BRX DO-MORE! MOTION CONTROL AND HIGH-SPEED I/O

In This Chapter...

Overview

The purpose of this chapter is to help the user understand the capabilities and flexibility of motion control, using high-speed inputs and outputs (HSIO) supported by the BRX MPU built-in I/O and the BX-HSIO1/BX-HSIO2/BX-HSIO4 (referred as BX-HSIO in this chapter for simplification) expansion modules. This section will show the user the steps needed to setup the I/O for use with the high-speed functions, provide various wiring examples, detailed programming examples and explain the available high-speed instructions.

On the BRX MPU built-in I/O, all DC Inputs and Outputs (standard and high-speed) can be selected for use in an HSIO application. However, the standard I/O will work at a lower response time (approximately 120Hz for standard inputs and 110Hz for standard outputs). This flexibility frees up the high-speed inputs and outputs. For example, in a BRX 18/18E MPU, when setting up a pulse train output (PTO) as step/direction control to a stepper motor, it is possible to select a high-speed output for the step signal and a standard output for the direction signal. By doing this, a high-speed output is made available that can be used for another high-speed output function, such as PWM. On the BX-HSIO expansion modules, the eight high-speed DC Inputs and eight high-speed DC Outputs available to the module can be selected for use in a HSIO application. It is not possible to select I/O points outside the BX-HSIO expansion module.

The high-speed inputs are very flexible to meet your application needs. They can work with an input DC frequency of up to 250kHz (2MHz for BX-HSIO4), as a high-speed counter as well as for simple Edge timers. High-speed counter values can be used as accurate position feedback and engineered values for rate and position. The high-speed inputs additionally can be triggers for Interrupt Service Routines (ISRs). The ISRs can be used to run logic based on events that are too fast for standard input triggers and normal PLC logic scan times.

NOTE: *Interrupts configured in the BX-HSIO expansion modules may have a latency of up to 50μs.*

The high-speed outputs are also very adaptable to your application needs. They can be switched as fast as 250kHz for BX-HSIO1 and BX-HSIO2, and up to 2MHz for BX-HSIO4. They also can be used for position and velocity motion control moves as well as provide a programmable Pulse Width Modulation (PWM) output. The high-speed outputs facilitate accurate position and velocity moves including homing, jogging, trapezoid and S-curve moves. Instructions include simple position and velocity moves, to more complicated electronic camming and gearing, providing additional ways to handle follower type moves. In addition to the frequency adjustment, the duty cycle can also be adjusted on the PWM output, allowing for more accurate control of the PWM output signal.

Overview, continued

The following tables show the BRX platform high-speed input (HSI) specifications.

HSI Specifications

Item	10/10E	18/18E	36/36E	BX-HSIO1	BX-HSIO2	BX-HSIO4
Input Type	Sink/Source					High Speed TTL Differential or Single Ended
Total Inputs [1]	6	10	20 [2]	8	8	8
High-Speed Inputs	6	10	10	8	8	8
Location	X0–X5	X0–X9	X0–X9	Slot dependent		
Frequency	0 to 250kHz					0 to 2MHz
Minimum Pulse width	0.5 µs					125ns
Off to On Response	< 2µs					125ns
On to Off Response	< 2µs					125ns

1. *Refer to the specific wiring chapter for the discrete input specs of the specific model you are using.*

2. *BRX MPU standard inputs may be used with high-speed functions, but at lower response frequencies of approximately 120Hz.*

High-speed Input Function

	Functions Available			Inputs Required[1]	User Selected Options					
	MPU	BX-HSIO1 BX-HSIO2	BX-HSIO4		Reset Input	Capture	Inhibit	Rotary	Position Scaling[2]	Rate Scaling[2]
Up Counter	Up to 3	Up to 4	Up to 4	1	1 Input is used	1 Input is used	1 Input is used	N/A	(optional)	
Down Counter				1					(optional)	
Quad Counter				2				(optional)		
Bidirectional Counter				2						
Up/Down Counter				2						
Edge Timer				1				N/A	N/A	(optional)
Edge Timer (Duration)				1						
Dual Edge Timer				2						
Pulse Catch				1	N/A	N/A	N/A			N/A
External Interrupt Triggers										
Event Trigger	4	4	4	Available inputs	N/A					
Timer Trigger	4	4	4	N/A						
Match Register	4	4	4							
Input Filters	Able to filter all inputs									

1. *BRX MPU standard inputs may be used with high-speed functions, but at lower response frequencies of approximately 120Hz.*

2. *Only one scaling option can be used at any given time. If Position scaling is used, Rate scaling is not available, and vice versa.*

Table Driven Inputs

	Functions Available			Inputs Required	Outputs Required[1]	Instructions
	MPU	BX-HSIO1 BX-HSIO2	BX-HSIO4			
Preset Table	Up to 4	Up to 4	Up to 4	Reference to (one) Axis Position or (one) High-Speed Counter/ Timer Accumulator	1	TDOPRESET
Programmable Limit Switch				Reference to (one) Axis Position or (one) High-Speed Counter/ Timer Accumulator	1	TDOPLS

1. *BRX MPU standard outputs may be used with high-speed functions, but at lower response frequencies of approximately 110Hz. Use of relay outputs is not recommended.*

Overview, continued

The following tables show the BRX platform high-speed output (HSO) specifications.

HSO Specifications

Item	10/10E	18/18E	36/36E	BX-HSIO1	BX-HSIO2	BX-HSIO4
Output Type	Sink or Source [2]			Sink	Source	High Speed TTL Differential or Single Ended
Total Outputs[1,2]	4	8	16 [2]	8	8	8
High-Speed Outputs	2	4	8	8	8	8
Location	Y0–Y1	Y0–Y3	Y0–Y7	Slot dependent		
Off to On Response	< 2μs					125ns
On to Off Response	< 2μs					125ns
Max Switching Frequency	1m cable; 250kHz 10m cable; 100kHz					2MHz

1. *Refer to the specific wiring chapter for the discrete output specs of the specific model you are using.*

2. *BRX MPU standard outputs may be used with high-speed functions, but at lower response frequencies of approximately 110Hz.*

High-speed Output Function

	Functions Available[2]			Outputs Required[1]	Profile/Instruction
	MPU	BX-HSIO1 BX-HSIO2	BX-HSIO4		
Axis/Pulse Output	Up to 4 (1 virtual and 3 axis)	Up to 4 (1 virtual and 3 axis)	Up to 4 (1 virtual and 3 axis)		
Virtual Axis	Up to 4	Up to 4	Up to 4	N/A	Trapezoid, Velocity, Electronic Camming, Electronic Gearing, Following, Homing
Step/Direction	Up to 3	Up to 3	Up to 3	2	
CW/CCW					
Quadrature					
Pulse Width Modulation (PWM)	Up to 3	Up to 4	Up to 4	1	N/A

1. *BRX MPU standard outputs may be used with high-speed functions, but at lower response frequencies of approximately 110Hz. Use of relay outputs is not recommended.*

2. *This is the total number of functions. On the BRX MPU, a combination of high-speed outputs and standard outputs may be used UP TO this total.*

Unsuitable Applications

There are situations where HSIO is not an appropriate control choice:

- Mechanical contacts used as counter or encoder inputs: Reliable readings are not possible using mechanical contacts. The bounce of mechanical contacts will cause the High-speed input to see more edges than intended.

- Direct connection to TTL, line driver or differential encoders, other than with the BX-HSIO4: A BX-HSIO1 or BX-HSIO2 high-speed input cannot accept these low voltage inputs directly. (Consider using a BX-HSIO4, or using the FC-ISO-C signal conditioner to be able to input these signal types to these modules.)

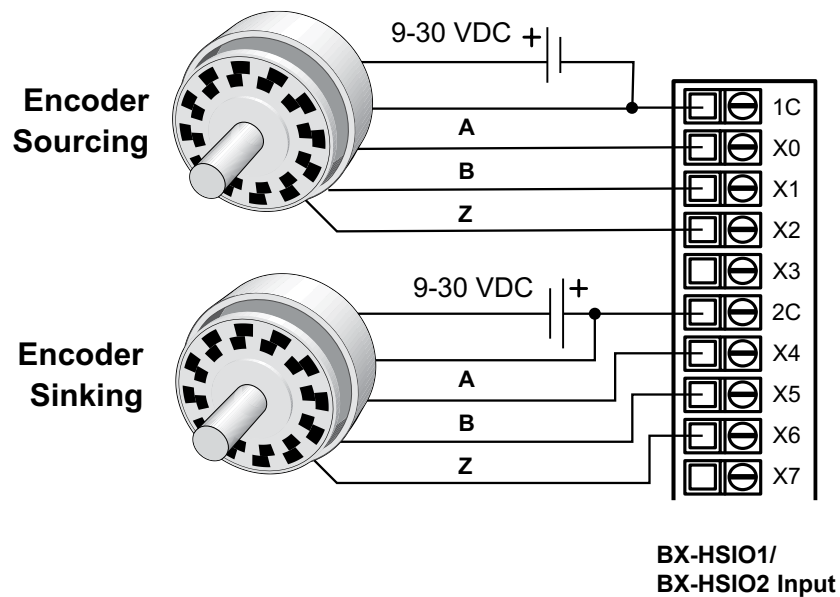- Absolute encoders are not suitable for use with the high speed inputs of the BRX.

BRX Wiring Examples: High-Speed Inputs

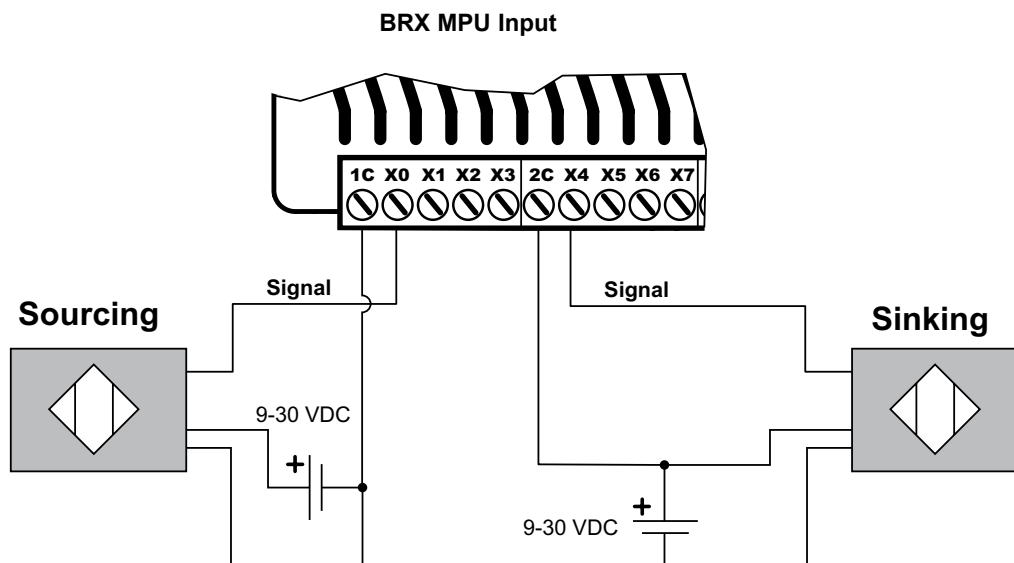## Quadrature Encoder Input

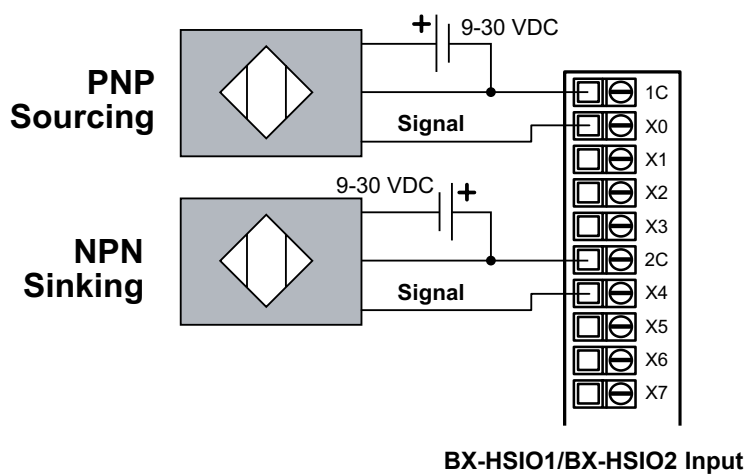**BRX MPU Wiring**
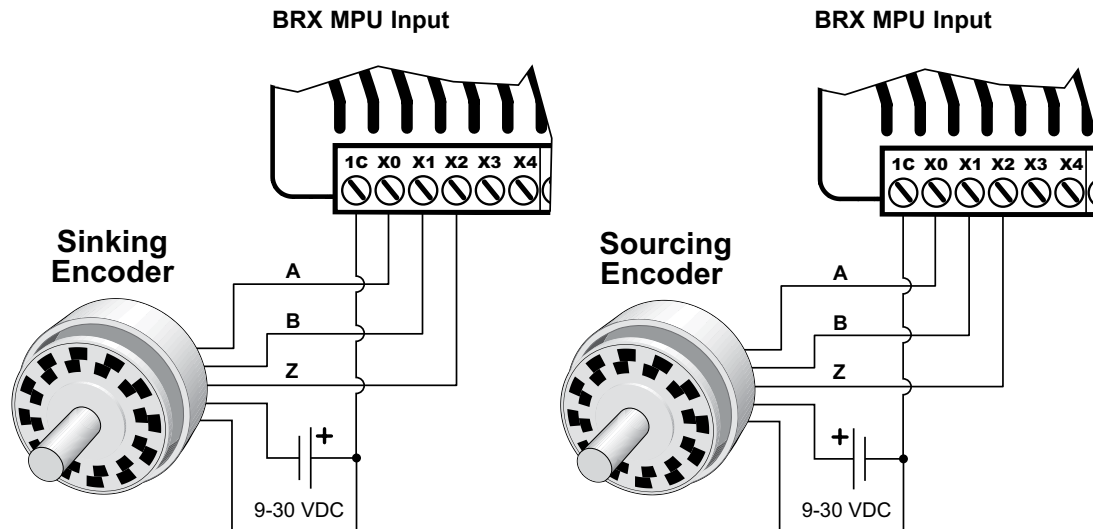
BX-HSIO1/BX-HSIO2 Wiring

BRX Wiring Examples: High-Speed Inputs, continued

NPN/PNP Input Example

BRX MPU Wiring

BRX MPU Input
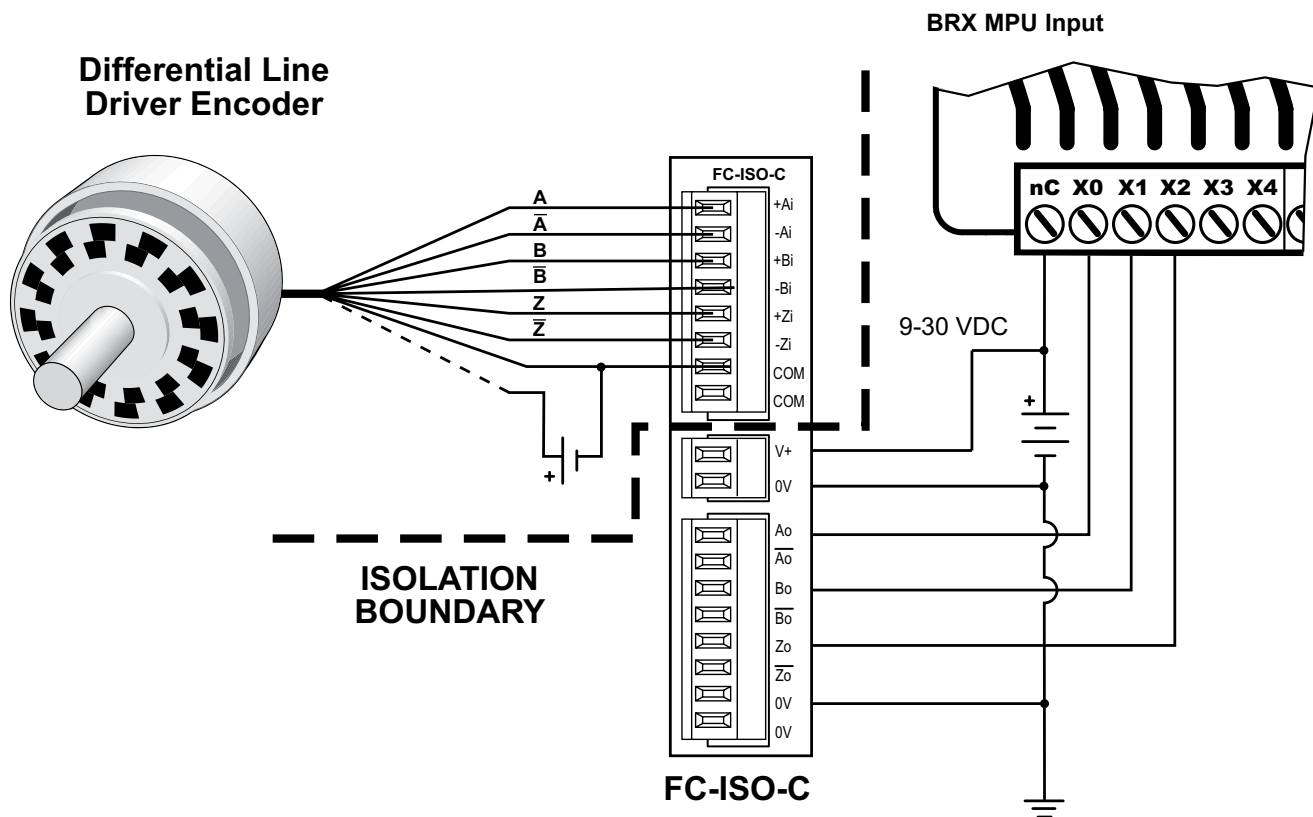

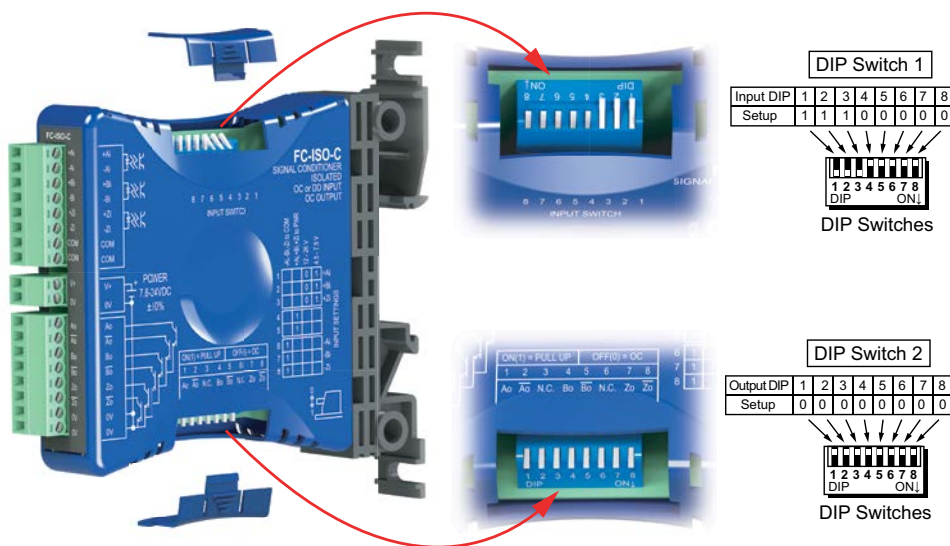### BX-HSIO1/BX-HSIO2 Wiring

BX-HSIO1/BX-HSIO2 Input

BRX Wiring Examples: High-Speed Inputs, continued

Differential Line Driver Encoder Input to BRX MPU

FC-ISO-C Dipswitch Settings

DIP Switch 1								
Input DIP	1	2	3	4	5	6	7	8
Setup	1	1	1	0	0	0	0	0

DIP Switches

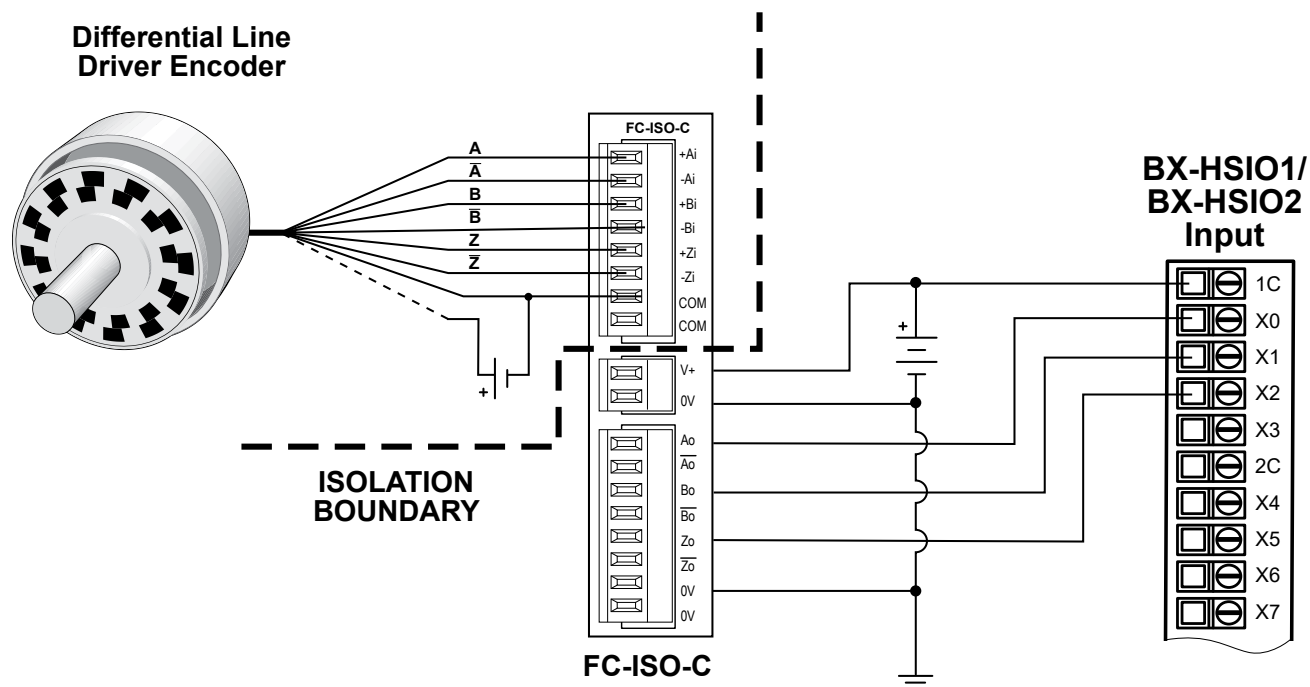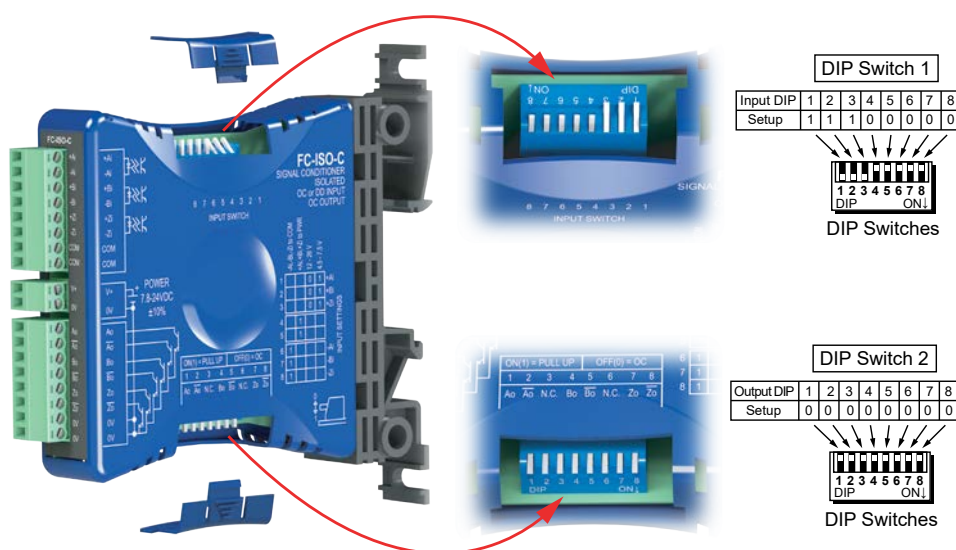| DIP Switch 2								
Output DIP	1	2	3	4	5	6	7	8
Setup	0	0	0	0	0	0	0	0

DIP Switches

BRX Wiring Examples: High-Speed Inputs, continued

Differential Line Driver Encoder Input to BX-HSIO1/BX-HSIO2

FC-ISO-C Dipswitch Settings

DIP Switch 1								
Input DIP	1	2	3	4	5	6	7	8
Setup	1	1	1	0	0	0	0	0

DIP Switches

DIP Switch 2								
Output DIP	1	2	3	4	5	6	7	8
Setup	0	0	0	0	0	0	0	0

DIP Switches

BRX Wiring Examples: High-Speed Inputs, continued

*ZIP*Link Terminal Block Wiring Connections for BX-HSIO4

Wiring Connections for ZL-RTB40 Terminal Block

MODULE	LABELS																				LEVEL
BX-HSIO4	IN 0-	IN 1-	IN 2-	IN 3-	COM	IN 4-	IN 5-	IN 6-	IN 7-	COM	OUT 0-	OUT 1-	OUT 2-	OUT 3-	COM	OUT 4-	OUT 5-	OUT 6-	OUT 7-	COM	UPPER
	IN 0+	IN 1+	IN 2+	IN 3+	COM	IN 4+	IN 5+	IN 6+	IN 7+	COM	OUT 0+	OUT 1+	OUT 2+	OUT 3+	COM	OUT 4+	OUT 5+	OUT 6+	OUT 7+	COM	LOWER
TERMINAL BLOCK LABEL SHEET FOR ZIPLINK CABLE ZL-BX-CBL-40-xS																				

Wiring Connections for ZL-RTB40-1 Terminal Block

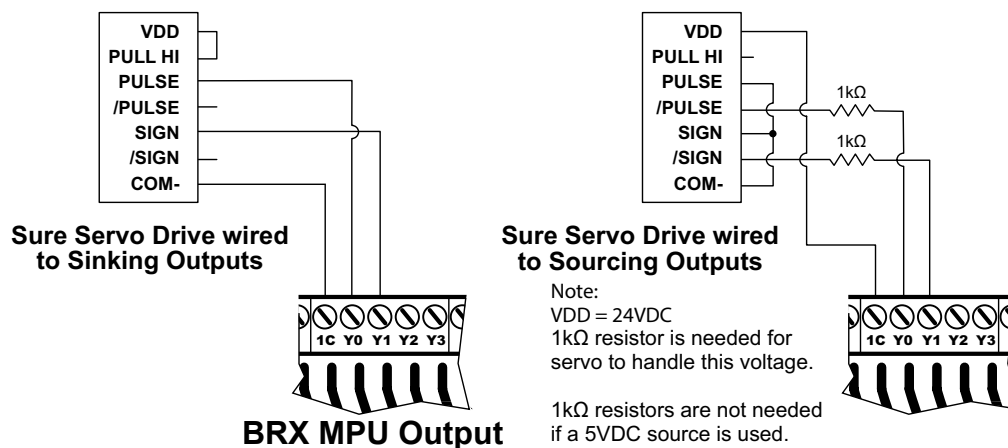| MODULE	LABELS																LEVEL
	COM		COM		COM		COM		COM		COM		COM		COM		UPPER
BX-HSIO4	IN 0-	IN 1-	IN 2-	IN 3-	IN 4-	IN 5-	IN 6-	IN 7-	OUT 0-	OUT 1-	OUT 2-	OUT 3-	OUT 4-	OUT 5-	OUT 6-	OUT 7-	MIDDLE
	IN 0+	IN 1+	IN 2+	IN 3+	IN 4+	IN 5+	IN 6+	IN 7+	OUT 0+	OUT 1+	OUT 2+	OUT 3+	OUT 4+	OUT 5+	OUT 6+	OUT 7+	LOWER
TERMINAL BLOCK LABEL SHEET FOR ZIPLINK CABLE ZL-BX-CBL-40-xS																

Differential 5V Encoder Input to BX-HSIO4

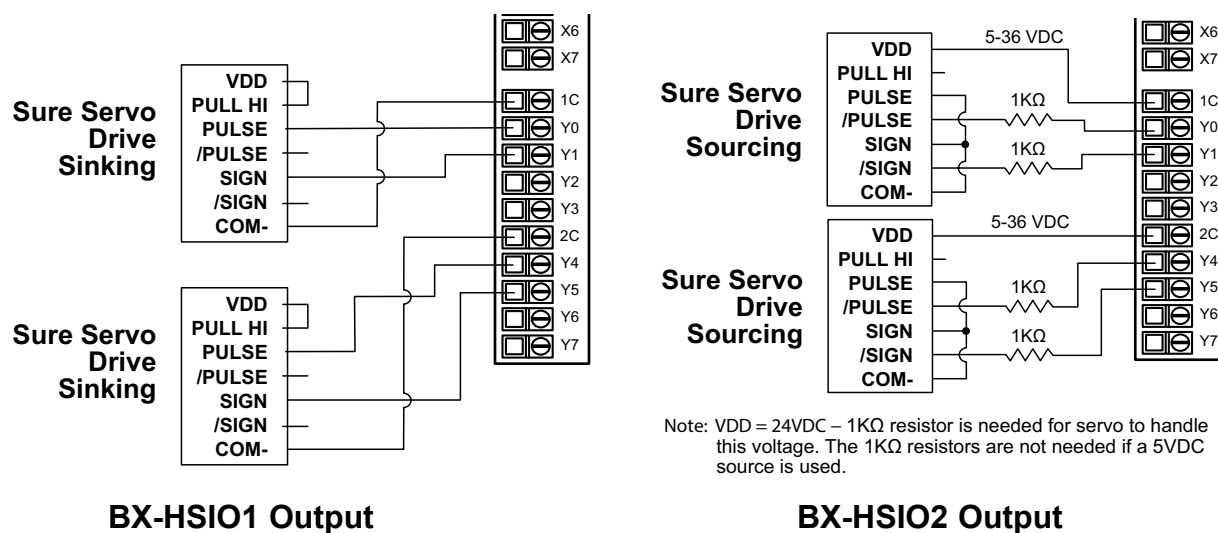To prevent damage to 5V inputs,
do not exceed 6.8V or 30 mA on inputs
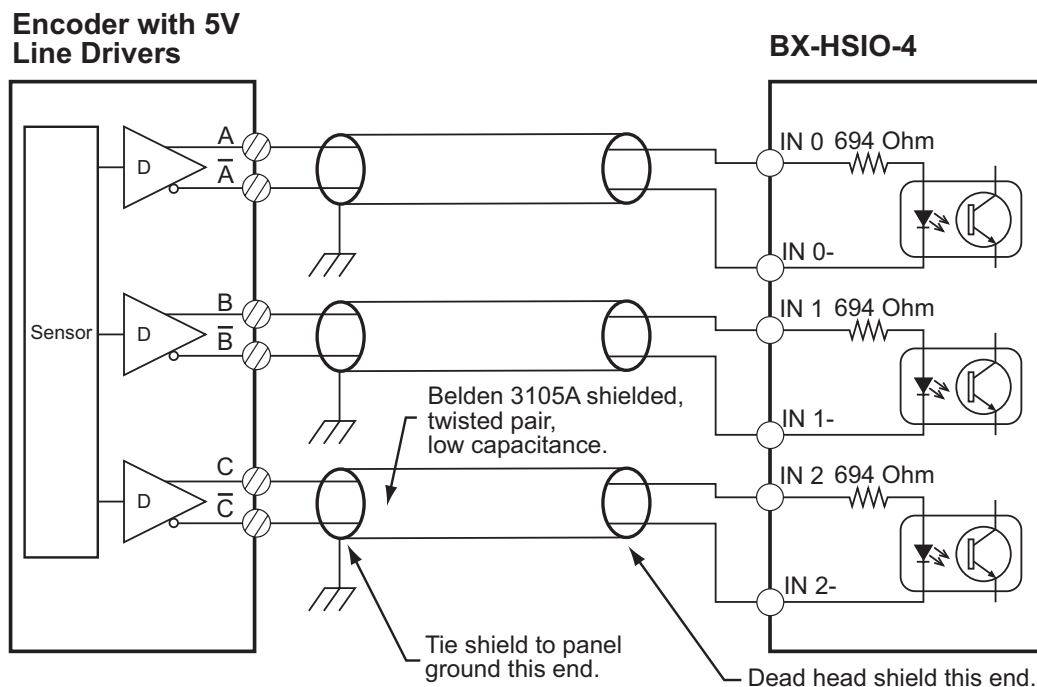
BRX Wiring Examples: High-speed Outputs

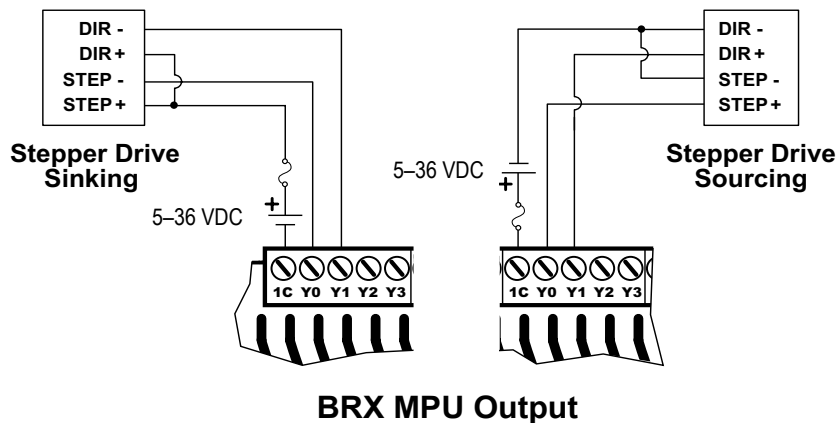## SureServo Driver Wiring Example

BRX MPU Wiring

**Sure Servo Drive wired
to Sinking Outputs**

**Sure Servo Drive wired
to Sourcing Outputs**

Note:
VDD = 24VDC
1kΩ resistor is needed for
servo to handle this voltage.

1kΩ resistors are not needed
if a 5VDC source is used.

BRX MPU Output

BX-HSIO1/BX-HSIO2 Wiring

**Sure Servo
Drive
Sinking**

**Sure Servo
Drive
Sinking**

**Sure Servo
Drive
Sourcing**

**Sure Servo
Drive
Sourcing**

Note: VDD = 24VDC – 1KΩ resistor is needed for servo to handle
this voltage. The 1KΩ resistors are not needed if a 5VDC
source is used.

BX-HSIO1 Output   **BX-HSIO2 Output**

NOTE: *Customer must consult SureServo documentation for specific details on the servo drive.*

BRX Wiring Examples: High-Speed Outputs, continued

Stepper Drive Wiring Example

BRX MPU Wiring

BRX MPU Output

BX-HSIO1/BX-HSIO2 Wiring
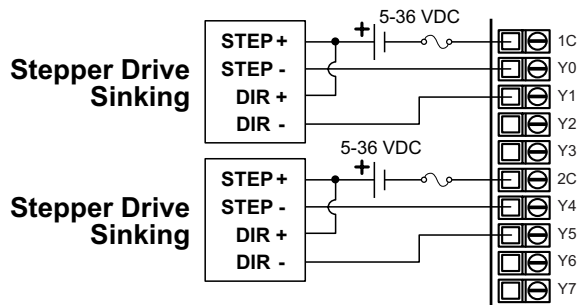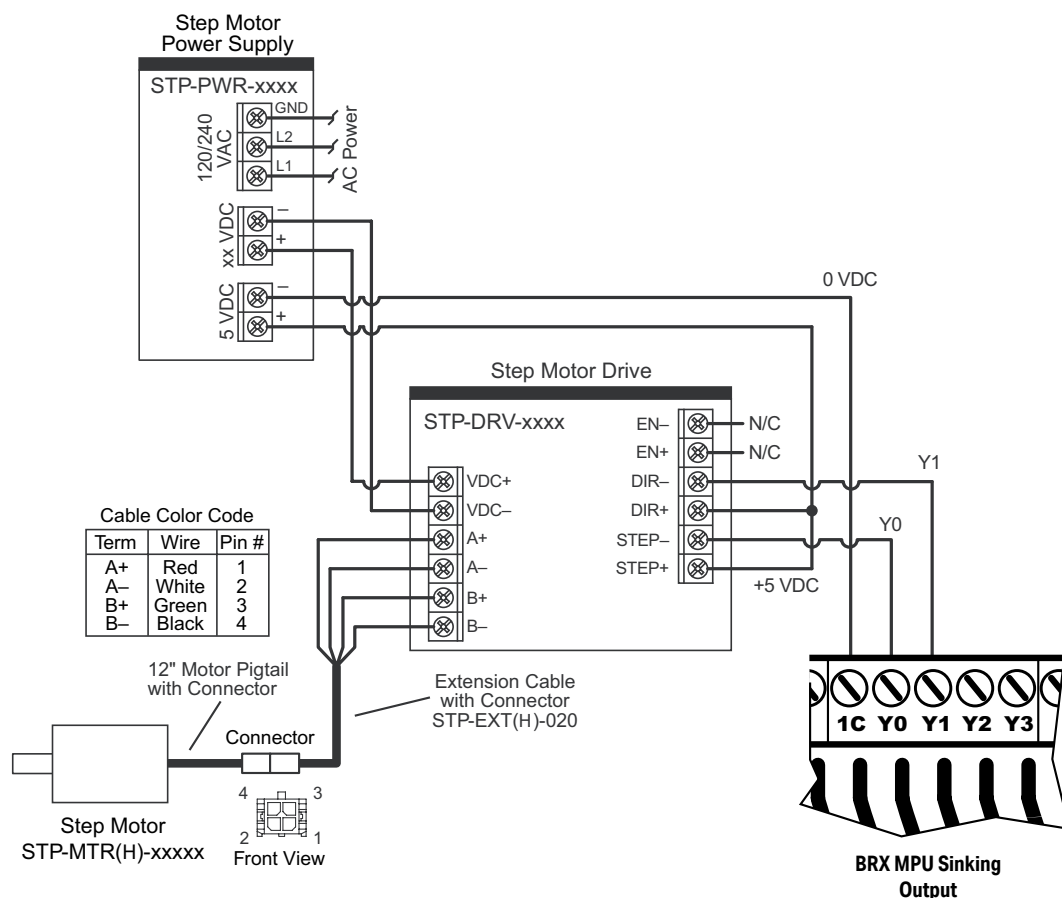

BX-HSIO1 Output
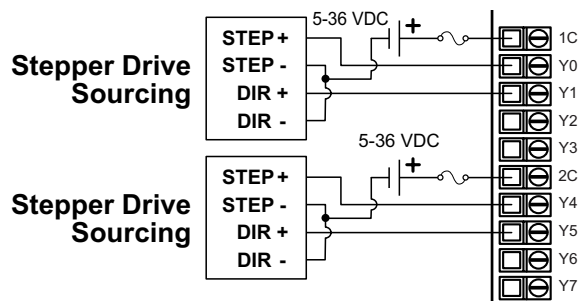
BX-HSIO2 Output

BRX Wiring Examples: High-Speed Outputs, continued

BRX MPU with Sinking Outputs to Stepper/Servo Drive Input

Detailed output wiring example between a BRX MPU and the SureStep Stepping System components.
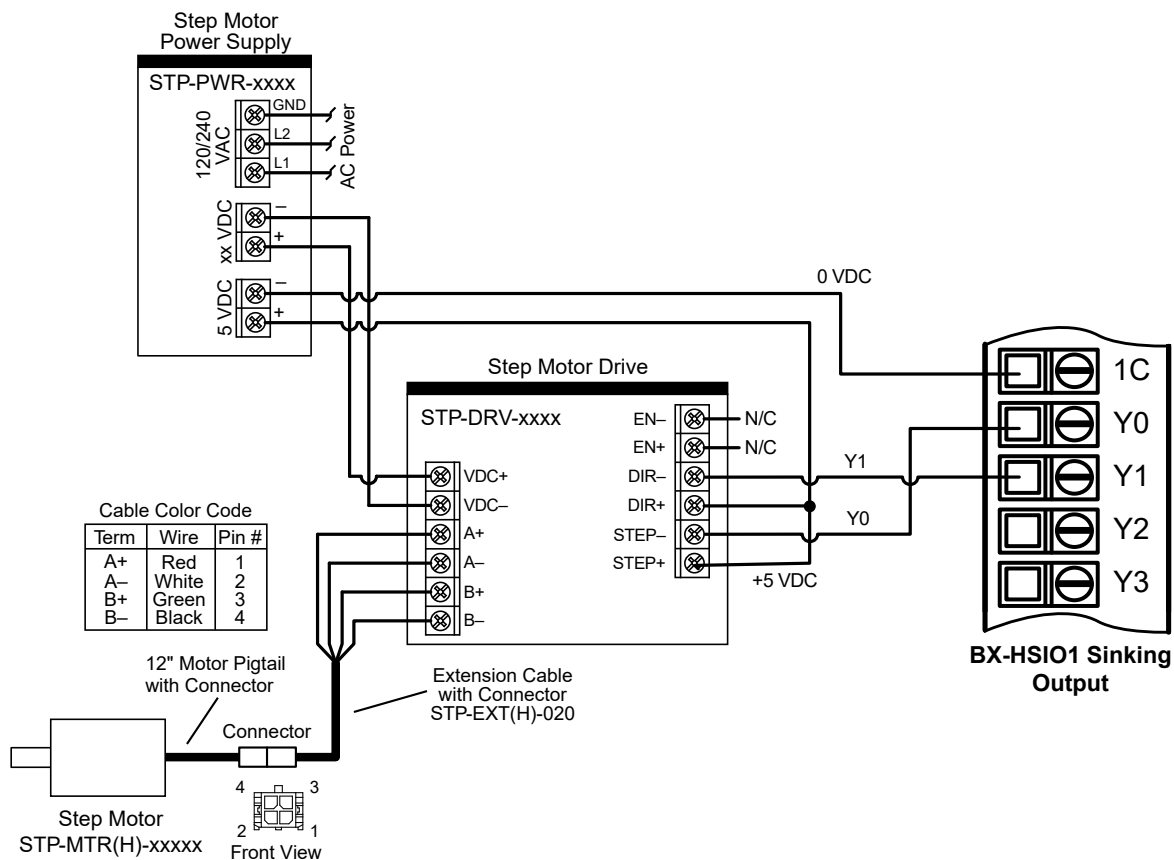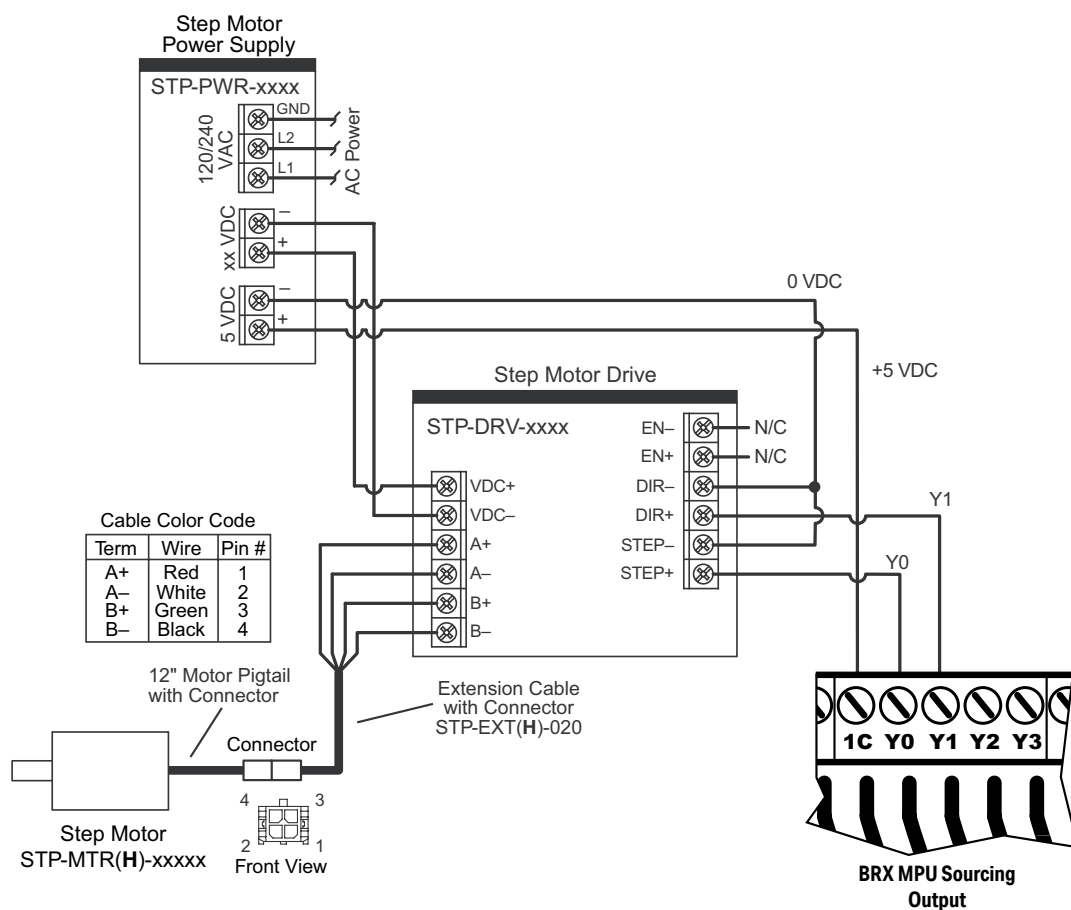
Cable Color Code

Term	Wire	Pin #
A+	Red	1
A–	White	2
B+	Green	3
B–	Black	4

Wiring Diagram for a BRX MPU Using the SureStep Stepping System Components.

BRX Wiring Examples: High-Speed Outputs, continued

## BX-HSIO1 with Sinking Outputs to Stepper/Servo Drive Input

Detailed output wiring example between a BX-HSIO1 and the SureStep Stepping System components.

Cable Color Code

Term	Wire	Pin #
A+	Red	1
A–	White	2
B+	Green	3
B–	Black	4

Wiring Diagram for a BX-HSIO1 Using the SureStep Stepping System Components.

BRX Wiring Examples: High-Speed Outputs, continued

BRX MPU with Sourcing Outputs to Stepper/Servo Drive Input

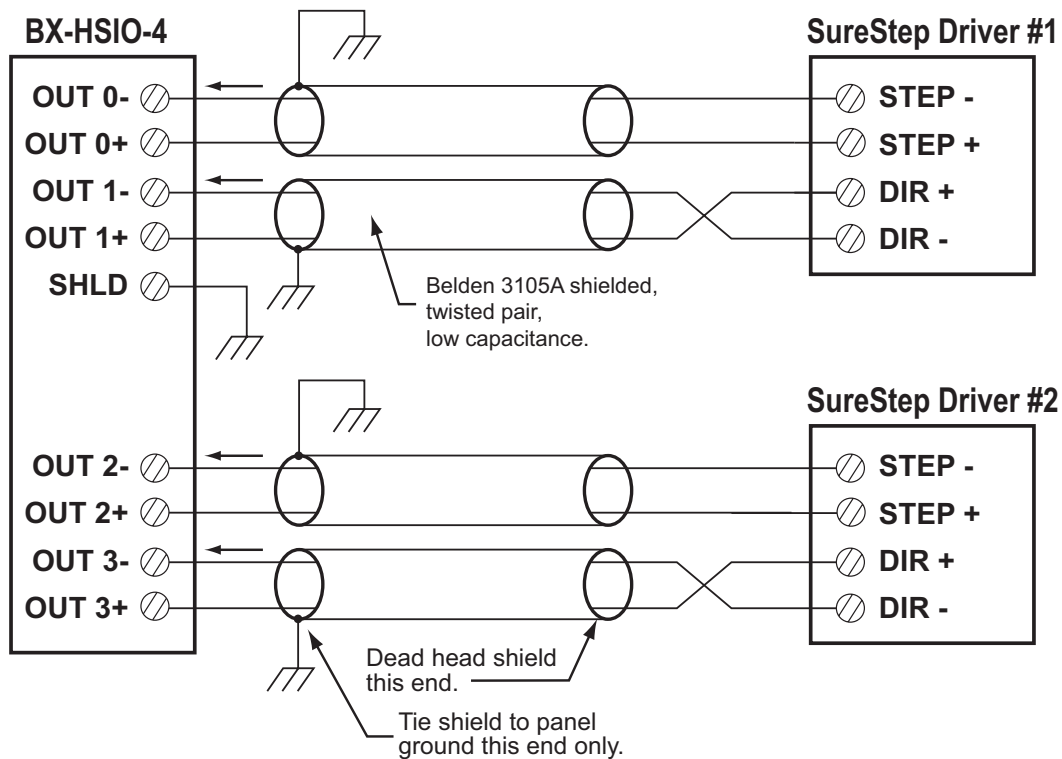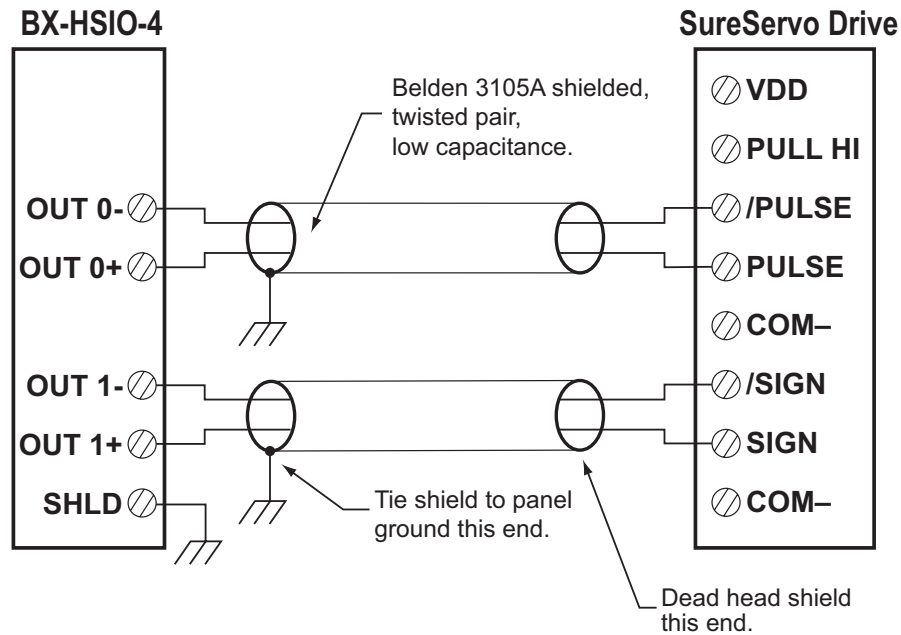Detailed output wiring example between a BRX MPU and the SureStep Stepping System components.

Cable Color Code

Term	Wire	Pin #
A+	Red	1
A–	White	2
B+	Green	3
B–	Black	4

Wiring Diagram for a BRX MPU Using SureStep Stepping System Components.

BRX Wiring Examples: High-Speed Outputs, continued

BX-HSIO2 with Sourcing Outputs to Stepper/Servo Drive Input

Detailed output wiring example between a BX-HSIO2 and the SureStep Stepping System components.

Cable Color Code

Term	Wire	Pin #
A+	Red	1
A–	White	2
B+	Green	3
B–	Black	4

Wiring Diagram for a BX-HSIO2 Using SureStep Stepping System Components.

BRX Wiring Examples: High-Speed Outputs, continued

Line Driver Pulse Output from BX-HSIO4
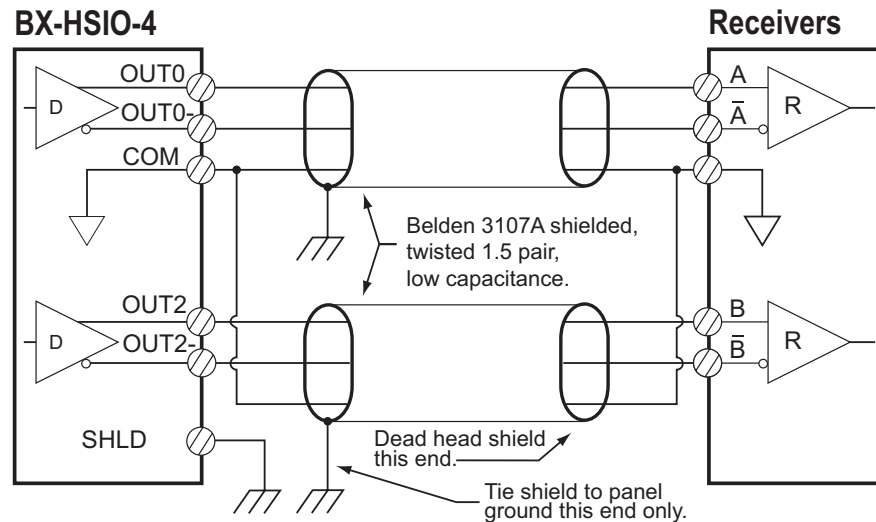
BX-HSIO-4 to SureStep

BRX Wiring Examples: High-Speed Outputs, continued

BX-HSIO-4 to SureServo

BX-HSIO-4   **SureServo Drive**

Belden 3105A shielded,
twisted pair,
low capacitance.

BX-HSIO-4	SureServo Drive
	VDD
	PULL HI
OUT 0-	/PULSE
OUT 0+	PULSE
	COM–
OUT 1-	/SIGN
OUT 1+	SIGN
SHLD	COM–

Tie shield to panel
ground this end.

Dead head shield
this end.

BX-HSIO-4 to Sourcing Input Device with Internal Power Supply

BX-HSIO-4   **Sourcing Input Device with Internal P/S**

Belden 3105A shielded,
twisted pair,
low capacitance.

BX-HSIO-4	Sourcing Input Device with Internal P/S
	VDD
OUT 0-	/PULSE
COM	COM–
	VDD
OUT 1-	/SIGN
COM	COM–
SHLD	

Tie shield to panel
ground this end.

Dead head shield
this end.
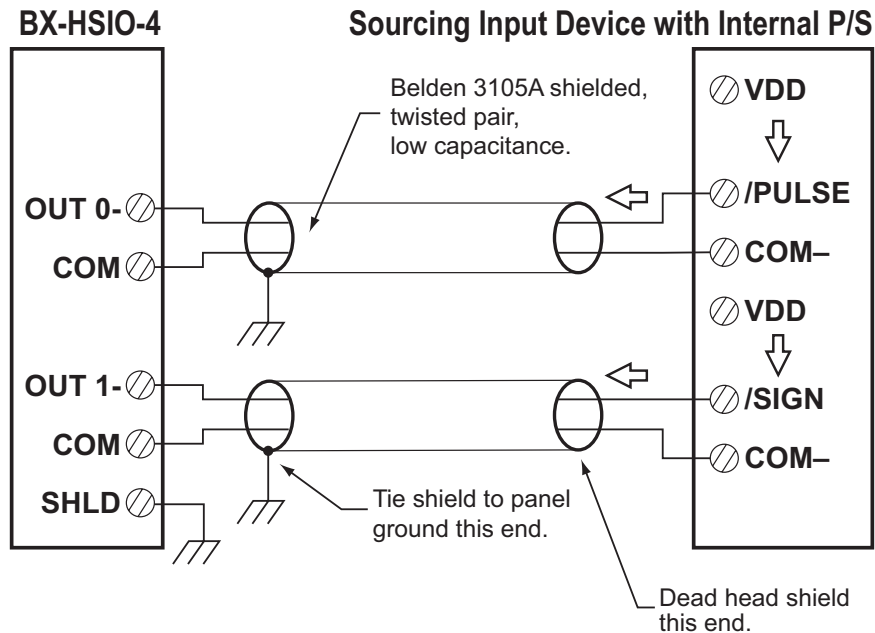
Available High-Speed Input and Output Features

The following High-Speed input and output features are available on the BRX Do-more! MPUs and the BX-HSIO specialty modules. Reference the specific numbered topic listed below for directions on configuring that feature.

1. Input Filters
2. Interrupt Setup (BRX MPUs Only)
    a. Setup Input Interrupts
    b. Setup Timer Interrupts
    c. Setup Match Register Interrupts
    d. Interrupt Instructions
3. High-Speed Counting, Timing and Pulse Catch
    a. Counters
    b. Timers
        i. Interval Scaling
    c. Pulse Catch
4. Table Driven Outputs
    a. Preset Tables
    b. Programmable Limit Switch (PLS)
5. Outputs
    a. Axis/Pulse Outputs
    b. PWM (Pulse Width Modulation)

Access Setup High Speed I/O page

BRX MPU

For High-Speed input and output setup:

- Go to Do-more! Designer (a) menu – PLC > System Configuration. In the *System Configuration* window or click on the (b) **Configure** icon, select (c) **BRX Local I/O** option under the *Configuration Entries* panel. Click on the (d) **High Speed I/O** button.

- From the system dashboard you can click (e) **HSIO** on the left side of the PLC image.
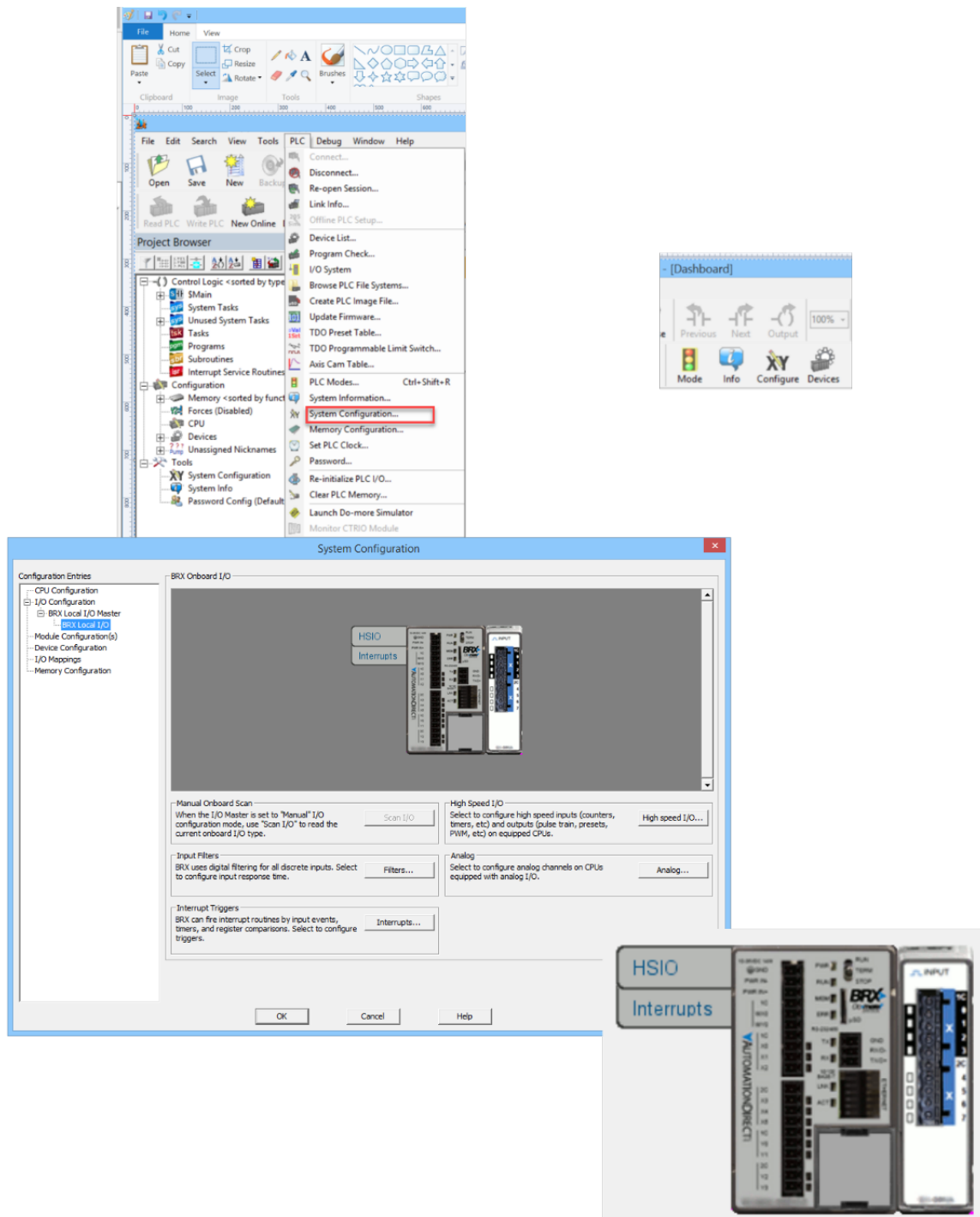
Access Setup High Speed I/O page, continued

The *Setup BRX High Speed I/O* page for the MPU built-in I/O comes up.

Setup BRX High Speed I/O

Input Functions (Counter/Timer/Pulse Catch)

Function 1:	Disabled
Function 2:	Disabled
Function 3:	Disabled

Axis/Pulse Outputs

Axis 0:	@Axis0 - Virtual Axis
Axis 1:	@Axis1 - Virtual Axis (Outputs disabled)
Axis 2:	@Axis2 - Virtual Axis (Outputs disabled)
Axis 3:	@Axis3 - Virtual Axis (Outputs disabled)

PWM Outputs

PWM 1:	Disabled
PWM 2:	Disabled
PWM 3:	Disabled

Table Driven Outputs

Table 1:	Disabled
Table 2:	Disabled
Table 3:	Disabled
Table 4:	Disabled

OK   Cancel

Access Setup High Speed I/O page, continued
BX-HSIO1/BX-HSIO2/BX-HSIO4

For High-Speed input and output setup:

- From the system dashboard left or right-click on the (a) BX-HSIO expansion module and click on the (b) **Configure Module** link.
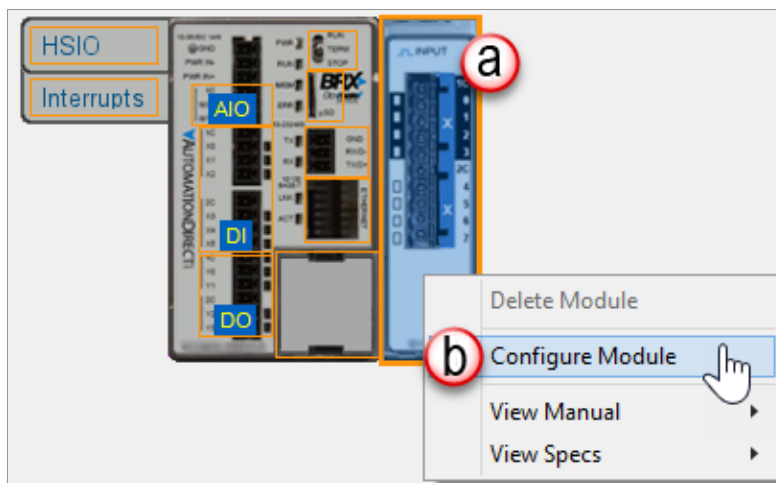
The *Setup BX-HSIO Module* page for the BX-HSIO expansion module I/O comes up.
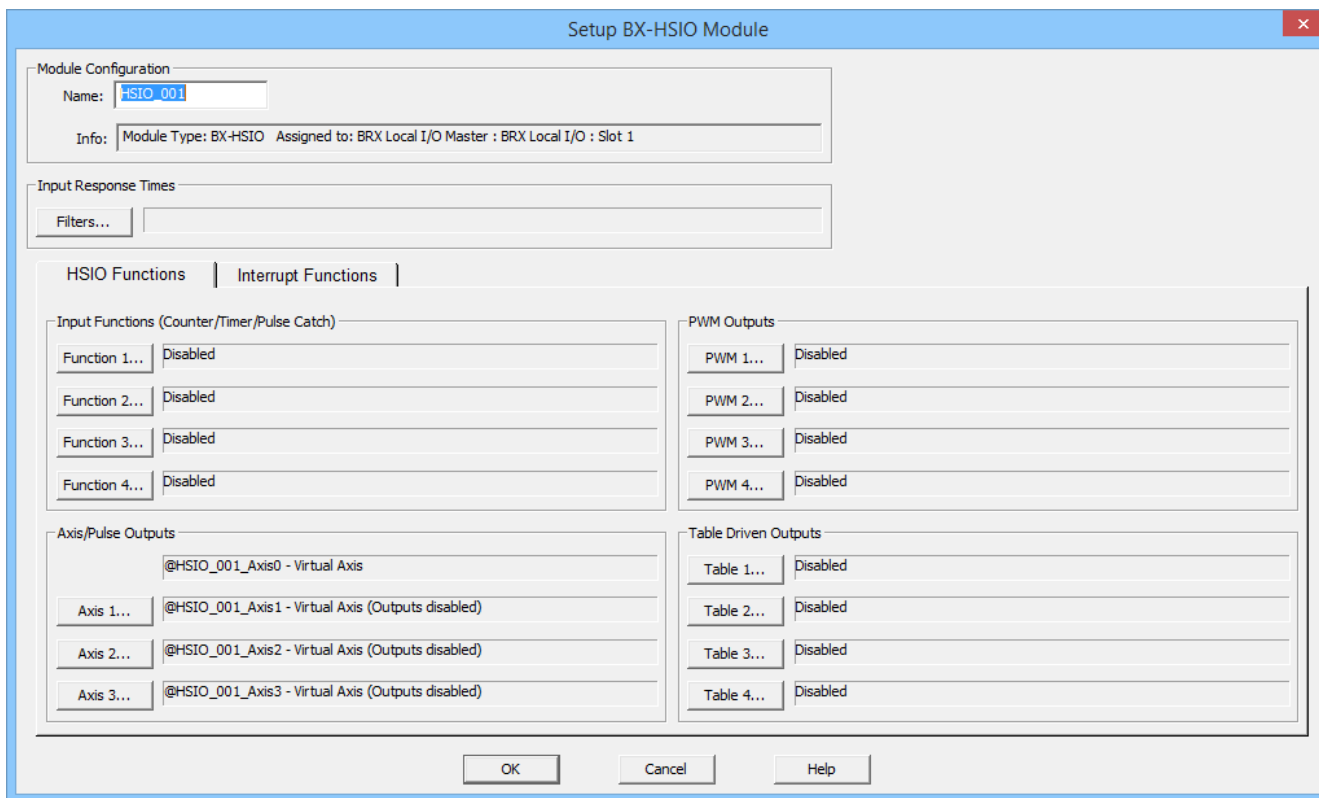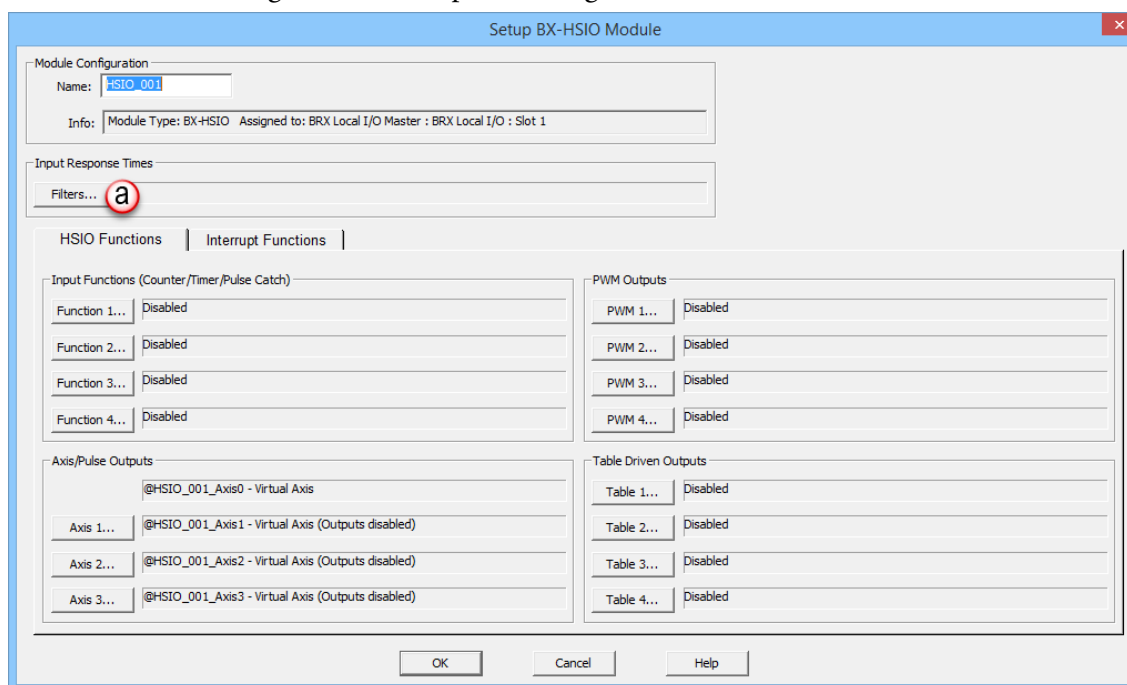
1. Input Filters

BRX MPU

In the *System Configuration* window, select (a) the *BRX Local I/O* option under the *Configuration Entries* panel. The on-board discrete inputs on all of the BRX hardware platforms can be configured to use (b) input filters.
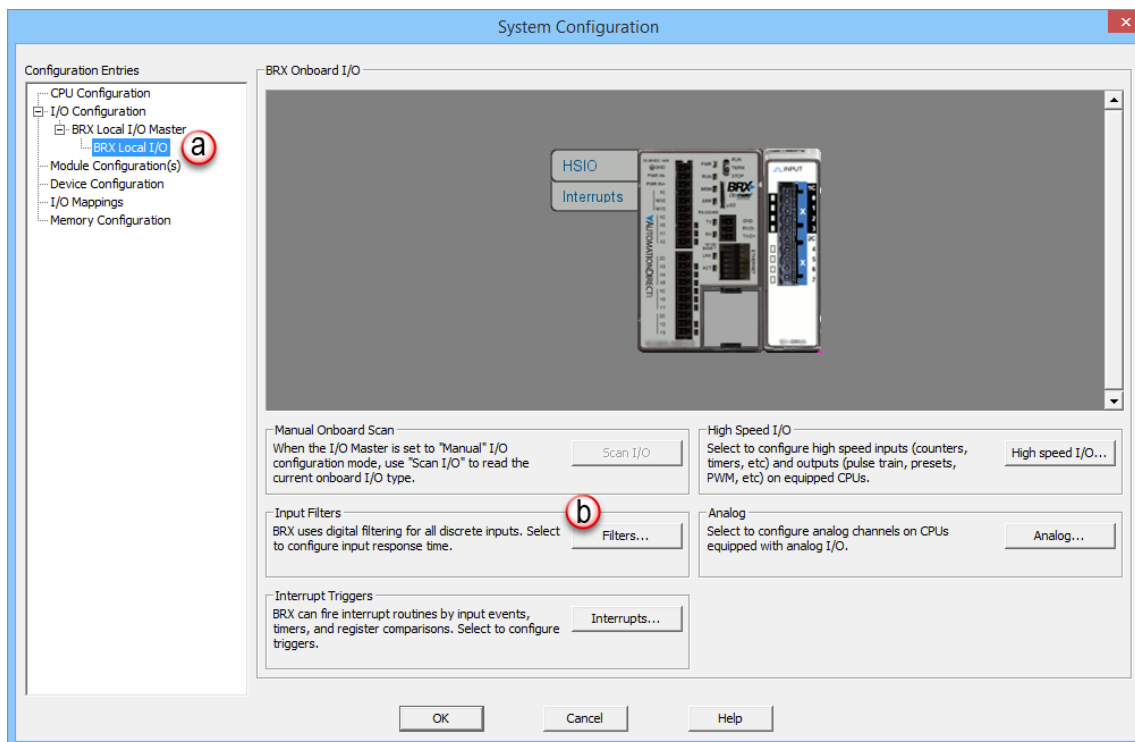
BX-HSIO1/BX-HSIO2/BX-HSIO4

In the *Setup BX-HSIO* window, select (a) the **Filters** button. The on-board discrete inputs on the BX-HSIO1 and BX-HSIO2 can be configured to use input filtering.

1. Input Filters, continued
Filter Setup

Filters are typically used on inputs that are operating in electrically noisy environments to remove "false positives". This is accomplished by requiring the input signal remain above the input hardware threshold level longer than the filter time so the CPU will see that input as ON. Once ON, it must be OFF for more than the filter time before the CPU will see that input as OFF.
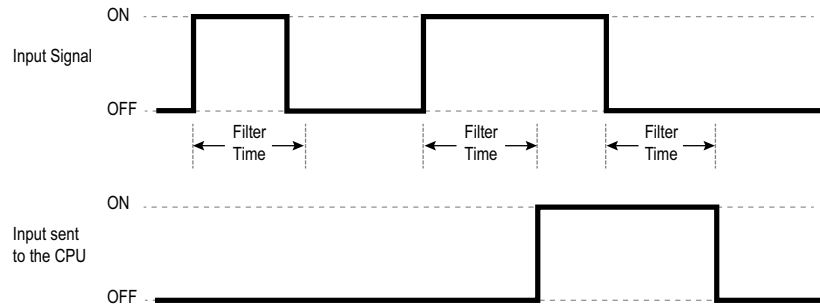
Clicking on **Filters** opens the *Setup Discrete Input Response Times* dialog box (below).

Choose Preferred Filter Scale – Sets format for all of the Inputs values entered in the form.

NOTE: *Be sure to select the Filter Scale before entering values in fields. If you change the Preferred Filter Scale after entering values then any values that are not valid in that scale will be set to 0.*

1. Input Filters, continued

The Filter Scale can be specified in the following formats:

- For BX-HSIO1 and BXHSIO2:

    - A frequency in the range of 0–250000 Hz.

    - A time value in milliseconds in the range of 0–112, microseconds in the range of 0–111848, or nanoseconds in the range of 0–111848093.

    - The number of 13.33 nanosecond clocks in the range of 0–8388607.

- For BX-HSIO4:

    - A frequency in the range of 0–2000000 Hz.

    - A time value in milliseconds in the range of 0–112, microseconds in the range of 0–111848, or nanoseconds in the range of 0–111848093.

    - The number of 13.33 nanosecond clocks in the range of 0–8388607.

An input filter value of 0 will use the default filter value of 1 microsecond. Selecting one format to specify the filter value will automatically show the filter value in the other two formats.
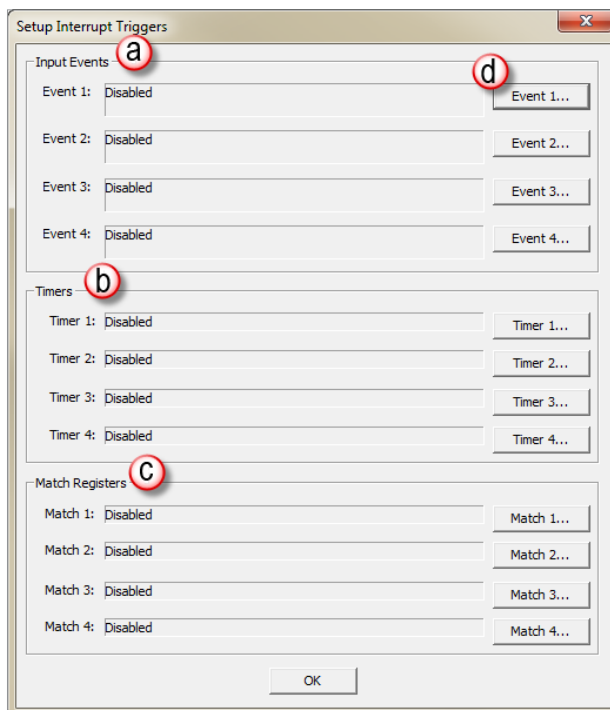
2. Interrupt Setup

BRX MPU

In the *System Configuration* window, select (a) the *BRX Local I/O* option under the *Configuration Entries* panel.
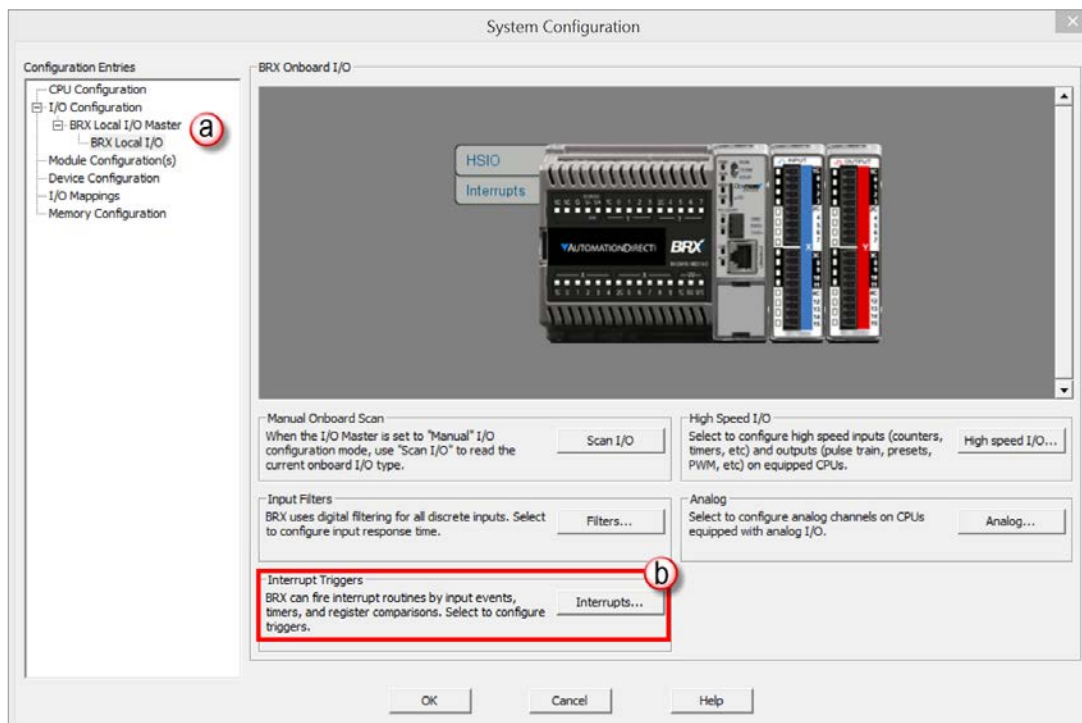
Clicking on (b) **Interrupts** opens the *Setup Interrupt Triggers* dialog box (below).

2. Interrupt Setup, continued
BX-HSIO1/BX-HSIO2/BX-HSIO4

From the system dashboard, left or right-click on the BX-HSIO expansion module and click on the **Configure Module** link. In the *Setup BX-HSIO* window, select (a) the **Interrupt Functions** tab.
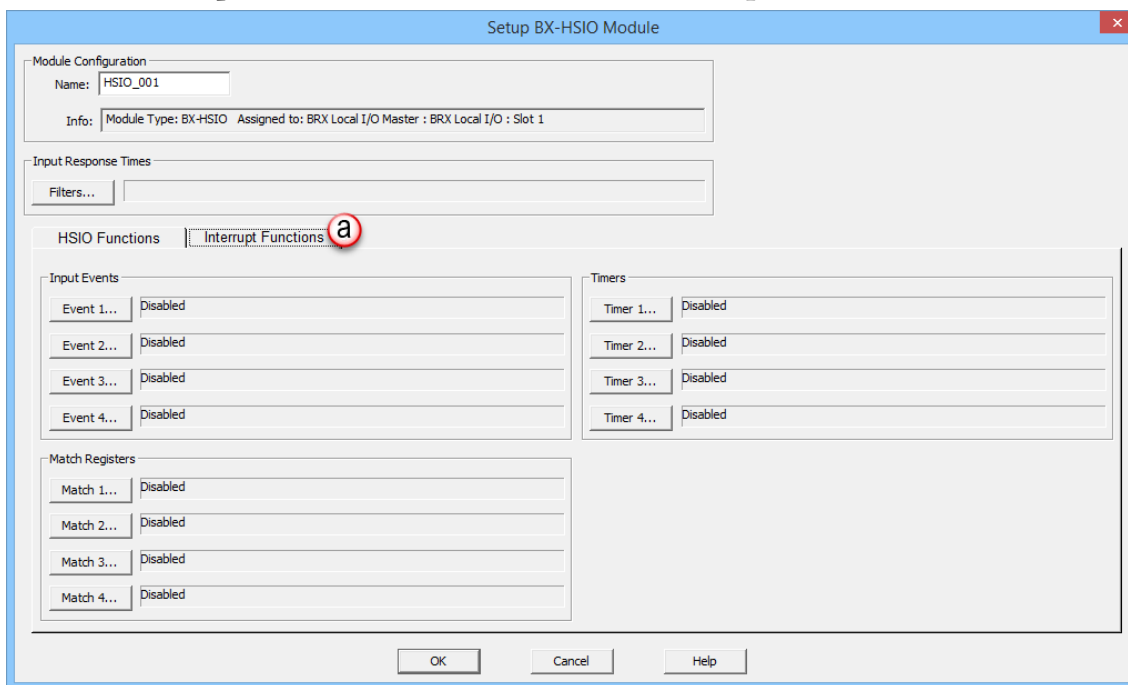
A PLC normally reads inputs at the top of the scan and writes outputs at the bottom of the scan. The ladder logic is solved after the inputs are read and after the ladder is solved the outputs are written. Because the PLC can change the amount of work it does from scan to scan, the PLC scan time will also change accordingly, which will directly affect how frequently inputs will be read and outputs will be written. Interrupts are a method of triggering an action or code segment immediately after the qualifying condition(s) becomes true, regardless of variations in the PLC scan time. In the BRX PLC, this can be accomplished by using hardware (a) *Input Events*, (b) *Timers* or (c) *Match Registers*, (matching a register count). There are 12 Interrupt triggers available: 4 Input Interrupts, 4 Timer Interrupts and 4 Match Register Interrupts.

Click on (d) **Event 1** to bring up the *Setup Input Event* dialog box.
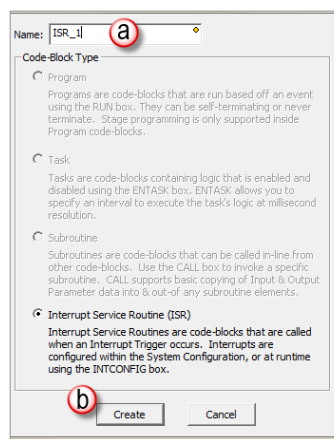
2a. Setup Input Interrupts

Input Interrupts can be used to respond to transitions of discrete inputs that occur during a PLC scan. Input Interrupts can be triggered in several different ways:

- Any (a) of the on-board discrete inputs

- Single Input (b, dropdown menu at right) OR Rising Edge, OR Falling Edge or OR Either

- Combinations of Inputs (c, dropdown menu at right) AND High Level or AND Low Level

Next, choose to assign the input event to (d) an existing *Interrupt Service Routine (ISR)* or (e) create a new ISR.
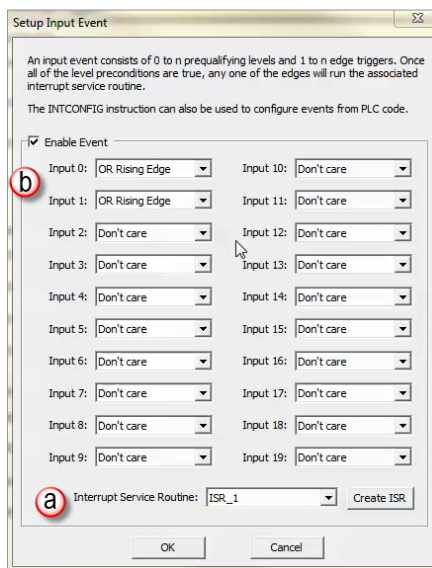
**Dropdown Menu
Interrupt Selections**
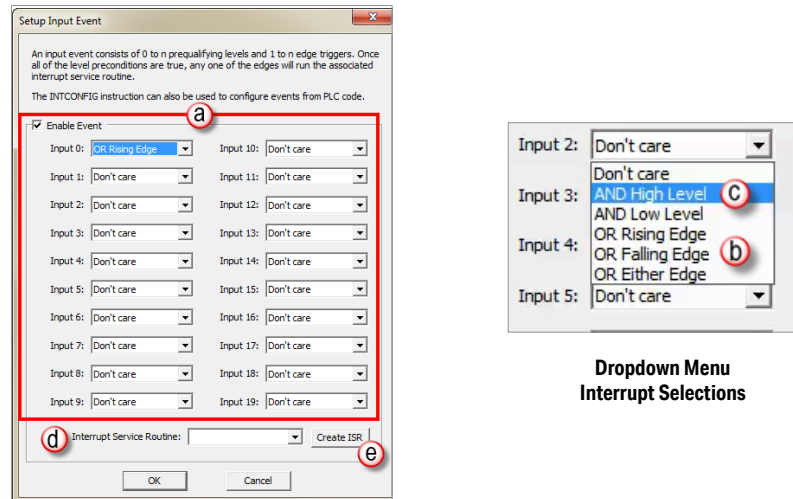
For this exercise we will click on "Create ISR" (e, above). Enter a name (a, below) for the ISR and click the Create button (b, below).

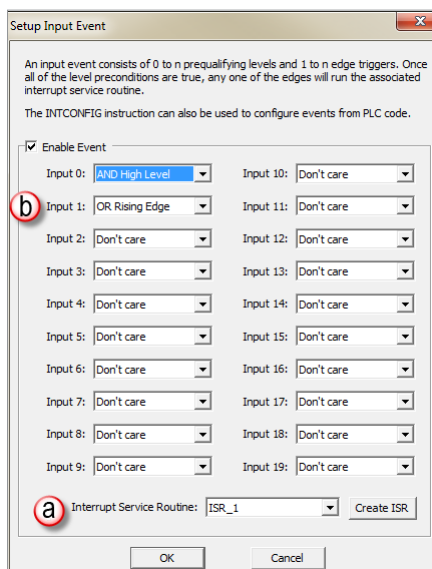## 2a. Setup Input Interrupts, continued

Input Interrupt Example 1

In the configuration below (a) of **ISR_1**, if (b) EITHER **Input 0** or **Input 1** goes from false to true, the Interrupt Service Routine **ISR_1** will run.
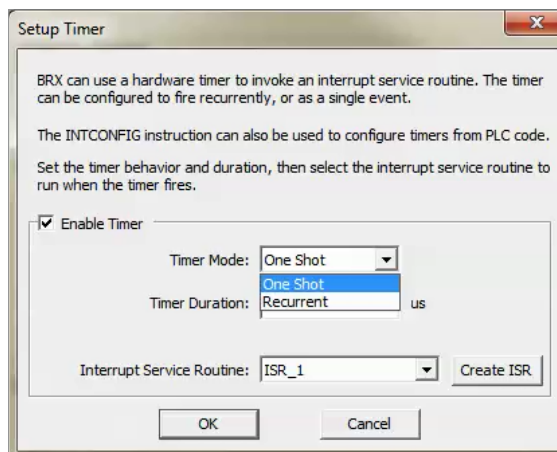

Input Interrupt Example 2

In the configuration below of (a) **ISR_1**, **Input 0** must be true and when (b) **Input 1** goes from false to true, the Interrupt Service Routine **ISR_1** will run.

2b. Setup Timer Interrupts

Timer Interrupts use a hardware timer to run an Interrupt Service Routine in situations where you need an action to occur at regular intervals that are not affected by variations in the PLC scan time. This can be a situation that require actions to occur at exact repeating intervals (Recurrent), or actions that occur after a precise amount of delay time (One Shot).

Timer Interrupts are fairly simple to setup. There are two modes: One Shot or Recurrent. The Timer Duration is in microseconds resolution. As with the Input Interrupt setup, an existing (a) ISR can be specified or a new one can be created from this dialog by clicking on (b) **Create ISR** button.
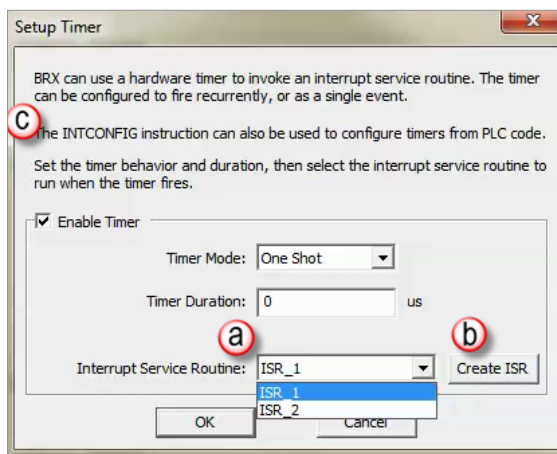

One Shot – The ISR will run only once at the time period specified after putting the PLC into Run. In order for this Interrupt to run again, the PLC will need to transition from Stop/Program to Run or you can use the (c) INTCONFIG instruction in ladder logic to trigger it again. The INTCONFIG instruction is very useful and will be discussed in more detail later.

Recurrent – the ISR will run continuously at the time period specified after putting the PLC in Run mode or it can be controlled (turned on or off) by using the INTCONFIG instruction.

NOTE: The Timer Interrupt will be triggered when the PLC goes from Stop/Program mode to Run mode. To stop this from taking place, you can use the ISRname.Inhibit bit to disable the ISR from triggering when the PLC first goes into Run mode.

2c. Setup Match Register
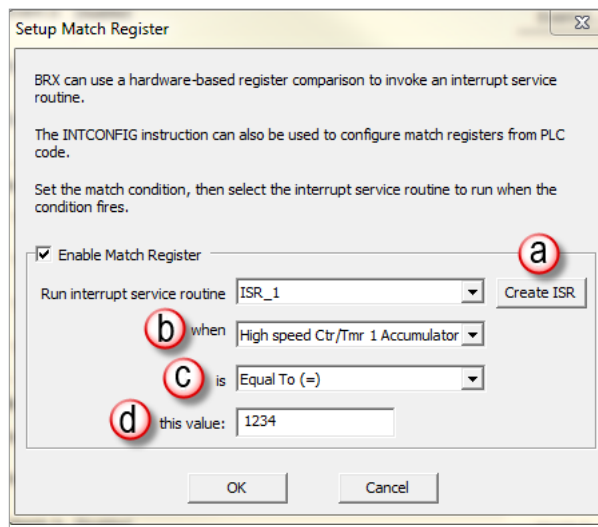
The Match Register function allows you to compare one of the hardware based registers to a value. When that condition is met, the specified Interrupt Service Routine will run. As with the other Interrupt functions, you can pick from a previously created ISR or you can create one from this dialog (a).

(b) **...when** – specifies which of the High-Speed input or output locations to use in the comparison. Choose from the following:

- High-Speed Ctr / Tmr 1 Accumulator
- High-Speed Ctr / Tmr 2 Accumulator
- High-Speed Ctr / Tmr 3 Accumulator
- Pulse Output 1 Position
- Pulse Output 2 Position
- Pulse Output 3 Position

(c) **...is** – specifies the math operator to use when performing the comparison. Choose from the following:

- Equal To
- Not Equal To
- Greater Than
- Greater Than or Equal To
- Less Than
- Less Than or Equal To

(d) **...this value** - specifies the 32-bit signed decimal constant value to compare to the register contents. This can be any constant value between -2147483648 and 2147483647.

The specified ISR will run once when the operand condition is met. For example: In the case of greater than, the ISR will run when the high-speed counter/timer or pulse output position is above the set point. It will not run again until the value has gone below the set point and then back above it again. The same is true for not equal to - the ISR will run the very first time after going into run mode if the high-speed counter/timer or pulse output position is not equal to the set point. It will not run again until the high-speed counter/timer or pulse output position becomes equal to the set point and then moves off of that value.

2d. Interrupt instructions

There are four Interrupt instructions:

- INTCONFIG (Configure Interrupt)

- INTDECONFIG (Deconfigure Interrupt)

- INTSUSPEND (Suspend Interrupts)

- INTRESUME (Resume Interrupts)

NOTE: *If you are doing ISRs, you should use one or more of the Immediate Output instructions:*
*OUTI (Out Immediate)*
SETI (Set Immediate)
RSTI (Reset Immediate).
These will help get a faster response from a Y in the ISRs.

A brief explanation of the use of these instructions will be given here. For full details on these instructions, refer to the help file.

For each instruction, there is an *Input Leg* action selection: **Power flow enabled** or **Edge triggered**. A **Power flow enabled** will lock the instruction on. So if an INTCONFIG instruction was configured as **Power flow enabled**, the rung became true and an INTDECONFIG instruction was enabled, the Interrupts would still occur. Choosing **Edge triggered** will invoke the action once and other instructions may change the current behavior.

NOTE: *If an Interrupt Trigger was created using the Interrupt Triggers setup in System Configuration, be aware that these instructions DO NOT change that configuration. The Interrupt Trigger configuration will return each time the BRX CPU transitions from PROGRAM to RUN mode.*
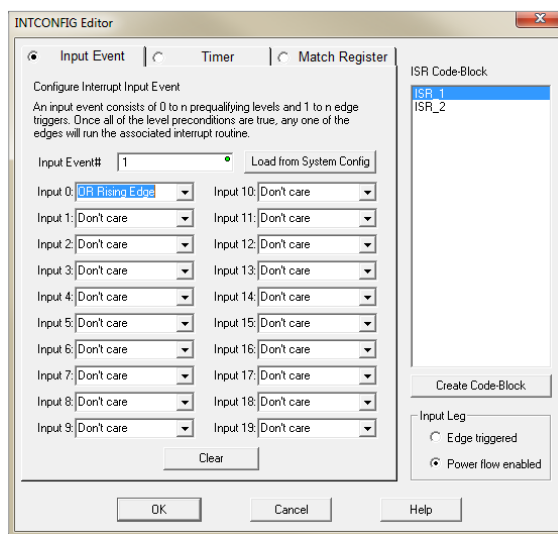
2d. Interrupt instructions, continued

INTCONFIG

The INTCONFIG instruction performs the same Interrupt setup located within the System Configuration. The instruction allows you to dynamically change the setup in applications where this may be required. One example is to calculate a Match Register value in Ladder Logic using the MATH instruction to D100, then use D100 in an INTCONFIG. If the INTDECONFIG instruction is used to disable Interrupts, this instruction is required to re-enable them.
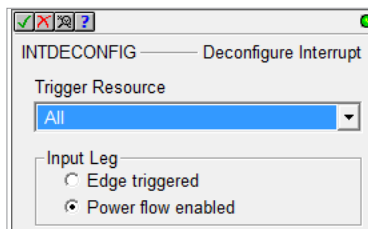

INTDECONFIG

This instruction disables the Interrupt(s) selected. Individual Interrupts may be selected as well as choosing to disable all of them. To re-enable Interrupts, the PLC must transition from Program to Run or the INTCONFIG instruction must be used. To temporarily disable Interrupts, use the INTSUSPEND instruction.

INTSUSPEND

The INTSUSPEND instruction temporarily suspends the interrupts and does not allow the ISRs to run. To enable the interrupts to function again, use the INTRESUME instruction. When interrupts are suspended using the INTSUSPEND instruction and one or more triggers take place, only one iteration of the interrupt will be available to be triggered after it is no longer suspended.
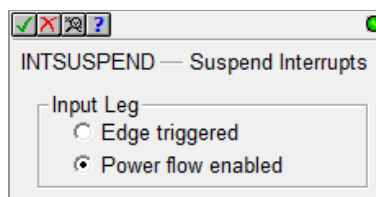
INTRESUME

The INTRESUME instruction resumes the normal processing of suspended interrupts. When the INTRESUME instruction is enabled, if one or more than one trigger took place while suspended, only one will be triggered when it resumes.
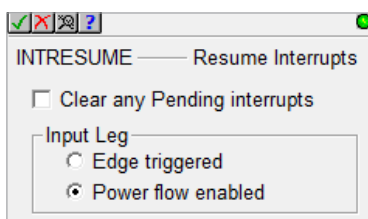
Clear any Pending interrupts: When enabled, it resets the trigger that took place while suspended. For example, if one or more triggers take place while suspended and the Clear any Pending Interrupts was not enabled, when interrupts are resumed, it will immediately execute the interrupt routine one time. If, however, the Clear any Pending Interrupts was enabled, any trigger that took place while suspended would be reset and when interrupts resume, it would start the normal processing of the interrupts.

2d. Interrupt instructions, continued
Interrupt Service Routines

Only a brief explanation will be given of the *Interrupt Service Routine*. This topic is defined in the help file as well.

When an *Interrupt Service Routine* (ISR) is created, a structure is created. The members of the structure are as follows:

- **.ExecutionTime** – Time, in microseconds, it took to run the ISR the last time it ran.
- **.HasRun** – Should be on if the ISR has run at least once since the last Program to Run transition.
- **.Inhibit** – Enabling this bit will prevent the ISR from running. The actions do NOT get queued when this bit is enabled and the hardware interrupt becomes true for this ISR.
- **.Latency** – Time in microseconds elapsed between when the hardware Interrupt occurred and when the ISR execution began.
- **.RunCounter** – Indicates how many times the ISR has run since the past Program to Run transition.

These structure members may be helpful in troubleshooting the process when using Interrupts.

3. High-Speed Counting, Timing and Pulse Catch

  a. Counters

  b. Timers

  c. Pulse Catch

  d. Table Driven Outputs

BRX MPU

There are three functions that can be configured as a counter, timer and/or pulse catch. To open the configuration window, click one of the **Function** buttons

3. High-Speed Counting, Timing and Pulse Catch, continued
BX-HSIO1/BX-HSIO2/BX-HSIO4

There are four functions that can be configured as a counter, timer and/or pulse catch. To open the configuration window, click one of the **Function** buttons

Setup BX-HSIO Module

Module Configuration

Name: HSIO_001

Info: Module Type: BX-HSIO   Assigned to: BRX Local I/O Master : BRX Local I/O : Slot 1

Input Response Times

Filters...

| HSIO Functions | Interrupt Functions |

Input Functions (Counter/Timer/Pulse Catch)

Function 1...	Disabled
Function 2...	Disabled
Function 3...	Disabled
Function 4...	Disabled

PWM Outputs

PWM 1...	Disabled
PWM 2...	Disabled
PWM 3...	Disabled
PWM 4...	Disabled

Axis/Pulse Outputs
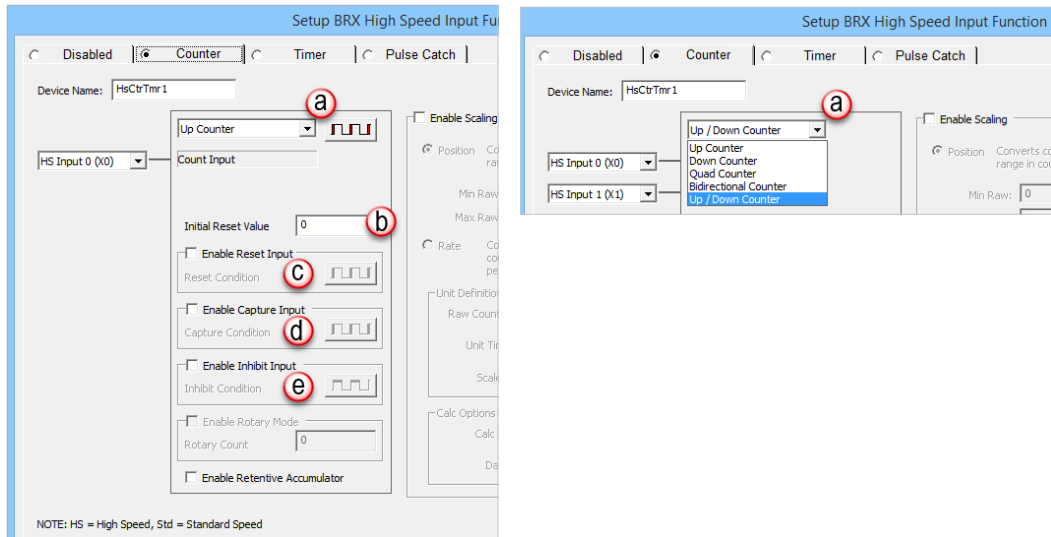
	@HSIO_001_Axis0 - Virtual Axis
Axis 1...	@HSIO_001_Axis1 - Virtual Axis (Outputs disabled)
Axis 2...	@HSIO_001_Axis2 - Virtual Axis (Outputs disabled)
Axis 3...	@HSIO_001_Axis3 - Virtual Axis (Outputs disabled)

Table Driven Outputs

Table 1...	Disabled
Table 2...	Disabled
Table 3...	Disabled
Table 4...	Disabled

OK   Cancel   Help

3a. Counters

A structure is created when this function is used. The name of the structure is configurable in the **Device Name** field. For the description of this function we will use the default name *HsCtrTmr1*.

(a) The BRX PLC allows for either 3 High-Speed Counters or 3 High-Speed Timers. Counters may be used in 5 different configurations with different options for the "Edge" to count:

Up Counter (Rising Edge, Trailing Edge or Both): Uses a single input and increments the count in the *$HsCtrTmr1.Acc* register.

Down Counter (Rising Edge, Trailing Edge or Both): Uses a single input and decrements the count in the *$HsCtrTmr1.Acc* register.

Quad Counter (1X, 2X or 4X): Uses 2 inputs and counts in both positive and negative directions based upon which of the inputs is 'leading'.

Bidirectional Counter: Uses 2 inputs. The first input is the count input. The second input determines whether the count is incrementing (when input is low) or decrementing (when input is high).

Up/Down Counter: Uses 2 inputs. The first input increments the count. The second input decrements the count.

(b) **Initial Reset Value**: The *$HsCtrTmr1.Acc* register will be loaded with the value configured in this field when a reset of the Counter occurs.

(c) **Reset Input**: Optionally, specify one High-Speed discrete input that will stop the current count operation. When this condition is met, the *$HsCtrTmr1.Acc* register will be loaded with the value specified in the (b) *Initial Reset Value* field. The Input can be configured to indicate a true condition when the Rising Edge occurs, the Falling Edge, High Level or Low Level. In addition to this hardware reset input, there is a software reset available that is performed through the structure fields named *$HsCtrTmr1.EdgeReset* or the *$HsCtrTmr1.LevelReset*. Any time the hardware level reset input is ON or the software level reset structure field is ON, the structure field *$DeviceName.AtResetValue* will be ON.

(d) **Enable Capture Input**: Optionally, specify one High-Speed discrete input that will capture the current count value when the specified input signal changes states. When this condition is met AND when the structure member *$HsCtrTmr1.EnableCapture* is true, the current value that is in the *$HsCtrTmr1.Acc* register will be loaded into the *$HsCtrTmr1.CapturedValue* register. When this occurs, the structure member *$HsCtrTmr1.CountCaptured* will be true. To capture again, the
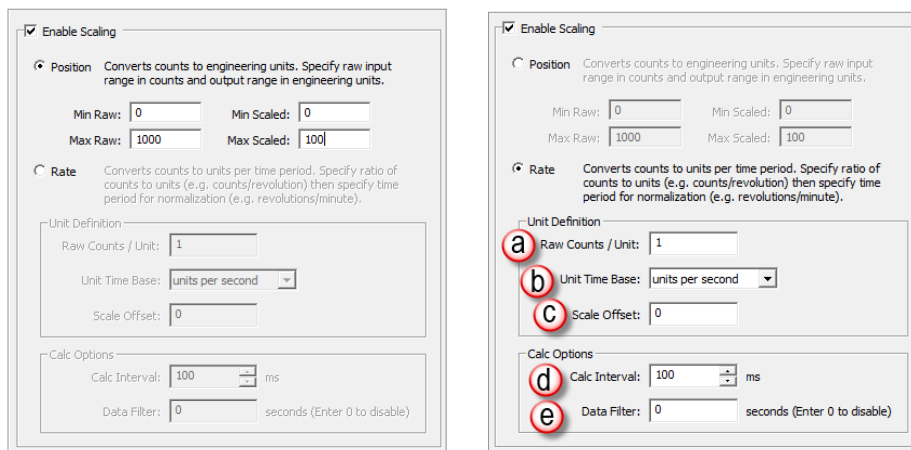
3a. Counters, continued

$HsCtrTmr1.EnableCapture bit must be turned OFF and back ON. This input can be configured to indicate a true condition on Rising Edge or Falling Edge.

(e) **Inhibit Input**: Optionally, specify one High-Speed discrete input that will cause the counter to stop counting input pulses. When the Inhibit Input is ON, the value in the *$HsCtrTmr1.Acc* register will be maintained, and any new pulses are not counted. When the Inhibit Input is no longer active, new pulses will be counted. The Input can be configured to indicate a true condition on High Level or Low Level.

(f) **Rotary Mode**: This mode is available only for Quad, Bi-Directional and Up/Down Counters. Input pulses that originate from a rotary source are expected to generate count values that wrap at a certain count value. One positive pulse at the maximum value will wrap the Current Count value to 0. One negative pulse at 0 will wrap the Current Count to the maximum value. The *Rotary Count* specifies the total number of counts in the rotary range. The range of count values will be from 0 to (Rotary Count – 1).

(g) **Enable Retentive Accumulator**: Optionally, make the current count value in the accumulator retentive, meaning that it will retains its current value through a loss of system power. **For this option to work correctly the high-speed counter's associated structure also must be marked as retentive in the Memory Configuration as shown below:**
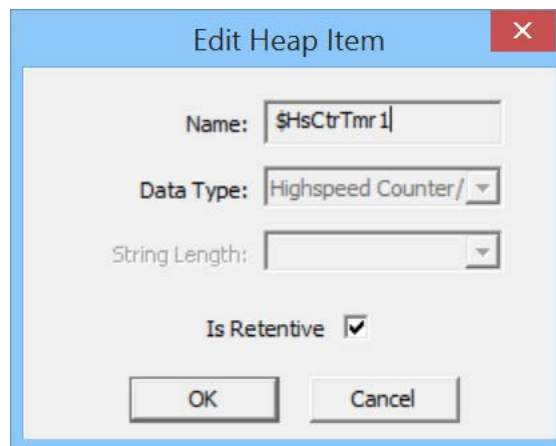
Counter Scaling

3a. Counters, continued

Position

Converts "raw" pulse count values to "engineering units" of distance using linear interpolation. Enter the **Min Raw**, **Max Raw**, **Min Scaled** and **Max Scaled** values and the scaling function will derive a distance or position value. That result will be placed in the *$HsCtrTmr1* structure member *$HsCtrTmr1.ScaledValue*.

Rate

Converts "raw" pulse count values to "engineering units" by sampling the accumulated count value over a period. Rate scaling of a pulse train is preferred over Interval Scaling of the time between pulses for frequencies over 5kHz. Because this form of scaling renders a velocity, the result of the scaling operation will be some distance unit (e.g. inches, feet, revolutions, etc.) per some time unit (seconds, minutes, hours). The scaled value will be placed in the structure member *$HsCtrTmr1.ScaledValue*.

NOTE: *The raw value is used for Match Register functionality. However, the Preset Command table and PLS table can use the scaled value.*

Rate scaling is a measurement of distance over time. The following parameters are used in calculating the *.ScaledValue*:

(a) **Raw Counts / Unit**: The raw count value that would comprise 1 scaled unit value. So for calculated RPMs of an encoder, this might be the Pulses Per Revolution of the encoder.

(b) **Unit Time Base**: A time base for the scaled unit value. In the example of calculating RPMs, this value would be units per minute.

(c) **Scale Offset**: Simply a value added to the resulting *.ScaledValue*.

(d) **Calc Interval**: Specifies how often (in milliseconds) the rate calculation is performed. The higher the value, the lower the impact on performance to the system. The calculation should be performed no faster than the process requires. If the application generates very slow pulse signals, consider using the Interval Scaling, discussed later in this chapter.

(e) **Data Filter**: Entering a value into this field will apply a time constant filter to a rolling average resulting in a 'smoother' value.

Rate Scale Example

An RPM (revolutions per minute) value is needed for a motor that has an 800-ppr (pulses per revolution) encoder that is wired to the High-Speed I/O inputs on a BRX MPU.

The encoder yields 800 pulses in 1 revolution. If it makes 1 revolution in 1 minute, that would yield 800 pulses in 1 minute's time. If the BRX MPU sees 800 pulses in the time span of 1 minute, that would be 1 RPM. The **Raw Counts / Unit** is equal to 800 (meaning 800 pulses per minute = 1 RPM).

Since the desired unit is RPM (revolutions per minute) a **Unit Time Base** of **Minutes** is selected.

Scale Offset is zero because when there are no pulses being generated we want the value to be 0 RPM.

3b. Timers

There are 2 types (a) of High-Speed Timers for the BRX PLC: **Edge Timer** (1 Input) and **Dual Edge Timer** (2 Inputs). Edge timers measure the amount of time between pulses. Using an edge timer instead of pulse counting for pulse rates below 5kHz is a better option when the desired result is a scaled value representing a speed or a rate.
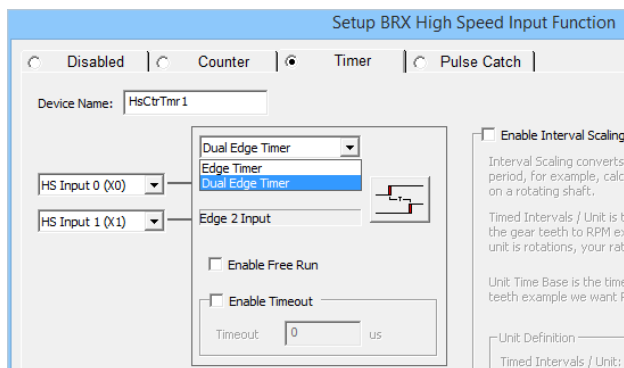
Edge Timer: With the Edge Timer, measurements can be taken in 4 possible ways:

- From rising edge of one pulse to the rising edge of the next pulse

- From rising edge to falling edge of the same pulse

- From falling edge of one pulse to the rising edge of the next pulse

- From falling edge of one pulse to the falling edge of the next pulse

Dual Edge Timer: With the Dual Edge Timer, measurements can be taken in 4 possible ways:

- From the rising edge of input 1 to the rising edge of the subsequent pulse of Input 2

- From the rising edge of input 1 to the falling edge of the subsequent pulse of Input 2

- From the falling edge of input 1 to the rising edge of the subsequent pulse of Input 2

- From the falling edge of input 1 to the falling edge of the subsequent pulse of Input 2

When a High-Speed Timer is configured, a structure is created. To use the timer, the *$HsCtrTmr1.EnableTimer* bit must be enabled. The BRX PLC will then look for the first pulse. When the first pulse is read, the timer begins. The current time value (in microseconds) can be seen in the *$HsCtrTmr1.Acc* register. When the second pulse has been read, the time value is moved into the *$HsCtrTmr1.LastTime* register and the timer stops. To run again, disable the *$HsCtrTmr1.EnableTimer* bit and re-enable. There are 2 other bits to indicate the state that the timer is in. The *$HsCtrTmr1.TimerStarted* indicates that the timer is active. The *$HsCtrTmr1.TimerComplete* bit indicates that the timer has completed.

Enable Free Run: When the (b) **Enable Free Run** checkbox is selected, the edge timer automatically re-arms itself after each timing measurement. This results in a continuous (moving average) measurement. With this mode selected, the *$HsCtrTmr1.EnableTimer* bit only needs to be enabled.

NOTE: *The timer will 'free run' and will not stop in between pulses. The BRX PLC is just grabbing snapshots of the configured pulse timing.*

3b. Timers, continued

Enable Timeout: This (c) feature gives an indication that the first edge has been received but the second edge has not been received within the time specified. When this takes place, the structure member *$HsCtrTmr1. Timeout* is set ON. The timeout value is entered in microseconds. When the timeout occurs, it will not successfully complete the timer even though the second edge may eventually arrive. The *$HsCtrTmr1. EnableTimer* bit must be disabled and re-enabled.

Interval Scaling

Interval scaling uses the time between input pulses to calculate the frequency of the input pulses. It then normalizes that frequency to a desired time base. Interval scaling is typically used for units of speed, flow, velocity, etc., and is preferred over Rate Scaling for input pulse frequencies lower than 5kHz. Because this form of scaling renders a velocity, result of the scaling operation will be some distance unit (e.g. inches, feet, revolutions, etc.) per some time unit (seconds, minutes, hours).

The Interval Scaling operation requires the following three parameters:

Timed Intervals / Unit: the number of pulses over a certain period, for example, the number of teeth on a gear, or the number of index marks on a belt.

Unit Time Base: the time unit you want the interval normalized to, either units per Second, Minute or Hour.

Scale Offset: a fixed value to add to each frequency calculation.

You can optionally enable a **Data Filter** that will perform a rolling average of the **Interval Scaled** values over the time (in seconds) specified for the **Data Filter**. A value of 0 disables the use of the **Data Filter**.

Interval Scale Example #1

A value for the RPM (revolutions per minute) is needed for a motor whose encoder is wired to the High-Speed I/O Inputs on a BRX CPU. The encoder produces 800 pulses per revolution.

The encoder yields 800 pulses in 1 revolution. Each time it makes 1 revolution in 1 minute, that would yield 800 pulses in 1 minute's time. If the input sees 800 pulses in the time span of 1 minute, that would be 1 RPM. The **Time Intervals / Unit** is equal to 800 (meaning 800 pulses per minute = 1 RPM).

Since the desired unit is RPM (revolutions per minute) a **Unit Time Base** of **Minutes** is selected.

Scale Offset is zero because when there are no pulses being generated we want the value to be 0 RPM.

3c. Pulse Catch

The BRX high-speed *Pulse Catch* will generate an output that can be seen by the PLC scan in response to input pulses that are too fast to reliably be seen otherwise. The output can be ON for one PLC scan or ON for a fixed number of milliseconds. Once the *Pulse Catch* has been configured its operation is completely automatic. The associated structure member *$HsCtrTmr1.PulseCatchOut* will come ON each time a qualifying *Pulse Catch* event is processed.
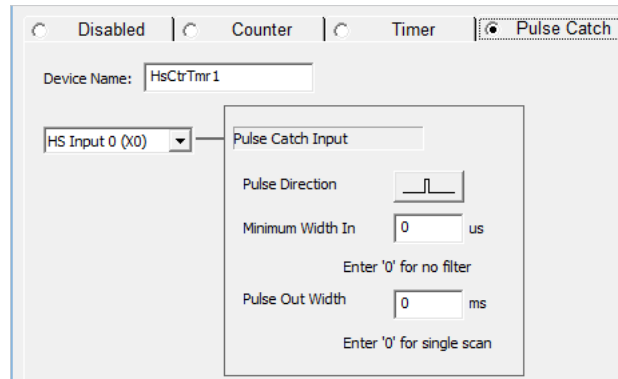
Device Name: is the name given to this high-speed I/O counter, timer or pulse catch function. This name will also be used as the name of the structure you will use to interact with this function in the ladder logic project.

Pulse Catch Input: selects which of the on-board inputs to use.

Note: You can select any of the on-board discrete inputs. Verify which inputs are high-speed for the MPU model you are using.

Pulse Direction button: specifies whether to monitor the input for positive pulses or negative pulses. Clicking the button cycles through the possible selections.

Minimum Width In: specifies the minimum pulse width (in microseconds) that will be considered a valid input to generate a Pulse Catch event.

Pulse Out Width: specifies how long the generated output, structure member *$HsCtrTmr1.PulseCatchOut*, will be ON each time a Pulse Catch event is generated. A value of 0 means the output will be ON for one complete PLC scan. Any other positive value means the output will be ON for that number of milliseconds. The structure member *$HsCtrTmr1.OutputTime* contains the amount of time remaining for the output to be ON.

3d. Table Driven Outputs

Table Driven Outputs are a method to control outputs at high speed based upon set points from a high-speed counter, timer or axis position. Typically, controlling outputs from a set point in ladder would incur 'jitter' delays from one scan to the next. When controlling outputs at High-Speed, the ladder scan variation may produce undesired changes in response from one scan to the next. Using table driven outputs will eliminate this ladder scan variation.
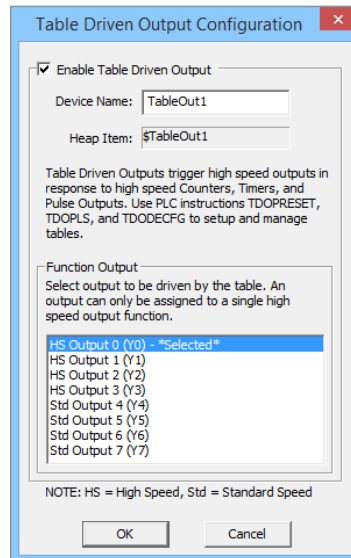
There are 2 methods of controlling table driven outputs: Preset Tables or Programmable Limit Switch (PLS). The *TDOPRESET* instruction is used for Preset Tables and the *TDOPLS* instruction is used for PLS.

Preset Tables

Preset Tables will always run in the order the steps appear in the table, from first to last. This means that it must always be known whether the count will increment or decrement and at what point it will do this. These steps compare the current count value of the specified Master Register to the Preset Count in the Step, and when the count values match, the step's action is performed on the selected discrete output and the next step in the table becomes the active step. If unexpected direction changes in count may occur, the Programmable Limit Switch function may be a better choice for that application.

As mentioned above, the table will always run steps from first (top of table) to last (bottom of table). To restart the table from the top, the **Reset Table & Acc** function should be used as one of the steps in the table.

There are 6 Preset functions to choose from in the table:

Set: This function will turn the Table Driven Output ON. A Reset must be used to turn the output OFF.

Reset: This function will turn the Table Driven Output OFF.

Pulse ON: This function will turn the Table Driven Output ON for the specified "Pulse Time" (in microseconds). At the end of the specified Pulse Time, the output will turn OFF.

Pulse OFF: If the output is ON, this function will turn the Table Driven Output OFF for the specified "Pulse Time" (in microseconds). At the end of the specified Pulse Time, the output will return to ON.

Toggle: This function will set the Table Driven Output to the opposite state from what it is currently at. If the output was ON, this function will turn it OFF. If the output was OFF, this function will turn it ON.

3d. Table Driven Outputs, continued

Reset Table & Acc: Performs a reset of the Master Register which sets its current count value to the Initial Reset Value specified in the Timer/Counter Function setup, and sets the current step in the Preset Table to Step 0.

When a Preset Table is configured, a structure becomes available for use in control and monitoring. The members of the structure are as follows:

.EnableOutput (Bit): This Bit is automatically set ON when the Preset Table is first loaded, and automatically turned OFF when the Table Driven Output is deconfigured. The ladder logic program can manually turn this Bit OFF to stop the table from writing it's state data to the Table Driven Output without having to use the TDODECFG instruction. While this bit is ON, the Preset Table updates the Table Driven Output.

.OutputState (Bit, Read Only): This bit is ON when the Table Driven Output is ON.

.StepNumber (Signed Byte): The zero-based step number from the table that is currently active. A step number of -1 indicates the Preset Table is either in Level Reset or is unconfigured.

.InputValOffset (Signed DWord): The current count value from the Master Register can be adjusted by a fixed amount before the comparison in the step is performed by entering that offset value here.

.ResetEdge (Bit): Turn this Bit ON to reset the Preset Table to Step 0, the PLC will automatically turn this bit back off.

.ResetLevel (Bit): Turn this Bit ON to reset the Preset Table to Step 0. Leaving the Bit ON will hold the Table in Reset until this Bit is turned OFF.

For more information on using the TDOPRESET instruction and the Preset Table function, reference the help file.

Programmable Limit Switch (PLS)

Unlike the Preset Table, the PLS (Programmable Limit Switch) will act upon the output whenever the High-Speed I/O Source Master Register is within the configured entry points no matter the direction it may have entered the range specified. The table contains a series of start and stop positions like the cams on a shaft. These cam positions are compared to the current count value of the specified Master Register. When the count value falls between any of the positions in the table the discrete output is turned ON or OFF as specified in the table.

The entry requires specifying the low range (**ON when Greater Than or Equal to**) and high range (**and Less Than**) fields. The **On when Greater than or equal to** field must always be a lower value than the **and Less Than** field.

Multiple entries can be configured to control the output at different ranges but they cannot overlap each other. The PLS can control the output so that its default state is ON and the entries configured will turn the output OFF within those ranges. The same structure that is used for the Preset Table is used for the PLS Table with a few minor changes in behavior. The explanations are given below:

.EnableOutput (Bit): This Bit is automatically set ON when the PLS Table is first loaded, and automatically turned OFF when the Table Driven Output is deconfigured. The ladder logic program can manually turn this Bit OFF to stop the table from writing its state data to the Table Driven Output without having to use a TDODECFG instruction. If this Bit is ON the PLS Table will update the Table Driven Output.

.OutputState (Bit, Read Only): This bit is true when the Table Driven Output being controlled by the PLS is currently true.

3d. Table Driven Outputs, continued

.StepNumber (Signed Byte): The zero-based step number from the table that is currently active. A step number of -1 indicates the PLS Table is between ON positions or is unconfigured.

.InputValOffset (Signed DWord): The current count value from the Master Register can be adjusted by a fixed amount before the comparison in the step is performed by entering that offset value here.

.ResetEdge and *.ResetLevel*: Not used with the PLS function.

For more information on using the TDOPLS and PLS function, reference the Help file.

Deconfigure Table Driven Output

Once a TDOPLS (Programmable Limit Switch on Table Drive Output) or TDOPreset (Run Preset Table on Table Drive Output) has been enabled for a Table-Driven Output, it will continue to control that output even if the input logic is no longer ON. Use the *Deconfigure Table Driven Output* (TDODECFG) instruction to stop the Preset table or PLS table from controlling the Table-Driven Output.

4. Outputs

    (a)   Axis/Pulse Outputs

    (b)   Pulse Width Modulation (PWM) Outputs

BRX MPU

BX-HSIO1/BX-HSIO2

4a. Axis/Pulse Outputs

The Axis/Pulse Output Configuration window allows you to specify the type of pulse train (mode) to output as well as the physical output points to use for this Axis. The software will indicate and warn if there are conflicts specified for the output points selected in this or other axis configurations.

There are four Pulse Output Modes:

1. **Virtual** (a): Axis can execute profiles for master/slave operations with other axes, or can trigger Table Driven Outputs or Match Register interrupts, but does not drive physical I/O.

NOTE: A **Virtual** axis will not generate pulses to physical outputs of the PLC. Convenient for testing.

---

2. **Step/Direction** (b): The output specified for **Function Output 1** will pulse at the speed and/or amount (Position) specified. The output specified for **Function Output 2** will be low for a positive position value move or high for a negative position value move.

4a. Axis/Pulse Outputs, continued

3. **CW/CCW** (c): The output specified for **Function Output 1** will pulse at the speed and/or amount (Position) specified when the Position value is positive. The output specified for **Function Output 2** will pulse at the speed and/or amount (Position) specified when the Position value is negative.

4. **Quadrature** (d): In Quadrature mode, both outputs specified will pulse at the speed and/or amount (Position) specified in 4X fashion (both leading and trailing edges are considered a pulse). If the position value specified is a positive value, the output specified for **Function Output 1** will lead. If the position value specified is a negative value, the output specified for **Function Output 2** will lead.

When an Axis is created, a structure is available for use in control and monitoring. Each member takes the form of *$Axisn.member*, where "*n*" is the Axis number (0 to 3) and "*member*" is the element word or bit referenced. Each member and an explanation of that structure are listed below.

.TargetVelocity (Signed DWord, Read/Write): When using an AXVEL instruction, this is the target Velocity (When using an AXPOSTRAP or AXPOSSCRV instruction, the velocity configured in the AXCONFIG instruction is the velocity that is used).

.CurRateOfChg (Signed DWord, Read Only): This is the current velocity the Axis is using during the ramp.

.TargetPosition (Signed DWord, Read Only): This is the target position that has been configured successfully by an AXPOSTRAP or AXPOSSCRV instruction.

.CurrentVelocity (Signed DWord, Read Only): This is velocity at which the axis is currently running.

.CurrentPosition (Signed DWord, Read Only): This is the pulse count where the Axis is currently located.

.FollowingError (Signed DWord, Read Only): When an Axis is configured to use encoder feedback as the Axis position, the FollowingError is the difference between the output pulse count (TargetPosition) and the encoder input value (CurrentPosition). This value is always reported in pulse counts, not in encoder count values.

.MstSlvCoordError (Signed DWord, Read Only): This error is the difference between the position of the Master Axis and the projected location of the Slave Axis when using the AXGEAR, AXFOLLOW and AXCAM instruction.

.Suspend (Bit, Read/Write): When this Bit transitions from OFF to ON, the Axis will decelerate to a Stop. A Bit transition from ON to OFF will resume the axis motion, accelerating back to the target velocity.

.MasterEnable (Bit, Read/Write): This Bit gets enabled automatically after the AXCONFIG has completed successfully. If the bit transitions from ON to OFF, the Axis will decelerate to a stop using the Axis' Fault Deceleration Rate.

.EnableOutput (Bit, Read/Write): This Bit is automatically set ON when the Axis is properly configured. If the bit transitions from ON to OFF the Axis will immediately stop sending output pulses to the high-speed output. If the Axis is moving it WILL NOT decelerate.

.Configured (Bit, Read Only): This bit is ON when the axis has been successfully configured by the AXCONFIG instruction.

.Active (Bit, Read Only): This bit is ON when the axis is enabled and in any mode other than idle.

.AtVelocity (Bit, Read Only): This bit is ON when the axis is generating the pulses at the frequency specified by an AXPOSTRAP or AXPOSSCRV, an AXVEL or other axis instructions (TargetVelocity = CurrentVelocity).

.AtPosition (Bit, Read Only): This bit is true when the position specified by an AXPOSTRAP or AXPOSSCRV has been achieved (TargetPosition = CurrentPosition).

4a. Axis/Pulse Outputs, continued

.ScriptBusy (Bit, Read Only): This Bit will be ON when an AXSCRIPT command is waiting on the Axis engine to complete an asynchronous function, like a Wait for Discrete Limit, Wait For Bit to be ON, etc.

.Fault (Bit, Read Only): Will be ON when either Fault Limit for an Axis is ON, or when the Axis' MasterEnable has been manually turned OFF. A Reset Axis Limit Fault (AXRSTFAULT) or an Axis Configuration (AXCONFIG) instruction must be executed to clear the Fault in the Axis before the Axis will operate again.

.Mode (Unsigned Byte, Read Only): This byte indicates the current execution mode of the Axis.

.ScriptStep (Unsigned Byte, Read Only): This byte indicates which step of an AXSCRIPT instruction is currently being executed. This value is a 1-based number in the range of 1 to 254. A step number of "-1" means Done.

.Timer (Signed DWord): If an AXSCRIPT instruction has a Start Timer command, this field will contain the time remaining (in milliseconds) as it times down.

.Counter (Signed DWord): If an AXSCRIPT instruction contains a For/Next loop, this field will contain the current loop count as it counts down.

4b. PWM (Pulse Width Modulation)

The PWM function has no associated instructions. In the *PWM Output Configuration*, specify the output to control. Once this function has been configured, a structure is created. The function is used by manipulating the members of that structure. The structure members are defined below:

```
                 PWM Output Configuration              [X]

         [✓] Enable PWM

            Device Name:  [PWMOut1              ]

              Heap Item:  [$PWMOut1             ]

         Use heap item's .Enable, .PeriodScale, .Period, and
         .DutyCycle fields to control the PWM from user
         program.

         ┌─ Function Output ──────────────────────┐
         │ Select output to be driven by the PWM. An │
         │ output can only be assigned to a single high │
         │ speed output function.                  │
         │                                         │
         │ HS Output 0 (Y0) - *Selected*          │
         │ HS Output 1 (Y1)                        │
         │ Std Output 2 (Y2)                       │
         │ Std Output 3 (Y3)                       │
         │                                         │
         └─────────────────────────────────────────┘

         NOTE: HS = High Speed, Std = Standard Speed

              [   OK   ]        [  Cancel  ]
```

.EnableOutput (Bit): Set this bit ON to generate output pulses, set the bit OFF to stop generating output pulses.

.PeriodScale (Bit): This specifies the time base for the output pulses. OFF = Microseconds (μs), ON = Milliseconds (ms).

.Period (Unsigned Word): Specifies the amount of time (in microseconds or milliseconds) for one complete pulse. This can be any positive constant value from 0–65535.

NOTE: *Remember, this value is NOT a frequency specified in Hz; this is the duration (milliseconds or microseconds) of one pulse. Because Frequency and Period are reciprocals of each other, the following formulas can be used to convert a value specified in Hz to a Period value in milliseconds or microseconds:*

*Converting Hz to millisecond period = (1/ Hz) * 1000. For example: 60Hz = (1 / 60) * 1000 = 17 milliseconds.*

*Converting Hz to microsecond period = (1/ Hz) * 1000000. For example: 60Hz = (1 / 60) * 1000000 = 16667 microseconds.*

.DutyCycle (Real): The DutyCycle determines the percentage of time that the output is high versus low. 10% Duty Cycle would mean that the output is high for 10% of the Period (or cycle) and low for 90% of the Period. This can be any Real value between 0.0 and 100.0.

BRX High-Speed Examples

This section includes brief descriptions of how to implement some common motion control solutions. The information provided should give the user a good understanding of what basic steps are required to implement the desired function. Later in the chapter, we will present detailed examples that will guide you step by step, on how to read a quadrature encoder value and how to generate a trapezoid profile using High-Speed outputs.

Get Position Using an Encoder

To read the position of an encoder, follow these basic steps in the Do-more! Designer software:

1. From the Dashboard page –
   a. For MPUs: Select *High-Speed I/O (HSIO)*.
   b. For BX-HSIO: Left-click or right-click on the BX-HSIO module. Click the *Configure module* link.
2. Counter/Timer/Pulse Catch Functions – Select *Function 1*.
3. Select *Counter* – Configure the counter for Quad Counter and select *X0* and *X1* as your inputs.
4. Optional – Setup scaling or enable rotary mode.
5. Download the configuration to the BRX CPU and set it to RUN.
6. While connected with the CPU, verify that the encoder counts are appearing in a *DataView* window. Use the address *$HsCtrTmr1.acc* to monitor the accumulated encoder pulses.

Get Rate Using an Encoder

To read the rate of an encoder, follow these basic steps in the Do-more! Designer software:

1. From the Dashboard page –
   a. For MPUs: Select *High-Speed I/O (HSIO)*.
   b. For BX-HSIO: Left-click or right-click on the BX-HSIO module. Click the *Configure module* link.
2. Counter/Timer/Pulse Catch Functions – Select *Function 1*.
3. Select *Counter* – Configure the counter for Quad Counter and select *X0* and *X1* as your inputs.
4. Enable Scaling – Select *Rate*.
5. Enter the conversion parameters.
6. Download the configuration and verify the CPU is in RUN mode.
7. While connected with the CPU, verify the encoder rate values are appearing in a *DataView* window. Use the address *$HsCtrTmr1.scaledvalue* to monitor the rate value.

Measure Timing Between Pulse Edges

To measure the time between edges of a pulse, follow these basic steps in the Do-more! Designer software:

1. From the Dashboard page –
   a. For MPUs: Select *High-Speed I/O (HSIO)*.
   b. For BX-HSIO: Left-click or right-click on the BX-HSIO module. Click the *Configure module* link.
2. Counter/Timer/Pulse Catch Functions – Select *Function 2*.

BRX High-Speed Examples, continued

3. Select *Timer* – Keep the default device name, *@HsCtrTmr2*. Select *Edge Timer* function. For this test, use one of the encoder inputs, i.e. *X0*. Select appropriate options (Free-run is suggested for testing since it does not require any ladder programming to function).

4. Optionally, setup scaling if needed.

5. Download the configuration and verify the CPU is in RUN mode.

6. While connected with the CPU, verify that the pulse measurements are showing up in a *DataView* window. Use the address *$HsCtrTmr2.LastTime* to monitor the previous amount of time between pulses.

Pulse Catch

To generate an output that can be seen by the PLC scan in response to a high-speed input that is too fast, follow these basic steps in the Do-more! Designer software:

1. From the Dashboard page –
   a. For MPUs: Select *High-Speed I/O (HSIO)*.
   b. For BX-HSIO: Left-click or right-click on the BX-HSIO module. Click the *Configure module* link.

2. Counter/Timer/Pulse Catch Functions – Select *Function 3*.

3. Select *Pulse Catch* – Keep the default name, *@HsCtrTmr3*.

4. Select the *Pulse Direction*.

5. Enter the *Minimum Width In* value in microseconds.

6. Enter the *Pulse Out Width* value in milliseconds. This will be how long the *.PulseCatchOut* structure field will remain on.

7. While connected with the CPU, verify that the pulse measurements are showing up in a *DataView* window. Use the address *$HsCtrTmr3.PulseCatchOut* to monitor the previous amount of time between pulses.

Table Driven Output using a Preset Table

Control a high-speed output based upon a high-speed counter, timer or axis position.

1. Configure a High-Speed input to read encoder values. See section on "Get Position using an Encoder".

2. From the Dashboard page –
   a. For MPUs: Select *High-Speed I/O (HSIO)*.
   b. For BX-HSIO: Left-click or right-click on the BX-HSIO module. Click the *Configure module* link.

3. Table Driven Output – Select *Table 1*.

4. Table Driven Output Configuration – Enable *Table Driven Output*.
   a. This is used to trigger High-Speed outputs in response to High-Speed counters, timers and pulse outputs.
   b. Use ladder instructions TDOPRESET, TDOPLS and TDODECFG to setup and manage tables.

5. Select output *Y3*.

6. Download changes to the CPU.

7. Add a rung with the instruction TDOPRESET and a rung with the instruction TDODECFG.

8. Once the TDOPRESET is running, it takes over the discrete output assigned to *Table1*.

9. To release the discrete output from being controlled by *Table1*, use TDODECFG.

BRX High-Speed Examples, continued

Pulse Train Output – Move to a Specific Position

The following steps will explain what is needed for the BRX CPU to prompt a stepper motor, for example, to move an absolute number of steps:

1. From the Dashboard page –
    a. For MPUs: Select *High-Speed I/O (HSIO)*.
    b. For BX-HSIO: Left-click or right-click on the BX-HSIO module. Click the *Configure module* link.

2. Axis/Pulse Outputs – Select *Axis 1*.

3. Axis 1 Configuration – Keep the default Device name, *@Axis1*. Select *Pulse Output*. Select the Output Type required by your controller. For this example, we are using Step/Direction with a SureStep stepper drive. Select *Y0* for Step and *Y1* for Direction.

4. Download the configuration to the CPU.

5. AXCONFIG ladder rung – Configure the AXCONFIG instruction for Device *@AXIS1*. Use the defaults for all the fields. The move instructions use the Accel/Decel and Min/Max frequencies configured in this instruction.

6. AXPOSTRAP ladder rung – Configure the AXPOSTRAP for Device *@Axis1*. For Move Type, select Single Move and Power Flow Enabled. For Target Type, select Relative and for Position Value use the default of 1000. For Linear vs Rotary, select Linear.

7. Download program and verify the CPU is in RUN mode.

8. Trigger the AXCONFIG ladder rung – When the AXCONFIG instruction is triggered, the *MasterEnable* and the *EnableOutput* heap items will be ON. The success bit will be ON.

9. Turn on the AXPOSTRAP ladder rung – Every time the rung is powered on and left on, the instruction will send out a pulse train that matches the target position value. The *CurrentPosition* will increase from 0 to 1000 the first time the rung is turned on. The target position will display 1000. *AtPosition* turns on when the target position is reached. If you turn the rung off and back on, the *CurrentPosition* will increase from 1000 to 2000.

10. A detailed example using the AXPOSTRAP will be provided later in this chapter. For detailed information of the instructions, review the Do-more! Designer Help topics.

Pulse Train Output – To Home an Output

The following steps will explain what is needed for the BRX CPU to prompt a stepper motor, for example, to find a Home position:

1. From the Dashboard page –
    a. For MPUs: Select *High-Speed I/O (HSIO)*.
    b. For BX-HSIO: Left-click or right-click on the BX-HSIO module. Click the *Configure module* link.

2. Axis/Pulse Outputs – Select Axis 1.

3. Axis 1 Configuration – Keep the default Device name, *@Axis1*. Select *Pulse Output*. Select the Output Type required by your controller. For this example, we are using *Step/Direction* with a SureStep stepper drive. Select *Y0* for Step and *Y1* for Direction.

4. Download the configuration to the CPU.

5. AXCONFIG ladder rung – Configure the AXCONFIG instruction for Device *@AXIS1*. Use the defaults for all the fields. The move instructions use the *Accel/Decel* and *Min/Max* frequencies configured in this instruction.

BRX High-Speed Examples, continued

6. AXHOME ladder rung – Configure the AXHOME for Device *@Axis1*. For this example, we use *X2* as our *Discrete Input Limit 1*. Enter a value for *Homing Velocity*. Select the Discrete input and event type. Select the *Homing Termination*. For this example, we will use Decelerate to 0 after reaching input limit 1 and zero out the position value when Home is reached. Edge-triggered is selected for the input leg.

7. Download program and verify the CPU is in RUN mode.

8. Trigger the AXCONFIG ladder rung – When the AXCONFIG instruction is triggered, the MasterEnable and the *EnableOutput* heap items will be ON. The AXCONFIG success bit will be ON.

9. Trigger the AXHOME ladder rung – Turning on the input to the AXHOME rung, will trigger the HOME move. The input does not have to remain on for the HOME move to continue when the edge-triggered input is selected. Pulse outputs will be generated while the discrete input limit 1 is not reached. Once the discrete input limit 1 is reached, the pulse output will decelerate to 0, as configured for this example. During the HOME move, the *CurrentPosition* will increase. Once the HOME position is reached, the *CurrentPosition* is reset to 0.

Output Pulse Width Modulated (PWM) Pulses

To generate PWM outputs, follow these basic steps:

1. From the Dashboard page –
   a. For MPUs: Select *High-Speed I/O (HSIO)*.
   b. For BX-HSIO: Left-click or right-click on the BX-HSIO module. Click the *Configure module* link.

2. PWM Outputs – Select *PWM 1*.

3. PWM 1 Configuration – Enable PWM. Keep the default Device name, *@PWMOut1*. Select output Y2, for example.

4. Download the program (no ladder instructions need to be added for this test) and verify the CPU is in Run.

5. Use *Data View* to control the PWM 1 output. Use heap items *.EnableOutput*, *.PeriodScale*, *.Period* and *.DutyCycle*.

6. *$PWMOut1.DutyCycle* – Specifies the percentage of one period where the output is ON, during the remaining portion the output will be OFF. This can be any Real value between 0.0 and 100.0.

7. *$PWMOut1.PeriodScale* – Selects the time base for the output pulses, OFF = microseconds (µs), ON = milliseconds (ms).

8. *$PWMOut1.Period* – Specifies the amount of time (in microseconds or milliseconds) for one complete pulse. This can be any positive constant value from 0–65535.

NOTE: *Remember, this value is NOT a frequency specified in Hz, this is the duration (milliseconds or microseconds) of one pulse. Because Frequency and Period are reciprocals of each other, the following formulas can be used to convert a value specified in Hz to a Period value in milliseconds or microseconds:*

*Converting Hz to millisecond period = (1/ Hz) * 1000. For example: 60Hz = (1 / 60) * 1000 = 17 milliseconds.*

*Converting Hz to microsecond period = (1/ Hz) * 1000000. For example: 60Hz = (1 / 60) * 1000000 = 16667 microseconds.*

9. *$PWMOut1.EnableOutput* - Set this ON to generate output pulses, set OFF to stop generating output pulses.

Detailed Example: Configure and Test a Quadrature Input

This example walks through the steps required to get the counts from a quadrature encoder. Phase A of the encoder is wired to input X0 and Phase B of the encoder is wired to input X1.

The Basic Steps

1. Wire the encoder and connect Do-more! Designer to the BRX CPU.
2. Configure a High-Speed Input as a quadrature counter.
3. Download the program changes to the CPU and place the CPU in Run.
4. Use DataView and Trend View to verify the encoder values are being read correctly by the BRX CPU.

Equipment Used

BRX MPU with DC inputs. A quadrature encoder (open collector or totem pole) properly powered and connected to inputs X0 and X1 of the BRX.

Launch Do-more! Designer. By default, the Tip of the Day will pop up. Close it.
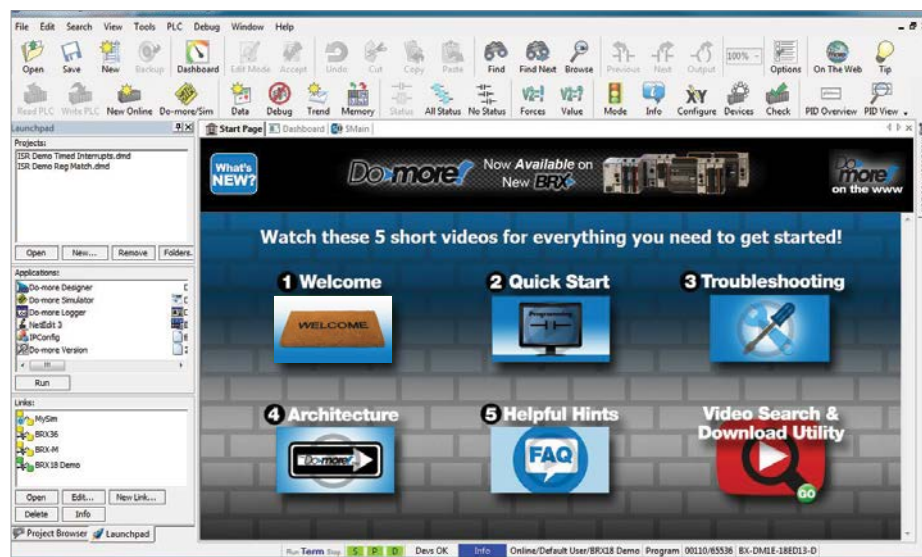
The *Select Project* window will be displayed next. Select **New Offline Project**. Launch the Do-more! Designer Software.
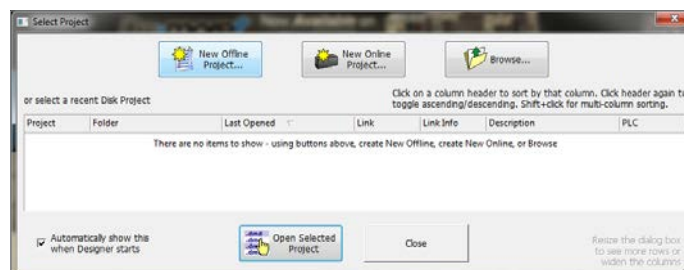
Detailed Example: Configure and Test a Quadrature Input, continued

In the *New Offline Project* window select the BRX MPU model you are working with. Rename the project as Quad Encoder in the **New Project Name** field (bottom left). Press the **OK** button to accept the selections.
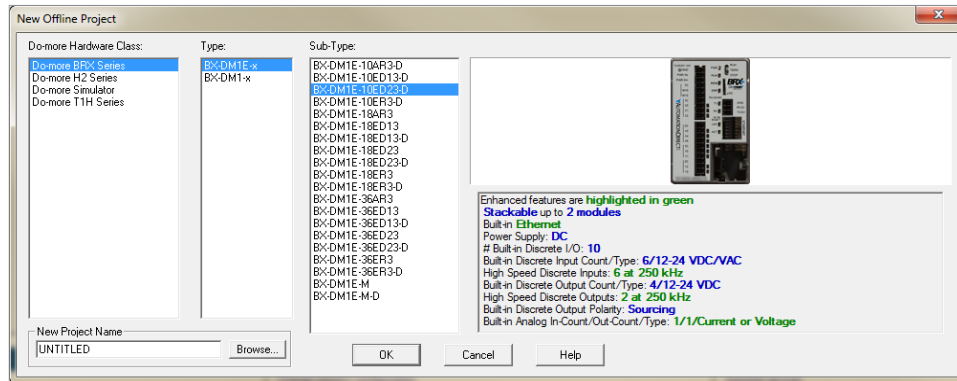
The Dashboard page is displayed and the screen will look as follows:
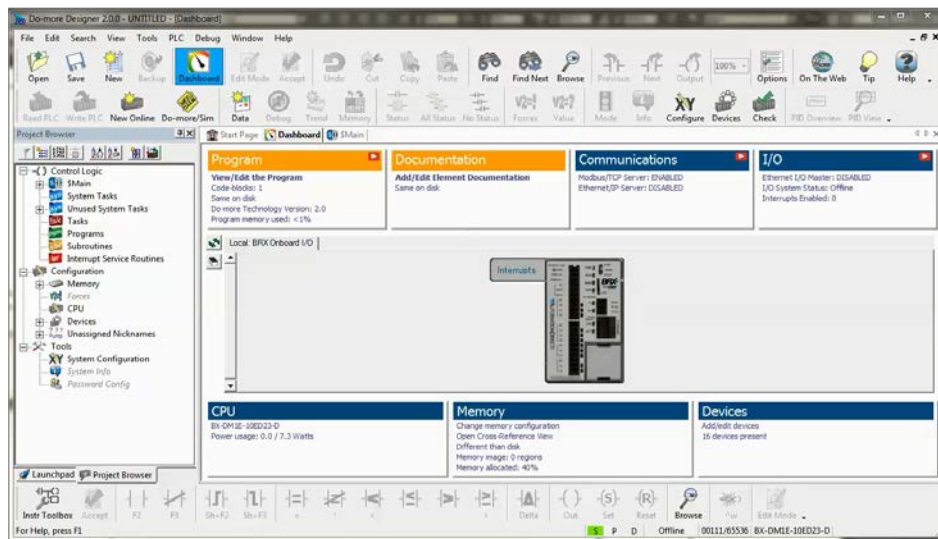
Mouse over the BRX MPU in the Dashboard BRX Onboard I/O view, orange blocks will appear around the various built-in MPU features. Select the inputs orange block (a). Select **Configure counter/timer functions** (b) to access the Setup BRX High-speed I/O page.
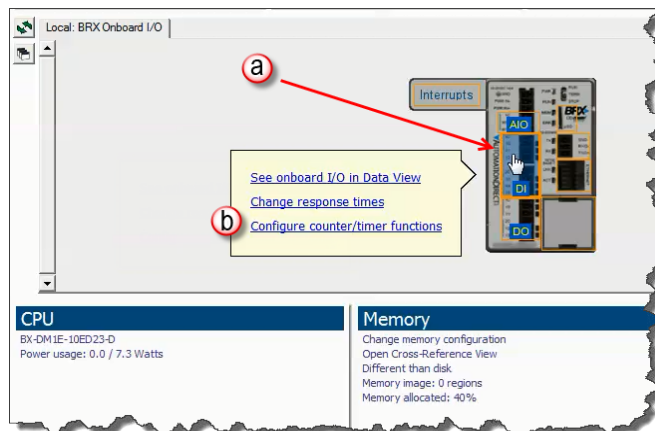
Detailed Example: Configure and Test a Quadrature Input, continued

From the Setup BRX High-speed I/O page, select **Function 1** under the *Counter/Timer Functions* section.
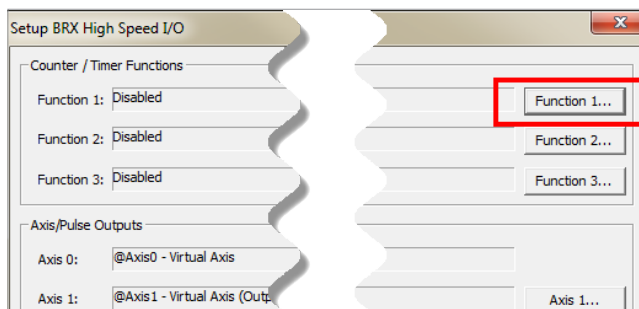
Selecting (a) **Counter** from the *Setup BRX Counter/Timer* dialog box. The dialog box will fill in with counter parameters. For this example, we will keep the default (b) **Device Name** of *HsCtrTmr1*. Select (c) **Quad Counter** from the dropdown menu, leaving it at (d) 1X counting resolution and select input (e) X0 (Phase A) and input (f) X1 (Phase B).
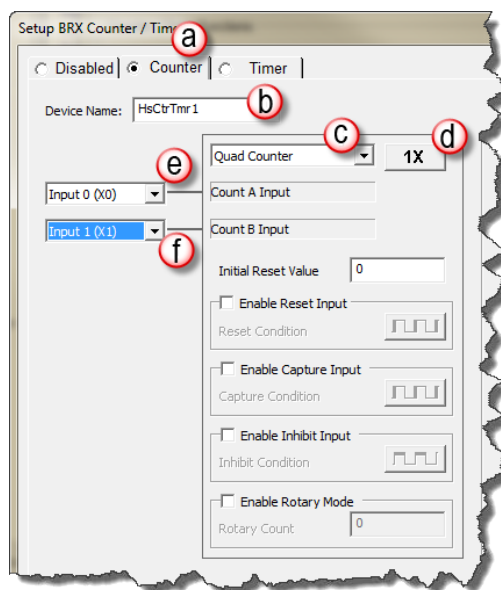
Click OK to return to the Dashboard screen. **Write** the changes to the PLC and verify you are in **Run** mode.

At this point, we have configured inputs X0 and X1 for quadrature counter high-speed inputs. No ladder is needed to monitor the counts from the encoder.

Detailed Example: Configure and Test a Quadrature Input, continued

Displaying Encoder Values via Data View

Create a New *Data View* window. From the menu select **Debug** -> **Data View** and **New**.
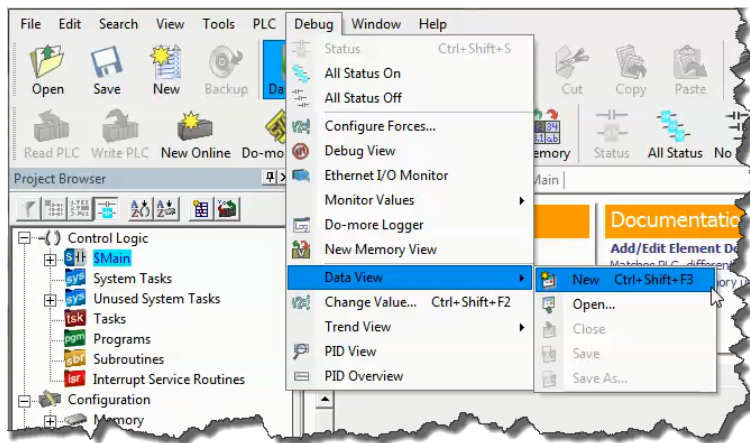
Alternately, you can use the *Data* icon on the toolbar to open a new *Data View* window.
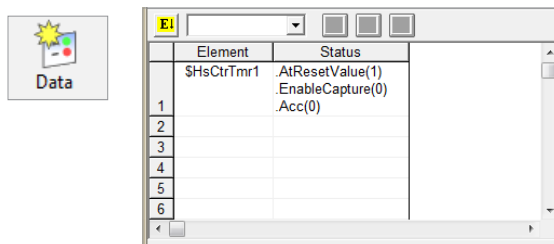
The *Data1* window is displayed. Start by typing *$HsCtrTmr1* on row 1 under the *Element* column. Make this *Data1* window wider by placing the mouse on the right edge of the window, left clicking and dragging to the right.

You can see under *Status* the Heap items *.AtResetValue*, *.EnableCapture*, and *.Acc*. The value in parenthesis is the current value. As you turn your encoder, the value for *.Acc* will change. If you move the encoder in the clockwise direction the values will increase. When the encoder moves in the counter-clockwise direction, the value decreases. (The reverse may be true depending on the wiring of the encoder phase A and phase B).

Detailed Example: Configure and Test a Quadrature Input, continued

It is possible to view one Heap item per Data View row. The available Heap items for the *$HsCtrTmr1* can be found under the **Project Browser** -> **Configuration** -> **Memory** -> **I/O** -> **Specialty** -> *$HsCtrTmr1* as shown below.
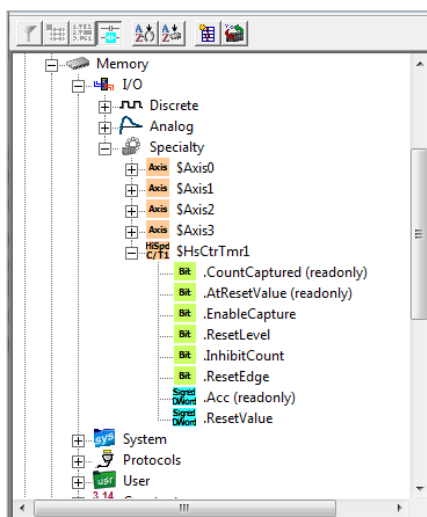
Displaying Encoder values using Trend View

Trend View is a very powerful utility, one that can be used to troubleshoot the control system. We highly recommend you use this tool to the fullest.

Create a new *Trend View* from the *Debug Menu* or the *Trend* icon on the toolbar, as shown below.
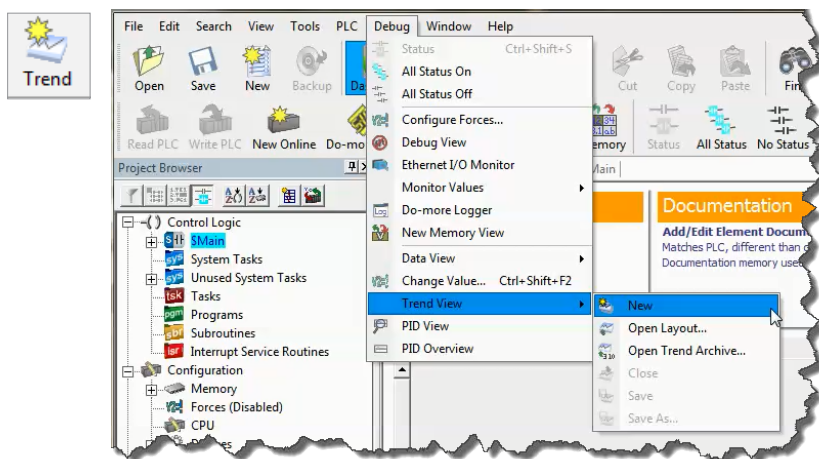
Detailed Example: Configure and Test a Quadrature Input, continued

In *Trend View*, using a single pane, add the accumulated value of the high-speed counter, *$HsCtrTmr1.Acc*. As the encoder is turned, you can see the accumulated value changing on the *Trend View* pane.
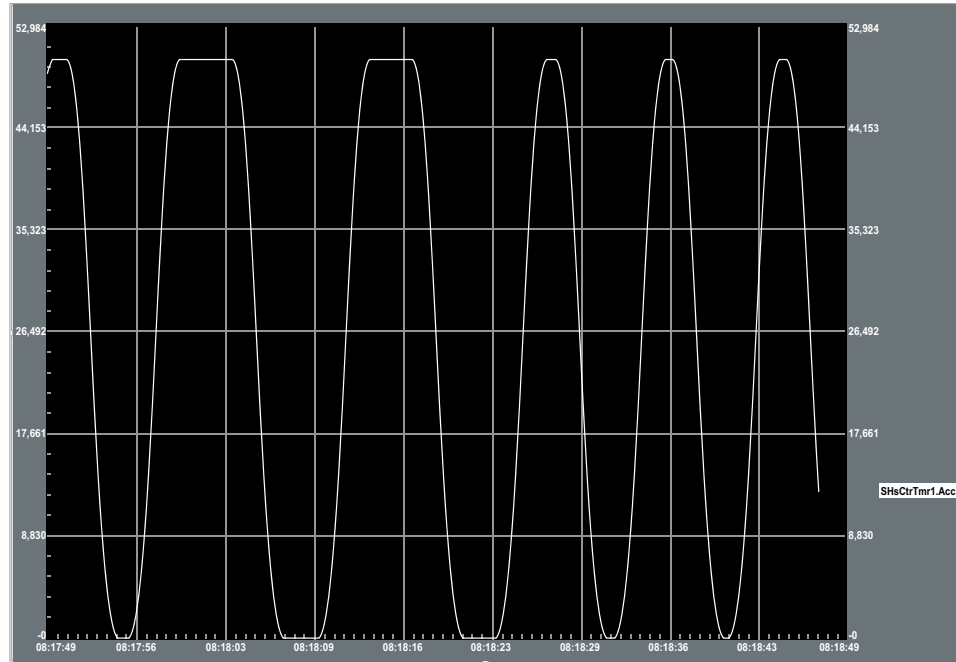
Detailed Example: Configure and Test a High-Speed Pulse Output with a Trapezoid Profile

This example walks through the steps required to use a BRX MPU to generate a pulsed output with a trapezoid profile using the built-in high-speed outputs.

The Basic Steps

1. Gather and wire up the hardware (not covered here).

2. Launch the Do-more! Designer software and connect to a BRX MPU with DC sinking or sourcing outputs.

3. Configure Axis 1 to use Y0 for Step and Y1 for Direction.

4. Write the necessary ladder program to generate the trapezoid profile.

5. Download and run the program.

6. Show how Data View and Trend View can be used to monitor and control the profile.

Equipment Needed

A BRX MPU with DC sinking or sourcing outputs, i.e. BX-DM1E-10ED23-D. A stepper drive and motor properly powered and wired to the BRX MPU outputs. Alternately, the output activity can be monitored in Data View or in Trend View, encase a stepper or similar hardware is not available.

Launch the Do-more! Designer Software. By default, the Tip of the Day will pop up. Close it.
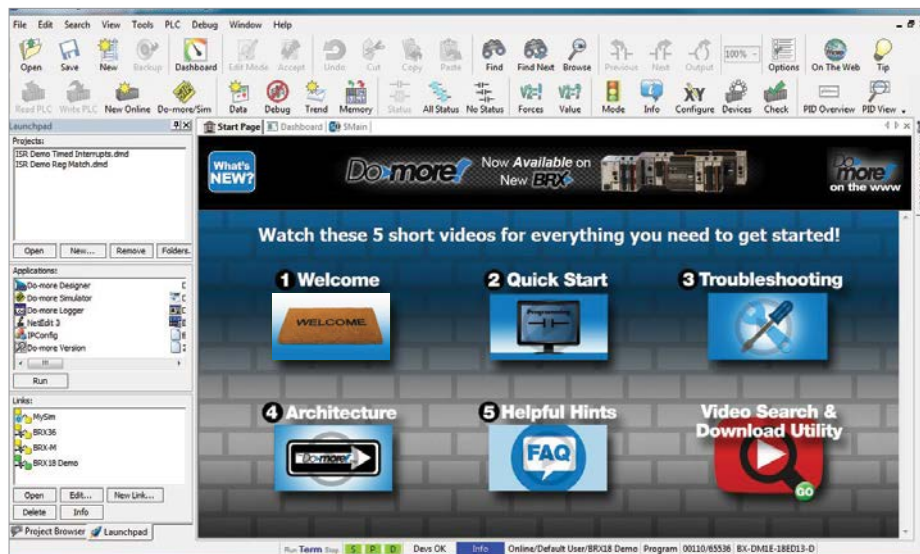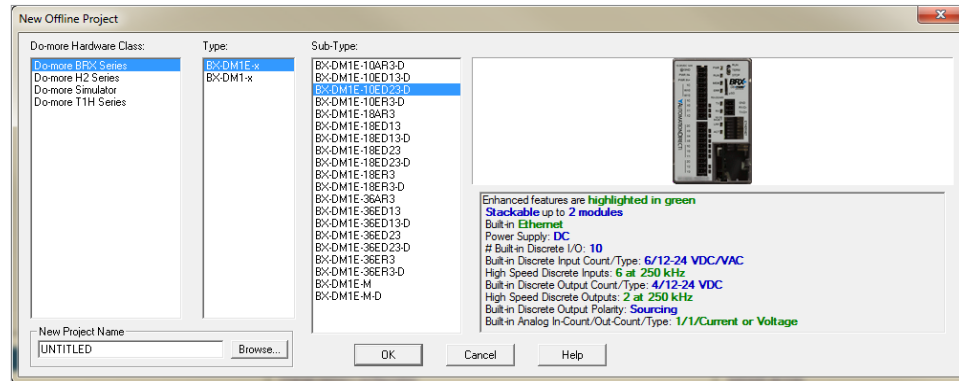

The *Select Project* window will be displayed next. Select **New Offline Project**.
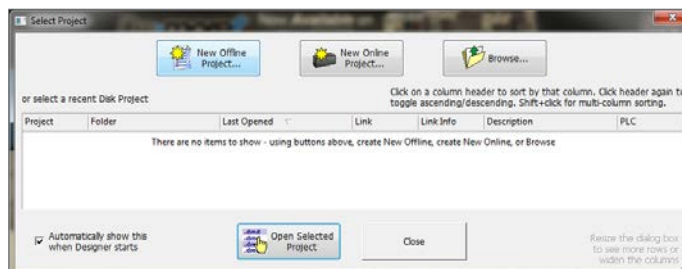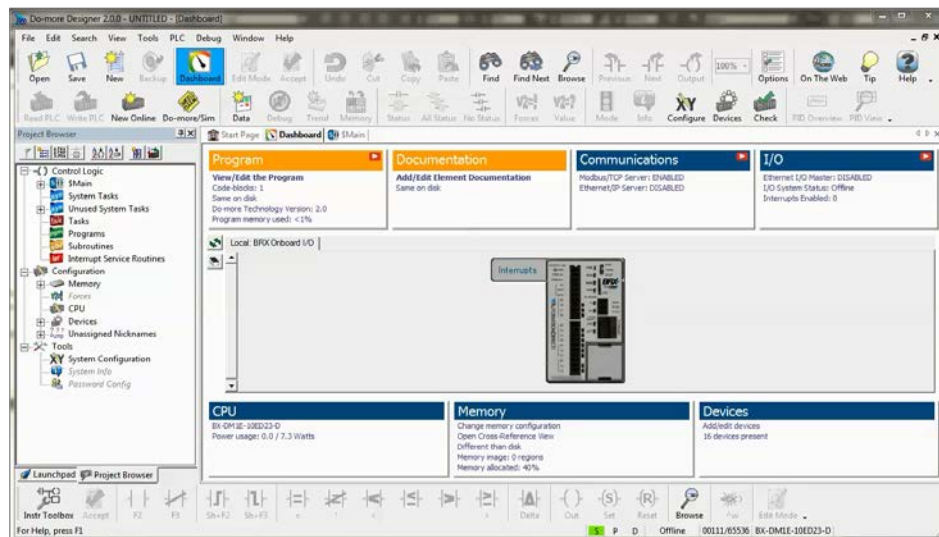
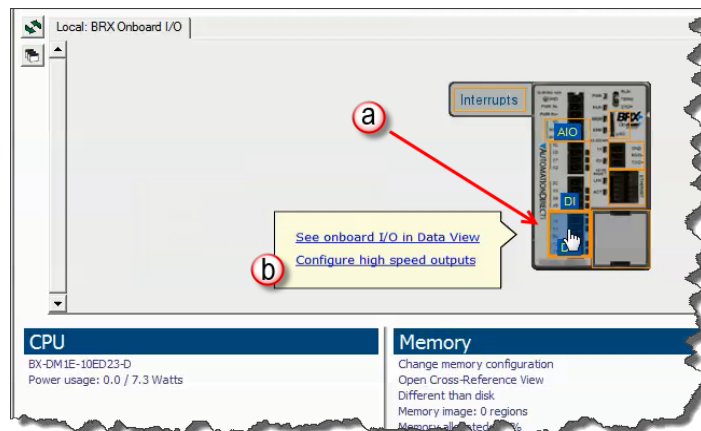## Detailed Example: Configure and Test a High-Speed Pulse Output, continued

In the *New Offline Project* windows select the BRX MPU model you are working with. Rename the project as Trapezoid, in the *New Project Name* field. Press the **OK** button to accept the selections.

The Dashboard page is displayed and the screen will look as follows:
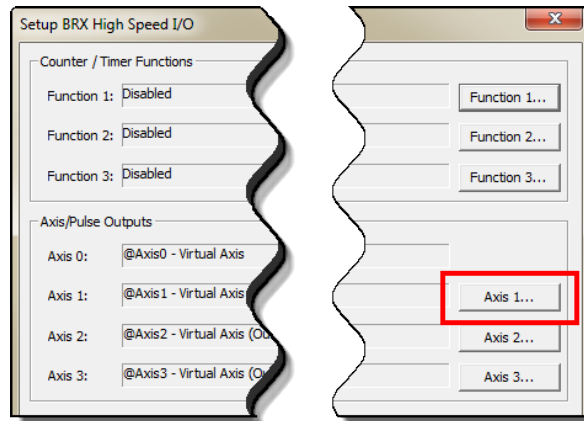

Mouse over the BRX MPU in the *Local BRX Onboard I/O* view. Select (a) the outputs orange block. Select **Configure high-speed outputs** (b) to access the *Setup BRX High-speed I/O* page.
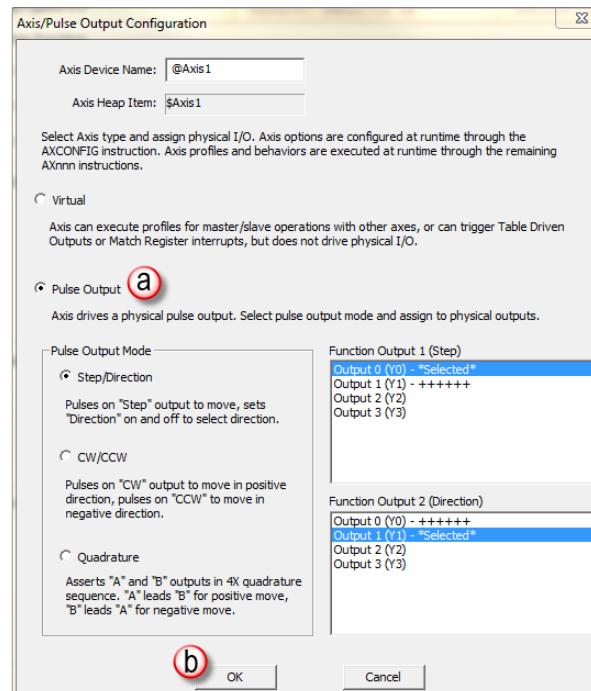
Detailed Example: Configure and Test a High-Speed Pulse Output, continued

From the *Setup BRX High-speed I/O* page, select **Axis 1** under the *Axis/Pulse Outputs* section.

The *Axis/Pulse Output Configuration* window comes up. Select (a) **Pulse Output**. Use the default device name, *$Axis 1*. For this exercise, select Pulse Output, Step/Direction, Y0 for Step and Y1 for Direction, as shown below.

Press (b) **OK** to return to the *Setup BRX High-Speed I/O* window. Press **OK** again to return to the *Dashboard* screen. Save the project at this time.


At this point, we have configured outputs Y0 and Y1 for Step and Direction high-speed outputs. Next, we will write the necessary ladder to generate a trapezoid profile with the configured high-speed outputs.
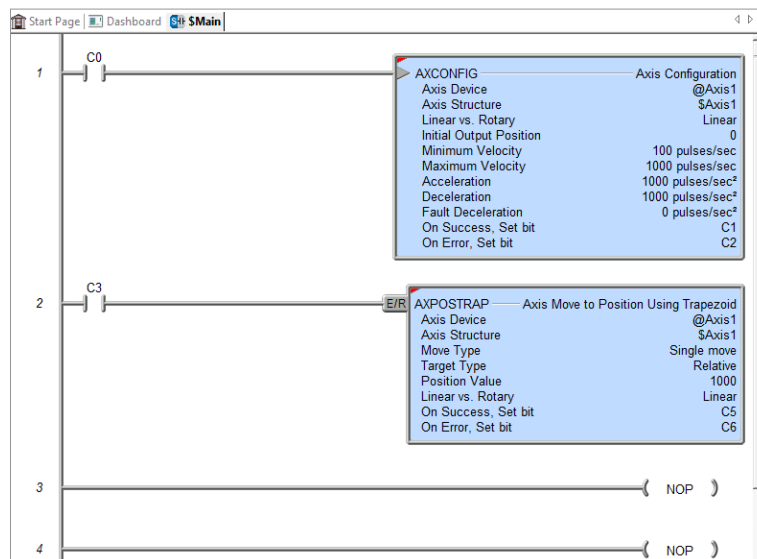
Detailed Example: Configure and Test a High-Speed Pulse Output, continued

Ladder Program for Trapezoid Profile

Start by opening the **$Main** code block (a), as shown below.
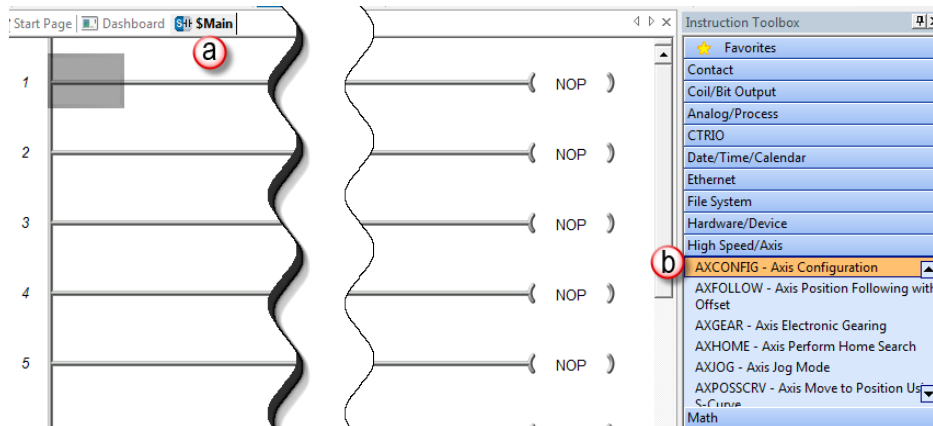
We will add two rungs to the ladder diagram by selecting two instructions from the *Instruction Toolbox* in the (b) *High-Speed/Axis* section. The ladder will look like the one shown below. On rung 1 we will place the AXCONFIG instruction and on rung 2 we will place the AXPOSTRAP instruction. The following page discusses setting the instruction parameters when placing these instructions.

Detailed Example: Configure and Test a High-Speed Pulse Output, continued

With the cursor on rung 1, Select AXCONFIG from the tool palette. AXCONFIG is needed to setup the runtime parameters for an Axis. An Axis has no default values and must be configured before it can be used. For this example, we will configure the AXCONFIG with the values as shown in the dialog box on the right. Since we have configured Axis 1 for Step/Direction, change the AXCONFIG Axis Device field to *@Axis1*. For the purpose of our example, leave all other fields at their default values.
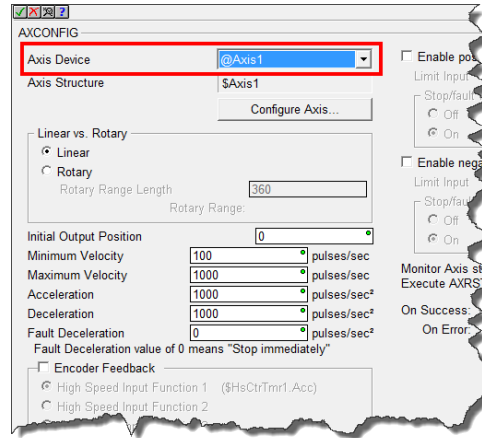
Move cursor to rung 2 and select AXPOSTRAP from the tool palette. AXPOSTRAP is used to move an Axis from its current position to a specified target position using the configured parameters to generate a trapezoid move profile.
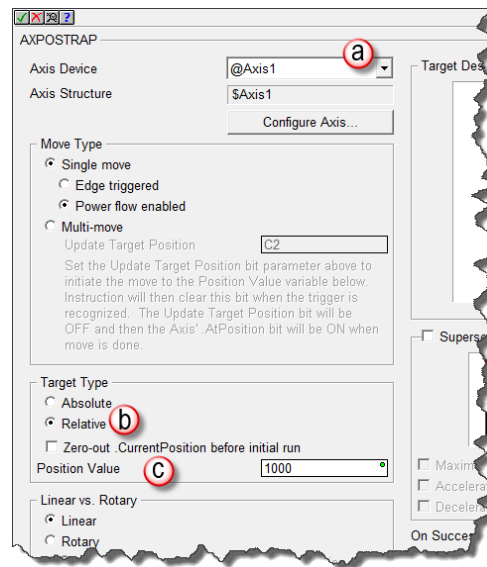

For the purpose of this example, change the (a) *Axis Device* to *@Axis1* and change the (b) *Target Type* to *Relative*, leaving the other parameters at their default values. You can change the (c) *Position Value* if your setup requires additional steps to approximate the trapezoid move. When you use the *Relative* target type, the profile will move the number of steps indicated by the (c) *Position Value* each time the AXPOSTRAP instruction is triggered.

Write the program to the BRX CPU and verify it is in **RUN** mode.

Detailed Example: Configure and Test a High-Speed Pulse Output, continued

Displaying and controlling the trapezoid profile move using Data View

From the menu select a new *Data View* under the **Debug -> Data View** menu. Alternately, you can use the **Data** icon (shown on right) on the toolbar ribbon to open a new *Data View* window.
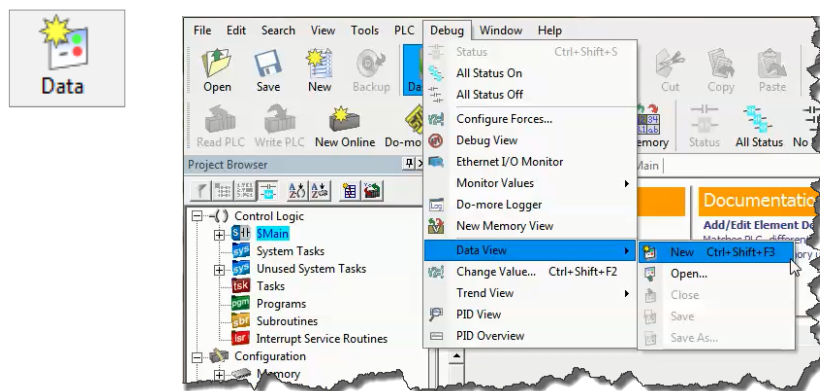
The *Data1* window is displayed. Start by typing C0 on row 1 and C3 on row 2, under the (a) *Element* column. Make the *Data1* window wider by placing the mouse on the right edge of the window (b), left clicking and dragging to the right. Select (c) the yellow **E** icon in order to display (d) the *Edits* column. Be sure **All Status On** (toolbar ribbon) is selected.
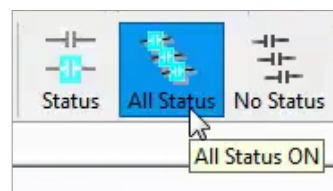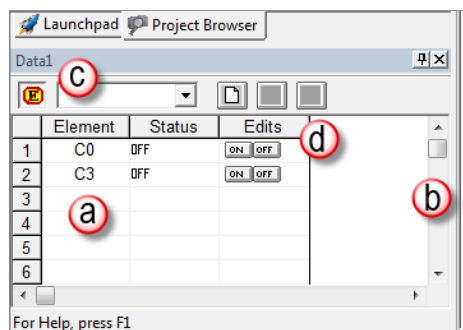
Under the *Edits* column (d), you will see **ON** and **OFF** buttons adjacent to C0 and C3. We will use these to trigger our two rungs of instructions. In Data View, to configure *$Axis1* parameters, double-left click the **ON** button for C0 element.

> **NOTE:** *The first time you use this feature, the program will ask you to confirm the writes. If you prefer check the box so that it will not ask you again.*

Detailed Example: Configure and Test a High-Speed Pulse Output, continued

After triggering C0, the ladder will display the following status as shown below.
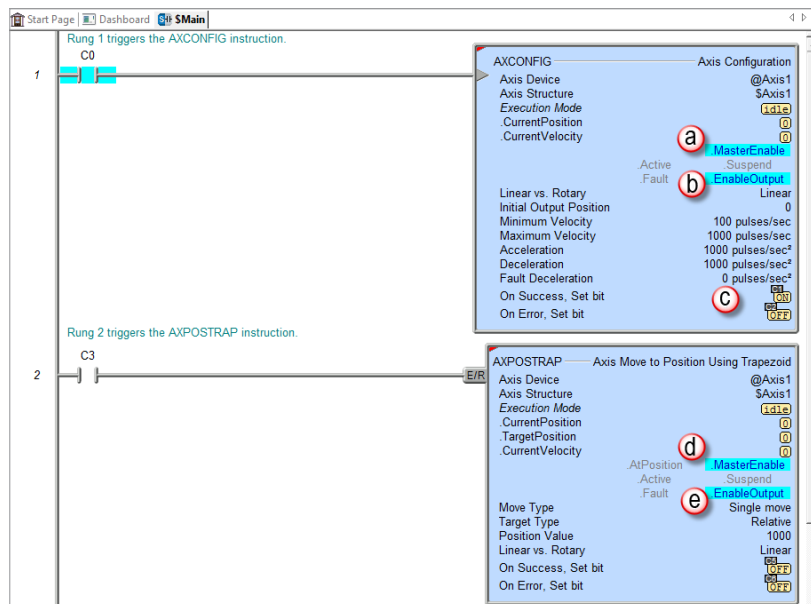
In the AXCONFIG instruction, we see (a) *.MasterEnable* and (b) *.EnableOutput* are highlighted in blue and C1 is set to ON. These indicate that *$Axis1* is configured and ready to be used.

The AXPOSTRAP instruction in Rung 2 shows *$Axis1* parameters have been configured (d) (*.MasterEnable* is highlighted) and it is ready to start sending pulses (e) (*.EnableOutput* is highlighted).

C0 can be turned off once the On Success bit (c), of the AXCONFIG turns on.

We are now ready to initiate our trapezoid move.

## Detailed Example: Configure and Test a High-Speed Pulse Output, continued

Double-left click on the C3 ON button in Data View. The trapezoid move will begin. Status indicators on (a) the AXPOSTRAP instruction will provide feedback about the move. The following graphic shows its status after the move is complete. Notice (b) the Current Position is now 1000 and the (c) Target Position is 1000. The field device should have moved 1000 steps.
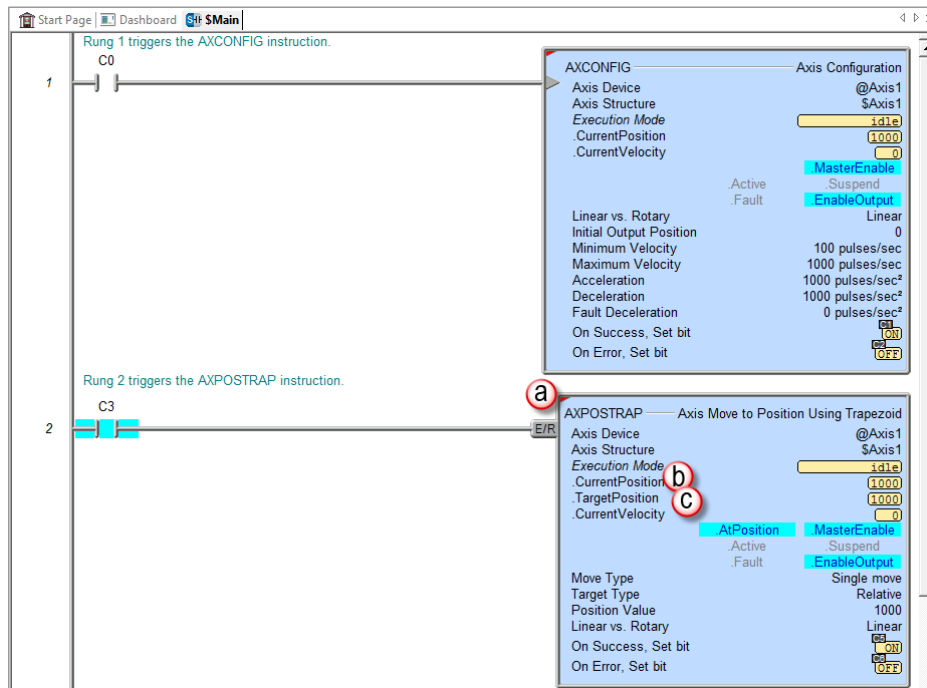
Displaying $Axis1 Current and Target values using Trend View

As mentioned before, Trend View is a very powerful utility, one that can be used to troubleshoot the control system.

Open **Trend View** from the **Debug Menu** or the **Trend** icon on the toolbar, as shown below.
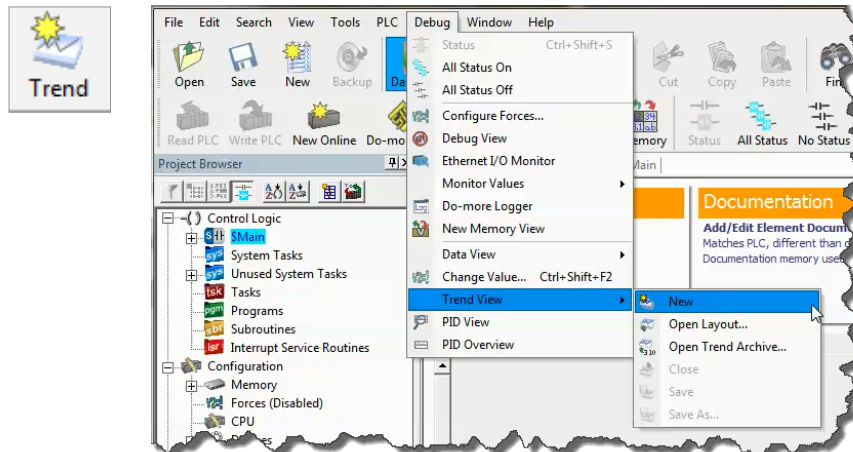
In Trend View, using a single pane, add *$Axis1.CurrentPosition* to a pane.

Every time the AXPOSTRAP instruction is triggered, 1000 pulses are generated. In the Trend View snapshot that follows, we can see the instruction was triggered twice. It started at 0 counts and reached 1000 counts the first trigger. On the second trigger, it started at 1000 and reached 2000 counts.

Detailed Example: Configure and Test a High-Speed Pulse Output, continued
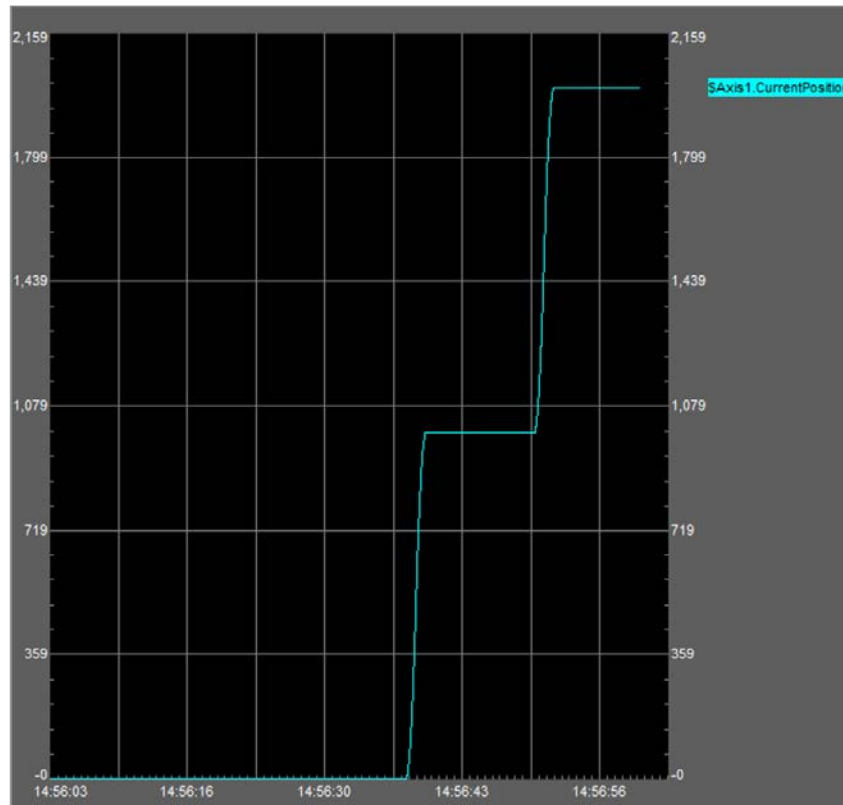
There are many settings available that allow the user to meet application needs. Please review the Do-more! Designer High-speed Instruction Help topics for detailed information.

BRX High-speed Instructions

In this section we will give an overview of the high-speed instructions available to the BRX Do-more! CPU. They will be presented in this section as in the Instruction Toolbox, in alphabetical order.

1. AXCAM – Axis Electronic Camming
2. AXCONFIG – Axis Configuration
3. AXFOLLOW – Axis Position Following with Offset
4. AXGEAR – Axis Electronic Gearing
5. AXHOME – Axis Perform Home Search
6. AXJOG – Axis Jog Mode
7. AXPOSSCRV – Axis Move to Position Using S-Curve
8. AXPOSTRAP – Axis Move to Position Using Trapezoid
9. AXRSTFAULT – Reset Axis Limit Fault
10. AXSCRIPT – Run a Sequence of Axis Commands
11. AXSETPROP – Set Axis Properties
12. AXVEL – Axis Set Velocity Mode
13. TDODECFG – Deconfigure Table Driven Output
14. TDOPLS – Load Programmable Limit Switch Table for Table Driven Output
15. TDOPRESET – Load Preset Table for Table Driven Output

An expanded discussion of each of these instructions can be found in the Do-more! Designer online Help files under the High-Speed/Axis topic.

Before Using the BRX High-Speed I/O Instructions

Remember the following before using the BRX high-speed i/o instructions.

1. Each of the instructions covered in this section will require one or more of the following to be setup prior to being able to use the instruction:

   – Slave Axis Device or Axis Device - specifies which axis to use as the slave axis. This can be any axes available to the MPU or to a BX-HSIO expansion module.

   – Master Register - selects the axis position or high-speed counter/timer accumulated value that provides the position source value. This value can come from any axes or high/speed counter/timer available to the MPU or to a BX-HSIO expansion module.

   – Table Driven Output Device - specifies which discrete output to use. This can be any of the outputs available to the MPU or available in the BX-HSIO expansion module.

   – AXCONFIG – this instruction configures parameters associated with a specific Axis. The Axis minimum and maximum Velocity, the Axis Acceleration and Decelaration are some of the configurable parameters that will affect how an Axis behaves. (See the AXCONFIG instruction later in this section for detailed information)

   Note: There can be more than one AXCONFIG instruction configured for the same Axis, however, the parameters associated with the last AXCONFIG run for an Axis will be used.

2. The Axis, the High-speed Counters/Timers and the Table Driven Outputs are configured in the MPUs *Setup BRX High-speed I/O* dialog or the BX-HSIO *Setup BX-HSIO Module* dialog. Please review the information under the "Access Setup High Speed I/O Page" section earlier in this chapter. It explains the different paths to opening these setup dialog page.

   Please review the section "Available High-Speed Input and Output Features" earlier in this chapter, which explains in detail how to configure the Axis/Pulse Outputs, the Counter/Timers, and the Table Driven Outputs.

   For example, from the MPUs *Setup BRX High-speed I/O* dialog or the BX-HSIO *Setup BX-HSIO Module* dialog:

   – Click the **Axis 1…** button to select the axis type: Virtual, Step/Direction, CW/CCW, Quadrature. Here you select the discrete output points to be used by an axis that generates physical pulses.

   – Click the **Function1…** button to configure a Counter or a Timer input function. Here you select the discrete input point(s) to be used by the input function.

   – Click the **Table1…** button to enable a Table Driven Output. Here you select the discrete output point to be used by the Table Driven Output function.

3. For instructions that require the use of an Axis, be sure to add at least one AXCONFIG instruction for the Axis used. Please see the detailed information on the AXCONFIG instruction later in this chapter.

AXCAM

The **Axis Electronic Camming** (AXCAM) instruction is used to establish a master/slave connection for an axis so that its movement is synchronized to another axis or to a high-speed counter/timer. The slave axis position is derived from the master position. Any time the master axis moves to a new position the slave axis will also move to a corresponding position as directed by the Cam Table.

This instruction requires the following:

- Axis configured to be used as the slave Axis.

- One of the following to be used as the Master Register:

  – Position value from a configured master axis.

  – Accumulated value from a configured high speed counter/timer function.

- AXCONFIG instruction configured for each axis associated with the instruction.

The (a) **Slave Axis Device** specifies which axis to use as the slave axis and the (b) **Master Register** selects the axis position or high-speed counter/timer accumulated value that provides the position source value. These can be any of the axis or high-speed counter/timers available to the MPU or available to a BX-HSIO expansion module.

The Axis and high-speed counters/timers are configured in the MPUs **Setup BRX High-speed I/O** dialog or the BX-HSIO **Setup BX-HSIO Module** dialog. Please see detailed information on these under the "Available High-Speed Input and Output Features" earlier in this chapter. These dialog boxes can be opened directly from the AXCAM instruction by clicking on the (c) **Configure Axis...** button and (d) **Configure Master Register Device...** button. These allow the user to:

- configure an axis or a counter/timer function for use with the instruction.

- verify or edit the physical I/O selected for an already configured axis or counter/timer function.
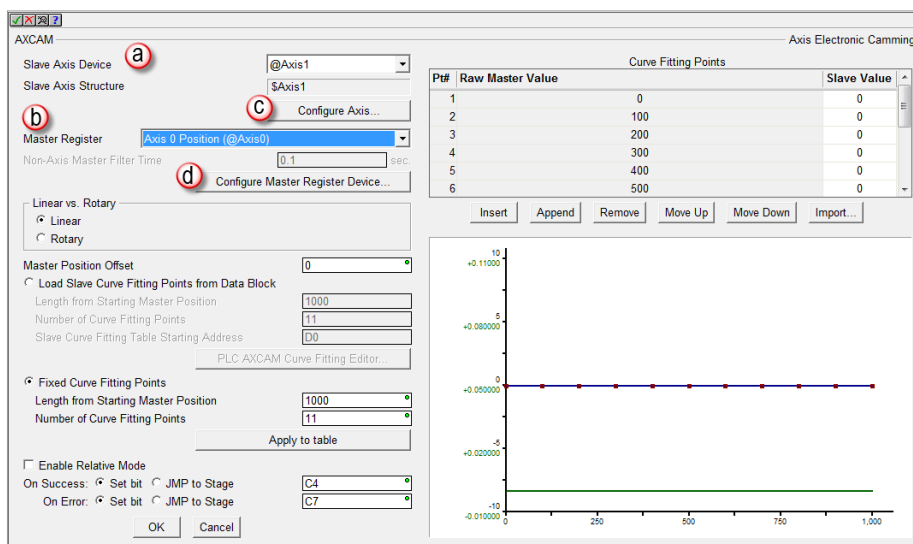
AXCAM, continued

Points and segments:

A Cam Table describes a motion profile that clearly assigns a slave position for each master position of a specific master position range.

Cam tables are entered as a series of up to 64 segments. All of the segments are the same size in that they are equal divisions of the total span of the Master Axis travel.

A segment describes the distance traveled and the position of the axis. It is equal to the difference between the previous and current positions. Each segment is executed over a defined distance traveled for the Master Axis. When executing these segments, the profiling software in the controller will fit a smooth curve to the data with an updated velocity profile every millisecond.

Curve Fitting

The Cubic Interpolation method fits a cubic trajectory between each pair (segment) of cam table entries.

All segment transitions are curved, even for a linear progression between the segments. The only exception is on the last 2 table segments. The last 2 segments will be a linear progression unless **Rotary** mode has been selected. In **Rotary** mode, there will be curves on the last position of the table to the first position of the table.
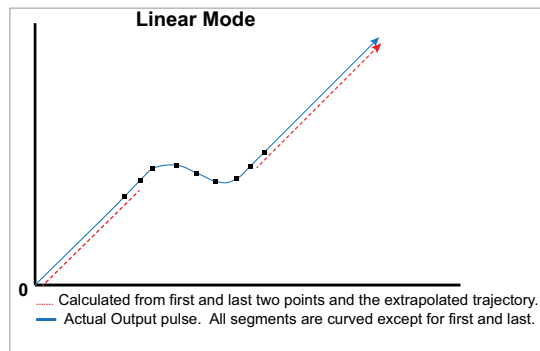
Master Register: This is the Axis, High-Speed Counter or Timer that provides the position source value. This can be any of the Axes or High-Speed Counter/Timers.

Non-Axis Master Filter Time: This parameter is only enabled if the Master Register is a High-Speed Counter/Timer. This value is the Filter Time Constant which specifies how often (in Seconds) the Slave's position is calculated. This can be any constant value or any numeric location.

Configure Master Register Device: This button will open the BRX High-Speed Input or Axis/Pulse Output dialog in the System Configuration.

Linear vs. Rotary: This setting selects the behavior that will occur when the end of the table has been reached.

- Linear: The first two and the last two table positions are used to calculate a linear trajectory (from 0 at the start of the table) and the CPU will continue to output pulses until the instruction is terminated.

Linear Mode

Calculated from first and last two points and the extrapolated trajectory.
Actual Output pulse. All segments are curved except for first and last.

AXCAM, continued

- **Rotary**: This selection will 'loop' the table to start at the beginning. The modulus of the Master Register is used as the table Master Value.
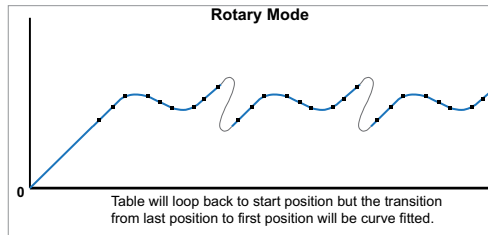
Master Position Offset: A pulse count value that is added to the Master Axis position before the associated Slave position value is calculated. This can be any constant value or any numeric location. If this value is a numeric location the value is read from that location only when the instruction is first enabled.

Load Slave Curve Fitting Points from Data Block: The data for the curve is located in PLC memory.

- **Length from Starting Master Position**: The total number of pulse counts the Master will move.

- **Number of Curve Fitting Points**: The number of segments to divide the Length from Master Position into. There must be at least 3 curve fitting points in the table with a maximum of 64.

- **Slave Curve Fitting Table Starting Address**: The beginning address in PLC memory where the curve data is stored.

Fixed Curve Fitting Points: The data for the curve is in the instruction table.

- **Length from Starting Master Position**: The total number of pulse counts the Master will move.

- **Number of Curve Fitting Points**: The number of segments to divide the **Length from Master Position** into. There must be at least 3 curve fitting points in the table with a maximum of 64.

Enable Relative Mode: If selected, the function is enabled. The Slave Axis' current position is stored internally and that position becomes the new 0 for the Slave Axis. As the Master Axis moves, the Slave position values are now relative to this stored position.

On Success:

- **Set Bit**: The bit will become FALSE when the instruction is enabled and will remain FALSE until the enable leg goes back OFF. Once the enable leg turns OFF, if the instruction's device/parameters were valid, this bit will turn ON once the .CurrentVelocity reaches 0.

- **JMP to Stage**: Similarly, the JMP will not occur until after the instruction is enabled, then disabled, and all the instruction device/parameters were valid and the .CurrentVelocity reaches 0.

On Error:

- **Set Bit**: The bit will become FALSE when the instruction is enabled and will become TRUE if there was a problem with the parameters configured in the instruction, with the devices specified or if the AXCAM move was interrupted before completion.

- **JMP to Stage**: If there was a problem with the parameters configured in the instruction, with the devices specified or the AXCAM move was interrupted before completion, the PLC will jump to that stage.
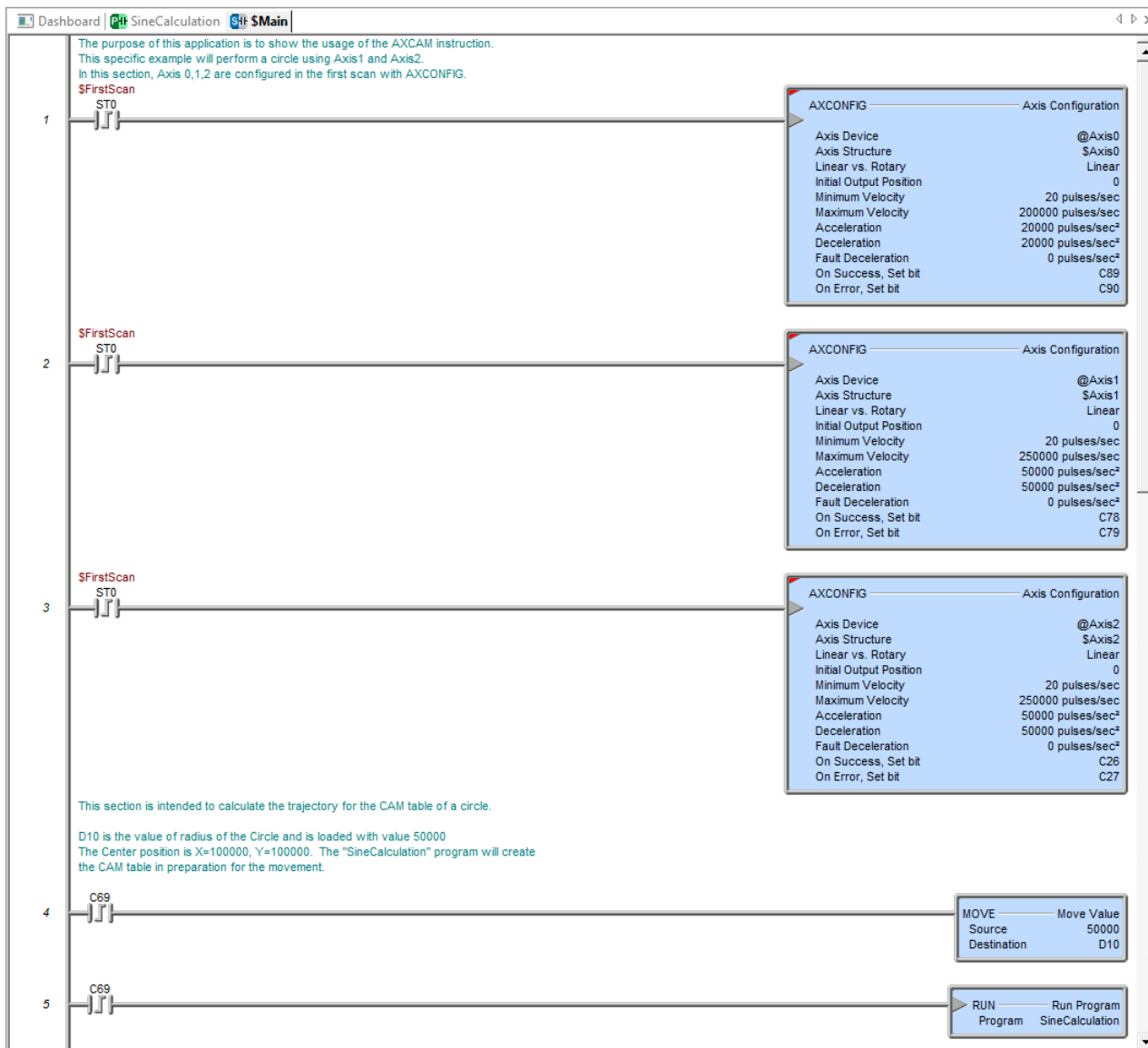
AXCAM, continued

Example AXCAM

In this example we will demonstrate the use of the AXCAM instruction in conjunction with the AXCONFIG instruction to create a circular path output using multiple axes.

The radius of the circle (50000) is loaded into D10. The center position is: X=100000, Y=100000. The path followed is defined in the program **SineCalculation**.

Axes 0,1 & 2 are defined with the AXCONFIG instruction. The AXCAM instruction creates the movement of the Axes to define the path traveled. From the $Main program, a program called **SineCalculation** is called to perform the necessary math calculations to create a circular path output. The **SineCalculation** is called for each segment in the camming table (61 segments).

The following pages show the ladder program that defines this operation.
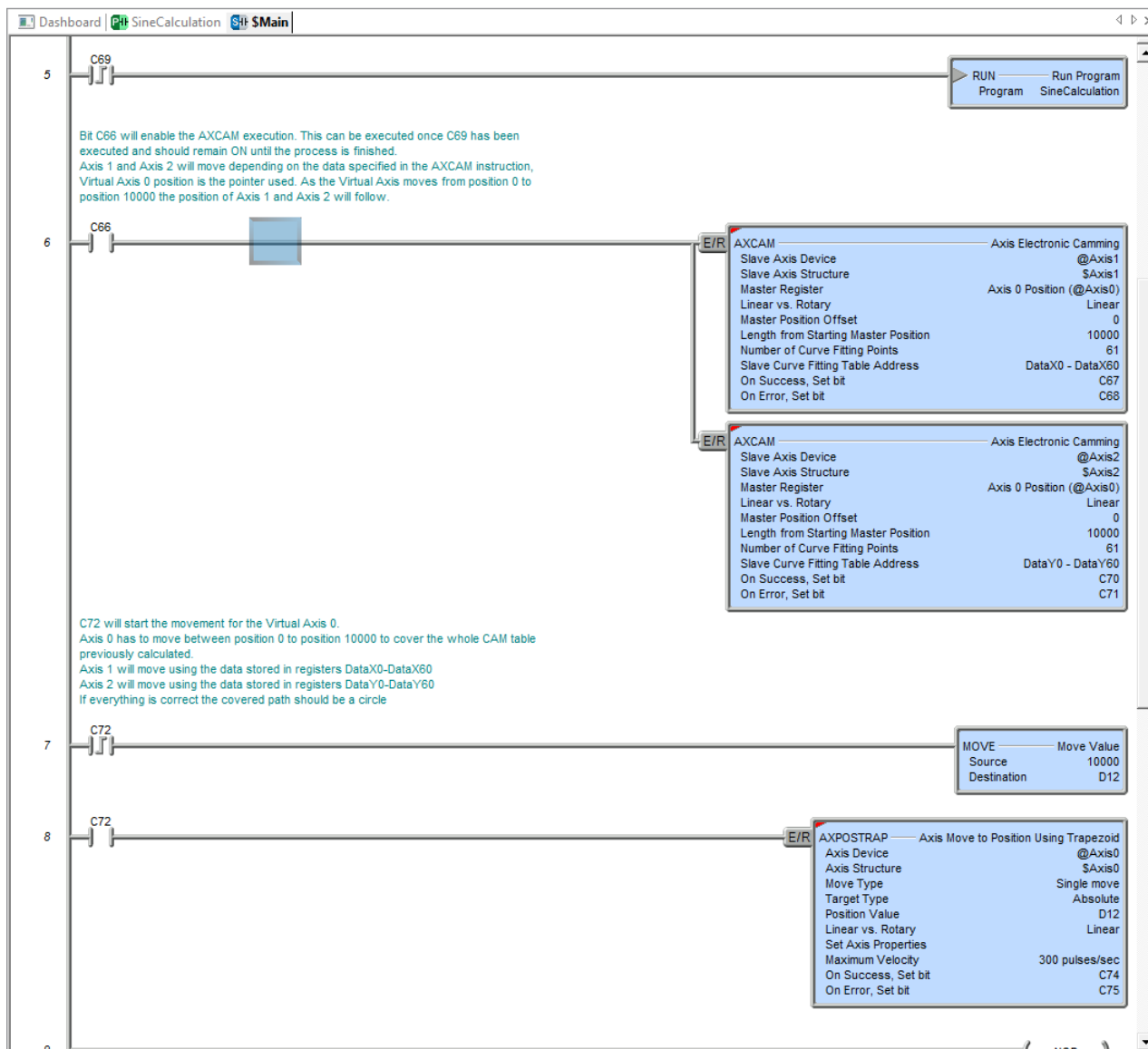
Here is the $Main program:
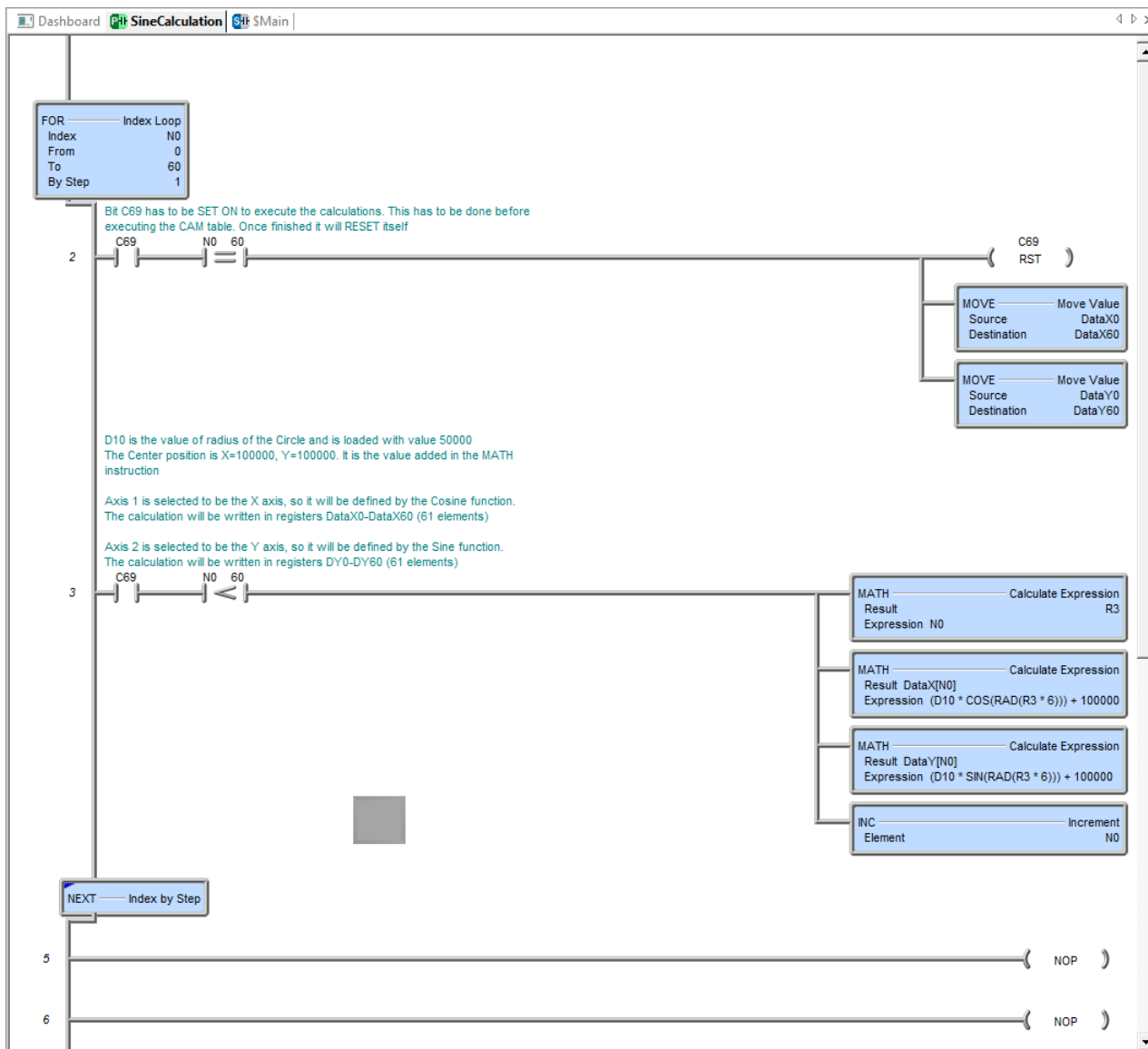

(Ladder continued on next page.)

AXCAM, continued

$Main program continued:

```
┌ Dashboard │ ▣ SineCalculation │ ▤ $Main                                          ◁ ▷ ✕ ┐

  5      C69                                                              ▷ RUN ──────── Run Program
       ──┤ ├──                                                             Program     SineCalculation

        Bit C66 will enable the AXCAM execution. This can be executed once C69 has been
        executed and should remain ON until the process is finished.
        Axis 1 and Axis 2 will move depending on the data specified in the AXCAM instruction,
        Virtual Axis 0 position is the pointer used. As the Virtual Axis moves from position 0 to
        position 10000 the position of Axis 1 and Axis 2 will follow.

  6      C66                                           E/R│ AXCAM ─────────────── Axis Electronic Camming
       ──┤ ├──                                             Slave Axis Device                    @Axis1
                                                           Slave Axis Structure                 $Axis1
                                                           Master Register            Axis 0 Position (@Axis0)
                                                           Linear vs. Rotary                     Linear
                                                           Master Position Offset                     0
                                                           Length from Starting Master Position   10000
                                                           Number of Curve Fitting Points            61
                                                           Slave Curve Fitting Table Address  DataX0 - DataX60
                                                           On Success, Set bit                      C67
                                                           On Error, Set bit                        C68

                                                      E/R│ AXCAM ─────────────── Axis Electronic Camming
                                                           Slave Axis Device                    @Axis2
                                                           Slave Axis Structure                 $Axis2
                                                           Master Register            Axis 0 Position (@Axis0)
                                                           Linear vs. Rotary                     Linear
                                                           Master Position Offset                     0
                                                           Length from Starting Master Position   10000
                                                           Number of Curve Fitting Points            61
                                                           Slave Curve Fitting Table Address  DataY0 - DataY60
                                                           On Success, Set bit                      C70
                                                           On Error, Set bit                        C71

        C72 will start the movement for the Virtual Axis 0.
        Axis 0 has to move between position 0 to position 10000 to cover the whole CAM table
        previously calculated.
        Axis 1 will move using the data stored in registers DataX0-DataX60
        Axis 2 will move using the data stored in registers DataY0-DataY60
        If everything is correct the covered path should be a circle

  7      C72                                                              MOVE ─────────── Move Value
       ──┤ ├──                                                            Source                   10000
                                                                          Destination                D12

  8      C72                                           E/R│ AXPOSTRAP ── Axis Move to Position Using Trapezoid
       ──┤ ├──                                             Axis Device                          @Axis0
                                                           Axis Structure                       $Axis0
                                                           Move Type                         Single move
                                                           Target Type                          Absolute
                                                           Position Value                           D12
                                                           Linear vs. Rotary                     Linear
                                                           Set Axis Properties
                                                           Maximum Velocity              300 pulses/sec
                                                           On Success, Set bit                      C74
                                                           On Error, Set bit                        C75
```

AXCAM, continued

Program: SineCalculation

Here is the SineCalculation program that is called from the $Main program.

AXCONFIG

The **Axis Configuration** (AXCONFIG) instruction configures the parameters for a specific Axis so that it may be used for all of the other AXIS commands. Pulse Outputs for a given AXIS cannot be commanded until the AXCONFIG instruction has been run successfully. When the AXCONFIG instruction has been successfully created and initiated, the **.MasterEnable** structure member for that AXIS will become true and the AXIS may be commanded by other Axis instructions.

NOTE: *There can be more than one AXCONFIG instruction configured for the same Axis, however, the parameters associated with the last AXCONFIG run for an Axis will be used.*

This instruction requires an Axis configured to be used as the slave Axis.

The (a) **Axis Device** specifies which Axis the parameters will be applied to. This can be any axis available to the MPU or available to a BX-HSIO expansion module.

The Axis is configured in the MPUs **Setup BRX High-speed I/O** dialog or the BX-HSIO **Setup BX-HSIO Module** dialog. Please see detailed information on these under the "Available High-Speed Input and Output Features" earlier in this chapter. This dialog box can be opened directly from the AXCONFIG instruction by clicking on the (b) **Configure Axis…** button. These allow the user to:

- configure an axis for use with the instruction.

- verify or edit the physical I/O selected for an already configured axis.

NOTE: *Axis0 is a 'virtual axis' and will not generate pulses to physical outputs of the PLC. Axis0 can be used for generating pulse output profile register values to be used for Table Driven Outputs (TDOPRESET or TDOPLS) or as the master for other axes in following-type applications (AXCAM, AXFOLLOW or AXGEAR). Axis0 is also good to use for testing.*

AXCONFIG, continued

Linear vs Rotary: This setting selects the behavior that will occur when the end of the table has been reached.

- **Linear**: Linear actuators move forward or backward on a fixed linear path. The movement of linear actuators are defined in linear units such as inches or millimeters. Since linear actuators only move in two directions on a fixed path, linear actuators are defined as finite, meaning they have a set distance that they can travel in either direction before they must stop.

- **Rotary**: Rotary actuators produce rotary motion, meaning that the actuator revolves on a circular path. Movement from this type of actuator is defined in rotary units, typically degrees. A rotary table doesn't have a fixed distance it can travel; it can keep spinning in the same direction indefinitely. The count for a rotary input is kept within a defined range (*Rotary Range Length*). Once the count has exceeded the high limit of the specified *Rotary Range Length*, it will reset to 0. If the Axis position is decreasing (negative velocity value), when it falls below 0, it will go to the high value specified in the *Rotary Range Length*.

Initial Output Position: The current count will be set to this value when the Axis is first enabled and any time the Axis is reset. This can be any constant value or any numeric location.

Minimum Velocity (pulses/sec): The slowest frequency of output pulses that will be generated when the output is enabled. This can be any positive constant from 10 to 250000 or any numeric location with a value in that range.

Maximum Velocity (pulses/sec): The fastest frequency of output pulses that will be generated when the output is enabled. This can be any positive constant from 10 to 250000 or any numeric location with a value in that range.

Acceleration Rate (pulses/sec^2): The rate at which the pulses will be generated when the axis is ramping up from a slower pulse rate to a higher pulse rate. This can be any positive constant or any numeric location with a value in that range.

Deceleration Rate (pulses/sec^2): The rate at which the pulses will be generated when the axis is ramping down from a faster pulse rate to a slower pulse rate. This can be any positive constant or any numeric location with a value in that range.

Fault Deceleration Rate (pulses/sec^2): Any time a Fault Limit is reached or the Axis .MasterEnable is turned OFF, the Axis will decelerate to 0 at this specified rate. A value of 0 will cause the Axis to immediately stop moving. This can be any positive constant or any numeric location with a value in that range.

Encoder Feedback: Enable this option if an incremental encoder is being used to provide the speed reference feedback. The Axis structure contains a *.FollowingError* that indicates the difference in the commanded position and the encoder feedback. This can indicate a stalled motor, mechanical slippage or other problem in the application.

- **High-speed Input Function 1/High-speed Input Function 2/High-speed Input Function 3**: Choose one of these options for the encoder feedback. The High-speed Counters must be setup in the *Setup BRX Counter/Timer* section of the *BRX Onboard I/O>High-speed I/O* section in the System Configuration. Click on the **Configure High-speed Input** button to access this directly.

- **Position Based on**:

  - **Encoder**: This option will place the specified High-speed Input Function into the *.CurrentPosition* member of the Axis configured in this instruction. The difference in the specified High-speed Input Function and the actual pulse output count of the Axis configured in this instruction will be indicated in the *.FollowingError*.

AXCONFIG, continued

- – **Pulse Output**: This option will indicate the actual pulse output count of the Axis configured in this instruction into the *.CurrentPosition* member and the *.FollowingError* will indicate the difference between *.CurrentPosition* and the High-speed Input Function that has been specified.
- **Pulse Output/Encoder Scale**: If the motor and the encoder have different pulse-per-revolution values, enter the scale value required to bring them into alignment.
- **Encoder Deadband** (counts): Having some deadband value around the encoder current position can prevent the pulse output from generating alternating small pulses trying to get the input value to an exact number. This value is applied both above and below the encoder value. For example: A value of 2 will be a deadband of 2 above and 2 below for a span of 4 counts.

Enable Positive/Clockwise Motion Fault-Limit: Enable this option to use a physical input that will cause the Axis to Fault (Stop) if the input becomes true or false (based on the setting below) when moving in the Clockwise or Positive direction. This is typically used as an over-travel limit switch.

If a Fault-Limit is tripped, use the Axis Reset Fault (AXRSTFLT) instruction or re-trigger the Axis Config (AXCONFIG) instruction to clear the fault. Any attempt to move in the same direction that caused the fault will immediately generate another fault condition. Movement of the Axis in the opposite direction is permitted.

- – **Limit Input**: This needs to be one of the onboard X discrete input elements. This is where the over travel limit switch would be wired to.
- – **Stop/fault when Limit is**: Choose OFF to cause the Axis to fault (stop) when the specified input goes from True to False. Choose ON to cause the Axis to fault (stop) when the specified input goes from False to True.

Enable Negative/Counter-Clockwise Motion Fault-Limit: Enable this option to use a physical input that will cause the Axis to Fault (Stop) if the input becomes true or false (based on the setting below) when moving in the Counter-Clockwise or Negative direction. This is typically used as an over-travel limit switch.

If a Fault-Limit is tripped, use the Axis Reset Fault (AXRSTFLT) instruction or re-trigger the Axis Config (AXCONFIG) instruction to clear the fault. Any attempt to move in the same direction that caused the fault will immediately generate another fault condition. Movement of the Axis in the opposite direction is permitted.

- – **Limit Input**: This needs to be one of the onboard "X" discrete input elements. This is where the over travel limit switch would be wired to.
- – **Stop/fault when Limit is**: Choose OFF to cause the Axis to fault (stop) when the specified input goes from True to False. Choose ON to cause the Axis to fault (stop) when the specified input goes from False to True.

On Success:

- **Set Bit**: The bit will become FALSE when the instruction is enabled and will become TRUE if the parameters configured in the instruction and proper devices were specified.
- **JMP to Stage**: If the parameters configured in the instruction and proper devices were specified, the PLC will jump to that stage.

On Error:

- **Set Bit**: The bit will become FALSE when the instruction is enabled and will become TRUE if there was a problem with the parameters configured in the instruction or with the devices specified.
- **JMP to Stage**: If there was a problem with the parameters configured in the instruction or with the devices specified, the PLC will jump to that stage.

AXFOLLOW

The **Axis Position Following with Offset** (AXFOLLOW) instruction is used to establish a Master/Follower connection for an Axis so that the Follower's movement is synchronized to the Master's movement. The Master can be another Axis or a High-Speed Counter/Timer.

Because the Follower Axis will need the ability to overtake the Master Axis during a Goto Relative Offset operation, ensure that the Maximum Velocity and Acceleration parameters of the Master and Follower Axes have been configured with enough capacity to allow this. The Follower Axis can be made more responsive by configuring it with higher Maximum Velocity or a faster Acceleration or both.

If the instruction is enabled with the Goto Offset Signal OFF, the Axis will behave in a Velocity-following manner. As soon as the Goto Offset Signal turns ON, the Axis will now behave in a Position-following manner and will remain so until the instruction is terminated. If the instruction is enabled with the Goto Offset Signal ON, the Axis will behave in a Position-following manner until the instruction is terminated.

To help keep track of the Axes movement relative to each other, any time the Axis is in Position Following mode the Follower Axis associated structure member, *.MstSlvCoordError*, will contain the difference (in pulse counts) between the Master Axis position and the Follower Axis' position.

This instruction requires the following:
- Axis configured to be used as the slave Axis.
- One of the following to be used as the Master Register:
  - Position value from a configured master axis.
  - Accumulated value from a configured high speed counter/timer function.
- AXCONFIG instruction configured for each axis associated with the instruction.

The (a) *Slave Axis Device* specifies which axis to use as the slave axis and the (b) *Master Register* selects the axis position or high-speed counter/timer accumulated value that provides the position source value. These can be any of the axis or high-speed counter/timers available to the MPU or available to a BX-HSIO expansion module.

The Axis and high-speed counters/timers are configured in the MPUs *Setup BRX High-speed I/O* dialog or the BX-HSIO *Setup BX-HSIO Module* dialog. Please see detailed information on these under the "Available High-Speed Input and Output Features" earlier in this chapter. These dialog boxes can be opened directly from the AXFOLLOW instruction by clicking on the (c) **Configure Axis...** button and (d) **Configure Master Register Device...** button. These allow the user to:

- configure an axis or a counter/timer function for use with the instruction.
- verify or edit the physical I/O selected for an already configured axis or counter/timer function.

AXFOLLOW, continued

> **NOTE:** *Axis0 is a 'virtual axis' and will not generate pulses to physical outputs of the PLC. Axis0 can be used for generating pulse output profile register values to be used for Table Driven Outputs (TDOPRESET or TDOPLS) or as the master for other axes in following-type applications (AXCAM, AXFOLLOW or AXGEAR). Axis0 is also good to use for testing.*

Master Register: This is the Axis, High-Speed Counter or Timer that provides the position source value. This can be any of the Axes or High-Speed Counter/Timers.

Non-Axis Master Filter Time: This parameter is only enabled if the Master Register is a High-Speed Counter/Timer. This value is the Filter Time Constant which specifies how often (in Seconds) the Slave's position is calculated. This can be any constant value or any numeric location.

Configure Master Register Device: This button will open the BRX High-Speed Input or Axis/Pulse Output dialog in the System Configuration.

Gear Ratio Multiplier: A multiplier that will be applied when the Follower's position is calculated. This can be any constant value or any numeric location. Unlike the similar parameter in the Electronic Gearing (AXGEAR) instruction, this value cannot be adjusted while the instruction is enabled.

Relative Offset Position: The Follower's position value will be adjusted by this pulse count value each time the Goto Offset Signal turns ON. This can be any constant value or any numeric location.

Relative Offset Velocity: The additional velocity the Follower Axis will use when moving to the Relative Offset Position. This value specifies how much faster (in pulses/second) the Follower Axis can move than the Master Axis when attempting to move to the Relative Offset Position. This value can be any constant between 0 and 250,000 or any numeric location containing a value in that range.

Goto Offset Signal: Each time this Bit transitions from OFF to ON, the Follower Axis will attempt to move to the Relative Offset Position using the Relative Offset Velocity. If the Relative Offset Position is reached the instruction will turn this Bit OFF. This can be any Bit memory address.

> **NOTE:** *If the Bit assigned to the Goto Offset Signal stays ON, it indicates the Follower Axis does not have the capacity to overtake the Master Axis. Make the Follower Axis more responsive by configuring it with higher Maximum Velocity or a faster Acceleration, or both.*

On Success:

- **Set Bit**: The bit will become FALSE when the instruction is enabled and will remain FALSE until the enable leg goes back OFF. Once the enable leg turns OFF, if the instruction's device/parameters were valid, this bit will turn ON once the .CurrentVelocity reaches 0.

- **JMP to Stage**: Similarly, the JMP will not occur until after the instruction is enabled, then disabled, and all the instruction's device/parameters were valid and the .CurrentVelocity reaches 0.
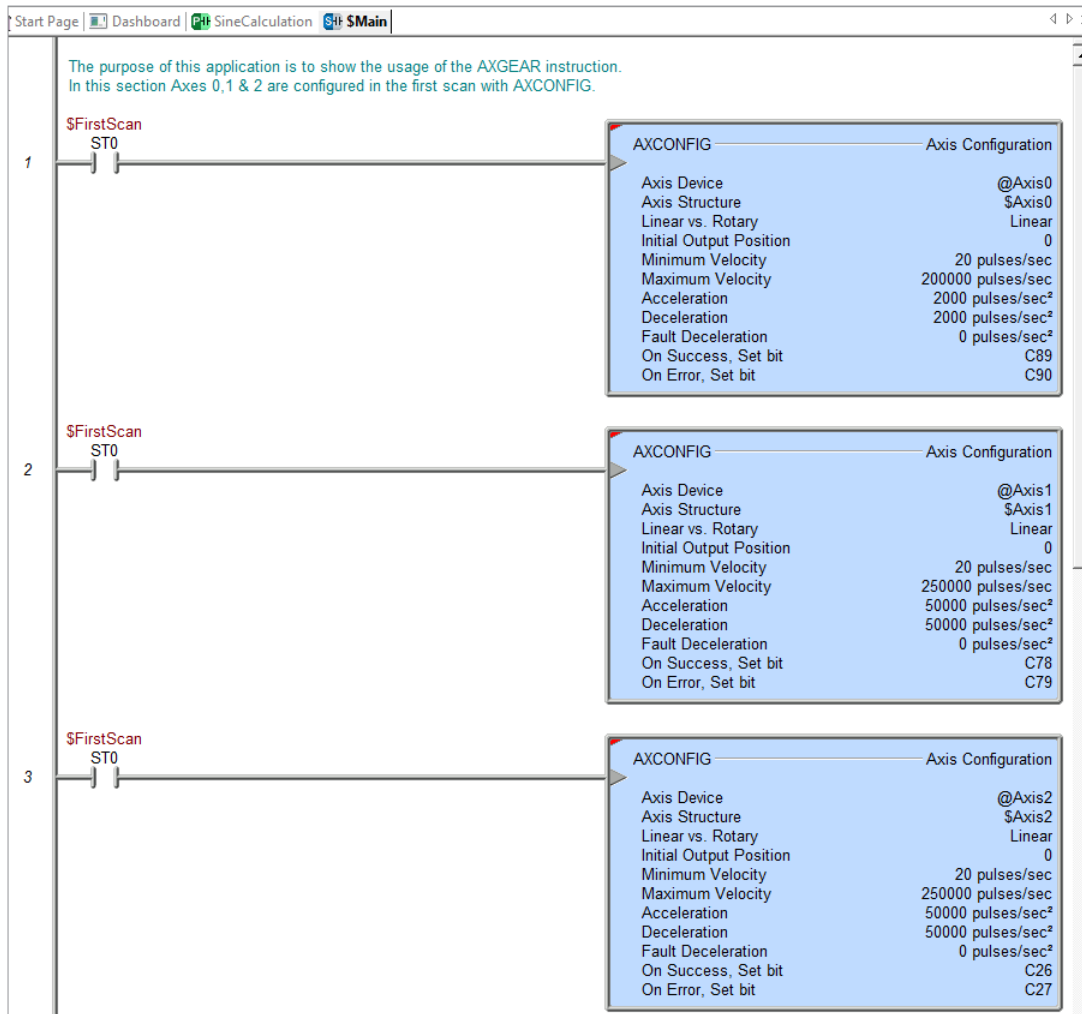
On Error:

- **Set Bit**: The bit will become FALSE when the instruction is enabled and will become TRUE if there was a problem with the parameters configured in the instruction, with the devices specified or if the AXFOLLOW move was interrupted before completion.

- **JMP to Stage**: If there was a problem with the parameters configured in the instruction, with the devices specified or the AXFOLLOW move was interrupted before completion, the PLC will jump to that stage.

AXFOLLOW, continued

Example AXFOLLOW ladder.

Rung 1: configures Axis 1 parameters

Rung 2: configures Axis 2 parameters

Rung 3: uses the AXVEL instruction to set the velocity structure

Rung 4: sets the AXFOLLOW instruction that will describe the motion of both axes
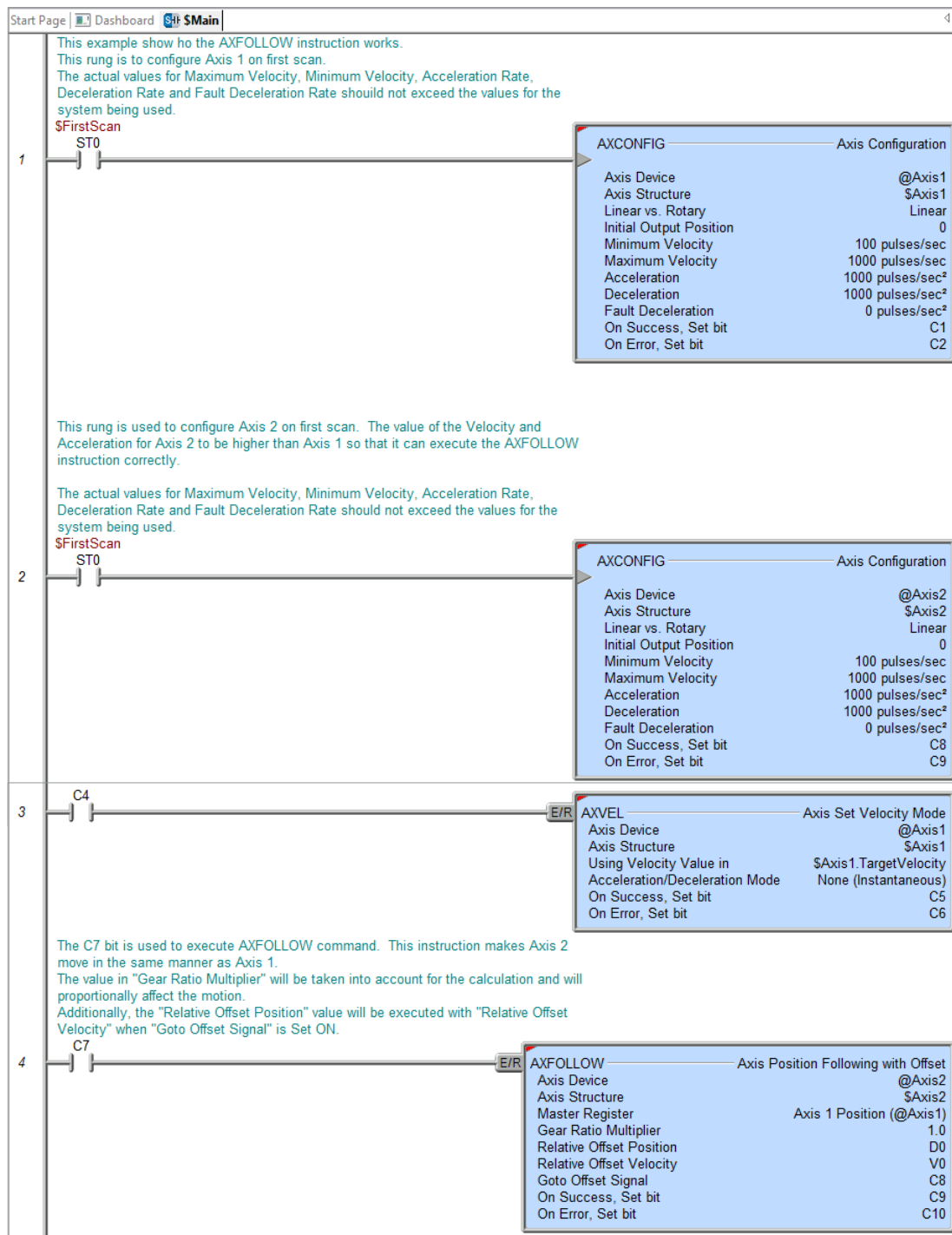
AXGEAR

The **Axis Electronic Gearing** (AXGEAR) instruction is used to create a type of Master/Slave connection that will synchronize the movement of one axis relative to another axis or a High-Speed Counter/Timer.

AXGEAR always behaves in a Position-Following mode. The Slave axis position is derived from the Master position so that any time the position of the Master Axis changes, the position of the Slave Axis will move to a position that has been modified by the user-supplied gear ratio. When the move to position operation is enabled, the AXGEAR instruction will use the current values of the Gear Ratio, the Axis Maximum Velocity, Accel, Decel and the Master's Target Position to calculate the move profile. Any changes to the Gear Ratio and Target Position will result in a trajectory change of the AXGEAR Axis.

To help keep track of the Axes movement relative to each other, any time the Axis is in Electronic Gearing mode the Slave Axis associated structure member, *.MstSlvCoordError*, will contain the difference (in pulse counts) between the Master Axis position and the Slave Axis position.

This instruction requires the following:

- Axis configured to be used as the slave Axis.
- One of the following to be used as the Master Register:
  - Position value from a configured master axis.
  - Accumulated value from a configured high speed counter/timer function.
- AXCONFIG instruction configured for each axis associated with the instruction.

The (a) *Slave Axis Device* specifies which axis to use as the slave axis and the (b) *Master Register* selects the axis position or high-speed counter/timer accumulated value that provides the position source value. These can be any of the axis or high-speed counter/timers available to the MPU or available to a BX-HSIO expansion module.

The Axis and high-speed counters/timers are configured in the MPUs *Setup BRX High-speed I/O* dialog or the BX-HSIO *Setup BX-HSIO Module* dialog. Please see detailed information on these under the "Available High-Speed Input and Output Features" earlier in this chapter. These dialog boxes can be opened directly from the AXGEAR instruction by clicking on the (c) **Configure Axis...** button and (d) **Configure Master Register Device...** button. These allow the user to:

- configure an axis or a counter/timer function for use with the instruction.
- verify or edit the physical I/O selected for an already configured axis or counter/timer function.

> **NOTE:** *Axis0 is a 'virtual axis' and will not generate pulses to physical outputs of the PLC. Axis0 can be used for generating pulse output profile register values to be used for Table Driven Outputs (TDOPRESET or TDOPLS) or as the master for other axes in following-type applications (AXCAM, AXFOLLOW or AXGEAR). Axis0 is also good to use for testing.*

AXGEAR, continued

Master Register: This is the Axis, High-Speed Counter or Timer that the Slave Axis will be electronically geared to. This can be any of the Axes or High-Speed Counter/Timers.

Non-Axis Master Filter Time: This parameter is only enabled if the Master Register is a High-Speed Counter/Timer. This value is the Filter Time Constant which specifies how often (in Seconds) the Slave's position is calculated. This can be any constant value or any numeric location.

Configure Master Register Device: This button will open the BRX High-Speed Input or Axis/Pulse Output dialog in the System Configuration.

Gear Ratio Multiplier: A multiplier that will be applied each time the Slave's position is calculated. A value of 0 will cause the Slave Axis to stop. This can be any constant value or any numeric location.

On Success:

- **Set Bit**: The bit will become FALSE when the instruction is enabled and will remain FALSE until the enable leg goes back OFF. Once the enable leg turns OFF, if the instruction's device/parameters were valid, this bit will turn ON once the .CurrentVelocity reaches 0.

- **JMP to Stage**: Similarly, the JMP will not occur until after the instruction is enabled, then disabled, and all the instruction's device/parameters were valid and the .CurrentVelocity reaches 0.

On Error:

- **Set Bit**: The bit will become FALSE when the instruction is enabled and will become TRUE if there was a problem with the parameters configured in the instruction, with the devices specified or if the AXGEAR move was interrupted before completion.

- **JMP to Stage**: If there was a problem with the parameters configured in the instruction, with the devices specified or the AXGEAR move was interrupted before completion, the PLC will jump to that stage.

Example AXGEAR ladder.

This example demonstrates the usage of the AXGEAR instruction. Axes 0, 1& 2 are configured in the first scan with AXCONFIG.

AXGEAR, continued

```
Start Page | Dashboard | SineCalculation | $Main
```

The purpose of this application is to show the usage of the AXGEAR instruction.
In this section Axes 0,1 & 2 are configured in the first scan with AXCONFIG.

1
$FirstScan
ST0

AXCONFIG — Axis Configuration

Axis Device	@Axis0
Axis Structure	$Axis0
Linear vs. Rotary	Linear
Initial Output Position	0
Minimum Velocity	20 pulses/sec
Maximum Velocity	200000 pulses/sec
Acceleration	2000 pulses/sec²
Deceleration	2000 pulses/sec²
Fault Deceleration	0 pulses/sec²
On Success, Set bit	C89
On Error, Set bit	C90

2
$FirstScan
ST0

AXCONFIG — Axis Configuration

Axis Device	@Axis1
Axis Structure	$Axis1
Linear vs. Rotary	Linear
Initial Output Position	0
Minimum Velocity	20 pulses/sec
Maximum Velocity	250000 pulses/sec
Acceleration	50000 pulses/sec²
Deceleration	50000 pulses/sec²
Fault Deceleration	0 pulses/sec²
On Success, Set bit	C78
On Error, Set bit	C79

3
$FirstScan
ST0

AXCONFIG — Axis Configuration

Axis Device	@Axis2
Axis Structure	$Axis2
Linear vs. Rotary	Linear
Initial Output Position	0
Minimum Velocity	20 pulses/sec
Maximum Velocity	250000 pulses/sec
Acceleration	50000 pulses/sec²
Deceleration	50000 pulses/sec²
Fault Deceleration	0 pulses/sec²
On Success, Set bit	C26
On Error, Set bit	C27

AXGEAR, continued

In this section when C66 goes high AXGEAR is executed. Axis 1 and Axis 2 will move proportionally depending on the data specified in gear ratio multiplier field of the AXGEAR instruction. Virtual Axis 0 position is the pointer used. As the Virtual Aixs 0 moves from position 0 to position 10000 the position of Axis 1 and Axis 2 will follow proportionally to the values in R0 and R1. Axis 2 rotation should be opposed to Axis 1.

(Ladder continued from previous page.)
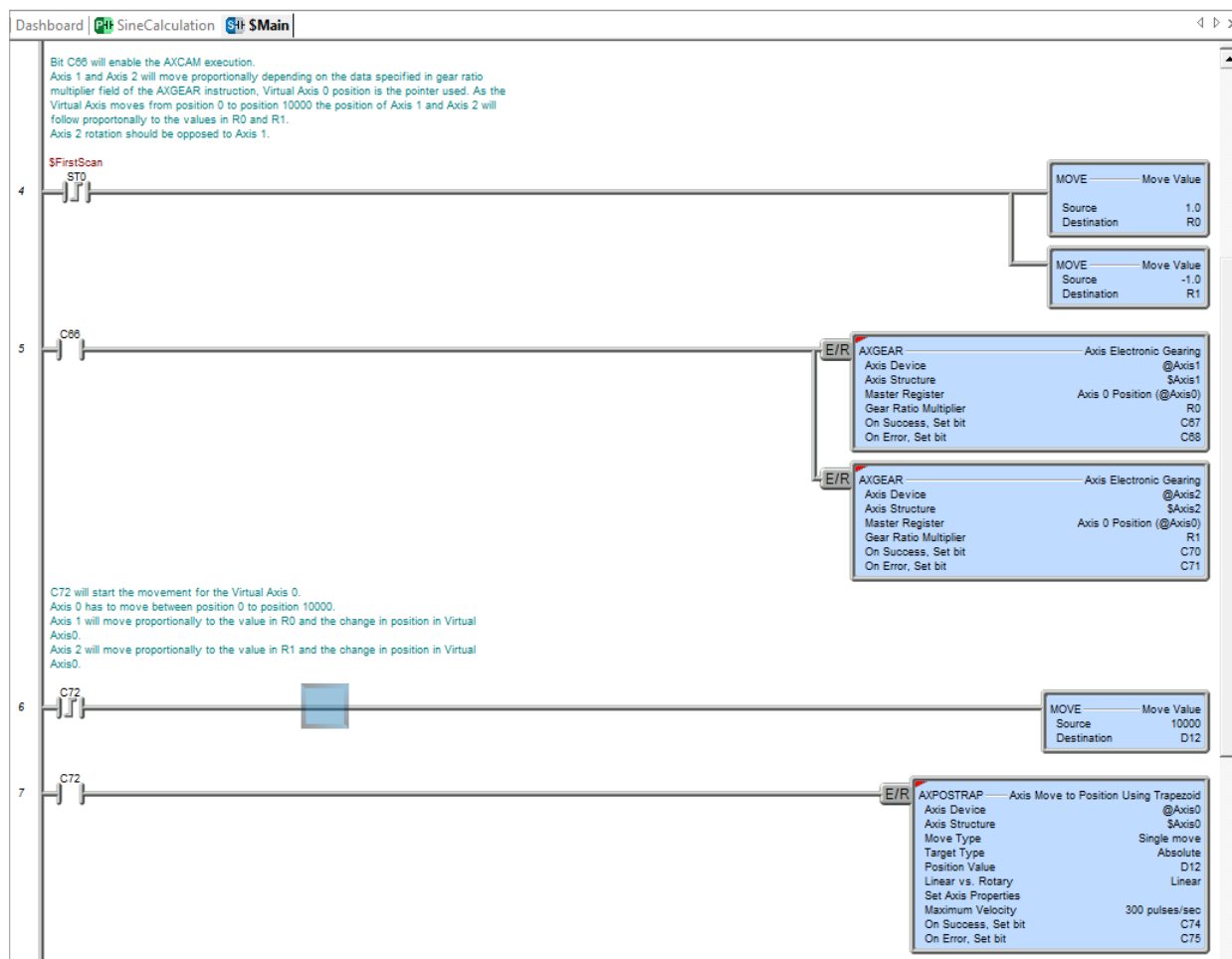
AXHOME

The **Axis Perform Home Search** (AXHOME) instruction is used to perform the necessary steps to move the specified Axis to a known starting position.

This instruction requires the following:

- Axis configured to be used as the slave Axis.
- AXCONFIG instruction configured for each axis associated with the instruction.

The (a) *Axis Device* specifies which axis to use as the slave axis. This can be any axis available to the MPU or available to a BX-HSIO expansion module.

The Axis is configured in the MPUs *Setup BRX High-speed I/O* dialog or the BX-HSIO *Setup BX-HSIO Module* dialog. Please see detailed information on these under the "Available High-Speed Input and Output Features" earlier in this chapter. These dialog boxes can be opened directly from the AXHOME instruction by clicking on the (b) **Configure Axis...** button. This allows the user to:

- configure an axis for use with the instruction.
- verify or edit the physical I/O selected for an already configured axis.

NOTE: *AxisO is a 'virtual axis' and will not generate pulses to physical outputs of the PLC. AxisO can be used for generating pulse output profile register values to be used for Table Driven Outputs (TDOPRESET or TDOPLS) or as the master for other axes in following-type applications (AXCAM, AXFOLLOW or AXGEAR). AxisO is also good to use for testing.*

AXHOME, continued

Homing Velocity (Signed): The maximum velocity the Axis will ramp up to when moving toward Discrete Input Limit 1. The Axis will ramp up and down using the Axis current Acceleration and Deceleration settings. This can be any constant value from -250000 to -10, 0, and 10 to 250000 or any numeric location containing a value in that range. The sign of the value will indicate the direction of travel. Positive numbers will move the Axis clockwise, negative numbers will move the Axis counter-clockwise.

Discrete Input Limit 1: The Home Search operation requires at least one discrete input.

- Discrete Input: The discrete input where the **Home** limit switch is connected. This must be one of the on-board discrete inputs.

- Event: Selects which of the following conditions will indicate the **Home** switch has been reached:

Rising Edge	⊓
Falling Edge	⊔
Rising OR Falling Edge	⊓
High Level	⊓
Low Level	⊔

Termination: Specifies what action to take after reaching the Home switch (Input Limit 1)

- **Position**: Use the Axis current positioning configuration to move the Axis to an absolute pulse count relative to the position of Discrete Input Limit 1.

- **Offset from Limit 1**: When Discrete Input Limit 1 is reached, the Axis marks the position of the switch and will then move the Axis toward the specified Position.

- **Creep to Second Position**: Move the Axis to a second discrete Input Limit switch.

  – Creep Velocity (Signed): The maximum frequency the Axis will ramp up to when moving toward Discrete Input Limit 2. The Axis will ramp up and down using the Axis current Acceleration and Deceleration settings. This can be any consta nt value between -250000 to -10, 0, and 10 to 250000 or any numeric location containing a value in that range. The sign of the value will indicate the direction of travel: positive numbers will move the Axis clockwise, negative numbers will move the Axis counter-clockwise.

  – Discrete Input Limit 2: Use a discrete input for Limit 2. This could be a second discrete input or the same input as Discrete Input Limit 1.

    Discrete Input: The discrete input where the Discrete Input Limit 2 switch is connected. This must be one of the on-board discrete inputs.

    Event: Selects which of the following conditions will indicate the Home switch has been reached:

Rising Edge	⊓
Falling Edge	⊔
Rising OR Falling Edge	⊓
High Level	⊓
Low Level	⊔

  – Decelerate to 0 Velocity: The Axis will decelerate from the Homing Velocity to 0.

AXHOME, continued

Zero Position At: Enable this option to have the Axis set its Current Position to 0 after the following step of the Home Search operation:

- Limit 1: Set the Current Position to 0 when the Discrete Input Limit 1 switch is reached. Any additional movement of the Axis during the Termination phase will be reflected in the Axis Current Position value.

- Home/When Done: Set the Current Position to 0 when all phases of the Home Search operation is complete.

Input Leg: Specifies how the instruction will be enabled to run:

- Edge Triggered: Each time the input logic transitions from OFF to ON this instruction will run to completion. This selection does not allow the Home Search operation to be interrupted once it has started except by manually turning OFF the .MasterEnable for the Axis, which will put the Axis into a fault condition that must be cleared before the Axis will be permitted to move. Use the AXRSTFAULT (Reset Axis Limit Fault) instruction to clear an Axis Fault.

- Power Flow Enabled: When the Input logic transitions from OFF to ON the instruction will begin to execute and will continue executing as long as the input logic remains ON. This selection allows the Home Search operation to be interrupted by turning the input logic OFF. This is an error condition but it does not put the Axis into a Fault condition.

On Success:

- **Set Bit**: The bit will become FALSE when the instruction is enabled and will become TRUE if the parameters configured in the instruction and proper devices were specified and the Home operation completes uninterrupted.

- **JMP to Stage**: If the parameters configured in the instruction and proper devices were specified, the PLC will jump to that stage once the Home operation completes.

On Error:

- **Set Bit**: The bit will become FALSE when the instruction is enabled and will become TRUE if there was a problem with the parameters configured in the instruction, with the devices specified or if the Home operation was interrupted before completion.

- **JMP to Stage**: If there was a problem with the parameters configured in the instruction, with the devices specified or the Home operation was interrupted before completion, the PLC will jump to that stage.

Examples of various AXHOME configurations are considered on the following pages.

AXHOME, continued

Example 1: Termination at Position of Limit 1
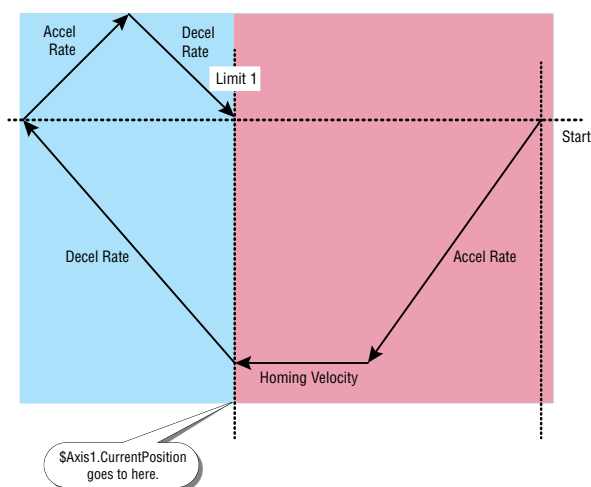
The Axis decelerates, changes direction and goes back to *$Axis.CurrentPosition* = 0 (Not looking at Limit 1 anymore).

Example 2: Termination Using Creep to Limit 2.
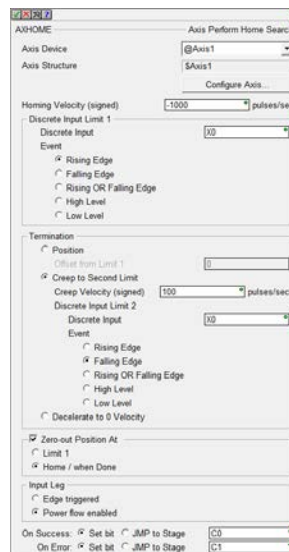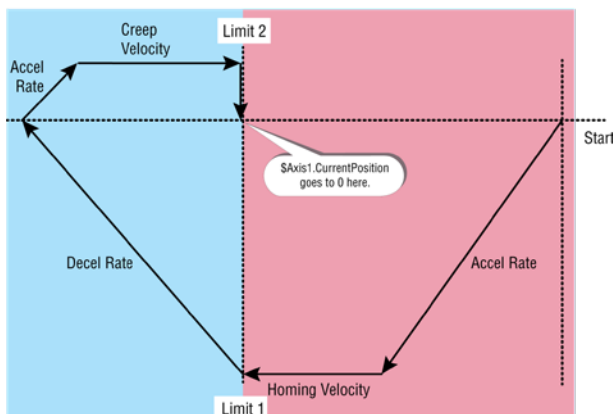
The Axis decelerates, changes direction and goes back to the falling edge of Limit 2 and the *$Axis. CurrentPosition* goes to 0 Take note that Limit 2 is the same physical switch in this situation. It could be a different one if it would make better sense for the application.

AXHOME, continued

Example 3: Using Level and Edge Limits, Sharing the Same Input.
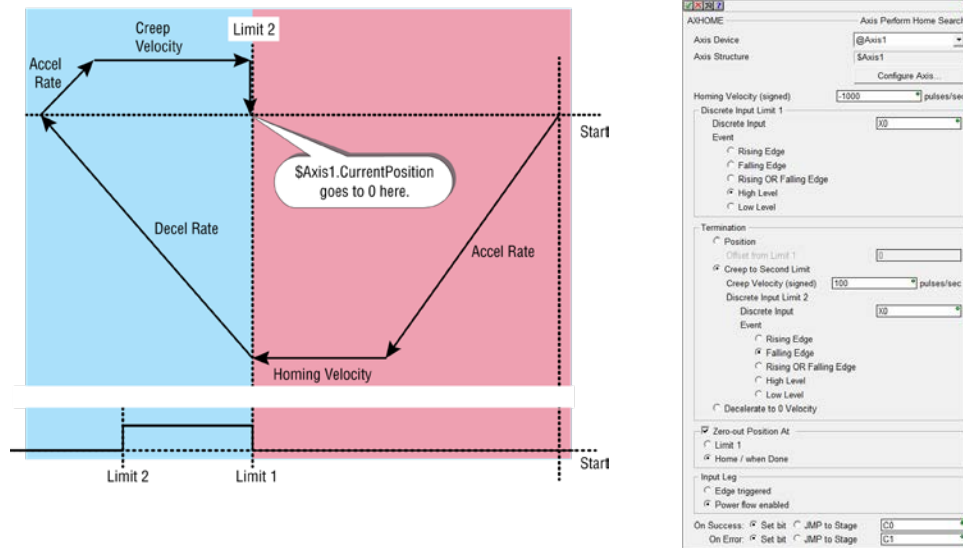
The Axis decelerates, changes direction and goes back to the Falling Edge of Limit 2 and the *$Axis. CurrentPosition* goes to 0. Take note that Limit 2 is the same physical switch in this situation. It could be a different one if it would make better sense for the application.

Example 4: Using Level and Edge Limits, Sharing the Same Input.
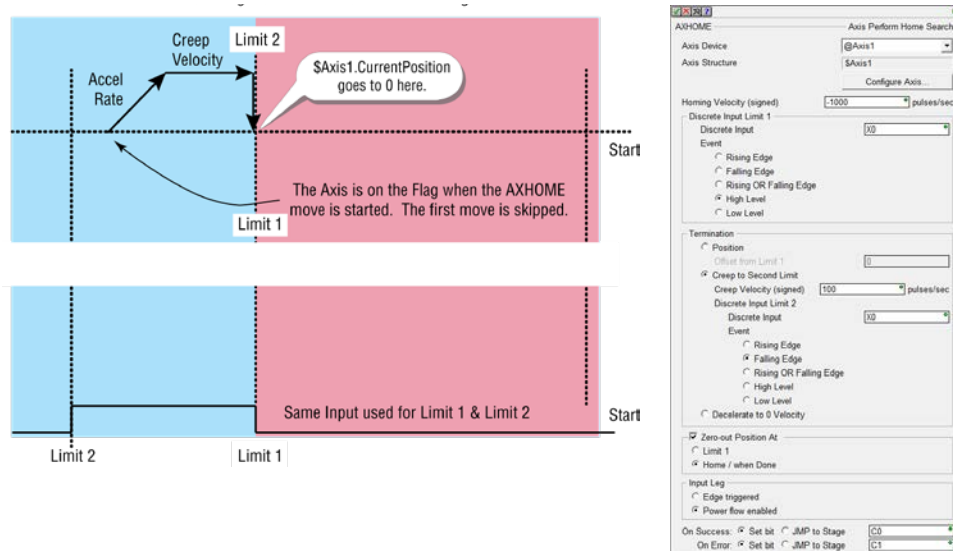
This is the same setup as the prior example but the Input for limit 1 on ON at the time of enabling the AXHOME instruction. The first movement does not occur in this case and the Axis creeps back to the Falling Edge of the switch.

AXHOME, continued

Example 5: Termination Decelerate to 0 Velocity.
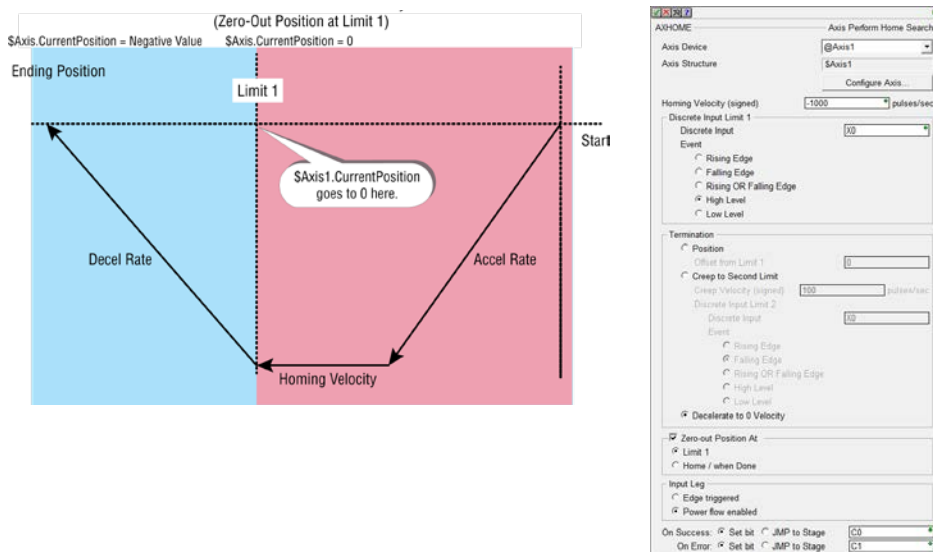
In this case, when the AXHOME operation is complete, the *$Axis1.CurrentPosition* will contain a negative value.

Example 6: Termination Decelerate to 0 Velocity.
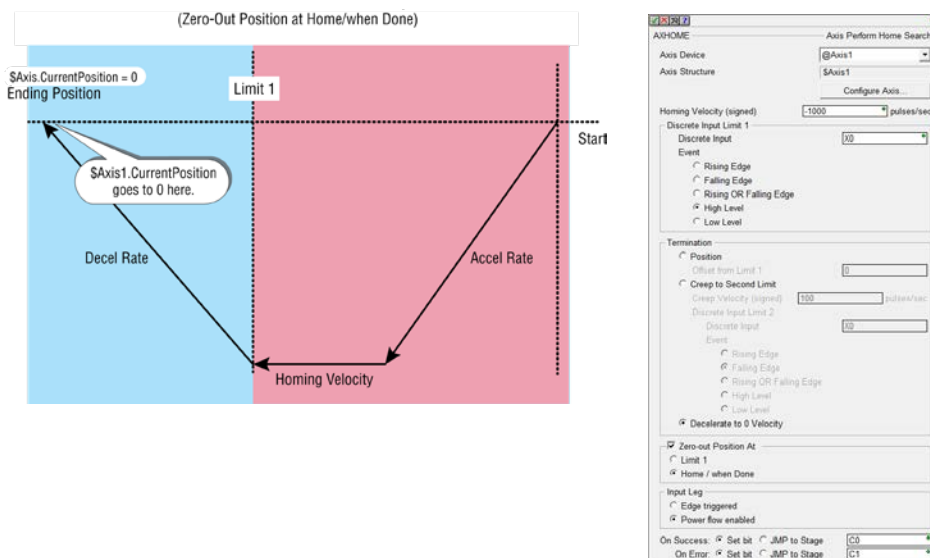
In this case, when the AXHOME operation is complete, the *$Axis1.CurrentPosition* will go to a 0 value.

AXJOG

The **Axis Jog** (AXJOG) instruction sets an Axis into a mode where its position can be manually adjusted by moving the Axis in the forward or reverse direction. Jogging is a simple velocity move that uses the Acceleration and Deceleration parameters specified in the Axis Configuration to ramp up to and ramp down from the Target Velocity specified in the AXJOG instruction or the Maximum Velocity specified in the Axis Configuration, whichever is lower.

This instruction requires the following:

- Axis configured to be used as the slave Axis.
- AXCONFIG instruction configured for each axis associated with the instruction.

The (a) *Axis Device* specifies which axis to use as the slave axis. This can be any axis available to the MPU or available to a BX-HSIO expansion module.

The Axis is configured in the MPUs *Setup BRX High-speed I/O* dialog or the BX-HSIO *Setup BX-HSIO Module* dialog. Please see detailed information on these under the "Available High-Speed Input and Output Features" earlier in this chapter. These dialog boxes can be opened directly from the AXJOG instruction by clicking on the (b) **Configure Axis...** button. This allows the user to:

- configure an axis for use with the instruction.
- verify or edit the physical I/O selected for an already configured axis.
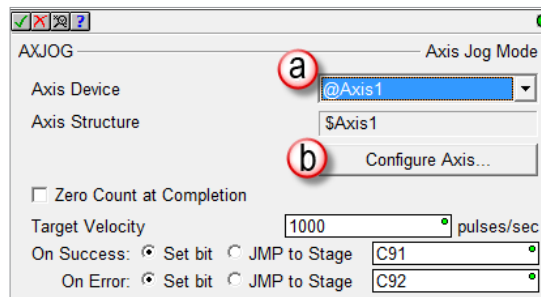
NOTE: *Axis0 is a 'virtual axis' and will not generate pulses to physical outputs of the PLC. Axis0 can be used for generating pulse output profile register values to be used for Table Driven Outputs (TDOPRESET or TDOPLS) or as the master for other axes in following-type applications (AXCAM, AXFOLLOW or AXGEAR). Axis0 is also good to use for testing.*
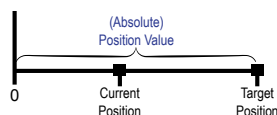
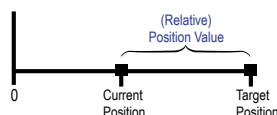**Zero Count at Completion**: When this option is enabled the Current Position of the Axis will be set to 0 when the Enable/Reset input transitions from ON to OFF. If this option is not enabled, the Axis will retain the contents of the Current Position after the Jog is complete.

Target Velocity: Specifies the speed that the Axis will output pulses when the Enable Leg of the AXJOG is ON and either the Forward or Reverse Leg is ON. If this value is higher than the Maximum Velocity specified in the Axis Configuration, the Axis Configuration value will be used. Forward and Reverse moves will use the Axis Configuration's Acceleration and Deceleration values when ramping up to and ramping down from the Maximum Velocity allowed.

AXJOG, continued

On Success:

- **Set Bit**: The bit will become FALSE when the instruction is enabled and will remain FALSE until the enable leg goes back OFF. Once the enable leg turns OFF, if the instruction's device/parameters were valid, this bit will turn ON once the .CurrentVelocity reaches 0.

- **JMP to Stage**: Similarly, the JMP will not occur until after the instruction is enabled, then disabled, and all the instruction's device/parameters were valid and the .CurrentVelocity reaches 0.

On Error:

- **Set Bit**: The bit will become FALSE when the instruction is enabled and will become TRUE when proper instruction parameters are NOT properly entered or the instruction IS NOT completed successfully. The specified bit is enabled with a SET operation, not an OUT operation. The On Error bit will remain ON even if the instruction input logic goes OFF.

- **JMP to Stage**: when proper instruction parameters are NOT properly entered or the instruction IS NOT completed successfully, the PLC will jump to that stage.

NOTE: Because this instruction puts the Axis into an operational mode (as opposed to performing a single operation), On Success is defined as getting the Axis into Jog mode with no errors. This means that the On Success indication will turn ON after the Enable/Reset input logic transitions from ON to OFF and the Axis' Current Velocity is at 0. When these conditions are met the Axis' Mode is "Idle". You should wait until the On Success indication turns ON before attempting to execute any other Axis instruction.

Example Ladder Logic:
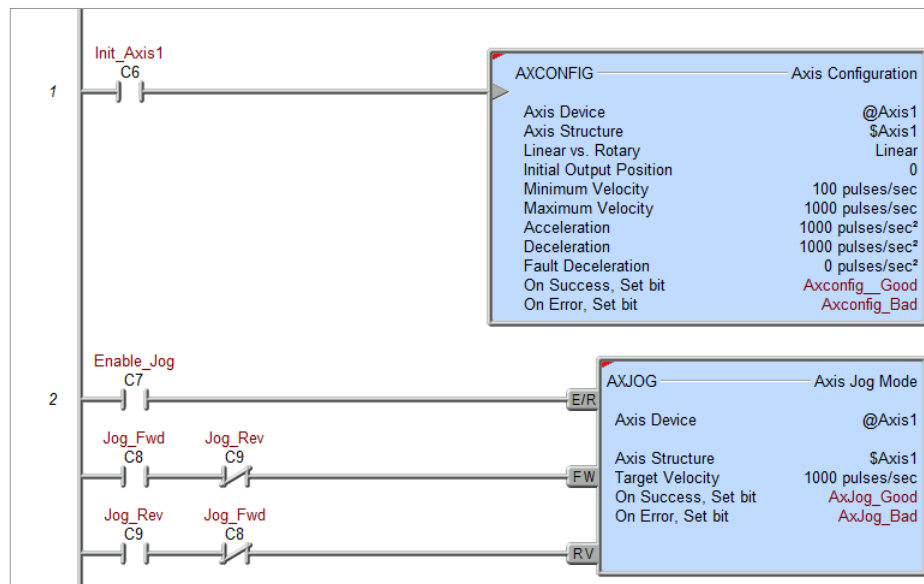
AXPOSSCRV

The **Axis Move to Position Using S-Curve** (AXPOSSCRV) instruction is used to move an Axis from its current position to a specified target position using the Axis configured parameters and the specified Jerk parameter which will yield an S-curve velocity profile.

This instruction requires the following:

- Axis configured to be used as the slave Axis.
- AXCONFIG instruction configured for each axis associated with the instruction.

The (a) *Axis Device* specifies which axis to use as the slave axis. This can be any axis available to the MPU or available to a BX-HSIO expansion module.

The Axis is configured in the MPUs *Setup BRX High-speed I/O* dialog or the BX-HSIO *Setup BX-HSIO Module* dialog. Please see detailed information on these under the "Available High-Speed Input and Output Features" earlier in this chapter. These dialog boxes can be opened directly from the AXPOSSCRV instruction by clicking on the (b) **Configure Axis...** button. This allows the user to:

- configure an axis for use with the instruction.
- verify or edit the physical I/O selected for an already configured axis.

> **NOTE:** *Axis0 is a 'virtual axis' and will not generate pulses to physical outputs of the PLC. Axis0 can be used for generating pulse output profile register values to be used for Table Driven Outputs (TDOPRESET or TDOPLS) or as the master for other axes in following-type applications (AXCAM, AXFOLLOW or AXGEAR). Axis0 is also good to use for testing.*

AXPOSSCRV, continued

Input Leg

- **Edge Triggered**: The move to position operation will be performed each time the input transitions from OFF to ON. Once a move to position operation is in progress it can only be stopped by manually setting the axis .MasterEnable member to OFF, which will put the axis into a Fault state.

- **Power flow Enabled**: The move to position operation will begin when the input transitions from OFF to ON and will continue to completion as long as the input remains ON. This selection has the benefit of being able to interrupt a move to position operation by setting the input state to OFF and NOT putting the Axis into a Fault state.

Target Type

- **Absolute**: Absolute moves are measured from the axis zero position. When an axis is initialized, its current position is set to 0. An absolute move to 10000 will generate 10000 pulses to move the axis forward 10000 pulses. A subsequent absolute move to -10000 will generate 20000 pulses to move the axis backward past 0 to the -10000 position. If you execute an absolute move with a Position Value that is the same as the Axis Current Position the axis will not move as the absolute position is already reached.

- **Relative**: Relative moves are measured from the Axis' current position. When an axis is initialized, its current position is set to 0. A relative move to 10000 will generate 10000 pulses to move the axis forward 10000 pulses. A subsequent relative move of -10000 will generate 10000 pulses to move the axis backward to that 0 position.

- **Zero-out Current Position Before Initial Move**: Enable this option to have the axis set its Current Position value to 0 before any move operation.

- **Position Value**: The target position (pulse count) to move the axis to. This can be any constant value, or any numeric location with a value in that range.

AXPOSSCRV, continued

Linear vs. Rotary

- **Linear**: The series of pulses will produce forward or backwards motion along a fixed linear path. If the Target Position value is a higher value than the Current Position value, the resulting move will be in the positive (increasing) direction. If the Target Position value is a lower value than the Current Position, the resulting move will be in the negative (decreasing) direction.

- **Rotary**: The series of pulses will produce Clockwise or CounterClockwise motion along a fixed circular path either as an absolute position value or relative position as noted in the discussion below.

  Move to Absolute Target in Clockwise Direction: Taking into account the Rotary Range specified in the AXCONFIG and the Target Position Value, the PLC will always generate pulses in an increasing (positive) direction. So if the Target Position value is lower than the Current Position, the PLC will 'roll over' in the clockwise direction to achieve the position. If the Target Position Value specified exceeds the Rotary Range, the Axis will move to the modulus result. For example: If the Rotary Range is 0–359 (360 degrees) and a Target Position Value of 500 was specified, the Axis will output 140 pulses.

  Move to Absolute Target in Counterclockwise Direction: Taking into account the Rotary Range specified in the AXCONFIG and the Target Position Value, the PLC will always generate pulses in a decreasing (negative) direction. So if the Target Position value is higher than the Current Position value, the PLC will 'roll over' in the Counter clockwise direction to achieve the position. If the Target Position Value specified exceeds the Rotary Range, the Axis will move to the modulus result. For example: If the Rotary Range is 0–359 (360 degrees) and a Target Position Value of 500 was specified, the Axis will output 220 pulses.

AXPOSSCRV, continued

Move to Absolute Target in Shortest Direction: Taking into account the Rotary Range specified in the AXCONFIG and the Target Position Value, the PLC will calculate the shortest distance between the Target Position Value and the Current Position value and go in either Clockwise or Counter Clockwise direction to achieve the target. If the Target Position Value specified exceeds the Rotary Range, the Axis will move to the modulus result. For example: If the Rotary Range is 0–359 (360 degrees) and a Target Position Value of 500 was specified, the Axis will output 140 pulses.

Relative Rotary Target Type, so sign of Position Value parameter specifies direction: A positive Position Value will move the Axis in a Clockwise direction and a negative Position Value will move the Axis in a Counter Clockwise direction. If the Target Position Value specified exceeds the Rotary Range, the Axis will move to the modulus result. For example: If the Rotary Range is 0–359 (360 degrees) and a Target Position Value of 500 was specified, the Axis will output 140 pulses.

Jerk: The Jerk term specifies how quickly the Axis is allowed to achieve maximum Acceleration and Deceleration on its way to reaching maximum Velocity. This value is specified in pulses/sec3.

Supersede Default Properties: These parameters allow this AXPOSSCRV instruction to override the values specified in the AXCONFIG instruction. They only temporarily change the values in the AXCONFIG for this movement. To permanently change the parameters in the AXCONFIG, use the Set Axis Properties (AXSETPROP) instruction.

- **Maximum Velocity** (pulses/sec): The fastest frequency of output pulses that will be generated during the move to position operation. This can be any positive constant from 10 to 250000, or any numeric location with a value in that range.

- **Acceleration Rate** (pulses/sec^2): The rate at which the pulses will be generated when the axis is ramping up from a slower pulse rate to a higher pulse rate. This can be any positive constant or any numeric location with a value in that range.

- **Deceleration Rate** (pulses/sec^2): The rate at which the pulses will be generated when the axis is ramping down from a higher pulse rate to a slower pulse rate. This can be any positive constant or any numeric location with a value in that range.

AXPOSSCRV, continued

On Success:

- **Set Bit**: The bit will become FALSE when the instruction is enabled and will become TRUE when instruction parameters are properly entered and the instruction completes successfully. The specified bit is enabled with a SET operation, not an OUT operation. The On Success bit will remain ON even if the instruction input logic goes OFF.

- **JMP to Stage**: When instruction parameters are properly entered and the instruction completes successfully, the PLC will jump to that stage.

On Error:

- **Set Bit**: The bit will become FALSE when the instruction is enabled and will become TRUE when proper instruction parameters are NOT properly entered or the instruction IS NOT completed successfully. The specified bit is enabled with a SET operation, not an OUT operation. The On Error bit will remain ON even if the instruction input logic goes OFF

- **JMP to Stage**: When proper instruction parameters are NOT properly entered or the instruction IS NOT completed successfully, the PLC will jump to that stage.

Example Usage:

AXPOSTRAP

The **Axis Move to Position Using Trapezoid** (AXPOSTRAP) instruction is used to move an Axis from its current position to a specified target position using the Axis' configured parameters which will yield a trapezoid velocity profile (linear acceleration and deceleration).

This instruction requires the following:

- Axis configured to be used as the slave Axis.
- AXCONFIG instruction configured for each axis associated with the instruction.

The (a) *Axis Device* specifies which axis to use as the slave axis. This can be any axis available to the MPU or available to a BX-HSIO expansion module.

The Axis is configured in the MPUs *Setup BRX High-speed I/O* dialog or the BX-HSIO *Setup BX-HSIO Module* dialog. Please see detailed information on these under the "Available High-Speed Input and Output Features" earlier in this chapter. These dialog boxes can be opened directly from the AXPOSTRAP instruction by clicking on the (b) **Configure Axis...** button. This allows the user to:

- configure an axis for use with the instruction.
- verify or edit the physical I/O selected for an already configured axis.
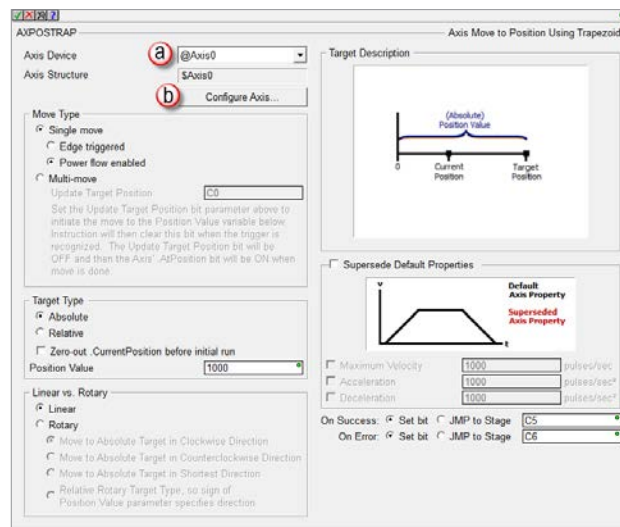
NOTE: *Axis0 is a 'virtual axis' and will not generate pulses to physical outputs of the PLC. Axis0 can be used for generating pulse output profile register values to be used for Table Driven Outputs (TDOPRESET or TDOPLS) or as the master for other axes in following-type applications (AXCAM, AXFOLLOW or AXGEAR). Axis0 is also good to use for testing.*

Move Type

- **Single move**: The axis will perform 1 move to the specified position.

    Edge Triggered: The move to position operation will be performed each time the input transitions from OFF to ON. Once a move to position operation is in progress it can only be stopped by manually setting the axis .MasterEnable member to OFF, which will put the axis into a Fault state.

    Power flow Enabled: The move to position operation will begin when the input transitions from OFF to ON and will continue to completion as long as the input remains ON. This selection has the

AXPOSTRAP, continued

benefit of being able to interrupt a move to position operation by setting the input state to OFF and NOT putting the axis into a Fault state.

- **Multi-move**: The axis can perform multiple moves by changing the "Position Value" register and triggering the "Update Target Position" bit.

- **Trigger Target Position**: Specify an internal bit for this field. Changing this bit from OFF to ON will result in the PLC changing its Target Position (without stopping) to the value loaded into the *Position Value* register. The PLC will automatically turn the *Update Target Position* bit back OFF after changing its target position. The *.AtPosition* bit member of the Axis structure can be monitored to ensure completion.

Target Type

- **Absolute**: Absolute moves are measured from the axis zero position. When an axis is initialized, its current position is set to 0. An absolute move to 10000 will generate 10000 pulses to move the Axis forward 10000 pulses. A subsequent absolute move to -10000 will generate 20000 pulses to move the Axis backward past 0 to the -10000 position. If you execute an absolute move with a Position Value that is the same as the axis Current Position the axis will not move as the absolute position is already reached.

- **Relative**: Relative moves are measured from the axis current position. When an axis is initialized, its current position is set to 0. A relative move to 10000 will generate 10000 pulses to move the axis forward 10000 pulses. A subsequent relative move of -10000 will generate 10000 pulses to move the Axis backward to that 0 position.

- **Zero Current Position Before Initial Move**: Enable this option to have the Axis set its Current Position value to 0 before any move operation.

  Single Move: Current Position is set to 0 before the move to position operation begins.

  Multi-Move: Current Position is set to 0 before the first move to position but does not happen on subsequent OFF to ON transitions of the Trigger Target Position.

- **Position Value**: The target position (pulse count) to move the Axis to. This can be any constant value, or any numeric location with a value in that range.

AXPOSTRAP, continued

Linear vs. Rotary:

- **Linear**: The series of pulses will produce forward or backwards motion along a fixed linear path. If the Target Position value is a higher value than the Current Position value, the resulting move will be in the positive (increasing) direction. If the Target Position value is a lower value than the Current Position, the resulting move will be in the negative (decreasing) direction.

- **Rotary**

  Move to Absolute Target in Clockwise Direction: Taking into account the Rotary Range specified in the AXCONFIG and the Target Position Value, the PLC will always generate pulses in an increasing (positive) direction. So if the Target Position value is lower than the Current Position, the PLC will 'roll over' in the clockwise direction to achieve the position. If the Target Position Value specified exceeds the Rotary Range, the Axis will move to the modulus result. For example: If the Rotary Range is 0–359 (360 degrees) and a Target Position Value of 500 was specified, the Axis will output 140 pulses.

  Move to Absolute Target in Counterclockwise Direction: Taking into account the Rotary Range specified in the AXCONFIG and the Target Position Value, the PLC will always generate pulses in a decreasing (negative) direction. So if the Target Position value is higher than the Current Position value, the PLC will 'roll over' in the Counter clockwise direction to achieve the position. If the Target Position Value specified exceeds the Rotary Range, the Axis will move to the modulus result. For example: If the Rotary Range is 0–359 (360 degrees) and a Target Position Value of 500 was specified, the Axis will output 220 pulses.

AXPOSTRAP, continued

Move to Absolute Target in Shortest Direction: Taking into account the Rotary Range specified in the AXCONFIG and the Target Position Value, the PLC will calculate the shortest distance between the Target Position Value and the Current Position value and go in either Clockwise or Counter Clockwise direction to achieve the target. If the Target Position Value specified exceeds the Rotary Range, the Axis will move to the modulus result. For example: If the Rotary Range is 0–359 (360 degrees) and a Target Position Value of 500 was specified, the Axis will output 140 pulses.

Relative Rotary Target Type, so sign of Position Value parameter specifies direction: A positive Position Value will move the Axis in a Clockwise direction and a negative Position Value will move the Axis in a Counter Clockwise direction. If the Target Position Value specified exceeds the Rotary Range, the Axis will move to the modulus result. For example: If the Rotary Range is 0–359 (360 degrees) and a Target Position Value of 500 was specified, the Axis will output 140 pulses.

Supersede Default Properties

Selecting these parameters allows the AXPOSTRAP instruction to override the values specified in the AXCONFIG instruction. They only temporarily change the values in the AXCONFIG for this movement. To permanently change the parameters in the AXCONFIG, use the Set Axis Properties (AXSETPROP) instruction.

- **Maximum Velocity** (pulses/sec): The fastest frequency of output pulses that will be generated during the move to position operation. This can be any positive constant from 10 to 250000, or any numeric location with a value in that range.

- **Acceleration Rate** (pulses/sec^2): The rate at which the pulses will be generated when the axis is ramping up from a slower pulse rate to a higher pulse rate. This can be any positive constant or any numeric location with a value in that range.

AXPOSTRAP, continued

- **Deceleration Rate** (pulses/sec^2): The rate at which the pulses will be generated when the axis is ramping down from a higher pulse rate to a slower pulse rate. This can be any positive constant or any numeric location with a value in that range.

On Success:

- **Set Bit**: The bit will become FALSE when the instruction is enabled and will become TRUE when instruction parameters are properly entered and the instruction completes successfully. The specified bit is enabled with a SET operation, not an OUT operation. The On Success bit will remain ON even if the instruction input logic goes OFF.

- **JMP to Stage**: When instruction parameters are properly entered and the instruction completes successfully, the PLC will jump to that stage.

On Error:

- **Set Bit**: The bit will become FALSE when the instruction is enabled and will become TRUE when proper instruction parameters are NOT properly entered or the instruction IS NOT completed successfully. The specified bit is enabled with a SET operation, not an OUT operation. The On Error bit will remain ON even if the instruction input logic goes OFF.

- **JMP to Stage**: When proper instruction parameters are NOT properly entered or the instruction IS NOT completed successfully, the PLC will jump to that stage.

Example Usage:

AXRSTFAULT

The **Reset Axis Limit Fault** (AXRSTFAULT) instruction is used to clear the fault in an axis that has either reached one of the configured Fault Limits while it was moving or has had its *.MasterEnable* manually reset.

After the instruction has cleared the fault state in an axis fault, any attempt to move the Axis in the same direction that caused the fault will immediately generate another fault condition. Movement of the axis in the opposite direction that caused the fault is permitted.

Triggering the AXCONFIG (Axis Configuration) instruction will also reset an Axis that is in a fault state.

This instruction requires the following:

- Axis configured to be used as the slave Axis.
- AXCONFIG instruction configured for each axis associated with the instruction.

The (a) *Axis Device* specifies which axis to use as the slave axis. This can be any axis available to the MPU or available to a BX-HSIO expansion module.

The Axis is configured in the MPUs *Setup BRX High-speed I/O* dialog or the BX-HSIO *Setup BX-HSIO Module* dialog. Please see detailed information on these under the "Available High-Speed Input and Output Features" earlier in this chapter. These dialog boxes can be opened directly from the AXRSTFAULT instruction by clicking on the (b) **Configure Axis...** button. This allows the user to:

- configure an axis for use with the instruction.
- verify or edit the physical I/O selected for an already configured axis.
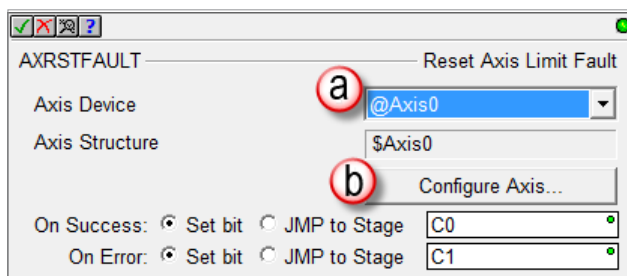
NOTE: Axis0 is a 'virtual axis' and will not generate pulses to physical outputs of the PLC. Axis0 can be used for generating pulse output profile register values to be used for Table Driven Outputs (TDOPRESET or TDOPLS) or as the master for other axes in following-type applications (AXCAM, AXFOLLOW or AXGEAR). Axis0 is also good to use for testing.

On Success:
- **Set Bit**: The bit will become FALSE when the instruction is enabled and will become TRUE when instruction parameters are properly entered and the instruction completes successfully. The specified bit is enabled with a SET operation, not an OUT operation. The On Success bit will remain ON even if the instruction input logic goes OFF.

- **JMP to Stage**: When instruction parameters are properly entered and the instruction completes successfully, the PLC will jump to that stage.

On Error:
- **Set Bit**: The bit will become FALSE when the instruction is enabled and will become TRUE when proper instruction parameters are NOT properly entered or the instruction IS NOT completed

AXRSTFAULT, continued

successfully. The specified bit is enabled with a SET operation, not an OUT operation. The On Error bit will remain ON even if the instruction input logic goes OFF.

- **JMP to Stage**: When proper instruction parameters are NOT properly entered or the instruction IS NOT completed successfully, the PLC will jump to that stage.

Example Usage
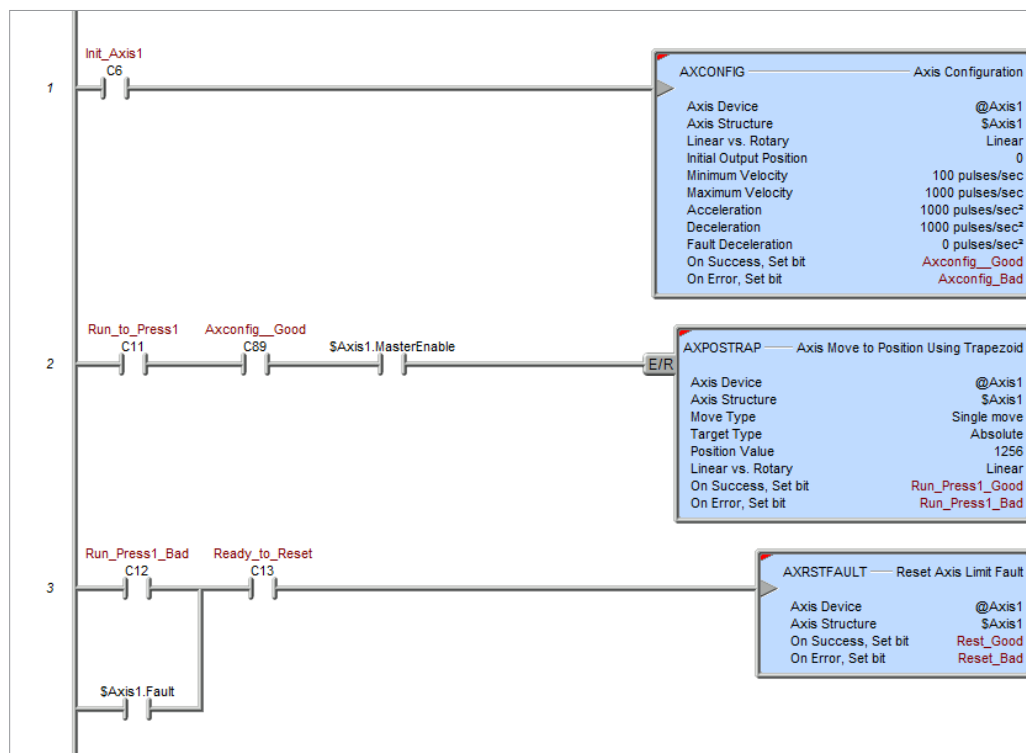
AXSCRIPT

The **Run a Sequence of Axis Commands** (AXSCRIPT) instruction is used to process a series of commands that can control an Axis, change PLC memory items, and interact with the ladder logic. This series of commands allow for Axis movements that cannot be done with the existing Axis instructions. You should only be using the AXSCRIPT instruction if the other Axis instructions don't give you the control that you need over that Axis. Although some of the Axis-control commands have similar sounding names to the existing Axis instructions, *they do not work exactly the same*.

Unlike the other Axis instructions which will leave the Axis in Idle mode when the instruction ends, the AXSCRIPT will leave the Axis in the last state that it was put into by the commands in the instruction when it ends. This can be used to your advantage by allowing you to "chain" AXSCRIPT instructions together. And because the other Axis instructions can start from any velocity / position, you can use an AXSCRIPT to start a move then finish the move with any of the other Axis instructions, which will end with the Axis in Idle mode.

If the parameters for any of the commands use a PLC memory location instead of a constant value, the instruction will use the value in those parameter locations when it gets to the step that contains them.

This instruction requires the following:

- Axis configured to be used as the slave Axis.
- AXCONFIG instruction configured for each axis associated with the instruction.

The (a) *Axis Device* specifies which axis to use as the slave axis. This can be any axis available to the MPU or available to a BX-HSIO expansion module.

The Axis is configured in the MPUs *Setup BRX High-speed I/O* dialog or the BX-HSIO *Setup BX-HSIO Module* dialog. Please see detailed information on these under the "Available High-Speed Input and Output Features" earlier in this chapter. These dialog boxes can be opened directly from the AXSCRIPT instruction by clicking on the (b) **Configure High Speed Resource...** button. This allows the user to:

- configure an axis for use with the instruction.
- verify or edit the physical I/O selected for an already configured axis.

NOTE: *Axis0 is a 'virtual axis' and will not generate pulses to physical outputs of the PLC. Axis0 can be used for generating pulse output profile register values to be used for Table Driven Outputs (TDOPRESET or TDOPLS) or as the master for other axes in following-type applications (AXCAM, AXFOLLOW or AXGEAR). Axis0 is also good to use for testing.*

AXSCRIPT, continued

Input Leg

- **Edge Triggered**: The AXSCRIPT will be performed each time the input transitions from OFF to ON. Once an AXSCRIPT operation is in progress it can only be stopped by manually setting the Axis' *.MasterEnable* structure member to OFF, which will put the Axis into a Fault state.

- **Power flow Enabled**: The AXSCRIPT will begin when the input transitions from OFF to ON and will continue to completion as long as the input remains ON. This selection has the benefit of being able to interrupt an AXSCRIPT operation by setting the input state to OFF and NOT putting the Axis into a Fault state.

The instruction's commands are listed in the order they will be executed. Each AXSCRIPT instruction can contain up to 50 commands.

- **Command**: is the Axis command or PLC command to execute. Refer to the Available Commands section below for details on the individual commands and their required parameters.

- **Parameter 1**: is the parameter required by the command.

- **Parameter 2**: is the optional 2nd parameter for any command that requires it.

The buttons below the table provide functions that are used to organize the rows in the table:

- **Add**: opens the row editor sub-dialog so that a new entry can be added to the end of the table.

- **Insert**: inserts an empty row before the currently selected row

- **Edit**: opens the currently selected row in the row editor sub-dialog

- **Remove**: deletes the currently selected row

- **Move Up / Move Down**: moves the currently selected row up one row or down one row respectively.

On Success:

- **Set Bit**: The bit will become FALSE when the instruction is enabled and will become TRUE when instruction parameters are properly entered and the instruction completes successfully. The specified bit is enabled with a SET operation, not an OUT operation. The On Success bit will remain ON even if the instruction input logic goes OFF.

- **JMP to Stage**: When instruction parameters are properly entered and the instruction completes successfully, the PLC will jump to that stage.

On Error:

- **Set Bit**: The bit will become FALSE when the instruction is enabled and will become TRUE when proper instruction parameters are NOT properly entered or the instruction IS NOT completed successfully. The specified bit is enabled with a SET operation, not an OUT operation. The On Error bit will remain ON even if the instruction input logic goes OFF.

- **JMP to Stage**: When proper instruction parameters are NOT properly entered or the instruction IS NOT completed successfully, the PLC will jump to that stage.

Available Commands:

The commands available for use in the AXSCRIPT instruction are listed in the following functional groups: Velocity, Position, Rotary, Follower, Events, Ladder Operations, and Misc.

AXSCRIPT, continued

These commands are entered in the table through the row editor that prompts for the command to execute and any parameters required by that command. And remember, if the parameters for any of the commands use a PLC memory location instead of a constant value, the instruction will use the value in the parameter location when it processes the step that contains the parameter reference.

After selecting the command and entering the parameters, use the buttons at the bottom of the dialog as follows:

- **Save**: will commit any changes that were made to the command and close the row editor.

- **Save / Next**: will commit any changes that were made to the command and create a new command entry for the next row in the table.

- **Save / Insert**: will commit any changes that were made to the command and create a new command entry for the previous row in the table.

- **Cancel**: will close the row editor without saving any changes that have been made.

Velocity commands are very similar to the Axis Set Velocity Mode (AXVEL) instruction:

The **Raw Velocity** command immediately sets the Axis' velocity to the specified Velocity value and immediately moves to the next step in the table.

The **Ramp to Velocity** command sets the Axis' Target Velocity to the specified Velocity value, and the Axis will begin to ramp toward that new Target Value using either an S-Curve (with Jerk) or Trapezoid profile as selected. If **Wait Until @Vel** is selected, the table will remain at this step until the Axis reaches the Target Velocity. If **Immediately go to next step** is selected, the table will immediately move to the next step without waiting on the Axis to reach the Target Velocity.

The **Go to Idle Mode (immediate stop)** command will instruct the Axis to immediately go to IDLE mode which takes the Axis out of the current execution mode. *If the Axis is moving it will immediately stop; it will NOT ramp to a stop.* To avoid potential machine damage, you should probably use a **Ramp to Velocity of 0** command (to bring the Axis to a controlled stop) before using the **Go to Idle Mode** command.

AXSCRIPT, continued

NOTE: *Unlike the other Axis instructions which will leave the Axis in Idle mode, the AXSCRIPT will leave the Axis in the last state that it was put into by the commands in the instruction. Unless you have a purpose for leaving the Axis in a state where it is potentially moving (not idle) when this instruction finishes, we recommend that you make sure the Axis is not moving when the AXSCRIPT instruction completes by adding a Go to Idle Mode command as the last step in the table.*

Position commands approximate the Axis Move to Position Using Trapezoid (AXPOSTRAP) and Axis Move to Position Using S-Curve (AXPOSSCRV) instructions when making **Linear position** moves.

The **Move to Absolute Pos w/Trap** command will use a Trapezoid profile to move the Axis to the specified absolute Target Position, optionally using a different Maximum Velocity. If **Wait 'til @Pos** is selected, the table will remain at this step until the Axis reaches the Target Position. If **Immediately go to next step** is selected, the table will immediately move to the next step without waiting on the Axis to reach the Target Position.

The **Move to Relative Pos w/Trap** command will use a Trapezoid profile to move the Axis to the specified relative Position Offset, optionally using a different Maximum Velocity. If **Wait 'til @Pos** is selected, the table will remain at this step until the Axis reaches the Position Offset. If **Immediately go to next step** is selected, the table will immediately move to the next step without waiting on the Axis to reach the Position Offset.

The **Move to Absolute Pos w/ SCurve** command will use an S-Curve profile to move the Axis to the specified absolute Target Position, optionally using a different Maximum Velocity. If **Wait 'til @Pos** is selected, the table

AXSCRIPT, continued

will remain at this step until the Axis reaches the Target Position. If **Immediately go to next step** is selected, the table will immediately move to the next step without waiting on the Axis to reach the Target Position.

The **Move to Relative Pos w/ SCurve** command will use an S-Curve profile to move the Axis to the specified relative Position Offset, optionally using a different Maximum Velocity. If **Wait 'til @Pos** is selected, the table will remain at this step until the Axis reaches the Position Offset. If **Immediately go to next step** is selected, the table will immediately move to the next step without waiting on the Axis to reach the Position Offset.

Rotary commands approximate the Axis Move to Position Using Trapezoid (AXPOSTRAP) instruction when making Rotary position moves.

The **Rotary Move to Relative Pos** command will use a Trapezoid profile to move the Axis to the specified relative Position Offset, optionally using a different Maximum Velocity. If **Wait 'til @Pos** is selected, the table will remain at this step until the Axis reaches the Position Offset. If **Immediately go to next step** is selected, the table will immediately move to the next step without waiting on the Axis to reach the Position Offset.

The **Rotary Move Clockwise** command will use a Trapezoid profile to move the Axis in the clockwise direction to the to the specified Target Position, optionally using a different Maximum Velocity. If **Wait 'til @Pos** is selected, the table will remain at this step until the Axis reaches the Target Position. If **Immediately**

AXSCRIPT, continued

go to next step is selected, the table will immediately move to the next step without waiting on the Axis to reach the Target Position.

The **Rotary Move Counter-Clockwise** command will use a Trapezoid profile to move the Axis in the counter-clockwise direction to the to the specified Target Position, optionally using a different Maximum Velocity. If **Wait 'til @Pos** is selected, the table will remain at this step until the Axis reaches the Target Position. If **Immediately go to next step** is selected, the table will immediately move to the next step without waiting on the Axis to reach the Target Position.

The **Rotary Move Shortest** command will use a Trapezoid profile to move the Axis in the direction that will reach the specified Target Position in the shortest distance, optionally using a different Maximum Velocity. If **Wait 'til @Pos** is selected, the table will remain at this step until the Axis reaches the Target Position. If **Immediately go to next step** is selected, the table will immediately move to the next step without waiting on the Axis to reach the Target Position.

Follower commands approximate the Axis Position Following with Offset (AXFOLLOW) instruction:

The **Follow Master** command will establish a Master / Follower connection for the Axis so that the Follower's movements are synchronized to the Master's movement, using the specified Gear Ratio. The Master can be another Axis or a High-Speed Counter / Timer. *Because the Follower Axis will need the ability to overtake the*

AXSCRIPT, continued

Master Axis, make sure the Maximum Velocity and Acceleration parameters of the Master and Follower Axes have been configured with enough capacity to allow this. The Follower Axis can be made more responsive by configuring it with higher Maximum Velocity or a faster Acceleration, or both.

If **Wait 'til Sync'd** is selected, the table will remain at this step until the velocity of the following Axis (including the Gear Ratio) matches the velocity of the master Axis. If **Immediately go to next step** is selected, the table will immediately move to the next step without waiting for the velocities to match.

The **Update Follower Offset** command causes the Follower Axis to attempt to move to the Relative Offset Position using the Relative Offset Velocity. *Because the Follower Axis will need the ability to overtake the Master Axis during this operation, make sure the Maximum Velocity and Acceleration parameters of the Master and Follower Axes have been configured with enough capacity to allow this.* The Follower Axis can be made more responsive by configuring it with higher Maximum Velocity or a faster Acceleration, or both.

Event commands will wait at the current step on an input event from the Axis.

The **Wait For Discrete Input Limit** command will wait at this step in the table until the selected event occurs on the specified discrete input. The discrete input must be a local I/O point if the Axis is using local discrete outputs, or a discrete input point on the same BX-HSIO module that the Axis is configured to use.

AXSCRIPT, continued

The **Wait For Register Match Condition** command will wait at this step in the table until the specified Register "matches" the specified Value, where the "matching condition" is one of the 6 numeric comparison operations. The choices available for the Match Register will depend on the Axis Device selected for the AXSCRIPT instruction.

If the Axis Device is a local Axis the Match Register can be:
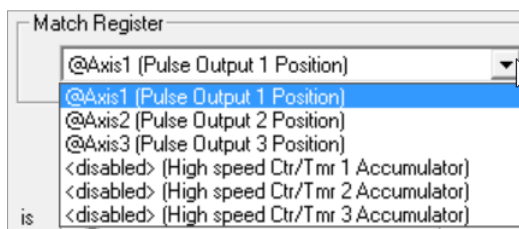
If the Axis Device is a BX-HSIO Axis the Match Register can be:

The **Wait Until At Velocity** command will wait at this step in the table until the Axis has reached its target velocity ($.CurrentVelocity = .TargetVelocity$).

The **Wait Until At Position** command will wait at this step in the table until the Axis has reached its target position ($.CurrentPosition = .TargetPosition$).

The **Wait Until Stopped** command will wait at this step in the table until the Axis has stopped moving. ($.CurrentVelocity = 0$).

Ladder Operation commands perform actions that interact with elements in the ladder logic program.

AXSCRIPT, continued

The **Set Stage** command will enable a Stage in the same Program that contains the AXSCRIPT instruction. This cannot be a Stage in a different Program code-block. If **Wait for Stage to Complete** is selected, the table will remain at this step until the specified Stage is disabled. If **Immediately go to next step** is selected, the table will immediately move to the next step without waiting on the Stage to be disabled.

The **Run Program** command will run the specified Program code-block. If **Wait for Program to Exit** is selected, the table will remain at this step until the specified Program's .Done bit comes ON. If **Immediately go to next step** is selected, the table will immediately move to the next step without waiting on the Program to end.

The **Start Timer** command will start running an internal Timer using the specified Preset value. *This is NOT a Timer in the ladder logic; it is a Timer that is internal to the AXSCRIPT instruction.* If **Wait for Timer to be Done** is selected, the table will remain at this step until the specified Timer has timed out. If **Immediately go to next step** is selected, the table will immediately move to the next step without waiting on the Timer to complete.

The **Set Bit** command will turn ON the specified bit location in the same way a Set Coil (SET) instruction does.

AXSCRIPT, continued

The **Reset Bit** command will turn OFF the specified bit location in the same way a Reset Coil (RST) instruction does.

The **Wait for Bit to be ON** command will wait at this step in the table until the specified Bit location turns ON. This does NOT wait for an OFF to ON transition of the specified Bit; if the Bit is ON when the table gets to this step, the condition will be met, and the table will not wait at this step.

The **Wait for Bit to be OFF** command will wait at this step in the table until the specified Bit location turns OFF. This does NOT wait for an ON to OFF transition of the specified Bit; if the Bit is OFF when the table gets to this step, the condition will be met and the table will not wait at this step.

The **Wait for Timer to be Done** command will wait at this step for the AXSCRIPT's internal Timer to complete. *This is NOT a Timer in the ladder logic; it is a Timer that is internal to the AXSCRIPT instruction.* This is only useful if the table contains a previously executed "Start Timer / immediately go to next step" command.

Miscellaneous commands

The **Set Axis Property** command will change the selected Axis parameter without having to re-run an Axis Configuration (AXCONFIG) instruction. Changes made to the Axis by this command will affect all subsequent instructions that reference this Axis, but they will not affect any Axis instructions that are currently running.

AXSCRIPT, continued

Example Usages:

The following example performs a **Blended Move**, where an Axis arrives at a target position at a velocity that is *NOT 0*. In this case, the use of the AXPOSTRAP instruction will not work for this application because the Axis arrives at the target position at a velocity of 0.

```
SG ──────────── Stage
  Stage        _do_AxisConfig
```

Configure Axis 2 for use by the AXScript instruction.

```
2 ─────────────────────────────────────────┐
                                            ► AXCONFIG                    Axis Configuration
                                              Axis Device                          @Axis2
                                              Axis Structure                       $Axis2
                                              Linear vs. Rotary                    Linear
                                              Initial Output Position                   0
                                              Minimum Velocity              10 pulses/sec
                                              Maximum Velocity         250000 pulses/sec
                                              Acceleration            150000 pulses/sec²
                                              Deceleration            175000 pulses/sec²
                                              Fault Deceleration      200000 pulses/sec²
                                              On Success, JMP to stage        _do_AXScript
                                              On Error, JMP to stage        _err_AxisConfig
```

```
SG ──────────── Stage
  Stage        _do_AXScript
```

The script will do the following:

Ramp Axis 2 up to 75000 pps

When Axis 2 is at position 100000 ramp Axis 2 up to 150000 pps

When Axis 2 is at position 300000 ramp Axis 2 down to 10000 pps

When Axis 2 is at 10000 pps Wait for Axis 2 Position to be 600000 then Run Program named Plunge_Route.

When Plunge_Route completes return Axis 2 to position 0 using Axis 2's defaults.

```
4 ─────────────────────┐
                        ► AXSCRIPT                          Run a Sequence of Axis Commands
                          Axis Device                                            @Axis2
                          Slave Axis Structure                                   $Axis2
```

#	Command	Parameter 1	Parameter 2
1	Ramp to Velocity	Target Vel (pps): 75000	Scurve Jerk (pps³) 7500
2	Wait for Register Match Condition	Reg: Pulse Output 2 Position	>= 100000
3	Ramp to Velocity	Target Vel (pps): 150000	Scurve Jerk (pps³) 17500
4	Wait for Register Match Condition	Reg: Pulse Output 2 Position	>= 300000
5	Ramp to Velocity, Wait until @Vel	Target Vel (pps): 10000	Scurve Jerk (pps³) 75000
6	Wait for Register Match Condition	Reg: Pulse Output 2 Position	>= 600000
7	Run Program, Wait for Program to EXIT	Program: Plunge_Route	
8	Move to Absolute Pos, Wait 'til @Pos	Target Pos: 0	
9	Go to Idle Mode (immediate stop)		

```
On Success, JMP to stage                                                _ok_AxisScript
On Error, JMP to stage                                                  _err_AxisScript
```

```
SG ──────────── Stage
  Stage        _ok_AxisScript
```

If the AXScript instruction completes successfully you will end up here.

```
6 ────────────────────────────────────────────────────( NOP )
```

```
SG ──────────── Stage
  Stage        _err_AxisConfig
```

If the AXConfig instruction does not complete successfully you will end up here.

```
8 ────────────────────────────────────────────────────( NOP )
```

```
SG ──────────── Stage
  Stage        _err_AxisScript
```

If the AXScript instruction does not complete successfully you will end up here.

```
10 ────────────────────────────────────────────────────( NOP )
```

AXSETPROP

The **Set Axis Properties** (AXSETPROP) instruction is used to make runtime changes to the configured parameters of an Axis.

This instruction requires the following:

- Axis configured to be used as the slave Axis.
- AXCONFIG instruction configured for each axis associated with the instruction.

The (a) *Axis Device* specifies which axis to use as the slave axis. This can be any axis available to the MPU or available to a BX-HSIO expansion module.

The Axis is configured in the MPUs *Setup BRX High-speed I/O* dialog or the BX-HSIO *Setup BX-HSIO Module* dialog. Please see detailed information on these under the "Available High-Speed Input and Output Features" earlier in this chapter. These dialog boxes can be opened directly from the AXSETPROP instruction by clicking on the (b) **Configure Axis...** button. This allows the user to:

- configure an axis for use with the instruction.
- verify or edit the physical I/O selected for an already configured axis.

NOTE: *Axis0 is a 'virtual axis' and will not generate pulses to physical outputs of the PLC. Axis0 can be used for generating pulse output profile register values to be used for Table Driven Outputs (TDOPRESET or TDOPLS) or as the master for other axes in following-type applications (AXCAM, AXFOLLOW or AXGEAR). Axis0 is also good to use for testing.*

Position: The current count will be set to this value when this instruction is enabled. This can be any constant value or any numeric location.

Minimum Velocity (pulses/sec): The slowest frequency of output pulses that will be generated when the output is enabled. This can be any positive constant from 10 to 250000 or any numeric location with a value in that range.

Maximum Velocity (pulses per second): The fastest frequency of output pulses that will be generated when the output is enabled. This can be any positive constant from 10 to 250000 or any numeric location with a value in that range.

Acceleration Rate (pulses/sec^2): The rate at which the pulses will be generated when the axis is ramping up from a slower pulse rate to a higher pulse rate. This can be any positive constant or any numeric location with a value in that range.

Deceleration Rate (pulses/sec^2): The rate at which the pulses will be generated when the axis is ramping down from a faster pulse rate to a slower pulse rate. This can be any positive constant or any numeric location with a value in that range.

AXSETPROP, continued

Fault Deceleration Rate (pulses/sec^2): Any time a Fault Limit is reached or the Axis .MasterEnable is turned OFF, the axis will decelerate to a velocity of 0 pulses/sec at this specified rate. A value of 0 will cause the Axis to immediately stop moving. This can be any positive constant or any numeric location with a value in that range.

Pulse Output/Encoder Scale: If the motor and the encoder have different pulse-per-revolution values, enter the scale value required to bring them into alignment.

Encoder Deadband (counts): Having some deadband value around the encoder current position can prevent the pulse output from generating alternating small pulses trying to get the input value to an exact number. This value is applied both above and below the encoder value. For example: A value of 2 will be a deadband of 2 above and 2 below for a span of 4 counts.

On Success:

- **Set Bit**: The bit will become FALSE when the instruction is enabled and will become TRUE when instruction parameters are properly entered and the instruction completes successfully. The specified bit is enabled with a SET operation, not an OUT operation. The On Success bit will remain ON even if the instruction input logic goes OFF.

- **JMP to Stage**: When instruction parameters are properly entered and the instruction completes successfully, the PLC will jump to that stage.

On Error:

- **Set Bit**: The bit will become FALSE when the instruction is enabled and will become TRUE when proper instruction parameters are NOT properly entered or the instruction IS NOT completed successfully. The specified bit is enabled with a SET operation, not an OUT operation. The On Error bit will remain ON even if the instruction input logic goes OFF.

- **JMP to Stage**: When proper instruction parameters are NOT properly entered or the instruction IS NOT completed successfully, the PLC will jump to that stage.

AXSETPROP, continued

Example Usage

Rung 1

Init_Axis1
C6

AXCONFIG	Axis Configuration
Axis Device	@Axis1
Axis Structure	$Axis1
Linear vs. Rotary	Linear
Initial Output Position	0
Minimum Velocity	100 pulses/sec
Maximum Velocity	1000 pulses/sec
Acceleration	1000 pulses/sec²
Deceleration	1000 pulses/sec²
Fault Deceleration	0 pulses/sec²
On Success, Set bit	Axconfig__Good
On Error, Set bit	Axconfig_Bad

Rung 2

Run_to_Press1   Axconfig__Good $Axis1.MasterEnable   E/R
C11   C89

AXPOSTRAP	Axis Move to Position Using Trapezoid
Axis Device	@Axis1
Axis Structure	$Axis1
Move Type	Single move
Target Type	Absolute
Position Value	1256
Linear vs. Rotary	Linear
On Success, Set bit	Run_Press1_Good
On Error, Set bit	Run_Press1_Bad

Rung 3

Jog_Mode $Axis1.MasterEnable
C19

AXSETPROP	Set Axis Properties
Axis Device	@Axis1
Axis Structure	$Axis1
Maximum Velocity	500 pulses/sec
Acceleration	250 pulses/sec²
Deceleration	250 pulses/sec²
On Success, Set bit	Axis1_ReadyJog
On Error, Set bit	Axis1_NotJog

Rung 4

Jog_Mode $Axis1.MasterEnable   Axis1_ReadyJog   E/R
C19   C15

Jog_Fwd   Jog_Rev   FW
C8   C9

Jog_Rev   Jog_Fwd   RV
C9   C8

AXJOG	Axis Jog Mode
Axis Device	@Axis1
Axis Structure	$Axis1
Target Velocity	500 pulses/sec
On Success, Set bit	Jog_Good
On Error, Set bit	Jog_Bad

AXVEL

The **Axis Set Velocity Mode** (AXVEL) instruction is used to put an axis into an operation mode where its movement is controlled by velocity rather than position.

The ladder logic input to this instruction is an Enable/Reset, which means that when the Input logic turns ON the axis will ramp to the axis structure's *.TargetVelocity* value, and when the Input logic turns OFF the axis will ramp down to a velocity of 0 and the instruction will end.

While the instruction is enabled, changing the axis associated structure member *.TargetVelocity* will dynamically change the output pulse frequency and will follow the specified Acceleration/Deceleration Mode of the instruction to achieve the speed change.

This instruction requires the following:

- Axis configured to be used as the slave Axis.
- AXCONFIG instruction configured for each axis associated with the instruction.

The (a) *Axis Device* specifies which axis to use as the slave axis. This can be any axis available to the MPU or available to a BX-HSIO expansion module.

The Axis is configured in the MPUs *Setup BRX High-speed I/O* dialog or the BX-HSIO *Setup BX-HSIO Module* dialog. Please see detailed information on these under the "Available High-Speed Input and Output Features" earlier in this chapter. These dialog boxes can be opened directly from the AXVEL instruction by clicking on the (b) **Configure Axis...** button. This allows the user to:

- configure an axis for use with the instruction.
- verify or edit the physical I/O selected for an already configured axis.
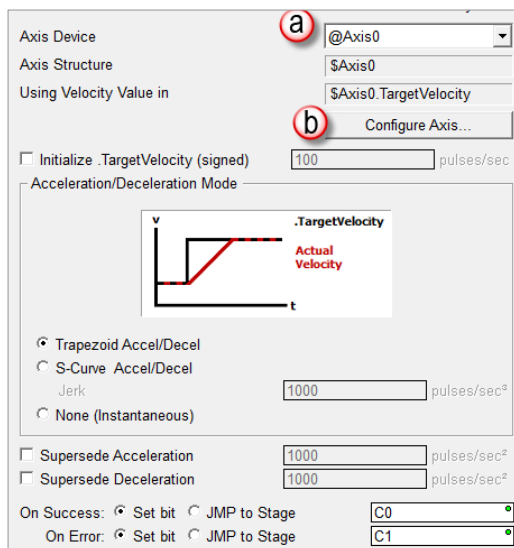
NOTE: *Axis0 is a 'virtual axis' and will not generate pulses to physical outputs of the PLC. Axis0 can be used for generating pulse output profile register values to be used for Table Driven Outputs (TDOPRESET or TDOPLS) or as the master for other axes in following-type applications (AXCAM, AXFOLLOW or AXGEAR). Axis0 is also good to use for testing.*

AXVEL, continued

Using Velocity Value in: This field indicates the proper axis structure member to use for the desired Velocity.

Initialize .TargetVelocity (signed): Enable this option to set the velocity of the axis when this instruction is enabled. A non-zero value will cause the axis to begin moving at the specified velocity as soon as this instruction is enabled. This value can be any constant between -250000 to -10, 0, and 10 to 250000 or any numeric location containing a value in that range. The sign of the value will indicate the direction of travel: positive numbers will cause the axis to move clockwise, negative numbers will cause the axis to move counter-clockwise. Any value that is below the axis **Configured Minimum Velocity** will result in the **Axis Minimum Velocity** being used.

Acceleration/Deceleration Mode: Specifies what level of Acceleration / Deceleration to use when the axis is changing to a new Target Velocity. The graphic will change with the selection to display what the velocity curve will look like for that selection.

- **Trapezoid Accel / Decel**: The axis will ramp from the Current Velocity to the new Target Velocity value using the axis currently configured Acceleration and Deceleration values which results in a trapezoid velocity path.

- **S-Curve Accel / Decel**: The axis will ramp from the Current Velocity to the new Target Velocity value using the axis currently configured Acceleration and Deceleration values in addition to the following Jerk parameter which results in an s-curve velocity path.

- **Apply Jerk to Accel / Decel** (pulses / sec^3): This selection is only available for S-Curve Accel / Decel. Whereas the Acceleration and Deceleration values specify how quickly the Axis is allowed to reach maximum velocity, the Jerk parameter specifies how quickly the Axis is allowed to achieve maximum Acceleration and Deceleration. This can be any positive constant greater than 0 or any numeric location with a value in that range.

- **None (Instantaneous)**: The axis will immediately begin moving at the specified Target Velocity value; there is no ramp up or ramp down to the new velocity.

Supersede Acceleration Rate (pulses/sec^2): When the axis is ramping up from a slower velocity to a higher velocity use the specified rate instead of the axis configured acceleration rate. This can be any positive constant or any numeric location with a value in that range.

Supersede Deceleration Rate (pulses/sec^2): When the axis is ramping down from a faster velocity to a slower velocity use the specified rate instead of the axis configured deceleration rate. This can be any positive constant or any numeric location with a value in that range.

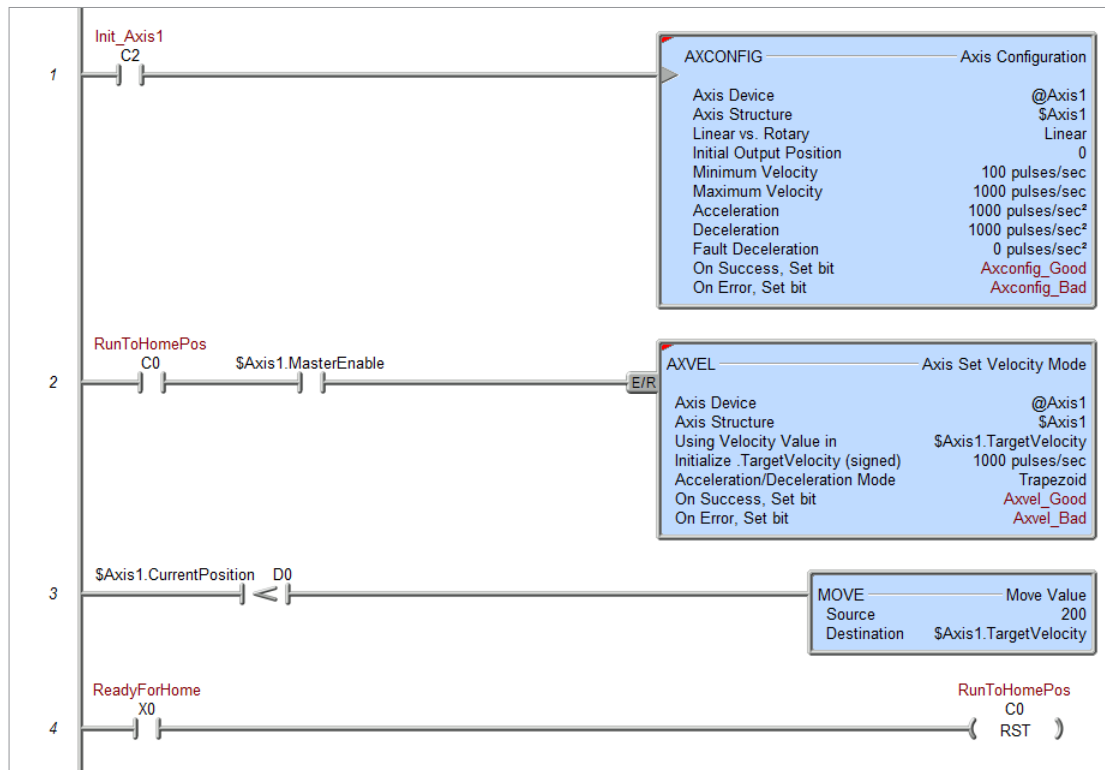## AXVEL, continued

On Success:

- **Set Bit**: The bit will become FALSE when the instruction is enabled and will remain FALSE until the enable leg goes back OFF. Once the enable leg turns OFF, if the instruction device/parameters were valid, this bit will turn ON once the .CurrentVelocity reaches 0.

- **JMP to Stage**: Similarly, the JMP will not occur until after the instruction is enabled, then disabled, and all the instruction device/parameters were valid and the .CurrentVelocity reaches 0.

On Error:

- **Set Bit**: The bit will become FALSE when the instruction is enabled and will become TRUE when proper instruction parameters are NOT properly entered or the instruction IS NOT completed successfully. The specified bit is enabled with a SET operation, not an OUT operation. The On Error bit will remain ON even if the instruction input logic goes OFF.

- **JMP to Stage**: When proper instruction parameters are NOT properly entered or the instruction IS NOT completed successfully, the PLC will jump to that stage.

> **NOTE:** Because this instruction puts the Axis into an operational mode (as opposed to performing a single operation), On Success is defined as getting the Axis into Jog mode with no errors. This means that the On Success indication will turn ON after the Enable/Reset input logic transitions from ON to OFF and the Axis' Current Velocity is at 0. When these conditions are met the Axis' Mode is "Idle". You should wait until the On Success indication turns ON before attempting to execute any other Axis instruction.

Example Usage

TDODECFG

Once a TDOPLS-Load Programmable Limit Switch Table for Table Driven Output or TDOPRESET-Load Preset Table for Table Driven Output has been enabled it will continue to control that output even if the input logic is no longer ON. Use the **Deconfigure Table Driven Output** (TDODECFG) instruction to stop the Preset Table or PLS table from controlling the Table Driven Output.
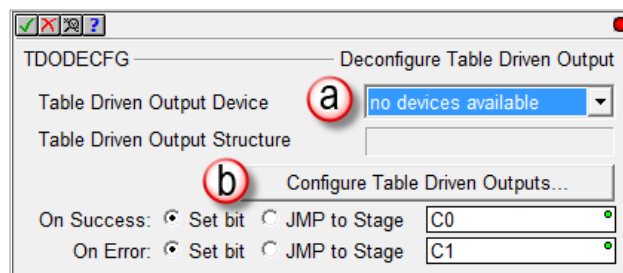
This instruction is useful for situations when the application may require changing control of the Table Driven Output from one instruction, such as a TDOPRESET, to another.

This instruction requires that a Table Driven Output be configured.

The (a) *Table Driven Output Device* specifies which discrete output will be used by the instruction. This can be any discrete output available to the MPU or available to a BX-HSIO expansion module.

The Table Driven Output is configured in the MPUs *Setup BRX High-speed I/O* dialog or the BX-HSIO *Setup BX-HSIO Module* dialog. Please see detailed information on these under the "Available High-Speed Input and Output Features" earlier in this chapter. These dialog boxes can be opened directly from the TDODECFG instruction by clicking on the (b) **Configure Axis...** button. This allows the user to:

- configure a Table Driven Output for use with the instruction.
- verify or edit the physical I/O selected for an already configured Table Driven Output function.


On Success:

- **Set Bit**: The bit will become FALSE when the instruction is enabled and will become TRUE when instruction parameters are properly entered and the instruction completes successfully. The specified bit is enabled with a SET operation, not an OUT operation. The On Success bit will remain ON even if the instruction input logic goes OFF.

- **JMP to Stage**: When instruction parameters are properly entered and the instruction completes successfully, the PLC will jump to that stage.

On Error:

- **Set Bit**: The bit will become FALSE when the instruction is enabled and will become TRUE when proper instruction parameters are NOT properly entered or the instruction IS NOT completed successfully. The specified bit is enabled with a SET operation, not an OUT operation. The On Error bit will remain ON even if the instruction input logic goes OFF.

- **JMP to Stage**: When proper instruction parameters are NOT properly entered or the instruction IS NOT completed successfully, the PLC will jump to that stage.

TDODECFG, continued

Example Usage
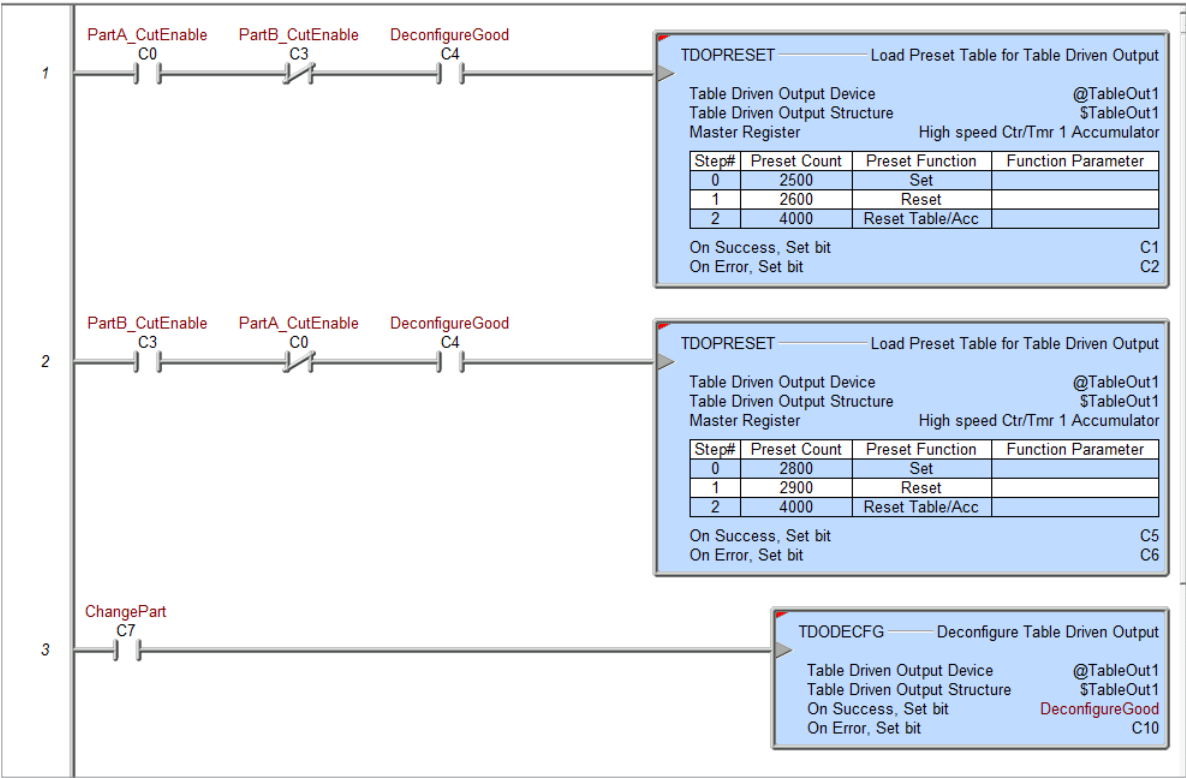
TDOPLS

A **Load Programmable Limit Switch Table for Table Driven Output** (TDOPLS) contains a table with a series of start and stop positions similar to the cams on a shaft. These cam positions are compared to the current count value of the specified Master Register, which can be a High-speed Counter, Timer or AXIS Current position value. When the count value falls between any of the positions in the table, the discrete output that is specified is turned ON or OFF according to the table configuration. Each table can have up to 64 cam positions. Each PLS table compares the count value of one Master Register counter and can drive one High-speed I/O discrete output.

Once a PLS table has been enabled for a Table Driven Output it will continue to control that output even if the input logic is no longer ON. Use the De-configure Table Driven Output (TDODECFG) instruction to stop the PLS table from controlling the discrete output associated with the selected Table Driven Output.

This instruction requires the following:

- Table Driven Output configured to be used by the instruction.
- One of the following to be used as the Master Register:
  - Position value from a configured master axis.
  - Accumulated value from a configured high speed counter/timer function.

The (a) *Table Driven Output Device* specifies which discrete output will be used by the instruction and the (b) *Master Register* selects the axis position or high-speed counter/timer accumulated value that provides the position source value. These can be any discrete output or high-speed counter/timer available to the MPU or available to a BX-HSIO expansion module.

The Table Driven Output and the high-speed counter/timer functions are configured in the MPUs *Setup BRX High-speed I/O* dialog or the BX-HSIO *Setup BX-HSIO Module* dialog. Please see detailed information on these under the "Available High-Speed Input and Output Features" earlier in this chapter. These dialog boxes can be opened directly from the TDOPLS instruction by clicking on the (c) **Configure Axis...** Button and (d) **Configure Master Register Device...** button. These allow the user to:

- configure a table function or a counter/timer function for use with the instruction.
- verify or edit the physical I/O selected for an already configured axis or counter/timer function.

TDOPLS, continued

Initialize .InputValOffset: *.InputValOffset* is used to adjust the current count value from the Master Register. The "calculated master register" value will be used for subsequent comparisons.

For example:

- A table is processed, expecting no deviation (delta value) in the position. If there is a deviation (an amount of error), this delta value can be written (by means of the ladder) to *.InputValOffset* in order for the system to behave as if there is no deviation.

- Write a fixed value into the *.InputValOffset* field in the instruction to take care of equipment changes, i.e. change an encoder with one with different pulses per revolution.

When the Master Register is configured as Linear or Rotary device, a positive value in the *.InputValOffset* field (a) means that the entries will act at a lower value than shown. A negative value means that the entries will act at a higher value than shown. For example: An offset of 500 is configured in the *.InputValOffset* field (a). The first entry (b, c) is configured to turn ON the output at 1000 and (d, e) turn OFF at 2000. When the table runs, the *.OutputState* will actually turn ON at a value of 500 and turn OFF at 1500. If the *.InputValOffset* was configured for -500, the first *.OutputState* would have turned ON at a value of 1500 and OFF at 2500.
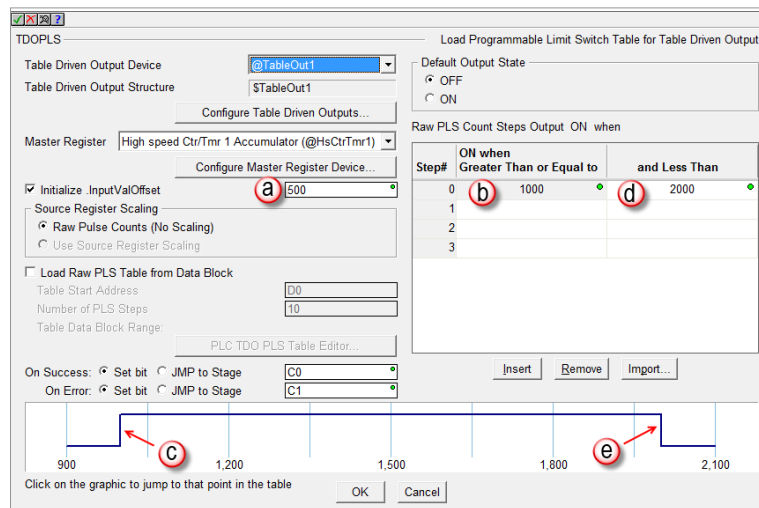
NOTE: *When the Master Register is configured as a Rotary device, the user must make sure the calculated range stays within the device's configured Rotary Range.*

TDOPLS will stop triggering the table *.OutputState* in two situations:

- A positive *.InputValOffset* value is greater than or equal to the configured "and Less Than" value.

- A negative *.InputValOffset* value has a magnitude greater than or equal to the configured "ON when Greater Than or Equal to" configured value.

The effect of *.InputValOffset* is illustrated by the following examples:

TDOPLS *.InputValOffset* Behavior									
Rotary Range: 0–200									
ON when Greater Than or Equal to: 100 and Less Than: 110									
.InputValOffset	-100	-99	-90	-50	0	50	90	99	105	110
Table .OutputState is ON	Stopped	199	190	150	100	50	10	1	0	Stopped
Table .OutputState is OFF		0	0	160	110	60	20	11	5	

TDOPLS, continued

> **NOTE:** *if the Master Register is an Axis' position, and the application is using either a servo controller or stepper controller with micro-stepping or smoothing, be aware that both controller types have a conversion time that will introduce some delay between the Axis' output position and the actual shaft position of the motor. The **.InputValOffset** can be used to compensate for the conversion time by adding the number of pulses that would occur during the conversion time to the current count of the Master Register. Multiplying the servo or stepper controller's conversion time by the current velocity will yield the number of pulses that would occur during that amount of time. Using this pulse count value in the **.InputValOffset** field will delay the comparison in the Steps by the same amount as the conversion time of the motor controller. Doing this will more closely sync the Table Driven Output's state to the actual shaft position of the motor.*

Source Register Scaling: The TDOPLS instruction can use the raw count value from the specified Master Register or, if scaling was enabled in the High-speed I/O setup, the scaled value to do its comparisons.

- **Raw Pulse Counts** (No Scaling): Select this option to enter the Preset Count values in the Table as raw count values. The image below shows (a) Raw Pulse Counts (No Scaling selected, indicating the Master Register scaling was not enabled. The (b) table entry shows that the values entered are based on raw counts.

- **Use Source Register Scaling**: If the Master Register has been configured to scale the current count value you can use the scaled values in the table by selecting this option and entering values that are scaled the same as the Master Register. The image below shows (a) Use Source Register Scaling selected, indicating the Master Register scaling was enabled. The (b) table entry shows that the entries are based on the scaled values.

TDOPLS, continued

Load Raw PLS Table from Data Block (a): When enabled, the values used by the PLS Table come from the values stored in the specified memory range. This allows for more dynamic control of the PLS function.

- **Table Start Address** (b): The beginning address in PLC memory where the PLS table data is stored.

- **Number of PLS Steps** (c): The number of steps in the PLS table that are stored in PLC memory. A PLS Table can have up to 64 steps.

- **Table Data Block Range** (d): Using the two values above, this shows the range of PLC memory that will be used as the PLS data table. With a starting address of D0 and a total of 2 steps, the memory range is D0 to D3.

NOTE: *The PLC TDO PLS Table Editor button is available only when the Do-more! Designer software is connected and Online with the BRX CPU.*

- **PLC TDO PLS Table Editor** button (e): Click this to open the *PLS Table Editor*. The editor allows you to edit the data values in the specified Data Block. The data configuration options on this dialog (Default Output State, value for "Greater Than or Equal", value for "Less Than") are the same as the main dialog. Detailed information of the PLC TDO PLS Table Editor dialog screen is discussed later in this section.

TDOPLS, continued

Default Output State (a): The Default Output state determines the starting state of the Output when the PLS instruction is enabled and the state that the Output is in when the count is NOT within the Entry comparisons. If the Default Output State is OFF, the Output will be ON when the count is within the Entry comparison values. If the Default Output State is ON, the Output will be OFF when the count is within the Entry comparison values.

Raw (or Scaled) PLS Count Steps Output ON when (b): A table of entries that determines the Output state when the TDOPLS is running.

- The entries in the table (b) can be constants or variables. If variables are used, the instruction must be disabled and re-enabled after the variable value has been changed. The values in the entry table cannot be overlapping.

- The values must be increasing with the Entry numbers. In other words, the values in (c) **And Less Than** must be larger than the value in (d) **Greater Than or Equal to**.

    Output On/Off when Greater Than or Equal to: The value that represents the lower edge of the cam position. This can be any value between -2,147,483,648 and 2,147,483,647.

    And Less Than: The value that represents the upper edge of the cam position. This can be any value between -2,147,483,648 and 2,147,483,647.

If invalid entry values are used when specifying constant values, the Do-more! Designer programming software will indicate an error (below) and not allow the instruction configuration to be completed.
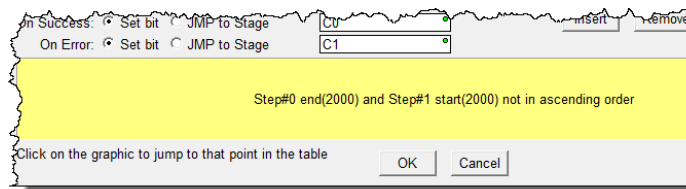
If variables are used and invalid entry values are specified, the **On Error** state will become true (*Set Bit* or *JMP to Stage*).

If the Scaling was configured for the Master Register counter and the Use Source Register Scaling option was specified, the entry values will be the scaled values and no conversion will be necessary.

Insert (e): Add an empty Step above the currently highlighted Step in the table.

Remove (f): Delete the currently highlighted Step from the table.

Import (g): Import the contents of the PLS Table from a CSV file. The format of the import file is two numbers per line, separated by a comma or whitespace, all numbers must be in ascending order, and a maximum of 64 lines. Please review CSV file formatting information later in this TDOPLS instruction section, under the section explaining the PLC TDO PLS Table Editor dialog.

On Success (h):

- **Set Bit**: The bit will become FALSE when the instruction is enabled and will become TRUE when instruction parameters are properly entered and the instruction completes successfully. The specified bit is enabled with a SET operation, not an OUT operation. The On Success bit will remain ON even if the instruction input logic goes OFF.

- **JMP to Stage**: If the parameters configured in the instruction and proper devices were specified, the PLC will jump to that stage.

TDOPLS, continued

On Error (i):

- **Set Bit**: The bit will become FALSE when the instruction is enabled and will become TRUE if there was a problem with the parameters configured in the instruction or with the devices specified.

- **JMP to Stage**: If there was a problem with the parameters configured in the instruction or with the devices specified, the PLC will jump to that stage.

PLC TDO PLS Table Editor: Allows you to edit the data values in the specified Data Block. The data configuration options on this dialog (Default Output State, value for "Greater Than or Equal", value for "Less Than") are the same as the main dialog.

> **NOTE:** Initial values for Default Output State, PLS Table Start Address, and Number of PLS Entries come from the main instruction editor. If these values are changed while editing the table data with this dialog those values will be updated on the main instruction editor when this dialog is closed.

> **NOTE:** If the PLC TDO PLS Table Editor dialog is opened when the Number of PLS Entries is a variable (memory address), the value in that location is read and the data in that number of rows will be read from the PLC to prefill the table. The Also Write Table Length to PLC selection will be enabled and that variable location will be prefilled here so that when the table data is written back to the PLC this location will be updated as well.

> **NOTE:** A graph of the current PLS Table configuration is displayed at the bottom of the editor which shows each of the entries. Clicking on the graph will place the cursor on the table entry that contains that location. If incorrect data values are entered, i.e. overlapping data, a message will be displayed instead of the graph.

- **PLC TDO PLS Table Editor Dialog**:

    Source Register Scaling (a): Same as the main dialog. The TDOPLS instruction can use the raw count value from the specified Master Register or, if scaling was enabled in the High-speed I/O setup, it can use the scaled value to do its comparisons.
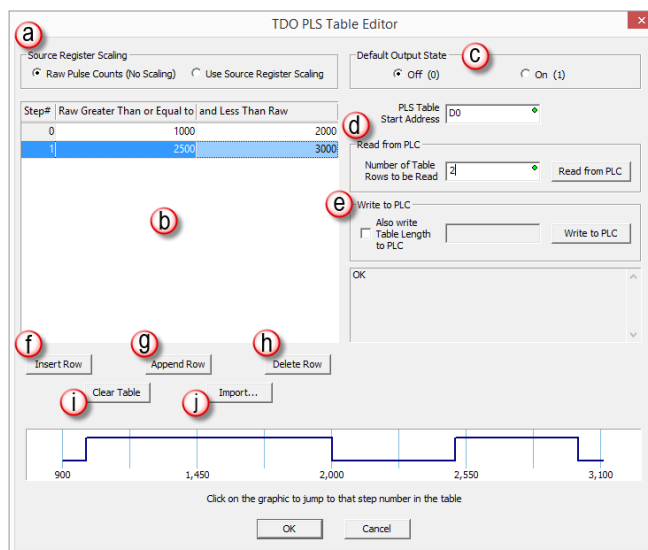
PLS Data Entry (b): Same as main dialog. Entries must be unique within the Table, the lower and upper positions of a step cannot overlap the other cam positions. The data has to be incremental.

- **Output On/Off when Greater Than or Equal to**: The value that represents the lower edge of the cam position. This can be any value between -2,147,483,648 and 2,147,483,647.

- **And Less Than**: The value that represents the upper edge of the cam position. This can be any value between -2,147,483,648 and 2,147,483,647.

TDOPLS, continued

Default Output State (c): Same as main dialog. Specifies the state of the first entry in the table.

– **OFF** (default): The output is OFF to start with. The first entry in the table will be a point which the output turns ON and subsequent entries will flip between OFF / ON / OFF / etc.

– **ON**: The output is ON to start with. The first entry in the table will be a point which the output turns OFF and subsequent entries will flip between ON / OFF / ON / etc.

Read from PLC button (d): Overwrite the contents of the table with values from the PLC memory location specified in the Load Raw PLS Table from Data Block section.

– Confirmation is requested before reading the PLC data. In this example the starting data block is D0, D10 holds a value of 5, which indicates 10 DWords will be read:
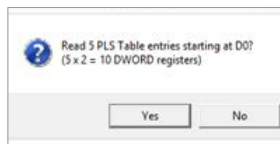
Write to PLC (e): Overwrite the contents of PLC memory at the location specified in the Load Raw PLS Table from Data Block section with the current contents of the table.

– Confirmation is requested before writing data to the PLC. In this example, there are 5 rows of data. A 5 is written o D10 and the 10 DWords of data is written to D0 to D9.
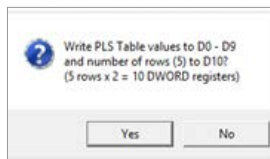
Insert Row button (f): (**Insert** button in main dialog) add an empty Step above the currently highlighted Step in the table.

Append Row (g): Add a new row after the current last step in the table.

Delete Row button (h): (**Remove** button in main dialog) delete the currently highlighted Step from the table.

Clear Table button (i): Will display a confirmation dialog explaining it will delete all rows and the table will end up with a single row with values of 0 for the **Greater Than or Equal to** and 0 for **and Less Than**.

Import button (j): (Import button in main dialog) Import the contents of the PLS Table from a CSV file. The format of the import file is two numbers per line, separated by a comma or whitespace, all numbers must be in ascending order, and a maximum of 64 lines.

NOTE: *Any available text editor or an Excel worksheet can be used to create a CSV file. Just save the file as a .CSV file type.*

	A	B	C	D
1	1000	1200	// Step 1	
2	2000	2300	//Step 2	
3	3000	3400	//Step 3	

TDOPLS, continued

Example of how data should be formatted in the CSV file:

a. Two integer values per row separated by comma (surrounded by any amount of white space) or just white space.

b. You can also have empty rows.

c. You can also use a line comment starting with two forward slashes: // This is a comment

Example on how to format the data per line in the CSV file:

a. Simple comma separated values 1 and 2. (See line one below.)

b. Some white space before and/or after the comma is OK (see line two below).

c. Whitespace before, between, and after is OK also (i.e. comma is optional, see line three below).

d. Comment line (no data allowed). (see line four below.)

```
Example csv data file.txt - Notepad

File  Edit  Format  View  Help

1,2
1,  2
  1  2
// 1  2 NONE OF THIS LINE IS DATA, it's a comment line
```
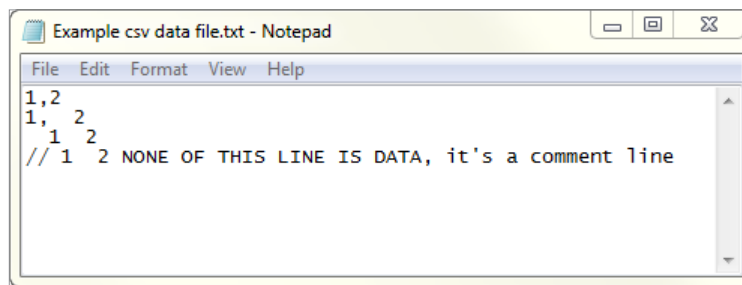
TDOPRESET

A **Load Preset Table for Table Driven Output** (TDOPRESET) instruction contains a series of steps that are processed in the order they appear in the table. These steps compare the current count value of the specified Master Register, which can be a High-speed Counter, Timer or AXIS Current position value, to the Preset Count in the Step, and when the count values match, the step action is performed on the selected discrete output and the next step in the table becomes the active step. Each Preset Table can have up to 64 steps. Each Preset Table compares the count value of one Master Register counter and can drive one High-speed I/O discrete output.

Once a Preset table has been enabled for a Table Driven Output, it will continue to control that output even if the input logic is no longer ON. Use the Deconfigure Table Driven Output (TDODECFG) instruction to stop the Preset table from controlling the Table Driven Output.
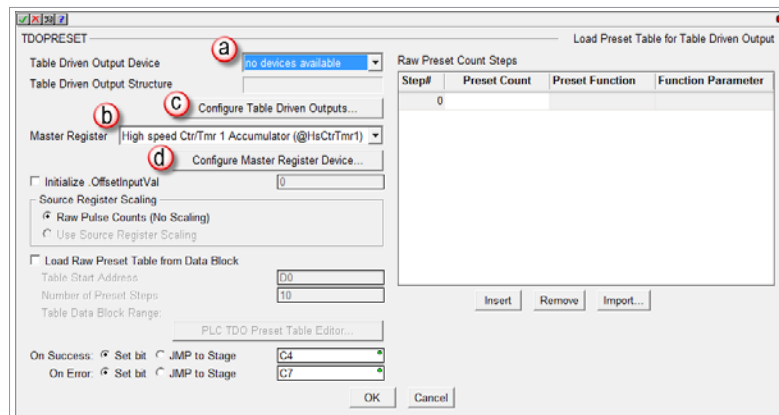
This instruction requires the following:

- Table Driven Output configured to be used by the instruction.
- One of the following to be used as the Master Register:
  - Position value from a configured master axis.
  - Accumulated value from a configured high speed counter/timer function.

The (a) *Table Driven Output Device* specifies which discrete output will be used by the instruction and the (b) *Master Register* selects the axis position or high-speed counter/timer accumulated value that provides the position source value. These can be any discrete output or high-speed counter/timer available to the MPU or available to a BX-HSIO expansion module.

The Table Driven Output and the high-speed counter/timer functions are configured in the MPUs *Setup BRX High-speed I/O* dialog or the BX-HSIO *Setup BX-HSIO Module* dialog. Please see detailed information on these under the "Available High-Speed Input and Output Features" earlier in this chapter. These dialog boxes can be opened directly from the TDOPRESET instruction by clicking on the (c) **Configure Axis...** Button and (d) **Configure Master Register Device...** button. These allow the user to:

- configure a table function or a counter/timer function for use with the instruction.
- verify or edit the physical I/O selected for an already configured axis or counter/timer function.

TDOPRESET, continued

Initialize .InputValOffset: *.InputValOffset* is used to adjust the current count value from the Master Register. The "calculated master register" value will be used for subsequent comparisons.

For example:

- A table is processed, expecting no deviation (delta value) in the position. If there is a deviation (an amount of error), this delta value can be written (by means of the ladder) to *.InputValOffset* in order for the system to behave as if there is no deviation.

- Write a fixed value into the *.InputValOffset* field in the instruction to take care of equipment changes, i.e. change an encoder with one with different pulses per revolution.

When the Master Register is configured as Linear or Rotary device, a positive value in this field means that the entries will act at a lower value than shown. A negative value means that the entries will act at a higher value than shown. For example: An offset of (a) 500 is configured in the *.InputValOffset* field. The first entry is configured to (b) SET the output at (c) 1000. When the table runs, the *.OutputState* will actually SET at a value of 500. If the *.InputValOffset* was configured for -500, the first *.OutputState* would have SET at a value of 1500.
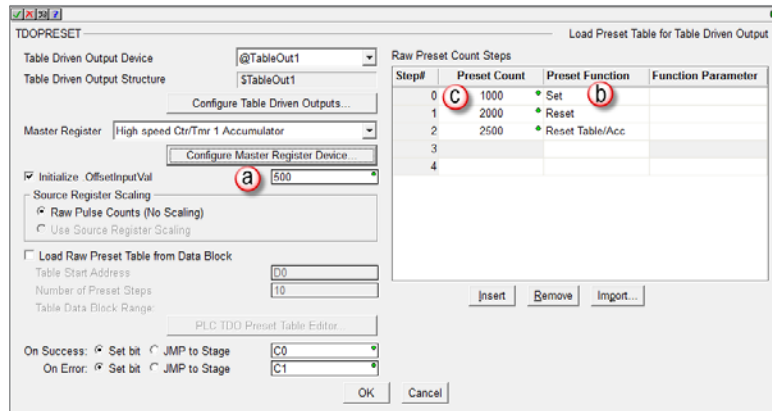
NOTE: *When the Master Register is configured as a Rotary device, the user must make sure the calculated range (Master Register value - .InputValOffset + 1) stays within the device's configured Rotary Range.*

TDOPRESET will stop triggering the table *.OutputState* when the Master Register is configured as a Rotary device, and either:

- A positive *.InputValOffset* value is at least 1 count greater than the configured "SET" value.

- A negative *.InputValOffset* value has a magnitude greater than or equal to the configured "SET" value.

The effect of *.InputValOffset* is illustrated by the following examples:

TDOPRESET *.InputValOffset* Behavior										
Rotary Range: 0–200										
SET at: 100 RESET at: 110										
.InputValOffset	-100	-99	-90	-50	0	50	90	99	100	101
Table .OutputState is SET	Stopped	199	190	150	100	50	10	1	0	Stopped
Table .OutputState is RESET		0	0	160	110	60	20	11	10	

TDOPRESET, continued

> **NOTE:** *if the Master Register is an Axis' position, and the application is using either a servo controller or stepper controller with micro-stepping or smoothing, be aware that both controller types have a conversion time that will introduce some delay between the Axis' output position and the actual shaft position of the motor. The **.InputValOffset** can be used to compensate for the conversion time by adding the number of pulses that would occur during the conversion time to the current count of the Master Register. Multiplying the servo or stepper controller's conversion time by the current velocity will yield the number of pulses that would occur during that amount of time. Using this pulse count value in the **.InputValOffset** field will delay the comparison in the Steps by the same amount as the conversion time of the motor controller. Doing this will more closely sync the Table Driven Output's state to the actual shaft position of the motor.*

Source Register Scaling: The TDOPRESET instruction can use the raw count value from the specified Master Register or, if scaling was enabled in the High-speed I/O setup, the scaled value to do its comparisons.

- **Raw Pulse Counts (No Scaling)** (a): Select this option to enter the Preset Count values in the Table as raw count values. The image above shows (a) **Raw Pulse Counts (No Scaling)** selected, indicating the Master Register scaling was not enabled. The (b) **Raw Preset Count Steps** table entry shows that the values entered are based on raw counts.

- **Use Source Register Scaling**: If the Master Register has been configured to scale the current count value you can use the scaled values in the table by selecting this option and entering values that are scaled the same as the Master Register. The image below shows (a) Use Source Register Scaling selected, indicating the Master Register scaling was enabled. The (b) Scaled Preset Steps table entry shows that the entries are based on the scaled values.

TDOPRESET, continued

Load Raw Preset Table from Data Block (a): When enabled, the values used by the Preset Table come from the values stored in the specified memory range. This allows for a more dynamic control of the Preset function.

- **Table Start Address** (b): The beginning address in PLC memory where the Preset table data is stored. This must be a block of Signed DWords.

- **Number of Preset Steps** (c): The number of steps in the Preset table that are stored in PLC memory. A Preset Table can have up to 64 steps.

- **Table Data Block Range** (d): Using the two values above, this shows the range of PLC memory that will be used as the Preset data table. With a starting address of D0 and a total of 10 steps, the memory range is D0 to D19. Data formatting will be discussed in the PLC TDO Preset Table Editor section found later in this section.

NOTE: *The PLC TDO Preset Table Editor button is available only when the Do-more! Designer software is connected and Online with the BRX CPU.*

- **PLC TDO Preset Table Editor** button (e): Click to open the PLC TDO Preset Table Editor which allows you to edit the data values in the specified Data Block. The data configuration options on this dialog (Preset Count, Preset Function, Function Parameter) are the same as the main dialog. Detailed information of the PLC TDO Preset Table Editor dialog screen is discussed later in this section.

TDOPRESET, continued

Raw (or Scaled) Preset Count Entries Output xx when: This is the table of entries for **Raw (or Scaled) Preset Count Steps** (a): This is the list of steps that determines the Table Driven Output state when the TDOPRESET is running.

- **Preset count**: The pulse count values in the table can be constants or variables. If variables are used, the instruction must be disabled and re-enabled after the variable value has been changed. This can be any value in the range of -2,147,483,648 and 2,147,483,647.

- **Preset Function** (b): Select the action to perform on the Table Driven Output when the Step is triggered.

  1. **Set**: Turns ON the Table Driven Output.

  2. **Reset**: Turns OFF the Table Driven Output.

  3. **Reset Table/Acc**: Performs a reset of the Master Register which sets its current count value to the Initial Reset Value specified in the Timer/Counter Function setup, and sets the current step in the Preset Table to Step 0.

  4. **Pulse On**: Turns ON the Table Driven Output for the amount of time specified in the Function Parameter, which is the duration of the output pulse in microseconds (1 to 16,777,215).

  5. **Pulse Off**: Turns OFF the Table Driven Output for the amount of time specified in the Function Parameter, which is the duration of the output pulse in microseconds (1 to 16,777,215).

  6. **Toggle**: Inverts the state of the Table Driven Output. If the Table Driven Output is currently ON, it is turned OFF, or it is turned ON if it is currently turned OFF.

- Several things to consider when using this instruction:

  1. Unlike the TDOPLS instruction, the values in the entry table do not have to be increasing with the Entry numbers. The table runs step by step in order of entry. Therefore, if the value in a higher entry is lower than a previous entry, the current count value must be decreasing in order for the step to be completed. For example, if Step 0 is configured as Set at 500 and Step 1 is configured as Reset at 0, the Output will be Set ON when the current counts reach 500 or more and the Output will be Reset to OFF when the current counts decrease to 0 or less.

  2. In order for the table to run continuously, the final entry function must be to (X) Reset Count. After the step with the Reset Count function is complete, the table will start over at Step 0. When the TDOPRESET instruction is disabled and re-enabled, it will start over at Step 0.

TDOPRESET, continued

3. If the Scaling was configured for the Master Register counter and the "Use Source Register Scaling" option was specified, the Step values will be entered as scaled values and no conversion will be necessary.

- **Insert** button (c): add an empty Step above the currently highlighted Step in the table.

- **Remove** button (d): delete the currently highlighted Step from the table.

- **Import** button (e): Import the contents of the Preset table from a CSV file. The format of the import file is two or three numbers per line, separated by a comma or whitespace, and a maximum of 64 lines. Please review the CSV file formatting information later in this TDOPRESET instruction section, under the section explaining the PLC TDO Preset Table Editor dialog.

On Success (f):

- **Set Bit**: The bit will become FALSE when the instruction is enabled and will become TRUE when instruction parameters are properly entered and the instruction completes successfully. The specified bit is enabled with a SET operation, not an OUT operation. The On Success bit will remain ON even if the instruction input logic goes OFF.

- **JMP to Stage**: When instruction parameters are properly entered and the instruction completes successfully, the PLC will jump to that stage.

On Error (g):

- **Set Bit**: The bit will become FALSE when the instruction is enabled and will become when proper instruction parameters are NOT properly entered or the instruction IS NOT completed successfully. The specified bit is enabled with a SET operation, not an OUT operation. The On Error bit will remain ON even if the instruction input logic goes OFF.

- **JMP to Stage**: When proper instruction parameters are NOT properly entered or the instruction IS NOT completed successfully, the PLC will jump to that stage.

NOTE: *Initial values for Preset Table Start Address, and Number of Table Rows to be Read (or Number of Preset Steps) come from the main instruction editor. If these values are changed while editing the table data with this dialog, those values will be updated on the main instruction editor when this dialog is closed.*

NOTE: *If the Number of Preset Steps is a variable (memory address), the value in that location is read and the data in that number of rows will be read from the PLC to prefill the table. The Also Write Table Length to PLC selection will be enabled and that variable location will be prefilled here so that when the table data is written back to the PLC this location will be updated as well.*

TDOPRESET, continued

PLC TDO Preset Table Editor
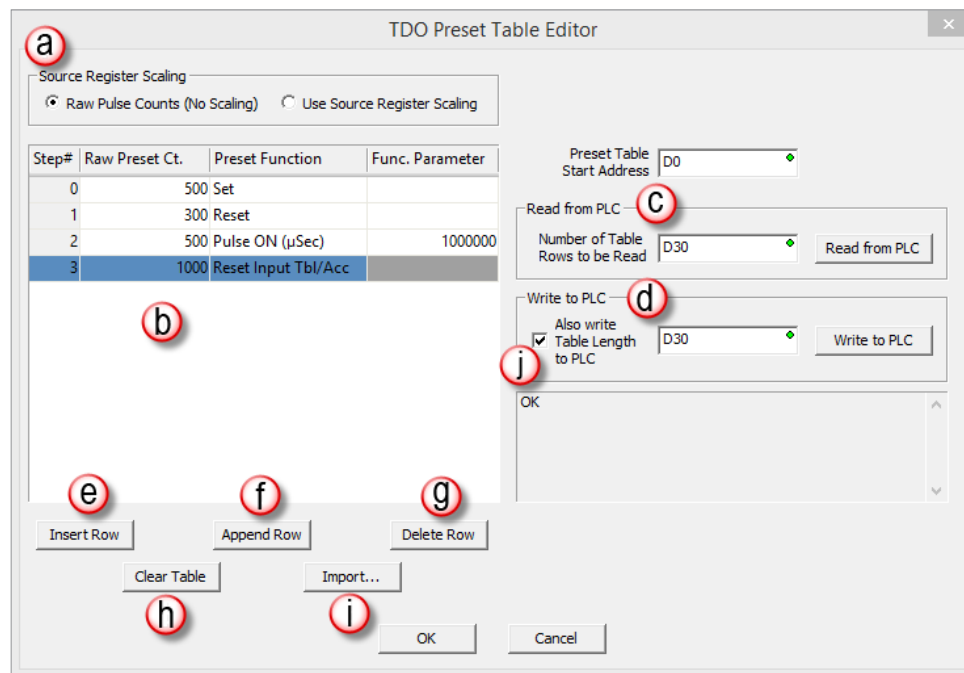
Source Register Scaling (a): Same as the main dialog. The TDOPRESET instruction can use the raw count value from the specified Master Register or, if scaling was enabled in the High-speed I/O setup, it can use the scaled value to do its comparisons.

Raw (or Scaled) Preset Count Steps (b): Same as main dialog. This is the list of steps that determines the Table Driven Output state when the TDOPRESET is running.

1. **Preset Count**: The PLC TDO Preset Table Editor is reading a block of D memory, it is editing the values that go into that block. The Preset Count in the PLC TDO Preset Table Editor can only be contant values. This value can be in the range of -2,147,483,648 and 2,147,483,647.

2. **Preset Functions**: Select the action to perform on the Table Driven Output when the Step is triggered.

   a. **Set**: Turns ON the Table Driven Output.

   b. **Reset**: Turns OFF the Table Driven Output.

   c. **Reset Table/Acc**: Performs a reset of the Master Register which sets its current count value to the Initial Reset Value specified in the Timer/Counter Function setup, and sets the current step in the Preset Table to Step 0.

   d. **Pulse On**: Turns ON the Table Driven Output for the amount of time specified in the Function Parameter, which is the duration of the output pulse in microseconds (1 to 16,777,215).

   e. **Pulse Off**: Turns OFF the Table Driven Output for the amount of time specified in the Function Parameter, which is the duration of the output pulse in microseconds (1 to 16,777,215).

   f. **Toggle**: Inverts the state of the Table Driven Output. If the Table Driven Output is currently ON, it is turned OFF, or it is turned ON if it is currently turned OFF.

3. **Read from PLC** button (c): Overwrite the contents of the table with values from the PLC memory location specified in the Load Raw Preset Table from Data Block section.

TDOPRESET, continued

Confirmation is requested before reading the PLC data. In this example the starting data block is D0, D30 is the memory address that stores the Number of Preset Steps. This has a value of 4. Eight DWords will be read from the PLC starting at D0:


4. **Write to PLC** (d): overwrite the contents of PLC memory at the location specified in the Load Raw Preset Table from Data Block section with the current contents of the table.

   Confirmation is requested before writing data to the PLC. The four rows of data are stored in D0 to D7.
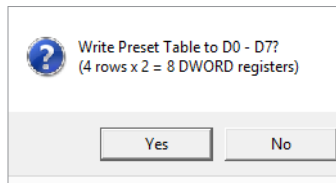
   Confirmation request before writing data to the PLC with the **Also write Table Length to PLC** enabled. In this example, there are 4 rows of data. A 4 is written to D30 and the 8 DWords of data is written to D0 to D7.
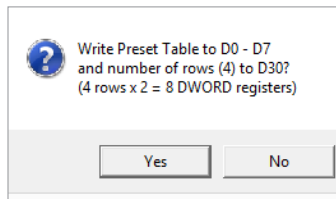
5. **Format of the Preset Data Read from PLC or Written to the PLC memory addresses**: When using the PLC TDO Preset Table Editor, the data values in the specified memory locations must be formatted properly. Improperly formatted data will generate a warning message. The following example shows a range of D0 to D7, which corresponds to a starting Preset Data Table address of D0 and 4 steps. Each step is comprised of two DWords. Each Preset table entry consists of 2 signed DWord locations:

   a. The first DWord location corresponds to the Preset Count where the specified command will take place. This can be any value between -2,147,483,648 and 2,147,483,647.

   b. The second DWord location contains the Function Code for the command to be performed and the additional data required by a command code.

      i. The upper BYTE (Byte 0) contains one of the following values to indicate the action:

         0: Set
         1: Reset
         2: Pulse ON
         3: Pulse OFF
         4: Toggle
         5: Reset Table & Acc

TDOPRESET, continued

    ii. When using the Pulse ON or Pulse OFF command, the lower 3 BYTEs of the second DWord location will contain the amount of time (in microseconds) the pulse will be ON or OFF respectively.

c. Example: This shows the Preset Table filled in with the desired values:

    Line 1   Step 0: Preset Count of 500. Set = Code 0.
    Line 2   Step 1: Preset Count of 300. Reset = Code 1.
    Line 3   Step 2: Preset Count of 500. Pulse ON = Code 2. Function Parameter of 1,000,000.
    Line 4   Step 3: Preset Count of 1000. Reset Table/Acc = Code 5.

0	500	Set	
1	300	Reset	
2	500	Pulse ON (µSec)	1000000
3	1000	Reset Input Tbl/Acc	

    i. The following Data View shows the Native values for the Data Block range of D0 to D7 associated with our example:

Data1

	Element	Status
1	D0	500
2	D1	0
3	D2	300
4	D3	16777216
5	D4	500
6	D5	34554432
7	D6	1000
8	D7	83886080
9		

    ii. The following Data View shows the second DWord of each step in BCD/HEX data type:

Data1

BCD/Hex

	Element	Status
1	D0	500
2	D1	0x00000000
3	D2	300
4	D3	0x01000000
5	D4	500
6	D5	0x020F4240
7	D6	1000
8	D7	0x05000000
9		

    D1 corresponds to SET (code 0) of Step 0.
    D3 corresponds to RESET (code 1) of Step 1.
    D5 corresponds to Pulse ON (code 2) of Step 2.
    D7 corresponds to Reset Table/Acc (code 5) of Step 3.

    iii. The upper Byte represents the Function Code. The lower three Bytes represent the Function Parameter value. 0Xuullllll.

    For Step 0, function code 0 (Set) is uu = 00.

    For Step 1, function code 1 (Reset) is uu = 01.

    For Step 2, function code 2 (Pulse ON) is uu = 02. Pulse ON has a function parameter of time in microseconds, which is llllll = 0F4240. Converting Hex 0F4240 to Decimal is 1,000,000.

TDOPRESET, continued

For Step 3, function code 5 (Reset Table/Acc) is uu = 05.

iv. If, for example, we want to change the Pulse ON to last 2,500,000 microseconds, D5 would have a BCD/HEX value = 0X022625A0.

6. **Insert Row** button (e): (Insert button in main dialog) add an empty Step above the currently highlighted Step in the table.

7. **Append Row** (f): Add a new row after the current last step in the table.

8. **Delete Row** button (g): (Remove button in main dialog) delete the currently highlighted Step from the table.

9. **Clear Table** button (h): Will display a confirmation dialog explaining it will delete all rows and the table will end up with a single row to Set at a Preset Value of 0.

10. **Import** button (i): (Import button in main dialog) Import the contents of the PLS Table from a CSV file. The format of the import file is two or three numbers per line, separated by a comma or whitespace, with a maximum of 64 lines.

Below is an example of how data should be formatted in the CSV file.

	A	B	C	D
1	1000	0		//Set ON @ 1000
2	2000	1		// Set OFF @ 2000
3	3000	2	1000000	//Pulse ON @ 3000 for 1,000,000 microseconds
4	4000	0		// Set ON @ 4000
5	5000	3	1500000	//Pulse OFF @ 5000 for 1,500,000 microseconds
6	6000	4		//Toggle @6000
7	7000	5		//Reset Count & Table @ 7000

Column A holds the Preset Count. This can be any value between -2,147,483,648 and 2,147,483,647

Column B holds the Preset Function code. The following table shows the available function codes.

Preset Function Codes		
0	SET	No function parameter needed
1	RESET	No function parameter needed
2	PULSE ON	Requires function parameter. Time in microseconds: 1 to 16,777,215
3	PULSE OFF	Requires function parameter. Time in microseconds: 1 to 16,777,215
4	TOGGLE	No function parameter needed
5	RESET TABLE & COUNT	No function parameter needed

Column C holds the Function Parameter, which is the amount of time (in microseconds) the pulse will be ON or OFF. This is required for the PULSE ON and PULSE OFF function codes.

Column D holds the comments, which is indicated by the "//" before the text.