOVE 2 "
2010 PRINT #1,MOVE2$
2015 ECHO$ = INPUT$(25,1)
2020 LF$ = 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.

### Multi-Axis Control

*Device-specific* commands require that a device address precede them. The IFX will not execute a device-specific command if there is no device address preceding the command, or if the device address setting in the IFX does not match the address preceding the command. *Universal* commands do not require device identifiers preceding them. A universal command with no device address will be executed regardless of the address setting of the IFX. If a device address does precede a universal command, it will only be executed by an IFX set to that particular address.

The **E** (Enable RS-232 communication) and **F** (Disable RS-232 communication) commands are useful in a daisy chain for locking out particular indexers from responding to universal commands with no preceding device address.

*NOTE: The F command will keep the IFX from executing any commands (except the E command) sent to it over the RS-232C interface, but will not prevent the command from being echoed.*

For Example, to lock-out the IFX unit set to device address 1, so that universal commands are only executed by the IFX's set to address 2 and 3, the following step is performed:

- Send the **1F** command over the RS-232C communication line, locking out the IFX at device address 1.

All universal commands (with no preceding device address) will now be executed only by the IFX's at Device addresses 2 and 3. Entering a **2F** command in addition to **1F** command would allow only the IFX at device address 3 to execute universal commands with no preceding device address. This eliminates the need to precede every command intended for a specific IFX (in this case, the IFX at device address 3) with a device address. Sending an **E** command over the RS-232C line will re-enable all drives previously disabled with the **F** command. Preceding the **E** command with a device address will re-enable only the IFX set to that particular device address.

When using the **XU** command to upload the contents of a specified sequence from an IFX in a daisy chain, it is necessary to disable all the drives in the chain that come after the drive being queried with the **F** command (as described above). This will prevent subsequent drives from executing the commands being sent back to the terminal. See the **XU** command description in Chapter 5, Software Reference .

For daisy-chain wiring instructions, refer to Chapter 2, Getting Started.

**Sample Application and Commands**

Example: Three indexers are on an RS-232C daisy chain. Send the following commands:

| <u>Command</u>     | <u>Description</u>  |
|--------------------|---|
| <b>MN</b>          | Sets unit to Preset mode  |
| <b>A 5</b>         | Sets acceleration to 5 revs/sec <sup>2</sup> for all three indexers |
| <b>V 1 0</b>       | Sets velocity to 10 rps for all three indexers                      |
| <b>1 D 8 0 0</b>   | Sets Axis 1 distance to 800 steps                                   |
| <b>2 D 2 0 0 0</b> | Sets Axis 2 distance to 2,000 steps                                 |
| <b>3 D 4 0 0 0</b> | Sets Axis 3 distance to 4,000 steps                                 |
| <b>G</b>           | Moves all axes.   |

Unit 1 moves 800 steps, unit 2 moves 2,000 steps, and unit 3 moves 4,000 steps. All three units use the same acceleration and velocity rates. Units 1, 2, and 3 will start at about the same time.

**Multi-Axis Interface Program Example**

The following program is very similar to IFX.BAS, except this program controls 2 IFX's on a daisy chain. This program assumes the device address of 2 IFX's to be 1 and 2 respectively. The program does the following:

- Executes the first move upon user input
- Waits for a line feed from the LX drive, which indicates the end of the move.
- Upon user input, executes the second move on axis 2
- Waits for a line feed from the second IFX, which indicates the end of the move. It then begins the process again.

```

1 '          IFX2.BAS PROGRAM
2 '
3 '
4 ' *
5 ' * This program controls the RS232 Communication line to execute 2
6 ' * different moves using 2 IFX units..
7 ' *
8 ' *
.....

```



**PLC Operation**

You can use a PLC to execute 7 different sequences that are stored in the IFX non-volatile memory. Three outputs from the PLC can be used to execute sequences, and two inputs to the PLC can be used to monitor IFX outputs.

**PLC Connections**

Assuming your PLC accepts open-collector outputs, connect the inputs and outputs as shown in Figure 4-4. If not, you will need to add pull-up resistors to the POBs (Pins 15 - 16 on the SD/IFX motherboard connector TB4).

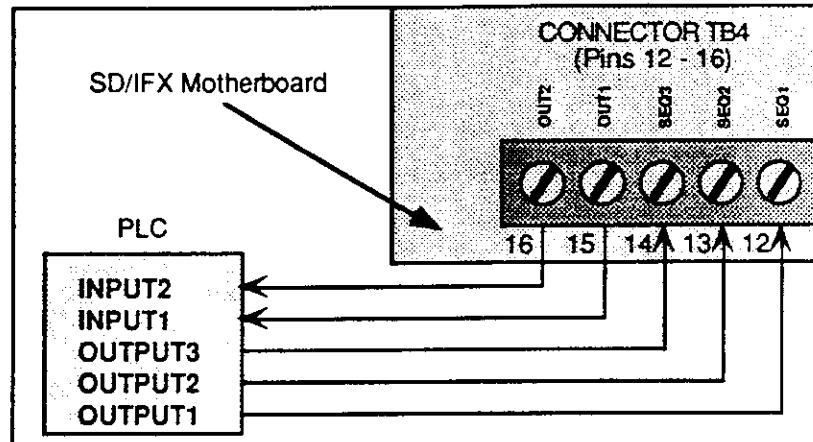


Figure 4-4. PLC Connections

**Scanning for Sequence Execution**

Changing the values of sequence input lines results in a new sequence being run that corresponds to the new value. As indicated in Table 4-2, the configuration of the values issued determines which sequence the indexer will run. For example, turning on SEQ2 and SEQ3 and turning off SEQ1 executes Sequence 2.

Issuing the **XP8** or the **XP9** commands causes the IFX to scan the sequence select inputs and execute the first valid sequence it encounters (on power-up). When in **XP8** mode, the IFX returns control to the RS-232C port after executing the first valid sequence. When in the **XP9** mode, the IFX will continue scanning inputs and executing valid sequences until you issue an **S** or a **K** command.

When you issue the **XP** command identifying sequences 1 - 7, this over-rides the sequence input configuration used when you issue the **XP8** and the **XP9** commands.

**Sample Applications and Commands**

This section provides step-by-step procedures to run sequences from your PLC. First, you need to enter the programs into the IFX. 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 outputs.

Using a computer or terminal, key in the following commands:

**STEP 1** Issue the **XP9** command.

**STEP 2** Define any sequences that your application may require.

| <u>Command</u> | <u>Description</u>                          |
|----------------|---|
| <b>XE1</b>     | Erases Sequence 1                           |
| <b>XD1</b>     | Defines Sequence 1                          |
| <b>A2</b>      | Sets acceleration to 2 rev/sec <sup>2</sup> |
| <b>V10</b>     | Set velocity to 10 rps                      |
| <b>D2000</b>   | Sets distance to 2,000 steps                |
| <b>G</b>       | Executes the move (Go)                      |
| <b>XT</b>      | Ends Sequence 1 definition                  |

| <u>Command</u> | <u>Description</u>                          |
|----------------|---|
| <b>XE2</b>     | Erases Sequence 2                           |
| <b>XD2</b>     | Defines Sequence 2                          |
| <b>A2</b>      | Sets acceleration to 2 rev/sec <sup>2</sup> |
| <b>V10</b>     | Set velocity to 10 rps                      |
| <b>D4000</b>   | Sets distance to 4,000 steps                |
| <b>G</b>       | Executes the move (Go)                      |
| <b>XT</b>      | Ends Sequence 2 definition                  |

| <u>Command</u> | <u>Description</u>                          |
|----------------|---|
| <b>XE3</b>     | Erases Sequence 3                           |
| <b>XD3</b>     | Defines Sequence 3                          |
| <b>A2</b>      | Sets acceleration to 2 rev/sec <sup>2</sup> |
| <b>V10</b>     | Set velocity to 10 rps                      |
| <b>D8000</b>   | Sets distance to 8,000 steps                |
| <b>G</b>       | Executes the move (Go)                      |
| <b>XT</b>      | Ends Sequence 1 definition                  |

**STEP 3** Verify that your programs were stored properly by uploading each entered sequence (**XU** command preceded by the device address and followed by the number of the sequence). If you receive responses that differ from what you programmed, re-enter those sequences.

**STEP 4** Cycle Power or enter the **Z** command. The IFX will go through the normal **XP9** mode operation. In **XP9** mode, the IFX scans the sequence select inputs and selects and executes the first available valid sequence according to the binary value of the three inputs (see Table 4-2). After the sequence is executed, the inputs are again scanned to execute another sequence. The IFX will continue to execute sequence in this fashion until you issue an **S** or a **K** command, or if an end-of-travel limit is encountered.

**STEP 5** To run sequence #1, ground sequence inputs 1, 2, and 3 at the same time (refer to Table 4-2 for sequence input configurations to run other sequences).