Micro830, Micro850, and Micro870 Programmable Controllers

Allen-Bradley

by **ROCKWELL AUTOMATION**

User Manual

Original Instructions

Important User Information

Read this document and the documents listed in the additional resources section about installation, configuration, and operation of this equipment before you install, configure, operate, or maintain this product. Users are required to familiarize themselves with installation and wiring instructions in addition to requirements of all applicable codes, laws, and standards.

Activities including installation, adjustments, putting into service, use, assembly, disassembly, and maintenance are required to be carried out by suitably trained personnel in accordance with applicable code of practice.

If this equipment is used in a manner not specified by the manufacturer, the protection provided by the equipment may be impaired.

In no event will Rockwell Automation, Inc. be responsible or liable for indirect or consequential damages resulting from the use or application of this equipment.

The examples and diagrams in this manual are included solely for illustrative purposes. Because of the many variables and requirements associated with any particular installation, Rockwell Automation, Inc. cannot assume responsibility or liability for actual use based on the examples and diagrams.

No patent liability is assumed by Rockwell Automation, Inc. with respect to use of information, circuits, equipment, or software described in this manual.

Reproduction of the contents of this manual, in whole or in part, without written permission of Rockwell Automation, Inc., is prohibited.

Throughout this manual, when necessary, we use notes to make you aware of safety considerations.

WARNING: Identifies information about practices or circumstances that can cause an explosion in a hazardous environment, which may lead to personal injury or death, property damage, or economic loss.

ATTENTION: Identifies information about practices or circumstances that can lead to personal injury or death, property damage, or economic loss. Attentions help you identify a hazard, avoid a hazard, and recognize the consequence.

IMPORTANT   Identifies information that is critical for successful application and understanding of the product.

These labels may also be on or inside the equipment to provide specific precautions.

SHOCK HAZARD: Labels may be on or inside the equipment, for example, a drive or motor, to alert people that dangerous voltage may be present.

BURN HAZARD: Labels may be on or inside the equipment, for example, a drive or motor, to alert people that surfaces may reach dangerous temperatures.

ARC FLASH HAZARD: Labels may be on or inside the equipment, for example, a motor control center, to alert people to potential Arc Flash. Arc Flash will cause severe injury or death. Wear proper Personal Protective Equipment (PPE). Follow ALL Regulatory requirements for safe work practices and for Personal Protective Equipment (PPE).

The following icon may appear in the text of this document.

Identifies information that is useful and can help to make a process easier to do or easier to understand.

Rockwell Automation recognizes that some of the terms that are currently used in our industry and in this publication are not in alignment with the movement toward inclusive language in technology. We are proactively collaborating with industry peers to find alternatives to such terms and making changes to our products and content. Please excuse the use of such terms in our content while we implement these changes.

Table of Contents

Hardware Overview

Chapter 1

About Your Controller

Chapter 2

Chapter 10

Use the High-speed Counter and Programmable Limit Switch

Chapter 11

Controller Security

Use PCCC Commands and MicroLogix Mapping

Appendix I

Index

Notes:

About This Publication

Use this manual if you are responsible for designing, installing, programming, or troubleshooting control systems that use Micro800™ controllers.

You should have a basic understanding of electrical circuitry and familiarity with relay logic. If you do not, obtain the proper training before using this product.

This manual is a reference guide for Micro800 controllers, plug-in modules, and accessories. It describes the procedures you use to install, wire, and troubleshoot your controller. This manual:

- Explains how to install and wire your controllers.
- Gives you an overview of the Micro800 controller system.

See the Online Help provided with Connected Components Workbench™ software for more information on programming your Micro800 controller.

Conformal Coated Catalogs

Catalog numbers with the suffix 'K' are conformal coated and their specifications are the same as non-conformal coated catalogs.

Download Firmware, AOP, EDS, and Other Files

Download firmware, associated files (such as AOP, EDS, and DTM), and access product release notes from the Product Compatibility and Download Center at rok.auto/pcdc.

Summary of Changes

This publication contains the following new or updated information. This list includes substantive updates only and is not intended to reflect all changes.

Topic	Page
Updated section Update Key Change Method and Authority Certification Key	88, 89
Added section Adjust for Daylight Saving	115
Updated topic Memory Allocation	128
Updated table Module Properties Parameters	137
Updated section Requested Packet Interval	138
Added information on disabling communication ports	225, 226
Updated topic Operation Mode	225
Added 0xF8A3 to table List of Error Codes for Micro800 Controllers	313
Added topic Retrieve a Core Dump on Major Fault	314
Updated table General Diagnostic Parameters	318
Updated section EtherNet/IP Overview Diagnostic Information	319
Updated section System Diagnostic Information	321
Updated table Controller Diagnostic Parameters	322
Added appendix Use PCCC Commands and MicroLogix Mapping	383

Additional Resources

These documents contain additional information concerning related products from Rockwell Automation. You can view or download publications at rok.auto/literature.

Additional Resources

Resource	Description
Micro800 Programmable Controller Family Selection Guide, publication 2080-SG001	Provides information to help you select the Micro800 controller, plug-ins, expansion I/O, and accessories, based on your requirements.
Micro800 Programmable Controllers Technical Data, publication 2080-TD001	Provides detailed specifications for Micro800 controllers, expansion I/O modules, plug-in modules, and accessories.
Micro800 Expansion I/O Modules User Manual, publication 2080-UM003	Information on features, configuration, wiring, installation, and specifications for the Micro800 expansion I/O modules and power supply.
Micro800 Plug-in Modules User Manual, publication 2080-UM004	Information on features, configuration, installation, wiring, and specifications for the Micro800 plug-in modules.
Micro800 Programmable Controllers General Instructions Reference Manual, publication 2080-RM001	Information on instruction sets for developing programs for use in Micro800 control systems.
Micro800 Programmable Controllers: Getting Started with Motion Control Using a Simulated Axis Quick Start, publication 2080-QS001	Provides quick start instructions for implementing a motion control project in Connected Components Workbench software.
Micro800 Programmable Controllers: Getting Started with CIP Client Messaging Quick Start, publication 2080-QS002	Provides quick start instructions for using CIP GENERIC and CIP Symbolic Messaging.
Micro800 Programmable Controllers: Getting Started with PanelView Plus Quick Start, publication 2080-QS003	Provides quick start instructions for using global variables for Micro800 controllers together with PanelView™ Plus HMI terminals.
Configuring Micro800 Controllers on FactoryTalk Linx Gateway Quick Start, publication 2080-QS005	Provides quick start instructions for configuring a Micro800 controller on FactoryTalk Linx Gateway.
Set up Micro800 Controllers for Implicit (Class 1) Comms with POINT I/O Adapters, publication 2080-QS006	Provides quick start instructions on how to set up Micro800 controllers to use Class 1 communications with POINT I/O™ adapters.
Micro800 Programmable Controller External AC Power Supply Installation Instructions 2080-IN001	Information on mounting and wiring the optional external power supply.
Micro800 Programmable Controllers Installation Instructions, publication 2080-IN013	Information on mounting and wiring Micro800 Controllers
Micro800 16-point and 32-point 12/24V Sink/Source Input Modules Installation Instructions, publication 2085-IN001	Information on mounting and wiring the expansion I/O modules (2085-IQ16, 2085-IQ32T)
Micro800 Bus Terminator Module Installation Instruction, publication 2085-IN002	Information on mounting and wiring the expansion I/O bus terminator (2085-ECR)
Micro800 16-point Sink and 16-point Source 12/24V DC Output Modules Installation Instructions, publication 2085-IN003	Information on mounting and wiring the expansion I/O modules (2085-OV16, 2085-OB16)
Micro800 8-point and 16-point AC/DC Relay Output Modules Installation Instructions, publication 2085-IN004	Information on mounting and wiring the expansion I/O modules (2085-OW8, 2085-OW16)
Micro800 8-point Input and 8-point Output AC Modules Installation Instructions, publication 2085-IN005	Information on mounting and wiring the expansion I/O modules (2085-IA8, 2085-IM8, 2085-OA8)
Micro800 4-channel and 8-channel Analog Voltage/current Input and Output Modules Installation Instructions, publication 2085-IN006	Information on mounting and wiring the expansion I/O modules (2085-IF4, 2085-IF8, 2085-OF4)
Micro800 4-channel Thermocouple/RTD Input Module Installation Instructions, publication 2085-IN007	Information on mounting and wiring the expansion I/O module (2085-IRT4)
Micro870 Programmable Controllers 24V DC Expansion Power Supply Installation Instructions, publication 2085-IN008	Information on mounting and wiring the optional external power supply for expansion I/O modules.
Micro800 RS-232/RS-485 Isolated Serial Port Plug-in Module Wiring Diagrams, publication 2080-WD002	Information on mounting and wiring the Micro800 RS-232/RS-485 Isolated Serial Port Plug-in Module.
Micro800 Non-isolated Unipolar Analog Input Plug-in Module Wiring Diagrams, publication 2080-WD003	Information on mounting and wiring the Micro800 Non-isolated Unipolar Analog Input Plug-in Module.
Micro800 Non-isolated Unipolar Analog Output Plug-in Module Wiring Diagrams, publication 2080-WD004	Information on mounting and wiring the Micro800 Non-isolated Unipolar Analog Output Plug-in Module.
Micro800 Non-isolated RTD Plug-in Module Wiring Diagrams, publication 2080-WD005	Information on mounting and wiring the Micro800 Non-isolated RTD Plug-in Module.
Micro800 Non-isolated Thermocouple Plug-in Module Wiring Diagrams, publication 2080-WD006	Information on mounting and wiring the Micro800 Non-isolated Thermocouple Plug-in Module.
Micro800 Memory Backup and High Accuracy RTC Plug-In Module Wiring Diagrams, publication 2080-WD007	Information on mounting and wiring the Micro800 Memory Backup and High Accuracy RTC Plug-In Module.
Micro800 6-Channel Trimpot Analog Input Plug-In Module Wiring Diagrams, publication 2080-WD008	Information on mounting and wiring the Micro800 6-Channel Trimpot Analog Input Plug-In Module.
Micro800 Digital Relay Output Plug-in Module Wiring Diagrams, publication 2080-WD010	Information on mounting and wiring the Micro800 Digital Relay Output Plug-in Module.
Micro800 Digital Input, Output, and Combination Plug-in Modules Wiring Diagrams, publication 2080-WD011	Information on mounting and wiring the Micro800 Digital Input, Output, and Combination Plug-in Modules.

Additional Resources (Continued)

Resource	Description
Micro800 High-speed Counter Plug-in Module Wiring Diagram, publication 2080-WD012	Information on mounting and wiring the High-speed Counter Plug-in module.
Micro800 DeviceNet Plug-in Module Wiring Diagram, publication 2080-WD013	Information on mounting and wiring the Micro800 DeviceNet® plug-in module.
EtherNet/IP Network Devices User Manual, publication ENET-UM006	Describes how to configure and use EtherNet/IP devices to communicate on the EtherNet/IP network.
Ethernet Reference Manual, publication ENET-RM002	Describes basic Ethernet concepts, infrastructure components, and infrastructure features.
System Security Design Guidelines Reference Manual, publication SECURE-RM001	Provides guidance on how to conduct security assessments, implement Rockwell Automation products in a secure system, harden the control system, manage user access, and dispose of equipment.
Industrial Components Preventive Maintenance, Enclosures, and Contact Ratings Specifications, publication IC-TD002	Provides a quick reference tool for Allen-Bradley® industrial automation controls and assemblies.
Safety Guidelines for the Application, Installation, and Maintenance of Solid-state Control, publication SGI-1.1	Designed to harmonize with NEMA Standards Publication No. ICS 1.1-1987 and provides general guidelines for the application, installation, and maintenance of solid-state control in the form of individual devices or packaged assemblies incorporating solid-state components.
Industrial Automation Wiring and Grounding Guidelines, publication 1770-4.1	Provides general guidelines for installing a Rockwell Automation® industrial system.
Product Certifications website, rok.auto/certifications	Provides declarations of conformity, certificates, and other certification details.

You can download the latest version of Connected Components Workbench software for your Micro800 controller at rok.auto/ccw.

Notes:

Hardware Overview

This chapter provides an overview of the Micro830®, Micro850®, and Micro870® controller hardware features.

Hardware Features

Micro830, Micro850, and Micro870 controllers are economical brick style controllers with embedded inputs and outputs. The controllers can accommodate from two to five plug-in modules, depending on the controller type. The Micro850 and Micro870 controllers have expandable features. The Micro850 controller supports up to four expansion I/O modules and the Micro870 controller supports up to eight expansion I/O modules.

IMPORTANT	For information on supported plug-in modules and expansion I/O modules, see the following publications:
	• Micro800 Expansion I/O Modules User Manual, publication 2080-UM003
	• Micro800 Plug-in Modules User Manual, publication 2080-UM004

The controllers also accommodate any class 2 rated 24V DC output power supply that meets minimum specifications, such as the optional Micro800 power supply.

For descriptions of status indicator operation for troubleshooting purposes, see Troubleshooting on page 307.

Micro830 Controllers

Micro830 10/16-point controllers and status indicators
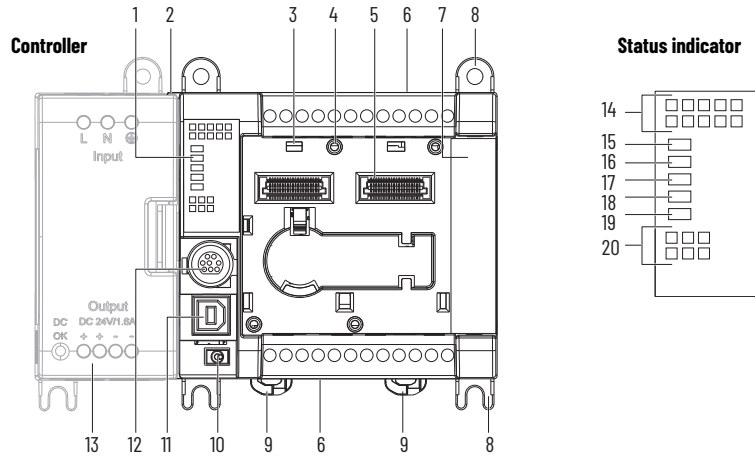
Micro830 24-point controllers and status indicators
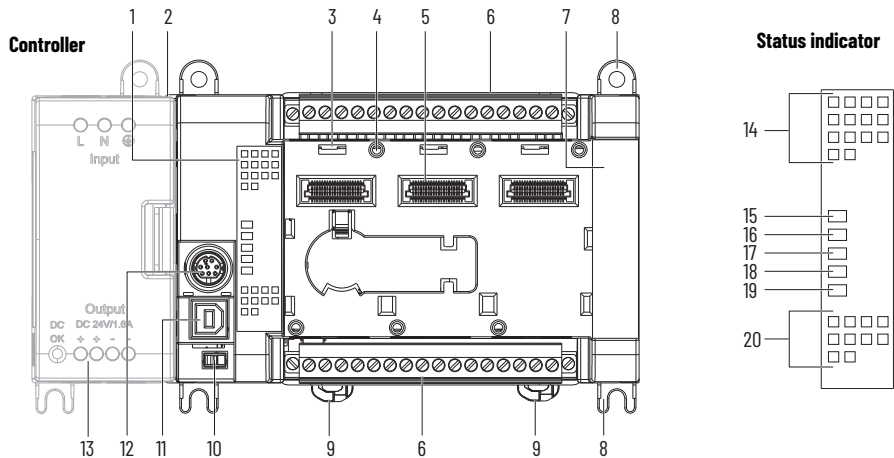
Micro830 48-point controllers and status indicators
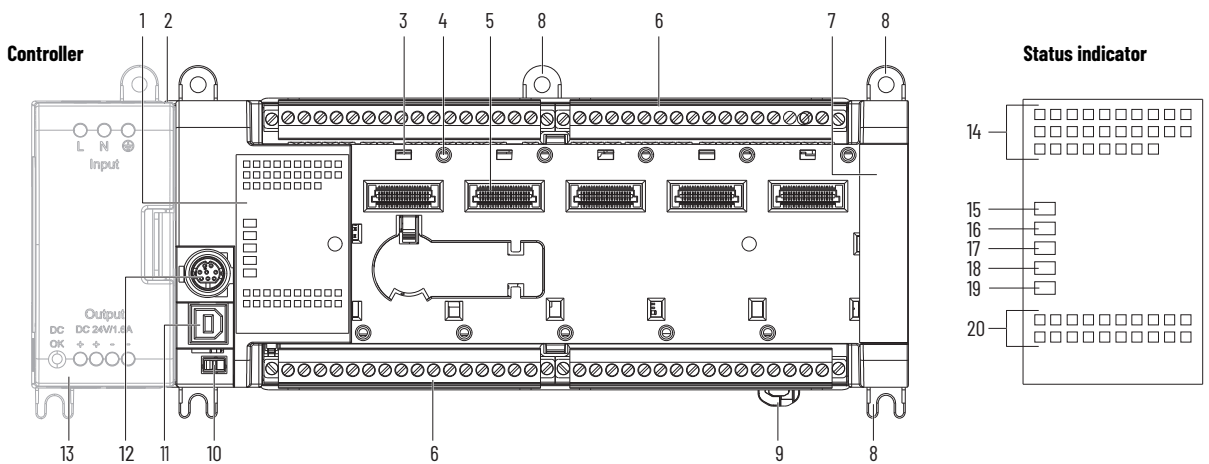
Micro830 Controller Description

	Description			Description
1	Status indicators		8	Mounting screw hole / mounting foot
2	Optional power supply slot		9	DIN rail mounting latch
3	Plug-in latch		10	Mode switch
4	Plug-in screw hole		11	Type B connector USB port
5	40-pin high-speed plug-in connector		12	RS-232/RS-485 non-isolated combo serial port
6	Removable I/O terminal block		13	Optional AC power supply
7	Right-side cover			

Micro830 Controller Status Indicator Description [(1)]

	Description			Description
14	Input status		18	Force status
15	Power status		19	Serial communications status
16	Run status		20	Output status
17	Fault status			

(1)   For detailed description of the different status LED indicators, see .

Micro850 Controllers

Micro850 24-point controllers and status indicators
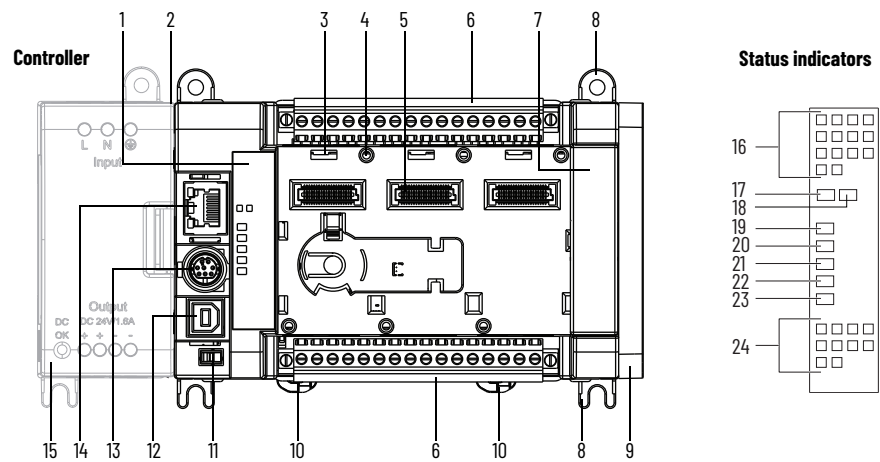
Micro850 48-point controllers and status indicators

Micro850 Controller Description

	Description		Description
1	Status indicators	9	Expansion I/O slot cover
2	Optional power supply slot	10	DIN rail mounting latch
3	Plug-in latch	11	Mode switch
4	Plug-in screw hole	12	Type B connector USB port
5	40-pin high-speed plug-in connector	13	RS-232/RS-485 non-isolated combo serial port
6	Removable I/O terminal block	14	RJ45 EtherNet/IP connector (with embedded yellow and green LED indicators)
7	Right-side cover	15	Optional AC power supply
8	Mounting screw hole/mounting foot		

Micro850 Controller Status Indicator Description [1]

	Description		Description
16	Input status	21	Fault status
17	Module status	22	Force status
18	Network status	23	Serial communications status
19	Power status	24	Output status
20	Run status		

(1)   For detailed description of the different status LED indicators, see .

> You can order the following replacement terminal blocks separately:
> - 2080-RPL24RTB for 24-point base controllers
> - • 2080-RPL48RTB for 48-point base controllers

Micro870 Controllers

Micro870 24-point controllers and status indicators

Micro870 Controller Description

	Description		Description
1	Status indicators	9	Expansion I/O slot cover
2	Optional power supply slot	10	DIN rail mounting latch
3	Plug-in latch	11	Mode switch
4	Plug-in screw hole	12	Type B connector USB port
5	40-pin high-speed plug-in connector	13	RS-232/RS-485 non-isolated combo serial port
6	Removable I/O terminal block	14	RJ45 EtherNet/IP connector (with embedded yellow and green LED indicators)
7	Right-side cover	15	Optional AC power supply
8	Mounting screw hole/mounting foot		

Micro870 Controller Status Indicator Description [1]

	Description			Description
16	Input status		21	Fault status
17	Module status		22	Force status
18	Network status		23	Serial communications status
19	Power status		24	Output status
20	Run status			

(1)   For detailed description of the different status LED indicators, see .

You can order replacement terminal blocks, catalog number 2080-RPL24RTB, separately.

Table 1 - Micro830 Controllers – Number and Types of Inputs/Outputs

Catalog Number	Inputs		Outputs			PTO Support	HSC Support
	110V AC	24V DC/V AC	Relay	24V Sink	24V Source		
2080-LC30-10QWB	–	6	4	–	–	–	2
2080-LC30-10QVB	–	6	–	4	–	1	2
2080-LC30-16AWB	10	–	6	–	–	–	–
2080-LC30-16QWB	–	10	6	–	–	–	2
2080-LC30-16QVB	–	10	–	6	–	1	2
2080-LC30-24QWB	–	14	10	–	–	–	4
2080-LC30-24QVB	–	14	–	10	–	2	4
2080-LC30-24QBB	–	14	–	–	10	2	4
2080-LC30-48AWB	28	–	20	–	–	–	–
2080-LC30-48QWB	–	28	20	–	–	–	6
2080-LC30-48QVB	–	28	–	20	–	3	6
2080-LC30-48QBB	–	28	–	–	20	3	6

Table 2 - Micro850 Controllers – Number and Types of Inputs/Outputs

Catalog Number	Inputs		Outputs			PTO Support	HSC Support
	120V AC	24V DC/V AC	Relay	24V Sink	24V Source		
2080-LC50-24AWB	14	–	10	–	–	–	–
2080-L50E-24AWB	14	–	10	–	–	–	–
2080-LC50-24QWB	–	14	10	–	–	–	4
2080-L50E-24QWB	–	14	10	–	–	–	4
2080-LC50-24QVB	–	14	–	10	–	2	4
2080-L50E-24QVB	–	14	–	10	–	2	4
2080-LC50-24QBB	–	14	–	–	10	2	4
2080-L50E-24QBB	–	14	–	–	10	2	4
2080-LC50-48AWB	28	–	20	–	–	–	–
2080-L50E-48AWB	28	–	20	–	–	–	–
2080-LC50-48QWB	–	28	20	–	–	–	6
2080-L50E-48QWB	–	28	20	–	–	–	6
2080-LC50-48QWBK	–	28	20	–	–	–	6
2080-L50E-48QWBK	–	28	20	–	–	–	6
2080-LC50-48QVB	–	28	–	20	–	3	6
2080-L50E-48QVB	–	28	–	20	–	3	6
2080-LC50-48QBB	–	28	–	–	20	3	6
2080-L50E-48QBB	–	28	–	–	20	3	6

Table 3 - Micro870 Controllers – Number and Types of Inputs/Outputs

Catalog Number	Inputs		Outputs			PTO Support	HSC Support
	120V AC	24V DC/V AC	Relay	24V Sink	24V Source		
2080-LC70-24AWB	14	–	10	–	–	–	–
2080-L70E-24AWB	14	–	10	–	–	–	–
2080-LC70-24QWB	–	14	10	–	–	–	4
2080-L70E-24QWB	–	14	10	–	–	–	4
2080-LC70-24QWBK	–	14	10	–	–	–	4
2080-L70E-24QWBK	–	14	10	–	–	–	4
2080-L70E-24QWBN	–	14	10	–	–	–	4
2080-L70E-24QWBNK	–	14	10	–	–	–	4
2080-LC70-24QBB	–	14	-	–	10	2	4
2080-L70E-24QBB	–	14	-	–	10	2	4
2080-LC70-24QBBK	–	14	-	–	10	2	4
2080-L70E-24QBBK	–	14	-	–	10	2	4
2080-L70E-24QBBN	–	14	-	–	10	2	4

Programming Cables

Micro800 controllers have a USB interface that lets you use standard USB cables as programming cables.

Use a standard USB type A-male to type B-male cable for programming the controller.

Embedded Serial Port Cables

Embedded serial port cables for communication are listed here. All embedded serial port cables must be 3 meters (9.8 feet) in length, or shorter.

Table 4 - Embedded Serial Port Cable Selection Chart

Connectors	Length	Catalog Number	Connectors	Length	Catalog Number
8-pin Mini DIN to 8-pin Mini DIN	0.5 m (1.5 ft)	1761-CBL-AM00[1]	8-pin Mini DIN to 9-pin D-shell	0.5 m (1.5 ft)	1761-CBL-AP00[1]
8-pin Mini DIN to 8-pin Mini DIN	2 m (6.5 ft)	1761-CBL-HM02[1]	8-pin Mini DIN to 9-pin D-shell	2 m (6.5 ft)	1761-CBL-PM02[1]
8-pin Mini DIN to 8-pin Mini DIN (with lock mechanism on both connectors)	2 m (6.5 ft)	1761-CBL-AH02	8-pin Mini DIN with lock mechanism to 9-pin D-shell	2 m (6.5 ft)	1761-CBL-PH02
—			8-pin Mini DIN to 6-pin RS-485 terminal block	30 cm (11.8 in.)	1763-NC01 series A

(1)   Series C or later for Class 1 Div 2 applications.

Embedded Ethernet Support

For Micro850 and Micro870 controllers, a 10/100 Base-T Port (with embedded green and yellow LED indicators) provides connection to an Ethernet network with any standard RJ45 Ethernet cable. The LED indicators serve as indicators for transmit and receive status.

RJ45 Ethernet Port Pin Mapping

Contact Number	Signal	Direction	Primary Function
1	TX+	OUT	Transmit data +
2	TX-	OUT	Transmit data -
3	RX+	IN	Differential Ethernet receive data +
4			Terminated
5			Terminated
6	RX-	IN	Differential Ethernet receive data -
7			Terminated
8			Terminated
Shield			Chassis ground

Yellow LED —
RJ45 connector —
Green LED —
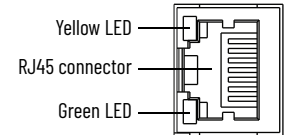
The yellow status LED indicates Link (steady yellow) or No Link (off).

The green status LED indicates activity (flashing green) or no activity (off).

Micro850 and Micro870 controllers support Ethernet crossover cables (2711P-CBL-EX04).

Ethernet Status Indication

Micro850 and Micro870 controllers also support two LEDs for EtherNet/IP™ to indicate the following:

- Module status
- Network status

For descriptions of Module and Network status indicators, see .

Notes:

About Your Controller

Programming Software for Micro800 Controllers

Connected Components Workbench software is a set of collaborative tools supporting Micro800 controllers. It is based on Rockwell Automation and Microsoft® Visual Studio® technology and offers controller programming, device configuration, and integration with an HMI editor. Use this software to program your controllers, configure your devices, and design your operator interface applications.

Connected Components Workbench software provides a choice of IEC 61131-3 programming languages (ladder diagram, function block diagram, and structured text) with user-defined function block support that optimizes machine control.

Obtain Connected Components Workbench Software

There are two editions for the Connected Components Workbench software:
- You can download Connected Components Workbench Standard Edition software for free at rok.auto/pcdc.
- To purchase Connected Components Workbench Developer Edition software, visit rok.auto/ccw.

Use Connected Components Workbench Software

To help you program your controller through the Connected Components Workbench software, you can see the Connected Components Workbench Online Help that comes with the software.

IMPORTANT	The new Micro850 (2080-L50E) and Micro870 (2080-L70E) controllers are only supported from Connected Components Workbench software version 20.01.00 onwards.

Controller Changes in Run Mode

Micro830, Micro850, and Micro870 controllers allow you to make certain changes while in Run mode by using the following features:
- Run Mode Change (RMC)

  Allows logic modifications to a running project without going to Remote Program mode.

  For more information, see .
- Run Mode Configuration Change (RMCC)

  Allows changing the address configuration of the controller to be made within a program during Run mode.

  For more information, see .

Using Run Mode Change (RMC)

Run Mode Change (RMC) is a productivity enhancement feature that is supported in Connected Components Workbench software for Micro830, Micro850, and Micro870 controllers. This feature saves you time by allowing logic modifications to a running project without going to Remote Program mode and without disconnecting from the controller.

> **IMPORTANT**   Micro830 and Micro850 controller firmware revision 8.011 or later is also required to use Run Mode Change.

RMC is useful during project development, when you add small changes incrementally to the logic and want to see the effects of the changes on the machine immediately. With RMC, since the controller stays in Remote Run mode, the controller logic and machine actuators do not have to reinitialize constantly, which can occur if the controller is switched to Remote Program mode (for example, the first scan bit is checked in the program logic to clear outputs).

When you edit, build, and download a project without using RMC, a full build of the entire controller project is performed and a full download of the project is performed. During RMC an incremental build is performed and only incremental changes are downloaded to the controller.

> **IMPORTANT**   Do not disconnect from the controller after you perform Run Mode Change, do a full build, and then try to reconnect. Connected Components Workbench software treats the project in the controller as different from the project in Connected Components Workbench software, even though the logic is identical, and asks to either upload or download the project.

RMC is performed incrementally at the end of every program scan to help prevent a large delay in the program scan. This process adds up to an additional 12 ms to the scan time. For example, if the program scan is normally 10 ms, it may increase to 22 ms during RMC until the update is finished. Similarly, user interrupts may be delayed.

Table 5 - Example of the Benefits of Using RMC – 20% Reduction in Download Time

Number of Changes	Time to Perform Conventional Download (seconds)	Time to Test Logic and Accept Changes (seconds)
1	36	29
5	180	130
10	360	255

Memory size of project used for comparison:
Data = 14,784 bytes; Program = 2,352 bytes
Note: The duration starts when connected to the controller and the RMC button is selected. The duration ends when the accept change process is finished.
For example:
1. When connected to the controller, select RMC.
2. Modify the program.
3. Select Test Logic.
4. Select Accept to finish, or select Test Logic to make another change.

> **ATTENTION:** Use extreme caution when you use Run Mode Change. Mistakes can injure personnel and damage equipment. Before using Run Mode Change:
> • Assess how machinery responds to the changes.
> • Notify all personnel about the changes.

A new global variable __SYSVA_PROJ_INCOMPLETE is added to indicate when Run Mode Changes are being made. Use this variable to notify personnel on the HMI that there are uncommitted changes in the controller.

Table 6 - Bit Definitions of Global Variable – __SYSVA_PROJ_INCOMPLETE

Bit	Definition
0	Set when the Run Mode Change process starts. Cleared once the Run Mode Change is written permanently to the controller (completion of Accept or Undo). This bit can be used to warn operators that a run mode change is in progress and that there are uncommitted changes in the controller.
1	Set if an error occurred while saving the changes to the flash memory or an integrity check failed during Run Mode Change. Cleared on the next successful Run Mode change.

When you perform a Test Logic Change, the value of the variable changes from zero to one. After you choose to accept or undo the changes, the value of the variable resets to zero.

IMPORTANT	When a Test Logic is performed, or undoing changes after the Test Logic is completed, any active communication instructions are aborted while the changes are downloaded to the controller.

Uncommitted Changes

Uncommitted changes are changes that are made in RMC that have not been accepted or undone after a Test Logic Change is performed.

If the controller power loses power while there are uncommitted changes, you cannot reenter RMC upon reconnection. You can choose to download the project again to keep the changes, or upload to discard the uncommitted changes.

If you choose to upload a project with uncommitted changes from the controller, you cannot enter RMC until you have done a full download.

RMC Memory

Run Mode Change (RMC) memory is used to store both the logic and user variable changes made during RMC. The default amount of memory that is allocated is 2 KB and can be increased up to 16 KB. However, there is still a limit of 2 KB for logic and user variables changes per Test Logic. To adjust the amount of RMC memory, the controller must be offline. After you have adjusted the amount, you must build the project and download it to the controller.

IMPORTANT	In a Connected Components Workbench software version 8 project, the available user data space was reduced by 6 KB to support optimal project settings for the new RMC feature.
	If you have a project that was developed before version 8, you may need to reduce the default "Allocated" 8 KB Temporary Variables section from the Memory page to compile the project successfully.

Figure 1 - Controller Memory Diagnostics Page in Connected Components Workbench Software
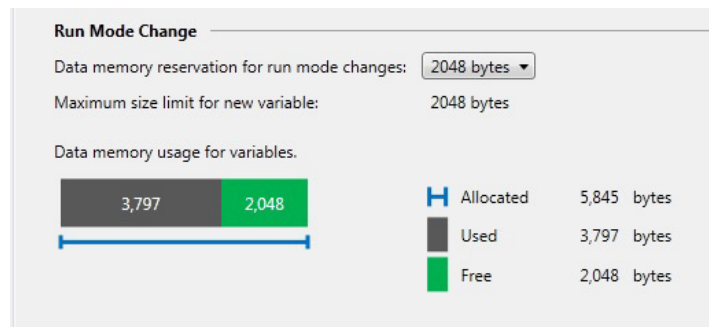
During RMC, an incremental build is performed and only incremental changes are downloaded to the controller until the RMC memory is filled.

Figure 2 - RMC Memory Usage Example
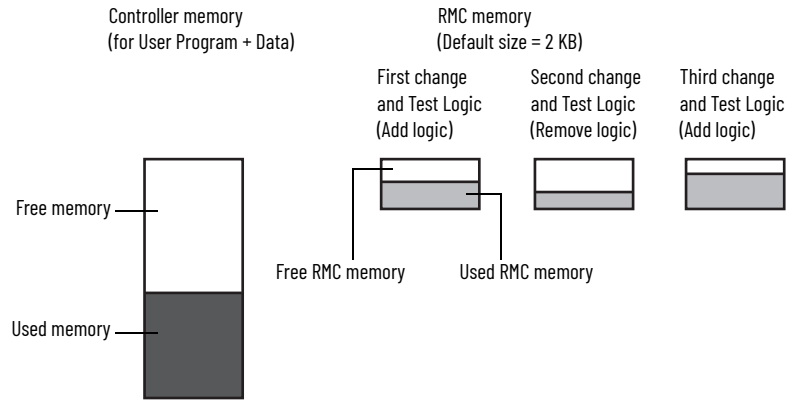
If insufficient RMC memory is available to make more changes (for example, a "not enough memory" error message appears during the RMC build or Test Logic), then you must perform a full download to transfer the incremental changes from the RMC memory to the standard user program and data memory.

Transfer Contents in RMC Memory to Controller Memory

The changes that you made during RMC are stored in the RMC memory and remain there until you perform a full build and download (while the controller is disconnected).

Figure 3 - RMC Memory Usage When Performing Full Build and Download Example
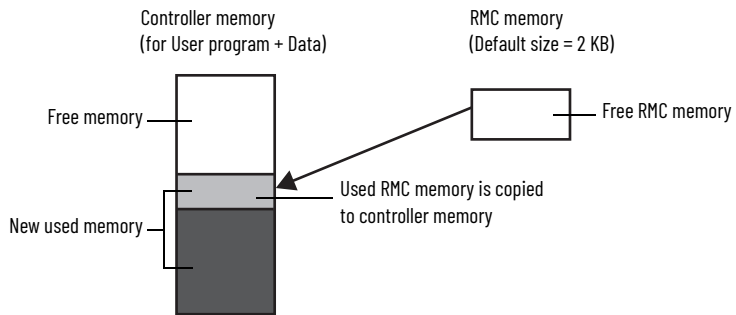
However, if the controller memory has insufficient space remaining to copy the contents of the RMC memory as shown in Figure 4, the operation fails and a "not enough memory" error message appears. Do not use RMC if you are near the limits of your controller memory.

Figure 4 - Insufficient Controller Memory Example
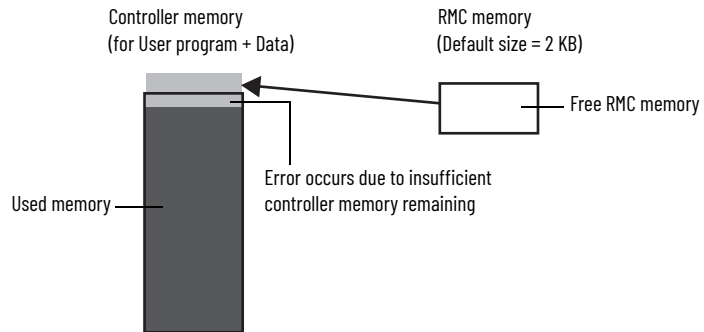
Limitations of RMC

Take note of the following limitations when using the Run Mode Change (RMC) feature:

- Configuration changes cannot be made (for example, change filter times).
- Up to 2 KB of logic (approximately 150 Boolean instructions[1]) and user variables can be added for each Test Logic.
- Total memory allocated for RMC (cumulative of all Test Logic Changes) can be increased from 2...16 KB, but the 2 KB limit for logic and user variables per Test Logic remains.
- Up to 20 Program Organizational Units (POU) can be added for each change (for example, if you currently have 5 POU, you can add 20 more for a total of 25 POU).
- If a user-defined function block is modified that changes the local variables, the local variables are reinitialized or reset to zero and a warning message is shown during the build. If you want to reapply the initial value, right-click on the UDFB and select Refactor > Reset Initial Values of Instances.
- If a new module is detected after doing a Discover Project operation, RMC is not possible because the configuration has changed.
- Exchange files cannot be imported when in RMC because it is considered a configuration change.
- Making changes to the display configuration (for example, to hide comments) are treated as logic changes and require you to build the project.
- Global variables cannot be deleted or modified in RMC, but can be added. To delete or modify a global variable, you must disconnect the controller from the Connected Components Workbench software.
- When using CIP™ messaging in RMC, setting the CIPTARGETCFG data type parameter ConnClose to TRUE has no effect. The Ethernet session does not close immediately upon successful messaging and you have to wait for the connection to time out after 60 seconds. This behavior applies to Connected Components Workbench software version 9 or earlier projects. For version 10 or later projects, the CIP connection timeout is configurable.

> ⚠️ **WARNING:** If you delete the output rung when in Run Mode Change and accept the changes, the output on the controller remains ON.

For an example of how to use this feature, see .

Using Run Mode Configuration Change (RMCC)

Run Mode Configuration Change (RMCC) is a productivity enhancement feature that is supported in Connected Components Workbench software for Micro830, Micro850, and Micro870 controllers. It allows you to reuse an identical program with multiple controllers by changing the address configuration of a controller within the program during Run mode. Micro830 and Micro850 controller firmware revision 9.011 or later is required to use this feature.

You can use RMCC to change the address configuration of the controller during Run mode, when the communication protocol is set to Modbus RTU for the serial ports or EtherNet/IP for the Ethernet port. RMCC uses a CIP Generic message, which can only be sent from within a controller program and not from an external device to the controller.

> **IMPORTANT**   During RMCC, the scan time may increase to close to 100 ms. Do not perform RMCC if the controller is performing time critical operations.

---

(1)   Approximately 85 Boolean instructions for Micro850 (2080-L50E) and Micro870 (2080-L70E) controllers.

Figure 5 - CIP Generic Message Instruction for Run Mode Configuration Change
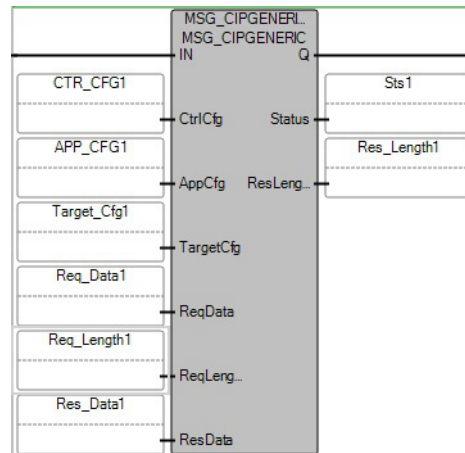
Only the controller that is sending the message can perform RMCC. To perform RMCC, you must configure the CIP Generic message as a loop-back message by setting the path to "0,0".

Figure 6 - Configure CIP Generic Message as a Loop-back Message

For Micro830, Micro850, and Micro870 controllers, the address configuration change is permanent and is retained when you cycle power to the controller.

Using Modbus RTU Communication

To use RMCC with the Modbus RTU communication protocol, the serial port must be set to the Modbus slave role. A CIP Generic message is sent from within a program with the following parameters.
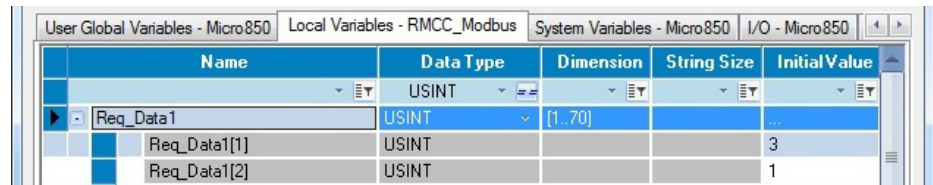
Table 7 - CIP Generic Message Parameters for RMCC using Modbus RTU

Parameter	Value
Service	16
Class	70
Instance	2 – Embedded serial port 5, 6, 7, 8, or 9 – Plug-in modules
Attribute	100
ReqData	New node address, 1
ReqLen	2

Figure 7 - RMCC Modbus Example – Set the Parameters
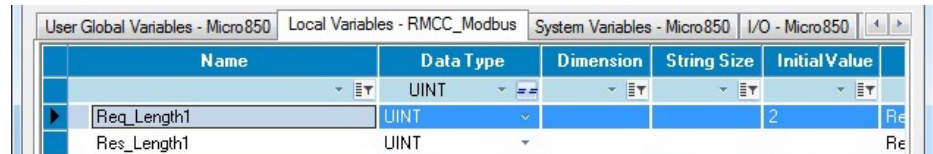
Figure 8 - RMCC Modbus Example – Set the New Node Address

The first byte indicates the new node address for the controller. For this example, the new node address is "3". The second byte must always be "1", this indicates that the Modbus role is configured as Slave.

Figure 9 - RMCC Modbus Example – Set the Message Length
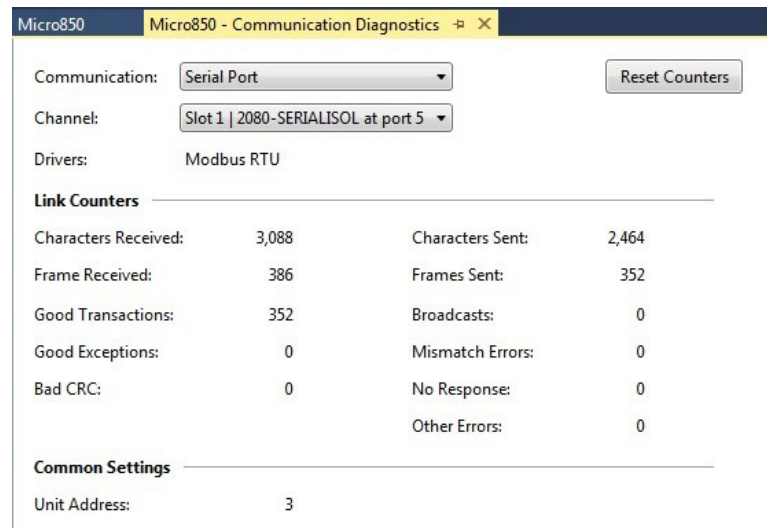
When the new node address is configured and applied, the port is not restarted.

IMPORTANT	You must verify that the new node address being configured is unique as it is not checked against existing node addresses of other devices.

You can verify that the node address has changed after performing RMCC by examining the Communication Diagnostics tab for the controller.

Figure 10 - RMCC Modbus Example – Verify Address Change

Using EtherNet/IP Communication

To use RMCC with the EtherNet/IP communication protocol, the controller must be configured to use a static IP address. If the controller is configured to use BOOTP or DHCP, the change is rejected. A CIP Generic message is sent from within a program with the following parameters.

Use RMCC when configuring the controller during commissioning. Immediately after changing the IP address, the cycle time may increase up to 100 ms for one program scan.

Table 8 - CIP Generic Message Parameters for RMCC using EtherNet/IP

Parameter	Value
Service	16
Class	245
Instance	1
Attribute	5
ReqData	IP address, subnet mask, Gateway address
ReqLen	22 bytes

Figure 11 – RMCC EtherNet/IP Example – Set the Parameters

Figure 12 – RMCC EtherNet/IP Example – Set the New IP Address

For this example, the new IP address is set to the following:

- IP address = 192.168.1.10
- Subnet mask = 255.255.255.0
- Gateway address = 192.168.1.1

Figure 13 - RMCC EtherNet/IP Example – Set the Message Length

After the new IP address is configured and applied, the controller disconnects from the Connected Components Workbench software if communication is through Ethernet.

> **IMPORTANT** Micro830 controllers do not support Run Mode Configuration Change using EtherNet/IP.

> **IMPORTANT** You should not perform IP address changes continuously. Allow an interval of at least six seconds before performing the next IP address change in order for duplicate address detection to work properly.

You can verify that the IP address has changed after performing RMCC by examining the Ethernet settings for the controller.

Figure 14 - RMCC EtherNet/IP Example – Verify Address Change

Safety Considerations

Safety considerations are an important element of proper system installation. Actively considering the safety of yourself and others, and the condition of your equipment, is of primary importance. We recommend reviewing the following safety considerations.

Disconnect Main Power

The main power disconnect switch should be located where the switch is quickly and easily accessible to operators and maintenance personnel. Besides disconnecting electrical power, all other sources of power (pneumatic and hydraulic) should be de-energized before working on a machine or process that is controlled by a controller.

> **WARNING:** Explosion Hazard
> Do not replace components, connect equipment, or disconnect equipment unless power has been switched off.

Safety Circuits

Circuits that are installed on the machine for safety reasons, like overtravel limit switches, stop push buttons, and interlocks, should always be hard-wired directly to the master control relay. These devices must be wired in series so that when any one device opens, the master control relay is de-energized, thus removing power to the machine. Never alter these circuits to defeat their function. Serious injury or machine damage could result.

> **WARNING:** Explosion Hazard
> Do not connect or disconnect connectors while the circuit is live.

Power Distribution

There are some points about power distribution that you should know:

- The master control relay must be able to inhibit all machine motion by removing power to the machine I/O devices when the relay is de-energized. It is recommended that the controller remains powered even when the master control relay is de-energized.

- If you are using a DC power supply, interrupt the load side rather than the AC line power. This avoids the additional delay of power supply turn-off. The DC power supply should be powered directly from the fused secondary of the transformer. Power to the DC input and output circuits should be connected through a set of master control relay contacts.

Periodic Tests of Master Control Relay Circuit

Any part can fail, including the switches in a master control relay circuit. The failure of one of these switches would most likely cause an open circuit, which would be a safe power-off failure. However, if one of these switches shorts out, it no longer provides any safety protection. These switches should be tested periodically to verify that they stop machine motion when needed.

Power Considerations

The following explains power considerations for the Micro800 controllers.

Isolation Transformers

You may want to use an isolation transformer in the AC line to the controller. This type of transformer provides isolation from your power distribution system to reduce the electrical noise that enters the controller, and is often used as a step-down transformer to reduce line voltage. Any transformer that is used with the controller must have a sufficient power rating for its load. The power rating is expressed in volt-amperes (VA).

Power Supply Inrush

During power-up, the Micro800 controller power supply allows a brief inrush current to charge internal capacitors. Many power lines and control transformers can supply inrush current for a brief time. If the power source cannot supply this inrush current, the source voltage may sag momentarily.

The only effect of limited inrush current and voltage sag on the Micro800 controller is that the power supply capacitors charge slower. However, the effect of a voltage sag on other equipment should be considered. For example, a deep voltage sag may reset a computer that is connected to the same power source.

The following considerations determine whether the power source must be required to supply high inrush current:

- The power-up sequence of devices in a system.
- The amount of the power source voltage sag if the inrush current cannot be supplied.
- The effect of voltage sag on other equipment in the system.

If the entire system is powered-up simultaneously, a brief sag in the power source voltage typically does not affect any equipment.

Loss of Power Source

The optional Micro800 controller AC power supply is designed to withstand brief power losses without affecting the operation of the system. The time the system is operational during power loss is called the program scan hold-up time after loss of power. The duration of the power supply hold-up time depends on the power consumption of the controller system, but is typically between 10 milliseconds and 3 seconds.

Input States on Power Down

The power supply hold-up time as described earlier is longer than the turn-on and turn-off times of the inputs. Because of this behavior, the controller may record the input state change from "On" to "Off" that occurs when power is removed before the power supply shuts down the system. It is important to understand this concept. You should write your program to account for this effect.

Other Types of Line Conditions

Occasionally, the power source to the system can be temporarily interrupted. It is also possible that the voltage level may drop substantially below the normal line voltage range for some time. Both of these conditions are considered to be a loss of power for the system.

Prevent Excessive Heat

For most applications, normal convective cooling keeps the controller within the specified operating range. Verify that the specified temperature range is maintained. Proper spacing of components within an enclosure is sufficient for heat dissipation.

In some applications, other equipment inside or outside the enclosure produces a substantial amount of heat. To help with air circulation and to reduce "hot spots" near the controller, place blower fans inside the enclosure.

Additional cooling provisions might be necessary when high ambient temperatures are encountered.

> Do not bring in unfiltered outside air. Place the controller in an enclosure to protect it from a corrosive atmosphere. Harmful contaminants or dirt can cause improper operation or damage to components. In extreme cases, you must use air conditioning to protect against heat build-up within the enclosure.

Master Control Relay

A hard-wired master control relay (MCR) provides a reliable means for emergency machine shutdown. Since the master control relay allows the placement of several emergency stop switches in different locations, its installation is important from a safety standpoint. Overtravel limit switches or mushroom-head push buttons are wired in series so that when any of them opens, the master control relay is de-energized. This removes power to input and output device circuits. See and .

> **WARNING:** Never alter these circuits to defeat their function since serious injury and/or machine damage can result.

> If you use an external DC power supply, interrupt the DC output side rather than the AC line side of the supply to avoid the additional delay of the power supply turn-off.
> The AC line of the DC output power supply should be fused.
> Connect a set of master control relays in series with the DC power supplying the input and output circuits.

Place the main power disconnect switch where the switch is quickly and easily accessible to operators and maintenance personnel. If you mount a disconnect switch inside the controller enclosure, place the switch operating handle on the outside of the enclosure, so that you can disconnect the power without opening the enclosure.

Whenever any of the emergency stop switches are opened, power to input and output devices should be removed.

When you use the master control relay to remove power from the external I/O circuits, power continues to be provided to the controller's power supply so that diagnostic indicators on the controller can still be observed.

The master control relay is not a substitute for a disconnect to the controller. It is intended for any situation where the operator must quickly de-energize I/O devices only. When you inspect or install terminal connections, replace output fuses, or work on equipment within the enclosure, use the disconnect to shut off power to the rest of the system.

> Do not control the master control relay with the controller. Provide the operator with the safety of a direct connection between an emergency stop switch and the master control relay.

Using Emergency Stop Switches

When using emergency stop switches, adhere to the following points:

- Do not program emergency stop switches in the controller program. Any emergency stop switch should turn off all machine power by turning off the master control relay.
- Observe all applicable local codes concerning the placement and labeling of emergency stop switches.
- Install emergency stop switches and the master control relay in your system. Verify that relay contacts have a sufficient rating for your application. Emergency stop switches must be easy to reach.
- In Figure 15 and Figure 16, input and output circuits are shown with MCR protection. However, in most applications, only output circuits require MCR protection.

Figure 15 and Figure 16 show the Master Control Relay that is wired in a grounded system.

> In most applications, input circuits do not require MCR protection; however, if you must remove power from all field devices, you must include MCR contacts in series with input power wiring.

Figure 15 - Schematic with IEC Symbols
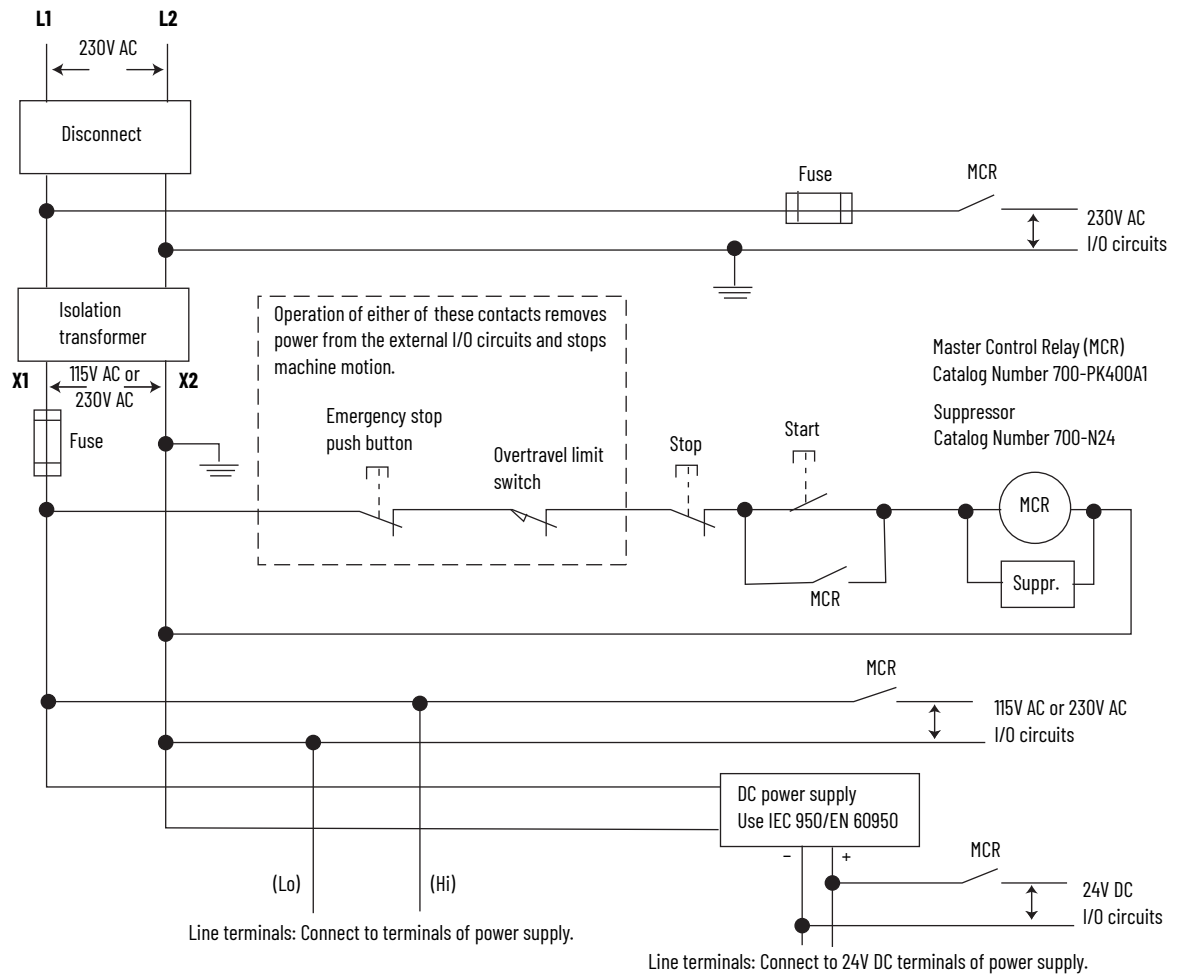
Figure 16 - Schematic with ANSI/CSA Symbols
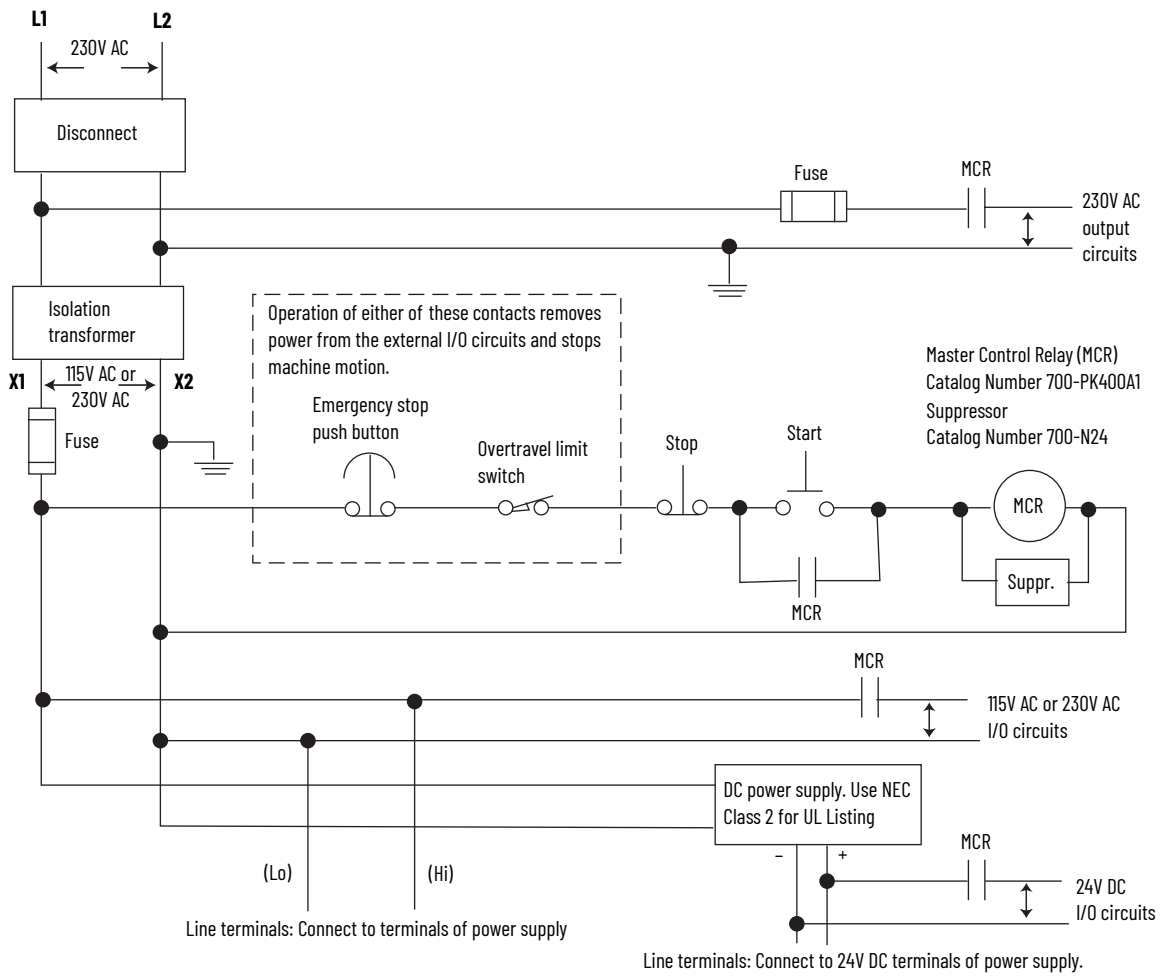
Install Your Controller

This chapter guides you on how to install the controller.

Controller Mounting Dimensions

Mounting Dimensions

Mounting dimensions exclude mounting feet or DIN rail latches.

Micro830 10-point and 16-point Controllers
2080-LC30-10QWB, 2080-LC30-10QVB, 2080-LC30-16AWB, 2080-LC30-16QWB, 2080-LC30-16QVB
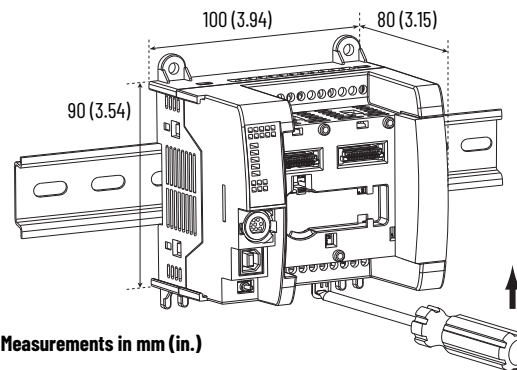
Measurements in mm (in.)

Micro830 24-point Controllers
2080-LC30-24QW8B, 2080-LC30-24QVB, 2080-LC30-24QBB

Measurements in mm (in.)

Micro830 48-point Controllers
2080-LC30-48AWB, 2080-LC30-48QWB, 2080-LC30-48QVB, 2080-LC30-48QBB

230 (9.06)   80 (3.15)

90 (3.54)

Measurements in mm (in.)

Micro850 24-point Controllers
2080-LC50-24AWB, 2080-LC50-24QWB, 2080-LC50-24QVB, 2080-LC50-24QBB,
2080-L50E-24AWB, 2080-L50E-24QWB, 2080-L50E-24QVB, 2080-L50E-24QBB

Micro870 24-point Controllers
2080-LC70-24AWB, 2080-LC70-24QWB, 2080-LC70-24QWBK, 2080-LC70-24QBB, 2080-LC70-24QBBK,
2080-L70E-24AWB, 2080-L70E-24QWB, 2080-L70E-24QWBK, 2080-L70E-24QWBN, 2080-L70E-24QWBNK,
2080-L70E-24QBB, 2080-L70E-24QBBK, 2080-L70E-24QBBN

158 (6.22)   80 (3.15)

90 (3.54)

Measurements in mm (in.)

Micro850 48-point Controllers
2080-LC50-48AWB, 2080-LC50-48QWB, 2080-LC50-48QWBK, 2080-LC50-48QVB, 2080-LC50-48QBB,
2080-L50E-48AWB, 2080-L50E-48QWB, 2080-L50E-48QWBK, 2080-L50E-48QVB, 2080-L50E-48QBB

238 (9.37)   80 (3.15)

90 (3.54)

Measurements in mm (in.)

Maintain spacing from objects such as enclosure walls, wireways, and adjacent equipment. Allow 50.8 mm (2 in.) of space on all sides for adequate ventilation. If optional accessories/modules are attached to the controller, such as the power supply 2080-PS120-240VAC or expansion I/O modules, make sure that there is 50.8 mm (2 in.) of space on all sides after attaching the optional parts.

DIN Rail Mounting

The module can be mounted using the following DIN rails: 35 x 7.5 x 1 mm (EN 50 022 - 35 x 7.5).

> For environments with greater vibration and shock concerns, use the panel mounting method, instead of DIN rail mounting.

Before mounting the module on a DIN rail, use a screwdriver in the DIN rail latch and pry it downwards until it is in the unlatched position.

1. Hook the top of the DIN rail mounting area of the controller onto the DIN rail, and then press the bottom until the controller snaps onto the DIN rail.

2. Push the DIN rail latch back into the latched position.
   Use DIN rail end anchors (Allen-Bradley® part number 1492-EAJ35 or 1492-EAHJ35) for vibration or shock environments.

To remove your controller from the DIN rail, pry the DIN rail latch downwards until it is in the unlatched position.

Panel Mounting

The preferred mounting method is to use four M4 (#8) screws per module. Hole spacing tolerance: ±0.4 mm (0.016 in.).

Follow these steps to install your controller using mounting screws.

1. Place the controller against the panel where you are mounting it. Make sure that the controller is spaced properly.

2. Mark the drilling holes through the mounting screw holes and mounting feet, then remove the controller.

3. Drill the holes at the markings, then replace the controller and mount it.
   Leave the protective debris strip in place until you are finished wiring the controller and any other devices.

IMPORTANT	For instructions on how to install your Micro800 system with expansion I/O, see the Micro800 Expansion I/O Modules User Manual, publication 2080-UM003.

Panel Mounting Dimensions

Micro830 10-Point and 16-Point Controllers
2080-LC30-10QWB, 2080-LC30-10QVB, 2080-LC30-16AWB, 2080-LC30-16QWB, 2080-LC30-16QVB
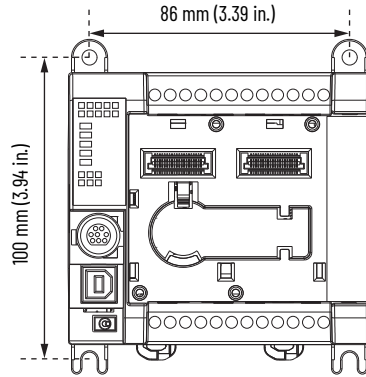
Micro830 24-Point Controllers
2080-LC30-24QWB, 2080-LC30-24QVB, 2080-LC30-24QBB

Micro850 24-point Controllers
2080-LC50-24AWB, 2080-LC50-24QWB, 2080-LC50-24QVB, 2080-LC50-24QBB,
2080-L50E-24AWB, 2080-L50E-24QWB, 2080-L50E-24QVB, 2080-L50E-24QBB

Micro870 24-point Controllers
2080-LC70-24AWB, 2080-LC70-24QWB, 2080-LC70-24QWBK, 2080-LC70-24QBB, 2080-LC70-24QBBK,
2080-L70E-24AWB, 2080-L70E-24QWB, 2080-L70E-24QWBK, 2080-L70E-24QWBN, 2080-L70E-24QWBNK,
2080-L70E-24QBB, 2080-L70E-24QBBK, 2080-L70E-24QBBN

Micro830 48-Point Controllers
2080-LC30-48AWB, 2080-LC30-48QWB, 2080-LC30-48QVB, 2080-LC30-48QBB

Micro850 48-point Controllers
2080-LC50-48AWB, 2080-LC50-48QWB, 2080-LC50-48QWBK, 2080-LC50-48QVB, 2080-LC50-48QBB
2080-L50E-48AWB, 2080-L50E-48QWB, 2080-L50E-48QWBK, 2080-L50E-48QVB, 2080-L50E-48QBB
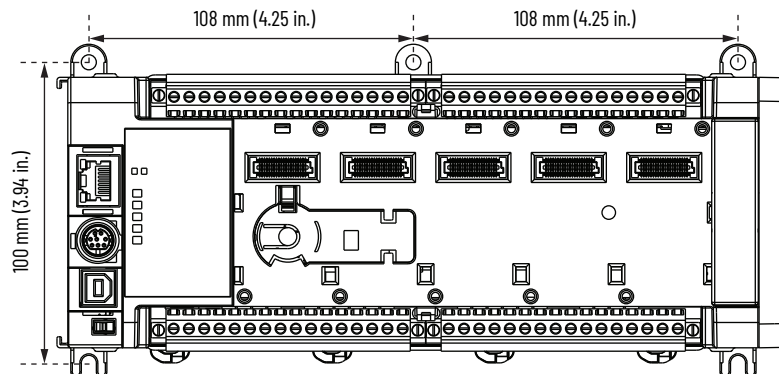
System Assembly

Micro830, Micro850, and Micro870 24-point Controllers (Front)

Micro830/Micro850/Micro870 24-pt controller with Micro800 power supply

Expansion I/O slots
(Applicable to Micro850 and Micro870 only)
Single-width (first slot)
Double-width (second slot)
2085-ECR (terminator)

Measurements in mm (in.)

Micro830, Micro850, and Micro870 24-point Controllers (Side)

Micro830/Micro850/Micro870 24-pt controller with Micro800 power supply

Expansion I/O slots
(Applicable to Micro850 and Micro870 only)
Single-width (first slot)
Double-width (second slot)
2085-ECR (terminator)

Measurements in mm (in.)

Micro830 and Micro850 48-point Controllers (Front)

Micro830/Micro850 48-pt controller with Micro800 power supply

Expansion I/O slots
(Applicable to Micro850 only)
Single-width (first slot)
Double-width (second slot)
2085-ECR (terminator)

Measurements in mm (in.)

Micro830 and Micro850 48-point Controllers (Side)

Micro830/Micro850 48-pt controller with Micro800 power supply

Expansion I/O Slots
(Applicable to Micro850 only)
Single-width (first slot)
Double-width (second slot)
2085-ECR (terminator)

Measurements in mm (in.)

Wire Your Controller

This chapter provides information on the Micro830, Micro850, and Micro870 controller wiring requirements.

Wiring Requirements and Recommendation

⚠️ **WARNING:** Before you install and wire any device, disconnect power to the controller system.

⚠️ **WARNING:** Calculate the maximum possible current in each power and common wire. Observe all electrical codes dictating the maximum current allowable for each wire size. Current above the maximum ratings may cause wiring to overheat, which can cause damage.
United States Only: If the controller is installed within a potentially hazardous environment, all wiring must comply with the requirements that are stated in the National Electrical Code 501-10 (b).

- Allow for at least 50 mm (2 in.) between I/O wiring ducts or terminal strips and the controller.
- Route incoming power to the controller by a path separate from the device wiring. Where paths must cross, their intersection should be perpendicular.

💡 Do not run signal or communications wiring and power wiring in the same conduit. Wires with different signal characteristics should be routed by separate paths.

- Separate wiring by signal type. Bundle wiring with similar electrical characteristics together.
- Separate input wiring from output wiring.
- Label wiring to all devices in the system. Use tape, shrink-tubing, or other dependable means for labeling purposes. In addition to labeling, use colored insulation to identify wiring based on signal characteristics. For example, you may use blue for DC wiring and red for AC wiring.

Table 9 - Wire Requirements

Controller	Wire Type	Wire Size		Rating
		Min	Max	
Micro830, Micro850, and Micro870 controllers	Solid	0.2 mm^2 (24 AWG)	2.5 mm^2 (12 AWG)	Rated @ 90 °C (194 °F) insulation max
	Stranded	0.2 mm^2 (24 AWG)	2.5 mm^2 (12 AWG)	

Use Surge Suppressors

Because of the potentially high current surges that occur when switching inductive load devices, such as motor starters and solenoids, the use of some type of surge suppression to protect and extend the operating life of the controllers output contacts is required. Switching inductive loads without surge suppression can significantly reduce the life expectancy of relay contacts. By adding a suppression device directly across the coil of an inductive device, you prolong the life of the output or relay contacts. You also reduce the effects of voltage transients and electrical noise from radiating into adjacent systems.

Figure 17 shows an output with a suppression device. We recommend that you locate the suppression device as close as possible to the load device.

Figure 17 - Output with Suppression Device
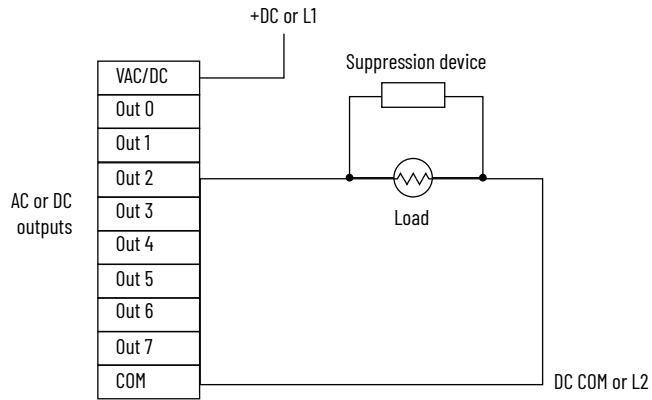
If the outputs are DC, we recommend that you use an 1N4004 diode for surge suppression, as shown in Figure 18. For inductive DC load devices, a diode is suitable. A 1N4004 diode is acceptable for most applications. A surge suppressor can also be used. See Recommended Surge Suppressors on page 47. These surge suppression circuits connect directly across the load device.

Figure 18 - DC Outputs with Surge Suppression
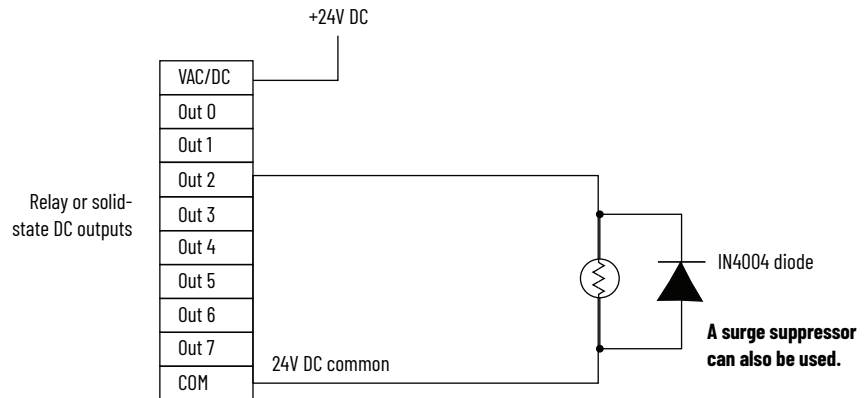
Suitable surge suppression methods for inductive AC load devices include a varistor, an RC network, or an Allen-Bradley® surge suppressor, which is shown in Figure 19. These components must be appropriately rated to suppress the switching transient characteristic of the particular inductive device. See Recommended Surge Suppressors on page 47 for recommended suppressors.

Figure 19 - Surge Suppression for Inductive AC Load Devices
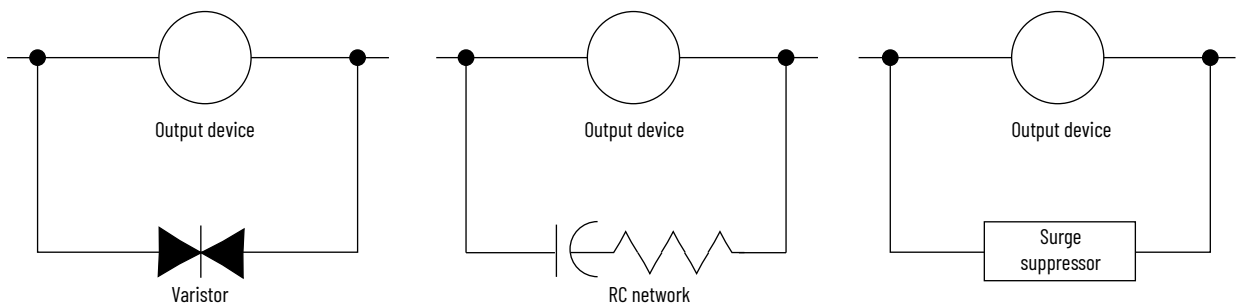
Recommended Surge Suppressors

Use the Allen-Bradley surge suppressors shown in [Table 10](#) with relays, contactors, and starters.

Table 10 - Recommended Surge Suppressors

Device	Coil Voltage	Suppressor Catalog Number	Type[1]
Bulletin 100/104K 700K	24...48V AC	100-KFSC50	RC
	110...280V AC	100-KFSC280	
	380...480V AC	100-KFSC480	
	12...55V AC, 12...77V DC	100-KFSV55	MOV
	56...136V AC, 78...180V DC	100-KFSV136	
	137...277V AC, 181...250 V DC	100-KFSV277	
	12...250V DC	100-KFSD250	Diode
Bulletin 100C (C09...C97)	24...48V AC	100-FSC48[2]	RC
	110...280V AC	100-FSC280[1]	
	380...480V AC	100-FSC480[1]	
	12...55V AC, 12...77V DC	100-FSV55[1]	MOV
	56...136V AC, 78...180V DC	100-FSV136[1]	
	137...277V AC, 181...250V DC	100-FSV277[1]	
	278...575V AC	100-FSV575[1]	
	12...250V DC	100-FSD250[1]	Diode
Bulletin 509 Motor Starter Size 0...5	12...120V AC	599-K04	MOV
	240...264V AC	599-KA04	
Bulletin 509 Motor Starter Size 6	12...120V AC	199-FSMA1[3]	RC
	12...120V AC	199-GSMA1[4]	MOV
Bulletin 700 R/RM Relay	AC coil	Not Required	
	24...48V DC	199-FSMA9	MOV
	50...120V DC	199-FSMA10	
	130...250V DC	199-FSMA11	
Bulletin 700 Type N, P, PK, or PH Relay	6...150V AC/DC	700-N24	RC
	24...48V AC/DC	199-FSMA9	MOV
	50...120V AC/DC	199-FSMA10	
	130...250V AC/DC	199-FSMA11	
	6...300V DC	199-FSMZ-1	Diode
Miscellaneous electromagnetic devices limited to 35 sealed VA	6...150V AC/DC	700-N24	RC

(1)   Do not use RC Type with Triac outputs. Varistor is not recommended for use on the relay outputs.
(2)   Catalog numbers for screwless terminals include the string 'CR' after '100-'.
       For example: Catalog number 100-FSC48 becomes 100-**CR**FSC48; Catalog number 100-FSV55 becomes 100-**CR**FSV55; and so on.
(3)   For use on the interposing relay.
(4)   For use on the contactor or starter.

Grounding the Controller

This product is intended to be mounted to a well grounded mounting surface such as a metal panel. See the Industrial Automation Wiring and Grounding Guidelines, publication [1770-4.1](#), for additional information.

> ⚠ **WARNING:** All devices connected to the RS-232/RS-485 communication port must be referenced to controller ground, or be floating (not referenced to a potential other than ground). Failure to follow this procedure may result in property damage or personal injury.

Wiring Diagrams

The following illustrations show the wiring diagrams for the Micro800 controllers. Controllers with DC inputs can be wired as either sinking or sourcing inputs. Sinking and sourcing does not apply to AC inputs.

High-speed inputs and outputs are indicated by ◯.

2080-LC30-10QWB

Input terminal block

COM0		I-01		I-03		I-04		NC		NC	
1	2	3	4	5	6	7	8	9	10	11	12
	I-00		I-02		COM1		I-05		NC		NC

Output terminal block

+DC24		CM0		CM1		CM2		CM3		NC	
1	2	3	4	5	6	7	8	9	10	11	12
	-DC24		O-00		O-01		O-02		O-03		NC

2080-LC30-10QVB

Input terminal block

COM0		I-01		I-03		I-04		NC		NC	
1	2	3	4	5	6	7	8	9	10	11	12
	I-00		I-02		COM1		I-05		NC		NC

Output terminal block

+DC24		+CM0		O-01		+CM1		O-03		NC	
1	2	3	4	5	6	7	8	9	10	11	12
	-DC24		O-00		-CM0		O-02		-CM1		NC

2080-LC30-16AWB, 2080-LC30-16QWB

Input terminal block

COM0		I-01		I-03		I-04		I-06		I-08	
1	2	3	4	5	6	7	8	9	10	11	12
	I-00		I-02		COM1		I-05		I-07		I-09

Output terminal block

+DC24		CM0		CM1		CM2		CM3		O-04	
1	2	3	4	5	6	7	8	9	10	11	12
	-DC24		O-00		O-01		O-02		O-03		O-05

💡 2080-LC30-16AWB has no high-speed inputs.

2080-LC30-16QVB

Input terminal block

COM0		I-01		I-03		I-04		I-06		I-08	
1	2	3	4	5	6	7	8	9	10	11	12
	I-00		I-02		COM1		I-05		I-07		I-09

Output terminal block

+DC24		+CM0		O-01		+CM1		O-03		O-04	
1	2	3	4	5	6	7	8	9	10	11	12
	-DC24		O-00		-CM0		O-02		-CM1		O-05
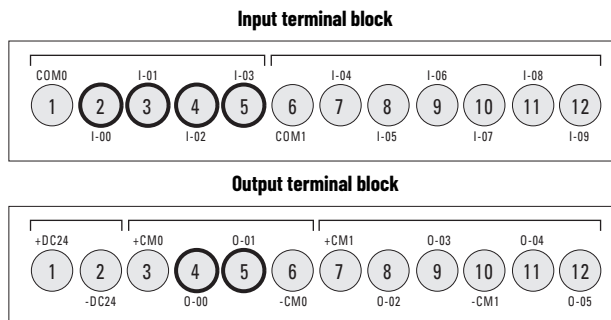
2080-LC30-24QWB, 2080-LC50-24AWB, 2080-LC50-24QWB, 2080-L50E-24AWB, 2080-L50E-24QWB, 2080-LC70-24AWB, 2080-LC70-24QWB, 2080-LC70-24QWBK, 2080-L70E-24AWB, 2080-L70E-24QWB, 2080-L70E-24QWBK, 2080-L70E-24QWBN, 2080-L70E-24QWBNK

Input terminal block

Output terminal block
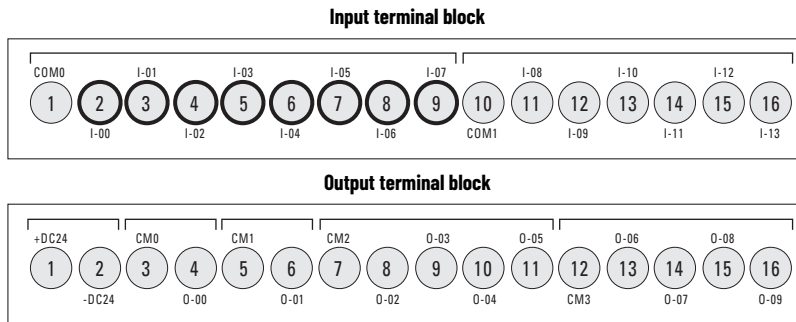
2080-LC30-24QWB, 2080-LC50-24QWB, 2080-L50E-24QWB, 2080-LC70-24QWB, 2080-LC70-24QWBK, 2080-L70E-24QWB, 2080-L70E-24QWBK, 2080-L70E-24QWBN, 2080-L70E-24QWBNK, DC Input Configuration

IMPORTANT	• Do not connect –DC24 (Output terminal 2) to Earth/Chassis Ground.
	• In Micro870 systems that use more than four Micro800 Expansion I/O modules, we recommend using a 1606-XLP60EQ power supply instead of a 2080-PS120-240VAC power supply. Make sure to wire both the Micro870 controller and the 2085-EP24VDC expansion power supply to the same 1606-XLP60EQ power supply.
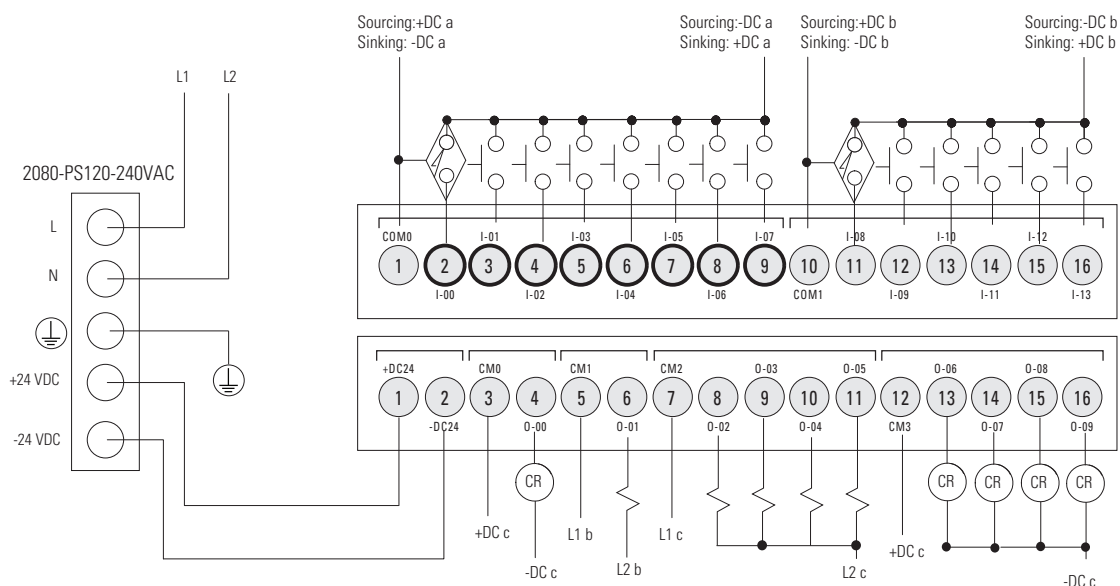
**2080-LC50-24AWB, 2080-L50E-24AWB, 2080-LC70-24AWB, 2080-L70E-24AWB,
DC Input Configuration**
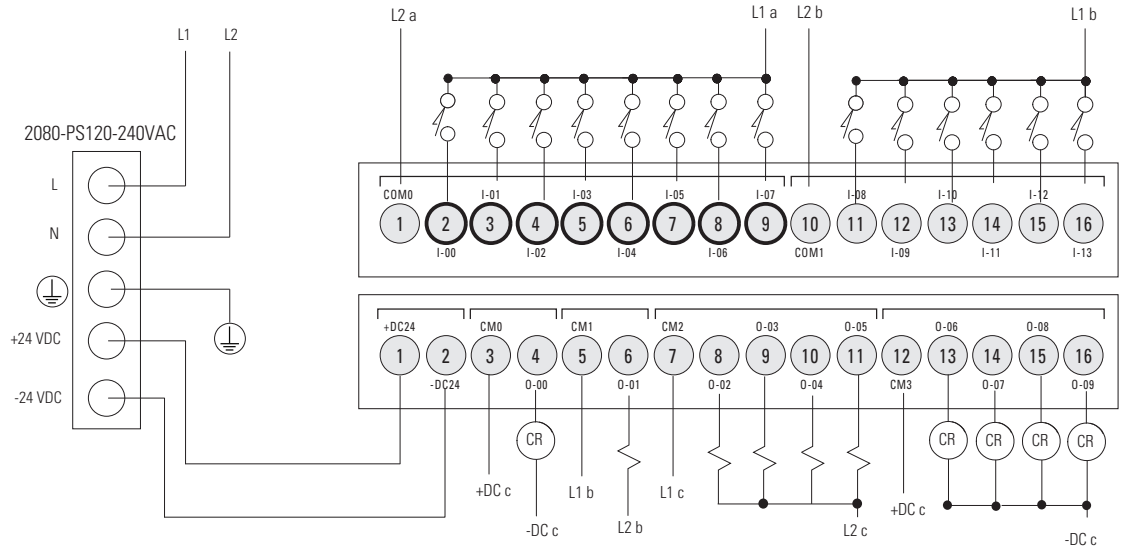
IMPORTANT   Do not connect –DC24 (Output terminal 2) to Earth/Chassis Ground.

**2080-LC30-24QVB, 2080-LC30-24QBB, 2080-LC50-24QVB, 2080-LC50-24QBB, 2080-L50E-24QVB,
2080-L50E-24QBB, 2080-LC70-24QBB, 2080-LC70-24QBBK, 2080-L70E-24QBB, 2080-L70E-24QBBK,
2080-L70E-24QBBN**

Input terminal block

Output terminal block
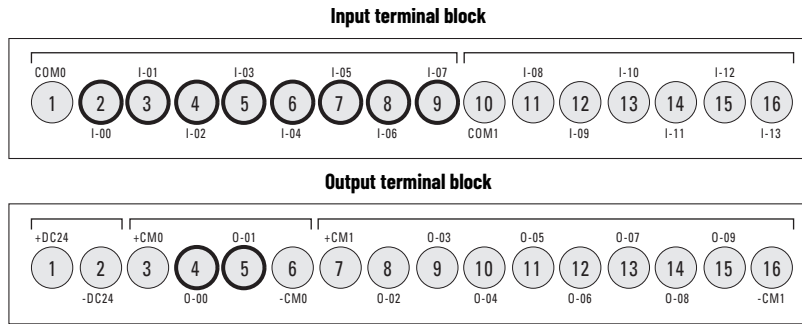
**2080-LC30-24QBB, 2080-LC50-24QBB, 2080-L50E-24QBB, 2080-LC70-24QBB, 2080-LC70-24QBBK,
2080-L70E-24QBB, 2080-L70E-24QBBK, 2080-L70E-24QBBN,
DC Input Configuration**
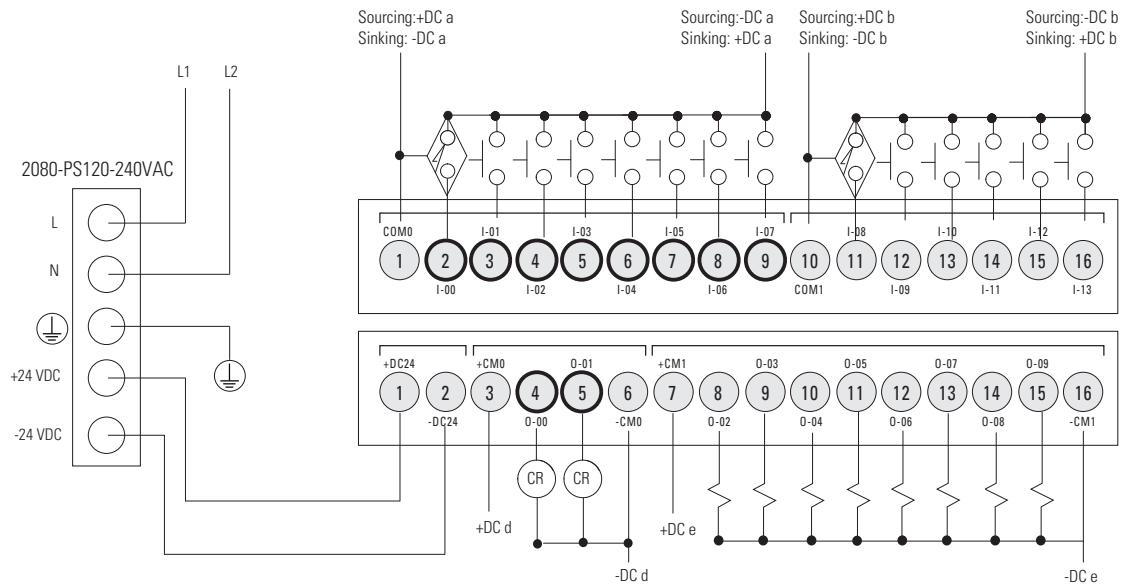
IMPORTANT   • Do not connect –DC24 (Output terminal 2) to Earth/Chassis Ground.
 • In Micro870 systems that use more than four Micro800 Expansion I/O modules, we recommend using a 1606-XLP60EQ power supply instead of a 2080-PS120-240VAC power supply. Make sure to wire both the Micro870 controller and the 2085-EP24VDC expansion power supply to the same 1606-XLP60EQ power supply.

2080-LC30-24QVB, 2080-LC50-24QVB, 2080-L50E-24QVB, DC Input Configuration
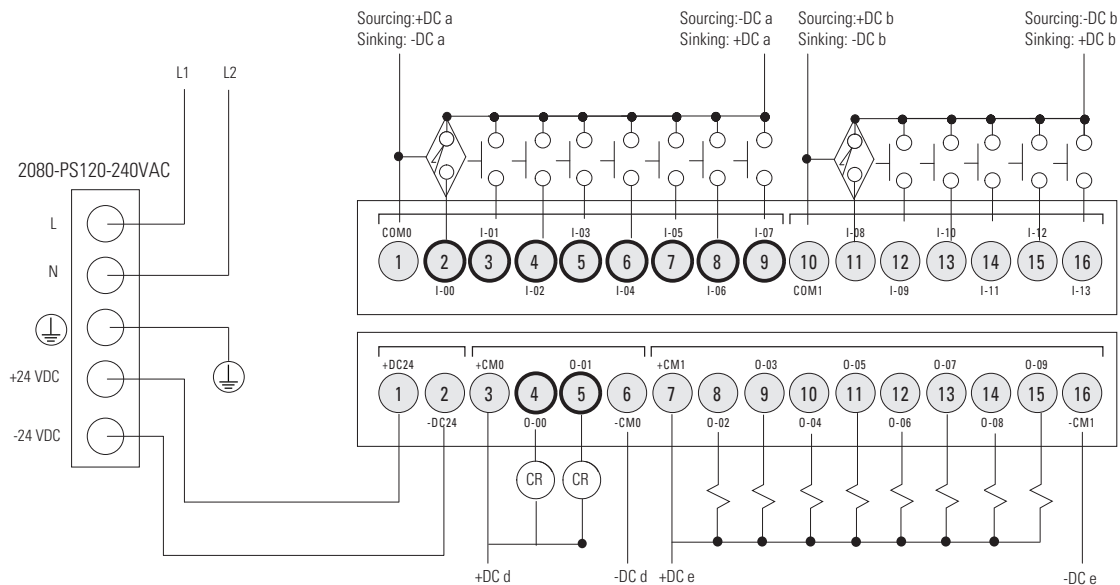
IMPORTANT   Do not connect –DC24 (Output terminal 2) to Earth/Chassis Ground.

2080-LC30-48AWB, 2080-LC30-48QWB, 2080-LC50-48AWB, 2080-LC50-48QWB, 2080-LC50-48QWBK, 2080-L50E-48AWB, 2080-L50E-48QWB, 2080-L50E-48QWBK

Input terminal blocks

Output terminal blocks

2080-LC30-48AWB has no high-speed inputs.

2080-LC30-48QVB, 2080-LC30-48QBB, 2080-LC50-48QVB, 2080-LC50-48QBB, 2080-L50E-48QVB, 2080-L50E-48QBB

Input terminal blocks
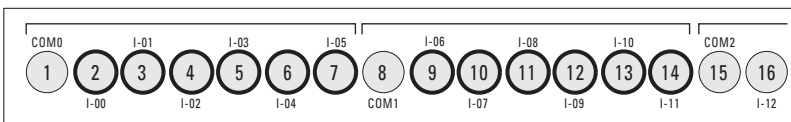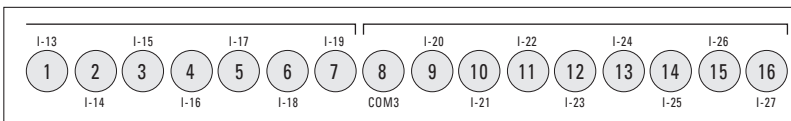
TERMINAL BLOCK 1

TERMINAL BLOCK 3

Output terminal blocks
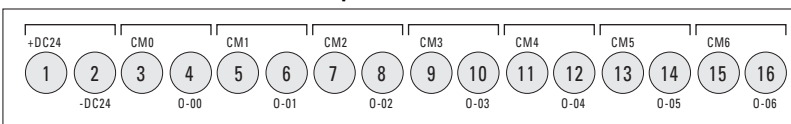
TERMINAL BLOCK 2

TERMINAL BLOCK 4
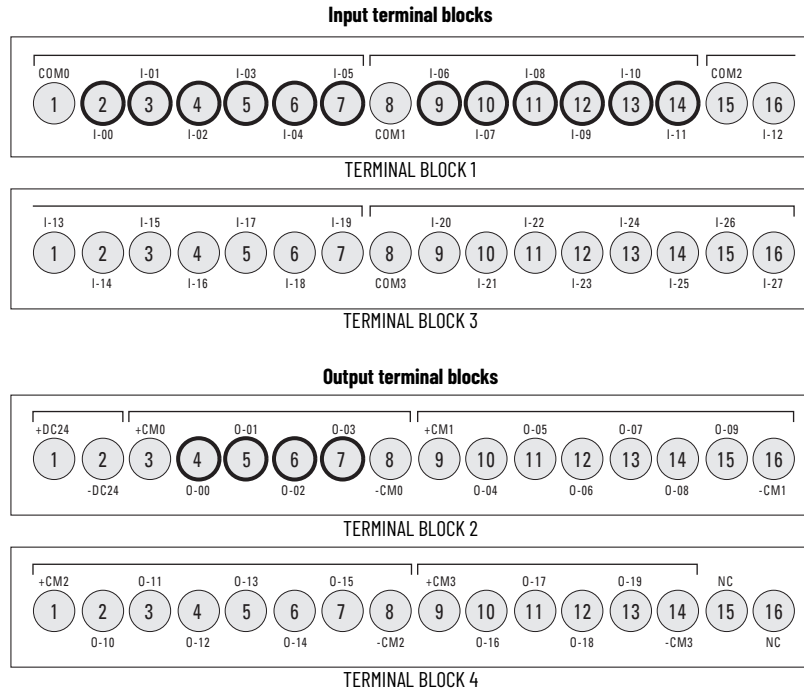
Controller I/O Wiring

This section contains some relevant information about minimizing electrical noise and also includes some wiring examples.

Minimize Electrical Noise

Because of the variety of applications and environments where controllers are installed and operating, it is impossible to achieve the removal of all environmental noise with input filters. To help reduce the effects of environmental noise, install the Micro800 system in a properly rated (for example, NEMA) enclosure. Make sure that the Micro800 system is properly grounded.

A system may malfunction due to a change in the operating environment after a period of time. We recommend that you periodically check the system operation, particularly when new machinery or other noise sources are installed near the Micro800 system.

Analog Channel Wiring Guidelines

Consider the following when wiring your analog channels:
- The analog common (COM) is not electrically isolated from the system, and is connected to the power supply common.
- Analog channels are not isolated from each other.
- Use Belden cable #8761, or equivalent, shielded wire.
- Under normal conditions, the drain wire (shield) should be connected to the metal mounting panel (earth ground). Keep the shield connection to earth ground as short as possible.
- To achieve optimum accuracy for voltage type inputs, limit overall cable impedance by keeping all analog cables as short as possible. Locate the I/O system as close to your voltage type sensors or actuators as possible.

Minimize Electrical Noise on Analog Channels

Inputs on analog channels employ digital high-frequency filters that significantly reduce the effects of electrical noise on input signals. However, because of the variety of applications and environments where analog controllers are installed and operated, it is impossible to achieve the removal of all environmental noise with input filters.

Several specific steps can be taken to help reduce the effects of environmental noise on analog signals:

- Install the Micro800 system in a properly rated enclosure, for example, NEMA. Make sure that the shield is properly grounded.
- Use Belden cable #8761 for wiring the analog channels, making sure that the drain wire and foil shield are properly grounded.
- Route the Belden cable separately from any AC wiring. Additional noise immunity can be obtained by routing the cables in a grounded conduit.

Grounding Your Analog Cable

Use shielded communication cable (Belden #8761). The Belden cable has two signal wires (black and clear), one drain wire, and a foil shield. The drain wire and foil shield must be grounded at one end of the cable.
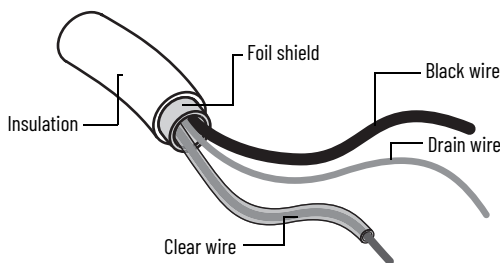
| IMPORTANT | Do not ground the drain wire and foil shield at both ends of the cable. |

Wiring Examples

Examples of sink/source, input/output wiring are shown as follows.

Figure 20 - Sink Output Wiring Example
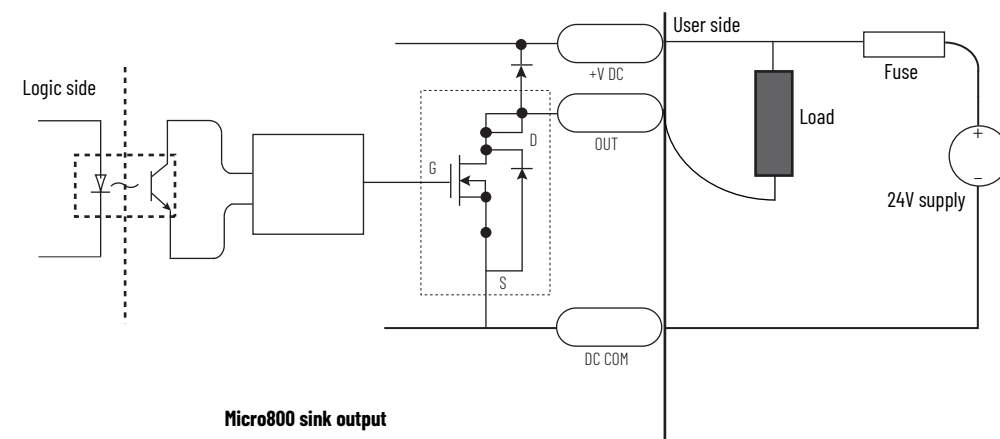
Figure 21 - Sink Input Wiring Example
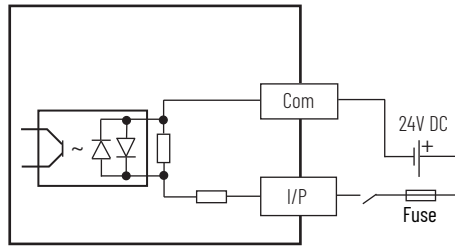
Figure 22 - Source Output Wiring Example

Micro800 source output
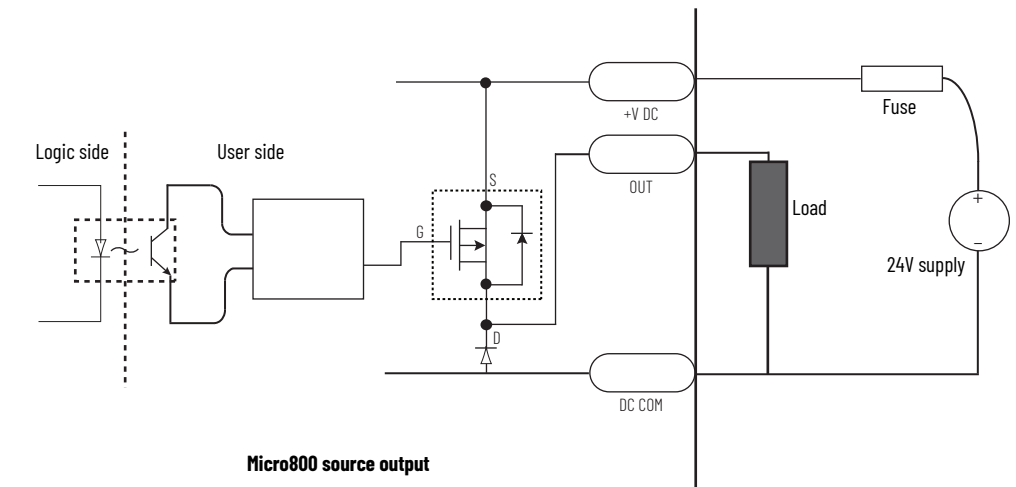
Figure 23 - Source Input Wiring Example
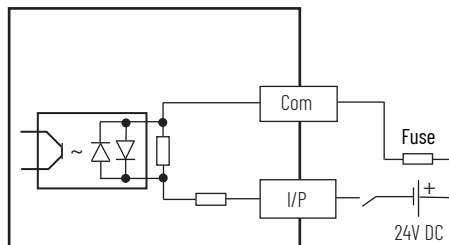
Embedded Serial Port Wiring

The embedded serial port is a non-isolated RS-232/RS-485 serial port, which is targeted to be used for short distances (<3 m [10 ft]) to devices such as HMIs.

See Embedded Serial Port Cables on page 22 for a list of cables that you can use with the embedded serial port 8-pin Mini DIN connector.

For example, the 1761-CBL-PM02 cable is typically used to connect the embedded serial port to PanelView™ 800 HMI using RS-232.

Figure 24 - Embedded Serial Port
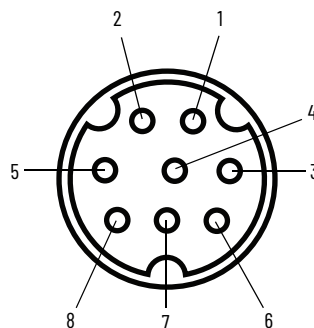
Table 11 - Pinout Explanations

Pin	Definition	RS-485 Example	RS-232 Example
1	RS-485+	B(+)	(not used)
2	GND	GND	GND
3	RS-232 RTS	(not used)	RTS
4	RS-232 RxD	(not used)	RxD
5	RS-232 DCD	(not used)	DCD
6	RS-232 CTS	(not used)	CTS
7	RS-232 TxD	(not used)	TxD
8	RS-485-	A(-)	(not used)

IMPORTANT	• Do not connect the GND pin of the serial port to Earth/Chassis Ground. The GND pin of the serial port is the DC common for the Serial Port Communication signals and is not intended for Shield Ground. • If the length of the serial cable is more than 3 meters (10 feet), use an isolated serial port, catalog number 2080-SERIALISOL.

Notes:

Communication Connections

Overview

This chapter describes how to communicate with your control system and configure communication settings. The method that you use and cabling that is required to connect your controller depends on what type of system you are employing. This chapter also describes how the controller establishes communication with the appropriate network.

The Micro830, Micro850, and Micro870 controllers have the following embedded communication channels:

- A non-isolated RS-232/RS-485 combo port
- A non-isolated USB programming port

In addition, the Micro850 and Micro870 controllers have an RJ45 Ethernet port.

Supported Communication Protocols

Micro830, Micro850, and Micro870 controllers support communication through the embedded RS-232/RS-485 serial port and any installed serial port plug-in modules. In addition, Micro850 and Micro870 controllers support communication through the embedded Ethernet port, and can be connected to a local area network for various devices providing 10 Mbps/100 Mbps transfer rate.

Micro830, Micro850, and Micro870 controllers support the following communication protocols:

- Modbus RTU Master and Slave
- CIP Serial Client/Server (DF1)
- CIP Symbolic Client/Server
- ASCII
- DNP3 [1]

Only Micro850 and Micro870 controllers support the following communication protocols:

- EtherNet/IP Client/Server
- Modbus TCP Client/Server
- DHCP Client
- Sockets Client/Server TCP/UDP

Table 12 - Connection Limits for Micro830/Micro850/Micro870 Controllers

Description		Micro830	Micro850/ Micro870
CIP Connections			
Total number of client plus server connections for all ports		16	24
Maximum number of client connections for all ports		15	16
Maximum number of server connections for all ports		16	24
Maximum number of EtherNet/IP connections	Client	—	16
	Server		23
Maximum number of USB connections	Client	—	—
	Server	15	23

(1)   DNP3 is only supported on 2080-L70E-24*QxB*N controllers.

Table 12 - Connection Limits for Micro830/Micro850/Micro870 Controllers (Continued)

Description		Micro830	Micro850/ Micro870
Maximum number of Serial connections	Client	15	16
	Server	15	23
TCP Connections			
Total number of client plus server connections			64
Maximum number for EtherNet/IP	Client		16
	Server	–	16
Maximum number for Modbus TCP	Client		16
	Server		16
Maximum number for User Programmable Sockets			8
User Programmable Sockets			
Total number of User Programmable Sockets (any combination of UDP plus TCP Client/Server)		–	8

> **IMPORTANT**   If all client/server connections are fully loaded, performance may be affected, such as data loss and intermittent delays during communication.

Here are some configuration examples, which are based on the limits that are described in the table above:

1. The maximum number of drives that can be controlled over EtherNet/IP is 16. This is due to the maximum limit of TCP Client connections being 16, and the maximum limit of EtherNet/IP Client connections being 16.

2. If you have 10 devices that are controlled over EtherNet/IP, the maximum number of devices that can be controlled over serial is six. This is due to the maximum limit of Client connections being 16.

3. The total number of UDP sockets plus TCP Client/Server sockets has a maximum limit of eight.

Modbus RTU

Modbus is a half-duplex, master-slave communications protocol. The Modbus network master reads and writes bits and registers. Modbus protocol allows one master to communicate with a maximum of 247 slave devices. Micro800 controllers support Modbus RTU Master and Modbus RTU Slave protocol. For more information on configuring your Micro800 controller for Modbus protocol, see the Connected Components Workbench Online Help. For more information about the Modbus protocol, see Modbus Protocol Specifications available from www.modbus.org.

> Use the MSG_MODBUS instruction to send Modbus messages over the serial port.

CIP Serial Client/Server – DF1

CIP Serial Client/Server allows CIP protocol to be used over a serial port. It is typically used with modems. The advantage over non-CIP serial protocols is that since the protocol is CIP, program downloads are supported including CIP pass-through from the serial port to Ethernet.

ASCII

ASCII provides connection to other ASCII devices, such as barcode readers, weigh scales, serial printers, and other intelligent devices. You can use ASCII by configuring the embedded or any plug-in serial RS-232/RS-485 port for the ASCII driver. See the Connected Components Workbench Online Help for more information.

To configure the serial port for ASCII, see .

Modbus TCP Client/Server

The Modbus TCP Client/Server communication protocol uses the same Modbus mapping features as Modbus RTU, but instead of the serial port, it is supported over Ethernet. Modbus TCP Server takes on Modbus Slave features on Ethernet.

No protocol configuration is required other than configuring the Modbus mapping table. For information on Modbus mapping, see .

> Use MSG_MODBUS2 instruction to send Modbus TCP messages over Ethernet port.

With Connected Components Workbench software version 12 or later, the Modbus TCP Server is disabled by default. If you want to use Modbus TCP, you can enable it from the Ethernet settings.
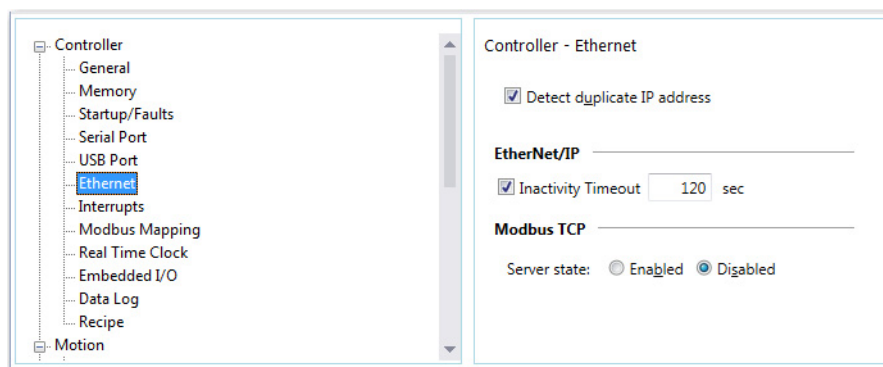
CIP Symbolic Client/Server

CIP Symbolic is supported by any CIP-compliant interface including Ethernet (EtherNet/IP) and serial port (CIP Serial). This protocol allows HMIs to connect easily to the Micro830, Micro850, and Micro870 controllers.

Micro850 and Micro870 controllers support up to 16 simultaneous EtherNet/IP client connections and 23 simultaneous EtherNet/IP Server connections.

CIP Serial, supported on Micro830, Micro850, and Micro870 controllers, uses the DF1 Full-duplex protocol, which provides point-to-point connection between two devices.

DF1 Half-duplex Master, DF1 Half-duplex Slave, and DF1 Radio Modem are supported in Micro850 (2080-L50E) and Micro870 (2080-L70E) controllers. These protocols provide connection to multiple devices over RS-485 or radio modems.

The Micro800 controllers support the protocol through RS-232 connection to external devices, such as computers running RSLinx® Classic software, PanelView™ Component terminals (firmware revisions 1.70 and above), PanelView 800 terminals, or other controllers that support CIP Serial over DF1 Full-Duplex, such as ControlLogix® and CompactLogix™ controllers that have embedded serial ports. Bulletins 2080-L50E and 2080-L70E also support the DF1 Half-duplex and Radio Modem protocols.

EtherNet/IP, supported on the Micro850 and Micro870 controller, uses the standard Ethernet TCP/IP protocol.

The Micro850 and Micro870 controller supports up to 23 simultaneous EtherNet/IP Server connections.

CIP Symbolic Addressing

You can access any global variables through CIP Symbolic addressing except for system and reserved variables.

One- or two-dimension arrays for simple data types are supported (for example, ARRAY OF INT[1...10, 1...10]) are supported but arrays of arrays (for example, ARRAY OF ARRAY) are not supported. Arrays of strings are also supported.

Table 13 - Supported Data Types in CIP Symbolic

Data Type[1]	Description
BOOL	Logical Boolean with values TRUE(1) and FALSE(0) (Uses up 8 bits of memory)
SINT	Signed 8-bit integer value
INT	Signed 16-bit integer value
DINT	Signed 32-bit integer value
LINT[2]	Signed 64-bit integer value
USINT	Unsigned 8-bit integer value
UINT	Unsigned 16-bit integer value
UDINT	Unsigned 32-bit integer value
ULINT[2]	Unsigned 64-bit integer value
REAL	32-bit floating point value
LREAL[2]	64-bit floating point value
STRING	character string (1 byte per character)
DATE[3]	Unsigned 32-bit integer value
TIME[3]	Unsigned 32-bit integer value

(1) Logix MSG instruction can read/write SINT, INT, DINT, LINT, and REAL data types using "CIP Data Table Read" and "CIP Data Table Write" message types.
     BOOL, USINT, UINT, UDINT, ULINT, LREAL, STRING, SHORT_STRING, DATE, and TIME data types are not accessible with the Logix MSG instruction.
(2) Not supported in PanelView Component or PanelView 800 terminals.
(3) Can be used by sending data to UDINT, mainly for use with PanelView Plus and PanelView 800 terminals.

CIP Client Messaging

CIP Generic and CIP Symbolic messages are supported on Micro800 controllers through the Ethernet and serial ports. The MSG_CIPSYMBOLIC and MSG_CIPGENERIC function blocks enable these client messaging features.

For more information and sample quick start project to help you use the CIP Client Messaging feature, see Micro800 Programmable Controllers: Getting Started with CIP Client Messaging, publication 2080-QS002.

Sockets Client/Server TCP/UDP

Sockets protocol is used for Ethernet communications to devices that do not support Modbus TCP and EtherNet/IP. Sockets support client and server, and TCP and UDP. Typical applications include communicating to printers, barcode readers, and PCs.

CIP Communications Pass-thru

The Micro830, Micro850, and Micro870 controllers support pass-thru on any communications port that supports Common Industrial Protocol (CIP) for applications such as program download. It does not support applications that require dedicated connections such as HMI. Micro830, Micro850, and Micro870 controllers support a maximum of one hop. A hop is defined to be an intermediate connection or communications link between two devices – in Micro800, the hop is through EtherNet/IP or CIP Serial or CIP USB.

Examples of Supported Architectures

USB to EtherNet/IP

For program download

You can download a program from the PC to controller1 over USB. Also, the program can be downloaded to controller2 and controller3 over USB to EtherNet/IP.
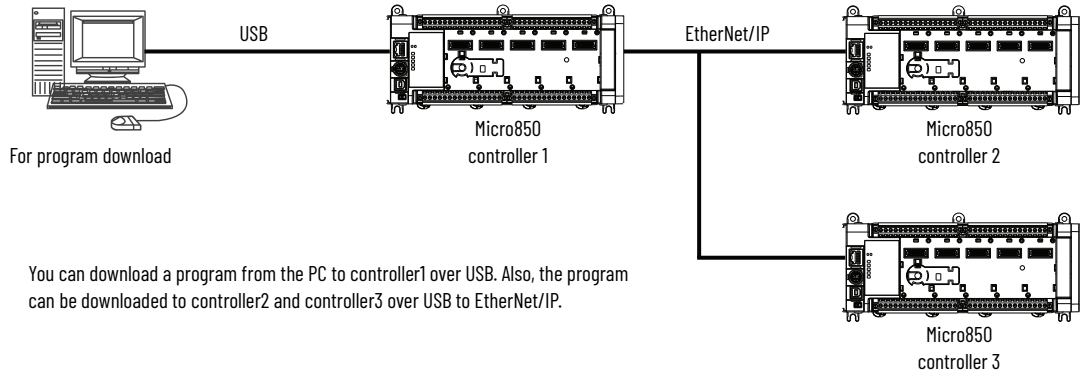
EtherNet/IP to CIP Serial

For program download

USB to DeviceNet

For program download
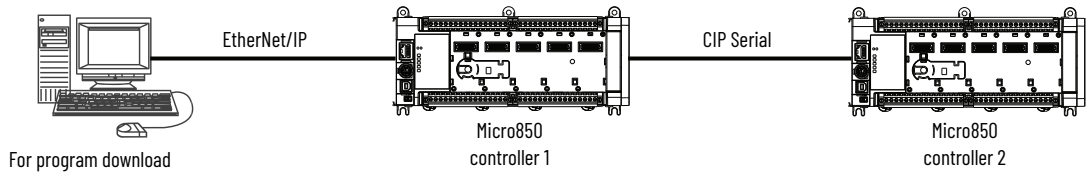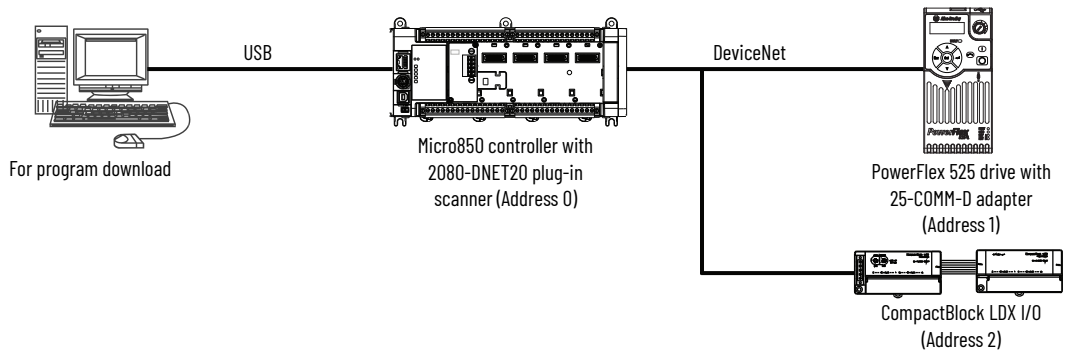
EtherNet/IP to DeviceNet

For program download

You can use Connected Components Workbench software to configure the PowerFlex drives.
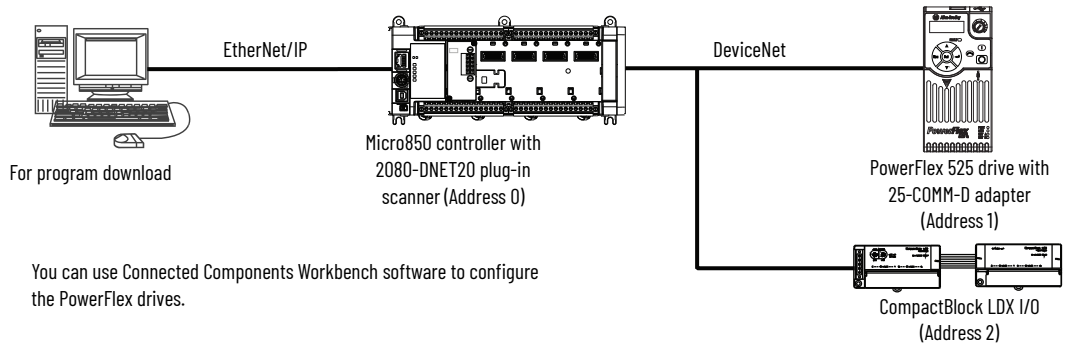
> **IMPORTANT**   Micro800 controllers do not support multiple hops (for example, from EtherNet/IP > CIP Serial > EtherNet/IP).

Use Modems with Micro800 Controllers

Serial modems can be used with the Micro830, Micro850, and Micro870 controllers.

Making a DF1 Point-to-point Connection

You can connect the Micro830, Micro850, and Micro870 programmable controller to your serial modem using an Allen-Bradley null modem serial cable (1761-CBL-PM02) to the controller's embedded serial port together with a 9-pin null modem adapter – a null modem with a null modem adapter is equivalent to a modem cable. The recommended protocol for this configuration is CIP Serial.

Construct Your Own Modem Cable

If you construct your own modem cable, the maximum cable length is 15.24 m (50 ft) with a 25-pin or 9-pin connector. shows the typical pinout for constructing a straight-through cable.

Figure 25 - Straight-through Cable Pinout Guide
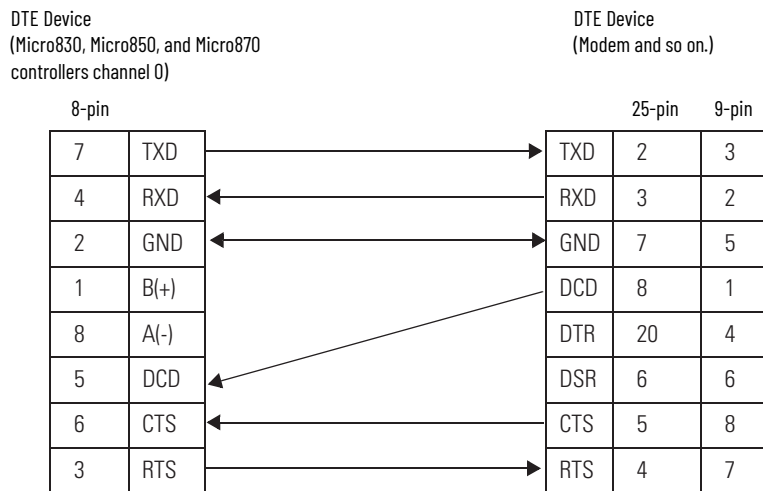
Configure Serial Port

You can configure the serial port driver as CIP Serial, Modbus RTU, ASCII, or Shutdown through the Device Configuration tree in the Connected Components Workbench software.

Configure CIP Serial Driver

1. Open your Connected Components Workbench project. On the device configuration tree, go to the Controller properties. Click Serial Port.

```
Controller
    General
    Memory
    Serial Port
    USB Port
    Ethernet
        Internet Protocol
        Port Settings
        Port Diagnostics
    Date and Time
    Interrupts
    Startup/Faults
    Modbus Mapping
    Embedded I/O
Plug-In Modules
    < Empty >
    < Empty >
    < Empty >
Expansion Modules
    < Empty >
    < Empty >
    < Empty >
    < Empty >
```
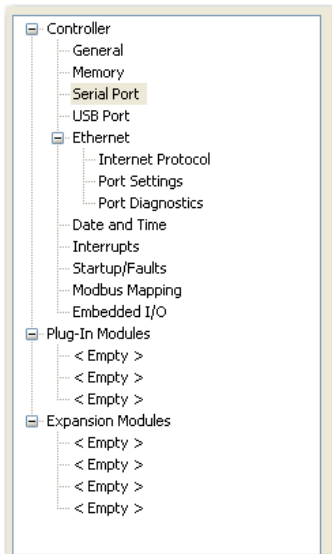
2. Select CIP Serial from the Driver field.

Controller - Serial Port

Common Settings

Driver:	CIP Serial
Baud Rate:	38400
Parity:	None
Station Address:	1

Protocol Control

DF1 Mode:	Full Duplex
Media:	RS232
Control Line:	No Handshake
Error Detection:	CRC
Embedded Responses:	After One Received
	☑ Duplicate Packet Detection
ACK Timeout:	50 ×20 ms ENQ Retries: 3
NAK Retries:	3

3. Specify a Baud Rate. Select a communication rate that all devices in your system support. Configure all devices in the system for the same communication rate. Default Baud Rate is set at 38,400 bps.

4. In most cases, parity and station address are left at the default settings.

5. Select Advanced Settings and set Advanced parameters.
See Table 14 for a description of the CIP Serial parameters.

Table 14 - CIP Serial Driver Parameters

Parameter	Options	Default
Baud Rate	Toggles between the communication rate of 1200, 2400, 4800, 9600, 19200, and 38400.	38400
Parity	Specifies the parity setting for the serial port. Parity provides additional message-packet error detection. Select Even, Odd, or None.	None
Station (Node) Address	Enter a value from 0…254. Enter 1 for DF1 Full-duplex.	1
DF1 Mode[1]	Defines the DF1 mode – Full-duplex, Half-duplex master, Half-duplex slave, Radio Modem[2].	Configured as Full-duplex by default.
Control Line	• Full-duplex: No Handshake, Full-duplex (RTS always ON). • Half-duplex slave: No Handshake, Half-duplex without continuous carrier (RTS/CTS). • Half-duplex master: No Handshake, Half-duplex without continuous carrier (RTS/CTS), Full-duplex (RTS always ON). • Radio Modem: No Handshake, Half-duplex without continuous carrier (RTS/CTS), Half-duplex with DCD Handshake.	Configured as No Handshake by default.
Duplicate Packet Detection	Detects and eliminates duplicate responses to a message. Duplicate packets may be sent under noisy communication conditions when the sender's retries are not set to 0. Toggles between Enabled and Disabled.	Enabled
Error Detection	Toggles between CRC and BCC.	CRC
Embedded Responses	To use embedded responses, choose Enabled Unconditionally. If you want the controller to use embedded responses only when it detects embedded responses from another device, choose After One Received. If you are communicating with another Allen-Bradley device, choose Enabled Unconditionally. Embedded responses increase network traffic efficiency.	After One Received
NAK Retries	The number of times the controller resends a message packet because the controller received a NAK response to the previous message packet transmission.	3
ENQ Retries	The number of enquiries (ENQs) that you want the controller to send after an ACK timeout occurs.	3
Transmit Retries	Specifies the number of times a message is retried after the first attempt before being declared undeliverable. Enter a value from 0…127.	3
ACK Timeout (x20 ms)	Specifies the amount of time after a packet is transmitted that an ACK is expected.	50
EOT Suppression	Enabled, Disabled When EOT Suppression is enabled, the slave does not respond when polled if no message is queued. This saves modem transmission power and time when there is no message to transmit.	Disabled
Poll Timeout (x20 ms)	0…65,535 (can be set in 20 ms increments) Poll Timeout only applies when a slave device initiates an MSG instruction. It is the amount of time that the slave device waits for a poll from the master device. If the slave device does not receive a poll within the Poll Timeout, an MSG instruction error is generated, and the ladder program must requeue the MSG instruction. If you are using an MSG instruction, it is recommended not to use a Poll Timeout value of zero. Poll Timeout is disabled when set to zero.	3000
RTS Off Delay (x20 ms)	0…65,535 (can be set in 20 ms increments) Specifies the delay time between when the last serial character is sent to the modem and when RTS is deactivated. Gives the modem extra time to transmit the last character of a packet.	0
RTS Send Delay (x20 ms)	0…65,535 (can be set in 20 ms increments) Specifies the time delay between setting RTS until checking for the CTS response. For use with modems that are not ready to respond with CTS immediately upon receipt of RTS.	0
Message Retries	0…255 Specifies the number of times a slave device attempts to resend a message packet when it does not receive an ACK from the master device. For use in noisy environments where message packets may become corrupted in transmission.	3

Table 14 - CIP Serial Driver Parameters (Continued)

Parameter	Options	Default
Priority Polling Range – High	Select the last slave station address to priority poll.	0
Priority Polling Range – Low	Select the first slave station address to priority poll. A value of 255 disables priority polling.	255
Normal Polling Range – High	Select the last slave station address to normal poll.	0
Normal Polling Range – Low	Select the first slave station address to normal poll. A value of 255 disables normal polling.	255
Normal Poll Group Size	Enter the quantity of active stations in the normal poll range that you want polled during a scan through the normal poll range before returning to the priority poll range. If no stations are configured in the Priority Polling Range, leave this parameter at 0.	0
Reply Message Wait Timeout	Defines the amount of time, in 20 ms increments, which the master station waits after receiving an ACK (to a master-initiated message) before polling the slave station for a reply. Choose a time that is, at minimum, equal to the longest time that a slave station requires to format a reply packet. This would typically be the maximum scan time of the slave station.	5
Polling Mode	If you want to: • Receive only one message from a slave station per its turn, choose STANDARD (SINGLE MESSAGE TRANSFER PER NODE SCAN). Choose this method only if it is critical to keep the poll list scan time to a minimum. • Receive as many messages from a slave station as it has, choose STANDARD (MULTIPLE MESSAGE TRANSFER PER NODE SCAN). • Accept unsolicited messages from slave stations, choose MESSAGE BASED (ALLOW SLAVES TO INITIATE MESSAGES) • Slave station-initiated messages are acknowledged and processed after all master station-initiated (solicited) messages. Slave stations can only send messages when they are polled. If the message-based master station never sends a slave station a message, the master station never sends the slave station a poll. Therefore, to obtain a slave station-initiated message regularly from a slave station, choose to use standard communication mode instead. • Ignore unsolicited messages from slave stations, choose MESSAGE BASED (DO NOT ALLOW SLAVES TO INITIATE MESSAGES) Slave station-initiated messages are acknowledged and discarded. The master station acknowledges the slave station-initiated message so that the slave station removes the message from its transmit queue, which allows the next packet that is slated for transmission into the transmit queue.	MESSAGE BASED (ALLOW SLABES TO INITIATE MESSAGES)

(1)   For more information on DF1 protocol, see <u>Connect to Networks using DF1 on page 335</u>
(2)   Half-duplex and Radio Modem DF1 modes are only supported on Micro850 (2080-L50E) and Micro870 (2080-L70E) controllers.

Configure Modbus RTU

1. Open your Connected Components Workbench project. On the device configuration tree, go to the Controller properties. Select Serial Port.
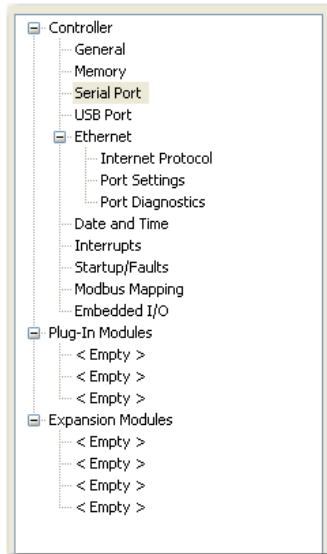
2. Select Modbus RTU on the Driver field.
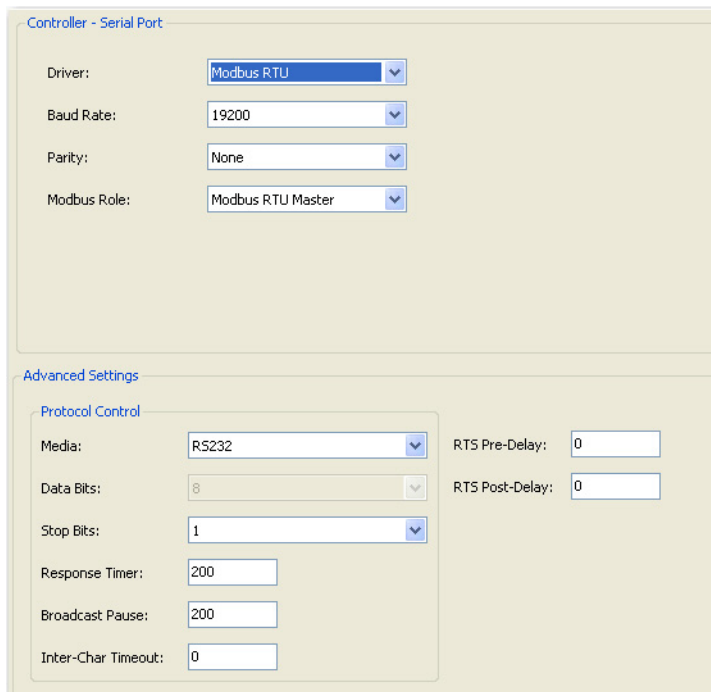
3. Specify the following parameters:
   - Baud Rate
   - Parity
   - Modbus Role

Table 15 - Modbus RTU Parameters

Parameter	Options	Default
Baud Rate	1200, 2400, 4800, 9600, 19200, 38400	19200
Parity	None, Odd, Even	None
Modbus Role	Master, Slave, Auto	Master

4. You can configure additional parameters under Advanced Settings.

Table 16 - Modbus RTU Advanced Parameters

Parameter	Options	Default
Media	RS-232, RS-232 RTS/CTS, RS-485	RS-232
Data Bits	Always 8	8
Stop Bits	1, 2	1
Response Timer	0...999,999,999 milliseconds	200
Broadcast Pause	0...999,999,999 milliseconds	200
Inter-char Timeout	0...999,999,999 microseconds	0
RTS Pre-delay	0...999,999,999 microseconds	0
RTS Post-delay	0...999,999,999 microseconds	0

Configure ASCII

1. Open your Connected Components Workbench project. On the device configuration tree, go to Controller properties. SelectSerial Port.
2. Select ASCII on the Driver field.
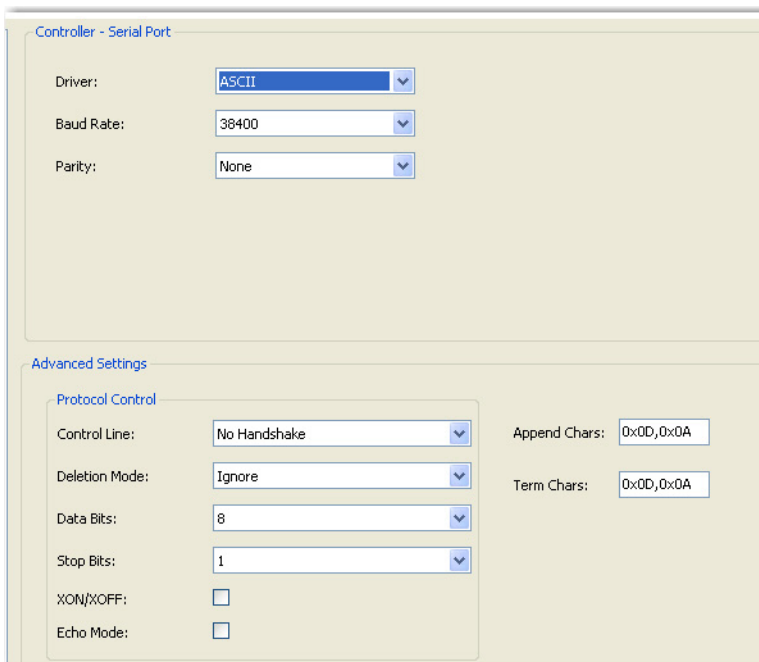
3. Specify the following parameters:
   - Baud Rate
   - Parity

Table 17 - ASCII Parameters

Parameter	Options	Default
Baud Rate	1200, 2400, 4800, 9600, 19200, 38400	19200
Parity	None, Odd, Even	None

4. You can configure advanced parameters under Advanced Settings.

Table 18 - ASCII Advanced Parameters

Parameter	Options	Default
Control Line	Full-duplex, Half-duplex with continuous carrier, Half-duplex without continuous carrier, No Handshake	No Handshake
Deletion Mode	CRT, Ignore, Printer	Ignore
Data Bits	7, 8	8
Stop Bits	1, 2	1
XON/XOFF	Enabled or Disabled	Disabled
Echo Mode	Enabled or Disabled	Disabled
Append Chars	0x0D,0x0A or user-specified value	0x0D,0x0A
Term Chars	0x0D,0x0A or user-specified value	0x0D,0x0A

Configure Ethernet Settings

1. Open your Connected Components Workbench project (for example, Micro850). On the device configuration tree, go to Controller properties. Select Ethernet.
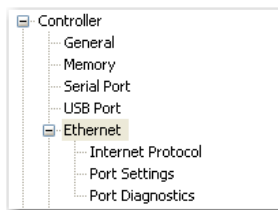
2. Under Ethernet, select Internet Protocol.
Configure Internet Protocol (IP) settings. Specify whether to obtain the IP address automatically using DHCP or manually configure IP address, subnet mask, and gateway address.
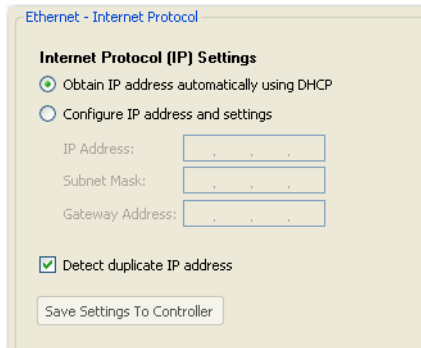
The Ethernet port defaults to the following out-of-the box settings:
- DHCP (dynamic IP address)
- Address Duplicate Detection: On

IMPORTANT	When a DHCP server fails, the Micro800 controller allocates IP addresses in the private range 169.254.0.1 to 169.254.255.254. The Micro800 controller verifies that its address is unique on the network using ARP. When the DHCP server is again able to service requests, the Micro800 controller updates its address automatically.

3. Select the checkbox Detect duplicate IP address to enable detection of duplicate address.

4. Under Ethernet, select Port Settings.
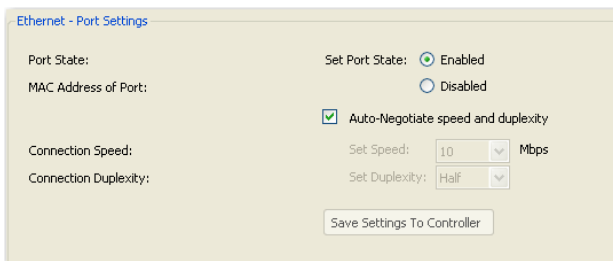
5. Set Port State as Enabled or Disabled.

6. To set the connection speed and duplexity manually, clear the check box Auto-Negotiate speed and duplexity. Then, set Speed (10 Mbps or 100 Mbps) and Duplexity (Half or Full) values.

7. Select Save Settings to Controller if you would like to save the settings to your controller.

8. On the device configuration tree, under Ethernet, select Port Diagnostics to monitor Interface and Media counters. The counters are available and updated when the controller is in Debug mode.

Validate IP Address

Modules must validate the incoming IP address configuration, whether it is obtained through explicit configuration or through DHCP.

The following rules must be obeyed when configuring the IP address:

- The IP address for the module cannot be set to zero, a multicast address, a broadcast address, or an address on the Class A loopback network (127.x.x.x).
- The IP address must not start with zero, and the IP address network ID must not be zero.
- The Network mask cannot be set to 255.255.255.255.
- The Gateway address must be on the same subnet as the IP address that is being configured.
- The Name Server address cannot be set to zero, a multicast address, a broadcast address, or an address on the Class A loopback network (127.x.x.x).

The valid range of static IPv4 IP addresses exclude:

- Broadcast or zero IP (255.255.255.255 or 0.0.0.0)
- IP addresses that start with 0 or 127 (0.xxx.xxx.xxx or 127.xxx.xxx.xxx)
- IP addresses that end with 0 or 255 (xxx.xxx.xxx.0 or xxx.xxx.xxx.255)
- IP addresses in the range 169.254.xxx.xxx (169.254.0.0 to 169.254.255.255)
- IP addresses in the range 224.0.0.0 to 255.255.255.255

Ethernet Host Name

Micro800 controllers implement unique host names for each controller, to be used to identify the controller on the network. The default host name consists of two parts: product type and MAC address, which are separated by a hyphen. For example: 2080-LC50-xxxxxxxxxxxx, where xxxxxxxxxxxx is the MAC address.

You can change the host name using the CIP Service Set Attribute Single when the controller is in Program/Remote Program mode.

Configure CIP Serial Driver

1. Open your Connected Components Workbench project. On the device configuration tree, go to the Controller properties. Click Serial Port.
2. Select CIP Serial from the Driver field.
3. Specify a Baud Rate. Select a communication rate that all devices in your system support. Configure all devices in the system for the same communication rate. Default Baud Rate is set @ 38,400 bps.
4. In most cases, parity and station address are left at default settings.
5. Select Advanced Settings and set Advanced parameters.

OPC Support Using FactoryTalk Linx

Support for Open Platform Communications (OPC) using CIP symbolic has been added from firmware release 7.011 onwards. This can be used in place of Modbus addressing.

FactoryTalk® Linx software version 5.70 (CPR9 SR7) or later and FactoryTalk® Linx Gateway software version 3.70 (CPR9 SR7) or later are required.

Micro870 Controller Distributed Network Protocol

This chapter describes how to configure the DNP3 communication settings.

Channel Configuration for DNP3 Slave

The default communication protocol for the serial ports is DF1 Full-Duplex. To communicate with Distributed Network Protocol (DNP3), the channel must be configured for DNP3 Slave.

The default communication protocol for the Ethernet channel in the controller is EtherNet/IP. To communicate with the DNP3 over IP protocol, select DNP3 over IP Enable in the Ethernet configuration page.

IMPORTANT	The DNP3 protocol is only supported in the following Micro870 controllers. • 2080-L70E-24QBBN • 2080-L70E-24QWBN, 2080-L70E-24QWBNK

To program the controller, use Connected Components Workbench software version 20.01.00 or later.

In Connected Components Workbench software, open the Micro870 controller project tree.
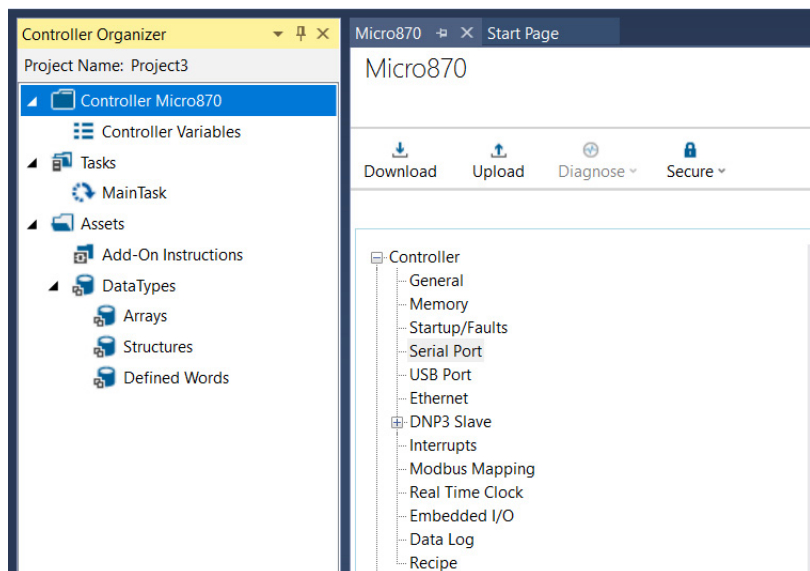
There are three configurations that are related to the DNP3 protocol in Connected Components Workbench software:
- Serial port configuration
- Ethernet port configuration
- DNP3 Slave Application Layer configuration

Serial Port Link Layer Configuration

Link Layer related configuration can be done in the Serial Port configuration page.

Ethernet Layer Configuration

To enable the DNP3 over IP protocol, select DNP3 over IP Enable in the Ethernet configuration page.

Link Layer related configuration can also be done in the Ethernet configuration page.

DNP3 Slave Application Layer Configuration

Application Layer related configuration can be done in the DNP3 Slave configuration page.

If multiple ports are configured for the DNP3 protocol, the serial and Ethernet ports share the DNP3 Slave configuration. Any changes in the DNP3 Slave configuration page affect all ports.

Serial Link Layer Configuration Parameters

Driver

Set this selection to DNP3 Slave to communicate with DNP3 protocol.

Node Address

This value is a node address of this DNP3 Slave.

The valid range is 0...65519. Default value is 1.

Baud

The selections can be 38400, 19200, 9600, 4800, 2400, and 1200. Default selection is 38400.

Parity

The selections can be NONE, EVEN, and ODD. Default selection is NONE.

Stop Bits

The selections can be 1 and 2. Default selection is 1.

Enable Master Address Validation

Valid selections are Enabled (Checked) and Disabled (Unchecked). Default value is Disabled (Unchecked).

When the selection is Disabled (Unchecked), the controller accepts the requests from any DNP3 Master.

When the selection is Enabled (Checked), the controller accepts the requests only from the DNP3 Master, which is configured in the Master Node0...Master Node4. The maximum number of Master Node Addresses for the Master Address Validation is 5.

Enable Self-Address

Valid selections are Enabled (Checked) and Disabled (Unchecked). Default value is Disabled (Unchecked).

When this bit is Disabled (Unchecked), any packets that contain the destination address 65532(FFFCh) are ignored.

When this bit is Enabled (Checked), any packets that contain the destination address 65532(FFFCh) are accepted and processed.

Any responses back to the DNP3 Master includes the actual configured DNP3 address of the Micro870 controller.

Master Node0

This value is used to:

- Validate the Master node address when the Enable Master Address Validation is Enabled (Checked).
- Vend Unsolicited Response when Unsolicited Response functionality is enabled. An Unsolicited Response is sent out to the DNP3 Master having this address.

The valid range is 0...65519. Default value is 0.

Master Node1, Master Node2, Master Node3, Master Node4

This value is used to check validation for Master node address when Enable Master Address Validation is Enabled (Checked).

The valid range is 0...65519. Default value is 0.

Control Line

The selections can be No Handshake and Half-duplex without continuous carrier (CTS/RTS). Default selection is No Handshake.

When the controller is connected to DNP3 Master using the RS-232 line directly, you must select No Handshake. If you want to use the Modem line in a half-duplex network, you must select Half-duplex without continuous carrier (CTS/RTS). If the controller is connected to an RS-485 network, you must select No Handshake.

Request LL Confirmation

Valid selections are Enabled (Checked) and Disabled (Unchecked). Default value is Disabled (Unchecked).

When the selection is Disabled (Unchecked), Primary Frames from the controller are sent out with the function code FC_UNCONFIRMED_USER_DATA (4).

When the selection is Enabled (Checked), Primary Frames from the controller are sent out with the function code FC_CONFIRMED_USER_DATA (3). In this case, the controller waits for the confirmation and may retry the Frame if it did not receive the confirmation from DNP3 Master within the time Confirmation Timeout (x1 ms).

Send LL Confirmation

Valid selections are Enabled (Checked) and Disabled (Unchecked). Default value is Disabled (Unchecked).

When the selection is Disabled (Unchecked), the optional Secondary Frame is not sent out with the function code FC_NACK (1) or FC_NOT_SUPPORTED (15).

When the selection is Enabled (Checked), the optional Secondary Frame is sent out with the function code FC_NACK (1) or FC_NOT_SUPPORTED (15).

IMPORTANT	Micro870 (2080-L70E-24QxBN) controllers support this function only when the DNP3 Master sends confirmed user data. This function is not supported when the DNP3 Master sends unconfirmed user data.

Confirmation Timeout (x1 ms)

When Request LL Confirmation is enabled, the controller waits to receive a confirmation frame until this timeout has expired.

The valid range is 1...65535. Default value is 20.

Message Retries

When Confirmation Timeout (x1 ms) has expired and this parameter was nonzero value, the controller tries to send retry packets.

The valid range is 0...255. Default value is 0.

Pre-transmit Delay (x1 ms)

The controller waits for the specified time before sending the packet.

The valid range is 0...65535. Default value is 0.

RTS Off Delay (x1 ms)

When the Control is set at Half-duplex Modem (CTS/RTS handshaking), this feature is enabled. This specifies a time delay between the end of a transmission and dropping of the RTS signal.

The valid range is 0...65535. Default value is 0.

RTS Send Delay (x1 ms)

When the Control is set at Half-duplex Modem (CTS/RTS handshaking), this entry is enabled. This specifies a time delay between the raising of the RTS and the initiation of a transmission.

The valid range is 0...65535. Default value is 0.

Max Random Delay (x1 ms)

This parameter is used with Pre-transmit Delay (x1 ms) for Collision Avoidance on RS-485 network. For more information, see <u>Collision Avoidance on page 113</u>.

The valid range is 0...65535. Default value is 0.

Ethernet Layer Configuration Parameters

The DNP3 over IP subsystem in the controller supports Listening End Point, TCP Dual End Point, and Datagram End Point types.

- Listening End Point type supports one TCP connection as a Server and UDP datagram.
- TCP Dual End Point type supports one TCP connection as a Server, and one TCP connection as a Client and UDP datagram.
- Datagram End Point type supports UDP datagram from DNP3 Masters. The default TCP and UDP port numbers are 20000 and the port numbers are configurable.

The parameter End Point Type determines the type of end point. According to the parameter, the controller works as different End Point types. See <u>Table 19 on page 77</u> for each configuration.

Table 19 - End Point Type Parameter Selections

End Point Type	Connection	Description
Listening	One TCP server connection	Any of the requests are accepted and the responses are transmitted through this connection. The unsolicited responses are transmitted through this connection when this connection is available.
	UDP datagram	Accepts only broadcast packets when the DNP3 destination node is one of 0xFFFD, 0xFFFE, and 0xFFFF in the request.
Dual	One TCP server connection	Any of the requests are accepted and the responses are transmitted through this connection. The unsolicited responses are transmitted through this connection when this connection is available. This connection has a higher priority than the Client connection.
	One TCP client connection	Any of the requests are accepted and the responses are transmitted through this connection. The unsolicited responses are transmitted through this connection when this connection is available. The controller does not request a TCP client connection to the DNP3 Master until an unsolicited response is generated.
	UDP datagram	Accepts only broadcast packets when the DNP3 destination node is one of 0xFFFD, 0xFFFE, and 0xFFFF in the request.
Datagram Only	UDP datagram only	Any of the requests are accepted and the responses are transmitted through this connection. All responses can be transmitted to the different DNP3 Master port according to the configuration of the parameters Remote UDP Port Number and Master IP Address0. If this parameter is not set to 0, the solicited responses are sent to the DNP3 Master port that is configured. If this parameter is set to 0, the solicited responses are sent to the DNP3 Master port that sent the request. TCP connection is not available in this configuration.

The parameter DNP3 over IP Enable is configured in the Ethernet configuration page and other parameters are configured in the DNP3 Slave configuration page. For more information, see Channel Configuration for DNP3 Slave on page 71.

DNP3 over IP Enable

The valid selections are Enabled (Checked) and Disabled (Unchecked). Default value is Disabled (Unchecked).

When the selection is Disabled (Unchecked), DNP3 service over Ethernet is disabled after the configuration is downloaded to the controller.

When the selection is Enabled (Checked), DNP3 service over Ethernet is enabled after the configuration is downloaded to the controller.

Enable Master Address Validation

The valid selections are Enabled (Checked) and Disabled (Unchecked). Default value is Disabled (Unchecked).

When the selection is Disabled (Unchecked), the controller accepts the requests from any DNP3 Master.

When the selection is Enabled (Checked), the controller accepts the requests only from the DNP3 Master Node Address, which is configured in the parameters Master Node0 on page 75, and Master Node1, Master Node2, Master Node3, Master Node4 on page 75. The maximum number of Master Node Addresses for the Master Address Validation is 5.

Enable Self-Address

The valid selections are Enabled (Checked) and Disabled (Unchecked). Default value is Disabled (Unchecked).

When this bit is Disabled (Unchecked), any packets that contain the destination address 65532 (FFFCh) are ignored.

When this bit is Enabled (Checked), any packets that contain the destination address 65532 (FFFCh) are accepted and processed.

Any responses back to the DNP3 Master includes the actual configured DNP3 address of the Micro870 controller.

Enable Access Control

The valid selections are Enabled (Checked) and Disabled (Unchecked). Default value is Disabled (Unchecked).

When the selection is Disabled (Unchecked), the controller accepts the requests from any DNP3 Master.

When the selection is Enabled (Checked), the controller accepts the requests only from the DNP3 Master IP Address, which is configured in the parameters Master IP Address0 to Master IP Address4. The maximum number of Master IP Addresses for the Access Control is 5.

End Point Type

The valid selections are Listening, Dual, and Datagram Only. Default is Listening End Point type.

Master Node0

This value is used to:
- Validate Master node address when the Enable Master Address Validation is Enabled (Checked).
- Send Unsolicited Response when Unsolicited Response functionality is enabled. An Unsolicited Response is sent out to the DNP3 Master having this address.

The valid range is 0...65519. Default value is 0.

Master Node1, Master Node2, Master Node3, Master Node4

This value is used for validation of the Master node address when the Enable Master Address Validation is Enabled (Checked). This value is only shown and valid when the Enable Master Address Validation is Enabled (Checked).

The valid range is 0...65519. Default value is 0.

Master IP Address0

This value is used to:
- Validate Master IP address when the Enable Access Control is Enabled (Checked).
- Send Unsolicited Response when Unsolicited Response functionality is enabled. An Unsolicited Response is sent out to the DNP3 Master having this address.

The valid value is an IP address. Default value is 0.0.0.0.

Master IP Address1, Master IP Address2, Master IP Address3, Master IP Address4

This value is used for validation of the Master IP address when the Enable Access Control is Enabled (Checked). This value is only shown and valid when the Enable Access Control is Enabled (Checked).

The valid value is an IP address. Default value is 0.0.0.0.

Master TCP Port Number (Unsol)

This value is used to configure Master TCP Port Number for Unsolicited Response.

The valid range is 0...65535. Default value is 20000.

Master UDP Port Number for Initial Unsolicited

This value is used to configure Master UDP Port Number for Initial Unsolicited Response if the parameter End Point Type is selected as Datagram Only.

The valid range is 0...65535. Default value is 20000.

Master UDP Port Number

This value is used to configure Master UDP Port Number if the parameter End Point Type is selected as Datagram Only.

The valid range is 0...65535. Default value is 20000.

Keep Alive Interval (x 1s)

This parameter specifies a time interval for TCP Keep Alive mechanism.

If the timer times out, the controller transmits a keep-alive message. The keep-alive message is a DNP Data Link Layer status request (FC_REQUEST_LINK_STATUS). If a response is not received to the keep-alive message, the controller deems that the TCP connection is broken and closes the TCP connection.

The valid range is 1...65535. Default value is 10.

Slave Node Address

This value is a node address of this DNP3 Slave.

The valid range is 0...65519. Default value is 1.

Local Port Number (UDP)

This value is used to configure the Local UDP Port Number, which is used for UDP socket listening.

The valid range is 0...65535. Default value is 20000.

Local Port Number (TCP)

This value is used to configure the Local TCP Port Number, which is used for TCP socket listening.

The valid range is 0...65535. Default value is 20000.

DNP3 Slave Application Layer Configuration Parameters

Channel for Unsolicited Response

Only channels that are already configured for DNP3 protocol appear in the Channel for Unsolicited Response dropdown menu. All Unsolicited Responses are transmitted through this selected channel.

Restore Events After Power Cycle

When the selection is Disabled (Unchecked), DNP3 events that are generated before a power cycle are flushed after a power cycle. When the option is Enabled (Checked), all DNP3 events are restored after a power cycle.

Valid selections are Enabled (Checked) and Disabled (Unchecked). Default value is Disabled (Unchecked).

Unsolicited Responses On Start Up

Valid selections are Enabled (Checked) and Disabled (Unchecked). Default value is Disabled (Unchecked).

When the selection is Disabled (Unchecked), the controller does not send any enabled Unsolicited Responses after a restart until it has received a FC_ENABLE_UNSOLICITED (20) command from the DNP3 Master.

When the selection is Enabled (Checked), the controller sends any enabled Unsolicited Responses after a restart to the DNP3 Master unconditionally.

Unsolicited Responses For Class1

Valid selections are Enabled (Checked) and Disabled (Unchecked). Default value is Disabled (Unchecked).

When the selection is Disabled (Unchecked), Unsolicited Response is disabled for Class 1 events. To prevent overflowing of the event buffer, the DNP3 Master should poll for Class 1 events.

When the selection is Enabled (Checked), Unsolicited Response is enabled for Class 1 events.

Unsolicited Responses For Class2

Valid selections are Enabled (Checked) and Disabled (Unchecked). Default value is Disabled (Unchecked).

When the selection is Disabled (Unchecked), Unsolicited Response is disabled for Class 2 events. To prevent overflowing of the event buffer, the DNP3 Master should poll for Class 2 events.

When the selection is Enabled (Checked), Unsolicited Response is enabled for Class 2 events.

Enable Unsolicited For Class3

Valid selections are Enabled (Checked) and Disabled (Unchecked). Default value is Disabled (Unchecked).

When the selection is Disabled (Unchecked), Unsolicited Response is disabled for Class 3 events. To prevent overflowing of the event buffer, the DNP3 Master should poll for Class 3 events.

When the selection is Enabled (Checked), Unsolicited Response is enabled for Class 3 events.

Send Initial Unsolicited Null Response On Start Up

Valid selections are Enabled (Checked) and Disabled (Unchecked). Default value is Disabled (Unchecked).

When the selection is Disabled (Unchecked), the controller does not send Unsolicited NULL Response with RESTART IIN bit on startup.

When the selection is Enabled (Checked), the controller sends Unsolicited NULL Response with RESTART IIN bit on startup.

This selection is also used for sending the Restart IIN bit during Driver and Channel configuration changes. For more information, see .

Enable Confirmation

Valid selections are Enabled (Checked) and Disabled (Unchecked). Default value is Disabled (Unchecked).

When the selection is Disabled (Unchecked), the controller sends Response packets with CON bit set in its header under the following conditions only:

- When the response has Event data
- When the response is multi-fragment response
- When the Unsolicited Response is sent

When the selection is Enabled (Checked), the controller always sends Response packets with the CON bit set in its header, which causes the DNP3 Master to send replies confirming that it received each Response packet without error.

Enable Time Synchronization

This parameter used with Time Synchronization Interval (x1 mins).

Valid selections are Enabled (Checked) and Disabled (Unchecked). Default value is Disabled (Unchecked).

When the selection is Disabled (Unchecked), the controller does not perform any time synchronization.

When the selection is Enabled (Checked), the controller sets the NEED_TIME Internal Indication bit (IIN1.4) on power-up and every interval that is configured in Time Synchronization Interval (x1 mins).

Time Synchronization Interval (x1 mins)

This parameter is used with Enable Time Synchronization. Only valid when Enable Time Synchronization is Enabled (Checked).

The valid range is 0...32767. Default value is 0. If the value is 0, the NEED_TIME Internal Indication (IIN1.4) bit are set at startup and then after every Time Synchronization Interval minute if the value is greater than 0.

When the parameter Enable Time Synchronization is Disabled (Unchecked), the IIN1.4 bit is never turned on.

Max Response Size

The controller sends an Application Layer frame to fit in Max Response Size. If the Response packet size is larger than this value, the controller fragments the Response packet.

The valid range is 27...2048 in bytes. Default value is 2048.

Confirmation Timeout (x1 ms)

When Enable Confirmation is enabled, the controller waits for Application Layer Confirmation until the Confirmation Timeout (x1 ms) has expired.

The valid range is 100...65535 in 1 ms increments. Default value is 10000.

Number of Retries

This parameter is used only for Unsolicited Response. If this value is set to the maximum of 65535, it means infinite retries of the Unsolicited Response.

The valid range is 0...65535. Default value is 0.

Number of Class1 Events

If the controller is configured not to initiate Class 1 Unsolicited Responses, this parameter is used to limit the maximum number of events, which is generated and logged into the event buffer for Class 1 events. In this case, the value 0 disables Class 1 event generation.

If the controller is configured to generate Unsolicited Responses, and the number of queued Class 1 events reaches this value, an Unsolicited Response is initiated.

The valid range is 0...10000. Default value is 10.
For more information, see DNP3 10K Event Logging on page 111.

Hold Time after Class1 Events (x 1s)

This parameter is only for a Class 1 Unsolicited Response. The controller holds the events during Hold Time after Class1 Events (x 1s) before initiating an Unsolicited Response.

The valid range is 0...65535. Default value is 5.

The value of 0 indicates that responses are not delayed due to this parameter.

The parameters Number of Class1 Events and Hold Time after Class1 Events (x 1s) are used together so that if either one of the criteria is met, an Unsolicited Response is transmitted.

By default, the Hold time is retriggered for each new event detected.

Number of Class2 Events

If the controller is configured not to initiate Class 2 Unsolicited Responses, this parameter is used to limit the maximum number of events, which is generated and logged into the event buffer for Class 2 events. In this case, value 0 disables Class 2 event generation.

If the controller is configured to generate Unsolicited Responses, and the number of queued Class 2 events reaches this value, an Unsolicited Response is initiated.

The valid range is 0...10000. Default value is 10.
For more information, see DNP3 10K Event Logging on page 111.

Hold Time after Class2 Events (x 1s)

This parameter is only for a Class 2 Unsolicited Response. The controller holds the events during Hold Time after Class2 Events (x 1s) before initiating an Unsolicited Response.

The valid range is 0...65535. Default value is 5.

The value of 0 indicates that responses are not delayed due to this parameter.

The Parameters Number of Class2 Events and Hold Time after Class2 Events (x 1s) are used together so that if either one of the criteria is met, an Unsolicited Response is transmitted.

By default, the Hold time is retriggered for each new event detected.

Number of Class3 Events

If the controller is configured not to initiate Class 3 Unsolicited Responses, this parameter is used to limit the maximum number of events, which is generated and logged into the event buffer for Class 3 events. In this case, value 0 disables Class 3 event generation.

If the controller is configured to generate Unsolicited Responses, and the number of queued Class 3 events reaches this value, an Unsolicited Response is initiated.

The valid range is 0...10000. Default value is 10.
For more information, see .

Hold Time after Class3 Events (x 1s)

This parameter is only for a Class 3 Unsolicited Response. The controller holds the events during Hold Time after Class3 Events (x 1s) before initiating an Unsolicited Response.

The valid range is 0...65535. Default value is 5.

The value of 0 indicates that responses are not delayed due to this parameter.

The parameters Number of Class3 Events and Hold Time after Class3 Events (x 1s) are used together so that if either one of the criteria is met, an Unsolicited Response is transmitted.

By default, the Hold time is retriggered for each new event detected.

Select Timeout (x 1s)

The valid range is 1...65535. Default value is 10.

This parameter is used for controlling CROB (Control Relay Output Block) and AOB (Analog Output Block). After receiving the request with the function code FC_SELECT(3), DNP3 Master should send the request with the function code FC_OPERATE(4) within this configured time.

DNP3 Object Data and Config
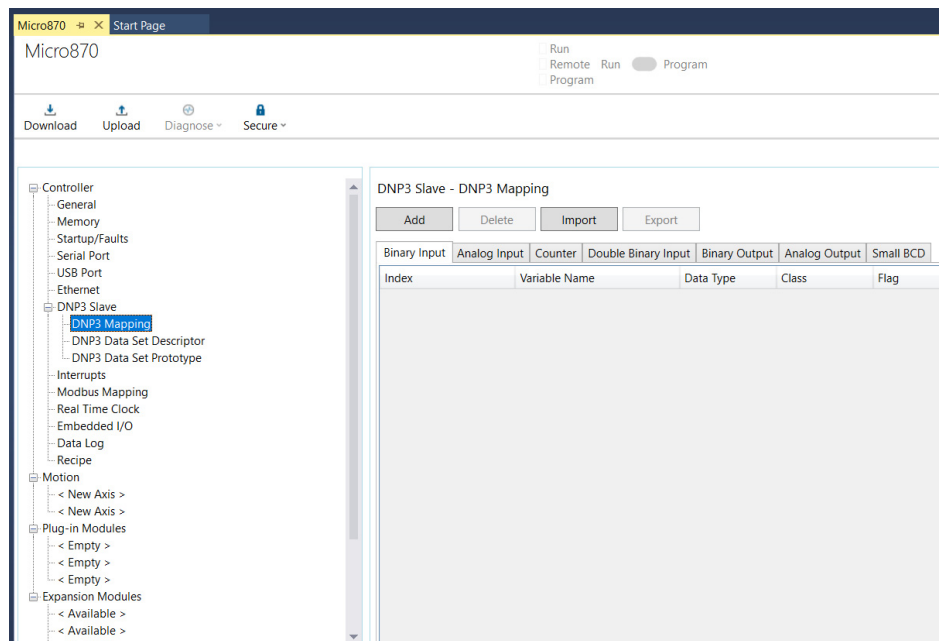
The DNP3 Mapping selection under DNP3 Slave in the controller properties allows you to define the mapping of the listed DNP3 object and object properties (class number, online/offline status, object quality flags, deadbands, and/or thresholds) to controller variables.

For more information, see .

DNP3 Secure Authentication

The controller implements DNP3 Secure Authentication, which is based on the DNP3 Specification, Supplement to Volume 2, Secure Authentication, Version 2.00 and 5.00.

DNP3 Secure Authentication has been implemented in the DNP3 Application Layer of the controller system. If you configure any parameters regarding DNP3 Secure Authentication in the DNP3 Slave Application Layer configuration, it affects all ports that are configured for the DNP3 protocol in the controller.

Enable Secure Authentication

The valid selections are Enabled (Checked) and Disabled (Unchecked). Default value is Disabled (Unchecked).

When the selection is Disabled (Unchecked), the controller disables the DNP3 Secure Authentication subsystem.

When the selection is Enabled (Checked), the controller enables the DNP3 Secure Authentication subsystem.

Secure Authentication Version

This parameter specifies the authentication version that this DNP3 slave controller uses.

Select 2 for Secure Authentication version 2 and select 5 for Secure Authentication version 5. Default value is 2.

Enable Aggressive Mode in Secure Authentication

The valid selections are Enabled (Checked) and Disabled (Unchecked). Default value is Disabled (Unchecked).

When the selection is Disabled (Unchecked), the controller disables DNP3 Aggressive Mode in Secure Authentication subsystem.

When the selection is Enabled (Checked), the controller enables DNP3 Aggressive Mode in Secure Authentication subsystem.

Critical Function Code in Secure Authentication

Critical Function Codes:

0	Non-Critical	8	Non-Critical	21	Critical	29	Critical
1	Non-Critical	9	Non-Critical	22	Non-Critical	30	Non-Critical
2	Critical	10	Non-Critical	23	Non-Critical	31	Critical
3	Critical	11	Non-Critical	24	Critical	129	Non-Critical
4	Critical	12	Non-Critical	25	Non-Critical	130	Non-Critical
5	Critical	13	Critical	26	Non-Critical		
6	Critical	14	Critical	27	Non-Critical		
7	Non-Critical	20	Critical	28	Non-Critical		

This critical function code in the DNP3 Slave configuration page is used to define the list of the critical function codes in Secure Authentication. A critical function code should be defined by selecting the number icon to change it between Critical and Non-critical.

Table 20 shows the default state of the function codes that are defined in the Connected Components Workbench software.

Table 20 - Function Codes

Function Code	Critical FCs	Function Code	Critical FCs
0 (0x00)	Non-critical	20 (0x14)	Critical
1 (0x01)	Non-critical	21 (0x15)	Critical
2 (0x02)	Critical	22 (0x16)	Non-critical
3 (0x04)	Critical	23 (0x17)	Non-critical
4 (0x04)	Critical	24 (0x18)	Critical
5 (0x05)	Critical	25 (0x19)	Non-critical
6 (0x06)	Critical	26 (0x1A)	Non-critical
7 (0x07)	Non-critical	27 (0x1B)	Non-critical
8 (0x08)	Non-critical	28 (0x1C)	Non-critical
9 (0x09)	Non-critical	29 (0x1D)	Critical
10 (0x0A)	Non-critical	30 (0x1E)	Non-critical
11 (0x0B)	Non-critical	31 (0x1F)	Critical
12 (0x0C)	Non-critical	129 (0x81)	Non-critical
13 (0x0D)	Critical	130 (0x82)	Non-critical
14 (0x0E)	Critical		

Expected Session Key Change Interval (x1 min) in Secure Authentication

This parameter is used for configuring the expected session key change interval in minutes.

The valid range is 0...120 (min). Default value is 15 mins.

When the DNP3 Master does not change the Session Key within the time that is configured, the controller invalidates the Session Key and its state for each user.

Expected Session Key Change Count in Secure Authentication

This parameter is used for configuring the expected session key change count.

The valid range is 1...10000. Default value is 1000.

Reply Timeout (x100 ms) in Secure Authentication

This parameter is used for configuring the reply timeout in 100 ms.

The valid range is 0...1200 (120 s). Default value is 20 (2 s).

Maximum Error Count in Secure Authentication

This parameter is used for configuring the maximum error count.

The valid range is 0...10. Default value is 2.

HMAC Algorithm in Secure Authentication

This parameter is used to configure the Hash-based Message Authentication Code (HMAC) algorithm.

- SHA-1
  - Truncated to 4 octets (serial)
  - Truncated to 10 octets (networked)
- SHA-256
  - Truncated to 8 octets (serial)
  - Truncated to 16 octets (networked)

Default value is SHA-256.

Update Key in Secure Authentication

This parameter is used to define user information Secure Authentication.

In Connected Components Workbench software, you can create the user keys in the DNP3 Slave configuration page.
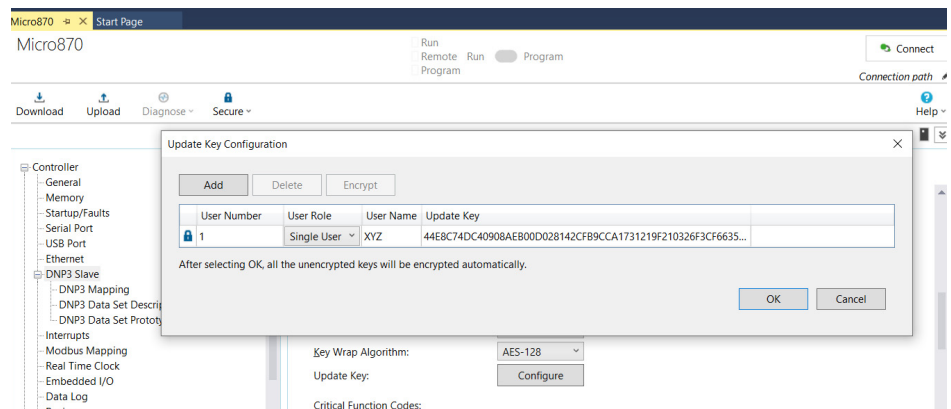
- User Number
  Valid range is 1...65535.
- User Role
  A dropdown selection of various roles that you can define for each user number (Viewer, Operator, Engineer, Installer, SECADM, SECAUD, RBACMNT, and single user).
- User Name
  Define the unique name for each user, up to 32 characters (numbers, alphabets, and symbols).
- Update Key
  The key to be used by each user, up to 32 hexadecimal digits.

To create a user, follow these steps.

1. Select Configure to open Update Key.

2. Select Add.

3. Enter the User Number, select the User Role, enter the User Name, and enter the Update Key.

4. Select Encrypt or OK to create the user.

5. Download the project to the controller to update the user information in the controller.

To remove an existing user, follow these steps.

1. Select Configure to open Update Key.

2. Select the user to remove and select Delete.

3. Select OK to close the window.

4. Download the project to the controller to update the user information in the controller.

Update Key Change Method and Authority Certification Key

IMPORTANT	This feature is only available in Secure Authentication Version 5.

An Authority Certification Key is a Symmetric (encrypted) or Public (not encrypted) key that is stored in the controller for authentication with the DNP3 master when a Key change request is processed.

Symmetric Key
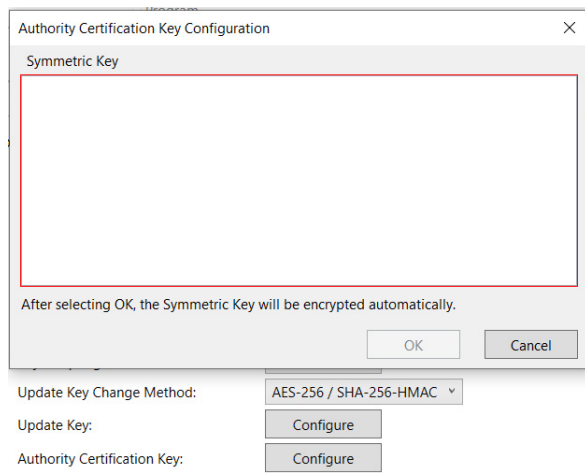
Public Key
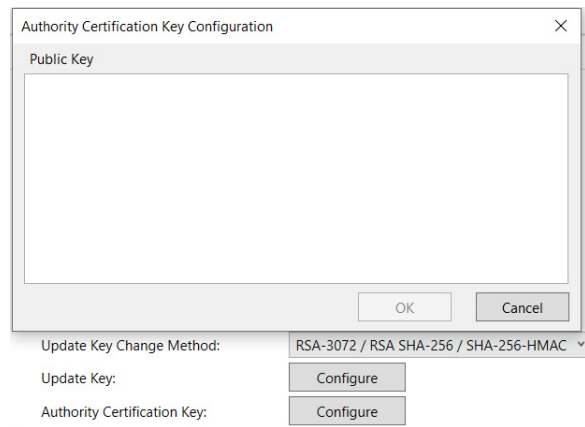
The type of key that is used in the certificate is based on the Update Key Change Method setting that you have selected in the configuration. To define the key, select one of the following settings.

- To use a Symmetric Key in authorization:
    - AES-128/SHA-1-HMAC
    - AES-256/SHA-256-HMAC
- To use a Public Key in authorization:
    - RSA-2048/RSA SHA-256/SHA-256/HMAC
    - RSA-3072/RSA SHA-256/SHA-256-HMAC

To configure the Authority Certification Key, select Configure.

To define a Symmetric Key, enter 32 or 64 characters into the field, depending on the Update Key Change Method that you have selected. Select OK to accept and encrypt the key.

To define a Public Key, you must generate an RSA-2048 or RSA-3072 Public Key, depending on the Update Key Change Method that you have selected, and enter it into the field. Select OK to accept the changes. Public Keys are not encrypted.

Example of a Public Key
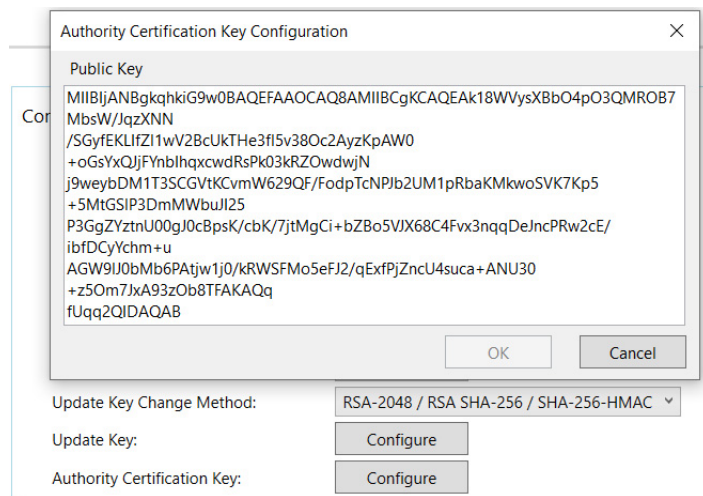
Default Variation Config

This configuration is used to define default variations in a response to a Class 0 poll request.

In Connected Components Workbench software version 20.01.00 or later, you can select Default Variation Table in the DNP3 Slave configuration page to change the configuration.
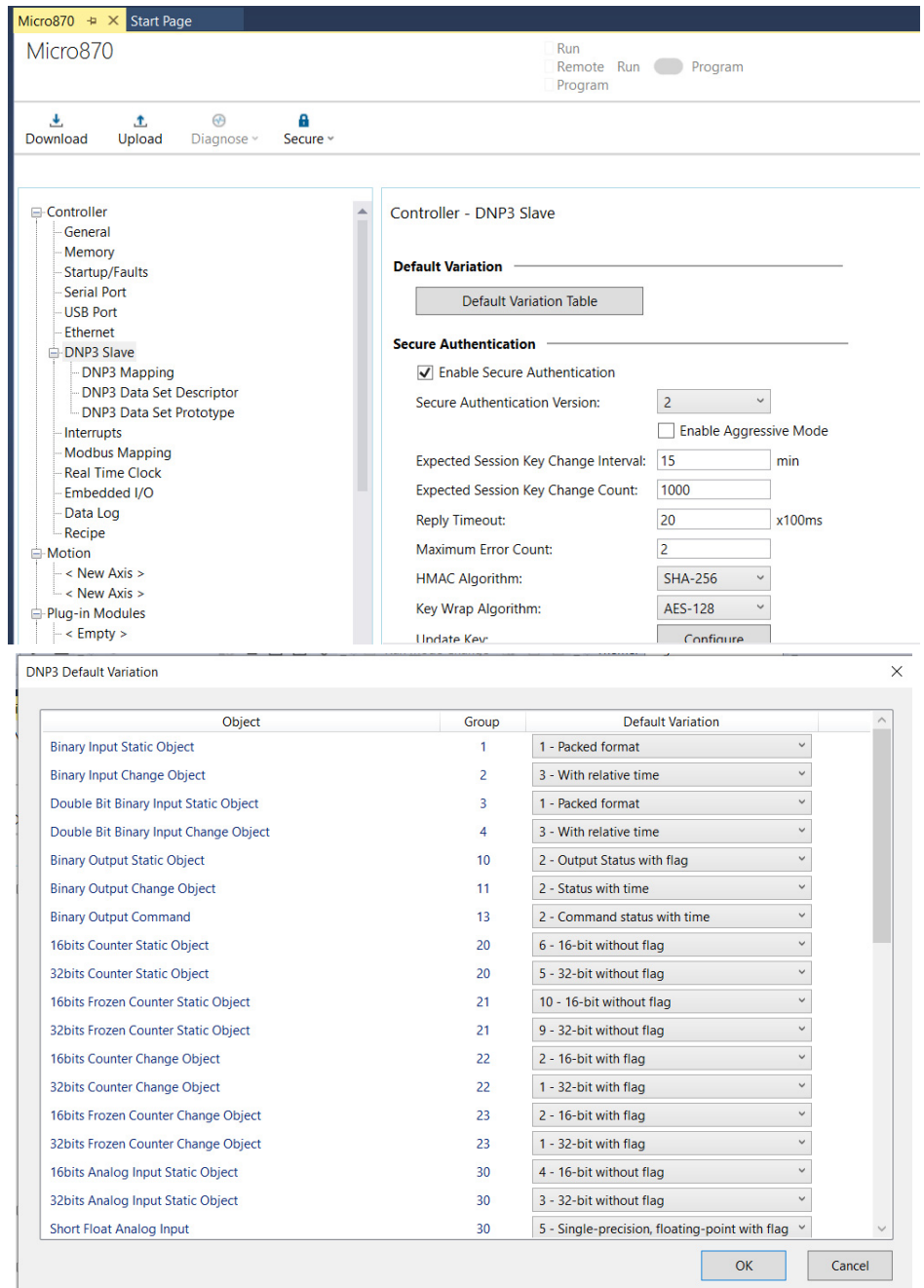
Table 21 shows the structure of the DNP3 Default Variation Configuration File.

Table 21 - DNP3 Default Variation Configuration File

Group	Default Variation for the Following Objects	Standard Default Variation	Alternate Default Variations
1	Binary Input Static Object	1 - Packed format	0 - All variation 2 - With flag
2	Binary Input Change Object	3 - With relative time	0 - All variation 1 - Without time 2 - With absolute time
3	Double Bit Binary Input Static Object	1 - Packed format	0 - All variation 2 - With flag
4	Double Bit Binary Input Change Object	3 - With relative time	0 - All variation 1 - Without time 2 - With absolute time
10	Binary Output Static Object	2 - Output status with flag	0 - All variation 1 - Packed format

Table 21 – DNP3 Default Variation Configuration File (Continued)

Group	Default Variation for the Following Objects	Standard Default Variation	Alternate Default Variations
11	Binary Output Change Object	2 - Status with time	0 - All variation 1 - Status without time
13	Binary Output Command	2 - Command status with time	0 - All variation 1 - Command status without time
20	16-bit Counter Static Object	6 - 16-bit without flag	0 - All variation 1 - 32-bit with flag 2 - 16-bit with flag 5 - 32-bit without flag
	32-bit Counter Static Object	5 - 32-bit without flag	0 - All variation 1 - 32-bit with flag 2 - 16-bit with flag 6 - 16-bit without flag
21	Frozen 16-bit Counter Static Object	10 - 16-bit without flag	0 - All variation 1 - 32-bit with flag 2 - 16-bit with flag 5 - 32-bit with flag and time 6 - 16-bit with flag and time 9 - 32-bit without flag
	Frozen 32-bit Counter Static Object	9 - 32-bit without flag	0 - All variation 1 - 32-bit with flag 2 - 16-bit with flag 5 - 32-bit with flag and time 6 - 16-bit with flag and time 10 - 16-bit without flag
22	16-bit Counter Change Object	2 - 16-bit with flag	0 - All variation 1 - 32-bit with flag 5 - 32-bit with flag and time 6 - 16-bit with flag and time
	32-bit Counter Change Object	1 - 32-bit with flag	0 - All variation 2 - 16-bit with flag 5 - 32-bit with flag and time 6 - 16-bit with flag and time
23	Frozen 16-bit Counter Change Object	2 - 16-bit with flag	0 - All variation 1 - 32-bit with flag 5 - 32-bit with flag and time 6 - 16-bit with flag and time
	Frozen 32-bit Counter Change Object	1 - 32-bit with flag	0 - All variation 2 - 16-bit with flag 5 - 32-bit with flag and time 6 - 16-bit with flag and time
30	16-bit Analog Input Static Object	4 - 16-bit without flag	0 - All variation 1 - 32-bit with flag 2 - 16-bit with flag 3 - 32-bit without flag 5 - Single-precision, floating point with flag 6 - Double-precision, floating point with flag
	32-bit Analog Input Static Object	3 - 32-bit without flag	0 - All variation 1 - 32-bit with flag 2 - 16-bit with flag 4 - 16-bit without flag 5 - Single-precision, floating point with flag 6 - Double-precision, floating point with flag
	Short Float Analog Input	5 - Single-precision, floating point with flag	0 - All variation 1 - 32-bit with flag 2 - 16-bit with flag 3 - 32-bit without flag 4 - 16-bit without flag 6 - Double-precision, floating point with flag

Table 21 – DNP3 Default Variation Configuration File (Continued)

Group	Default Variation for the Following Objects	Standard Default Variation	Alternate Default Variations
32	16-bit Analog Input Change Object	2 - 16-bit without time	0 - All variation 1 - 32-bit without time 3 - 32-bit with time 4 - 16-bit with time 5 - Single-precision, floating point without time 6 - Double-precision, floating point without time 7 - Single-precision, floating point with time 8 - Double-precision, floating point with time
	32-bit Analog Input Change Object	1 - 32-bit without time	0 - All variation 2 - 16-bit without time 3 - 32-bit with time 4 - 16-bit with time 5 - Single-precision, floating point without time 6 - Double-precision, floating point without time 7 - Single-precision, floating point with time 8 - Double-precision, floating point with time
	Short Float Analog Input Change Object	5 - Single-precision, floating point without time	0 - All variation 1 - 32-bit without time 2 - 16-bit without time 3 - 32-bit with time 4 - 16-bit with time 6 - Double-precision, floating point without time 7 - Single-precision, floating point with time 8 - Double-precision, floating point with time
33	Frozen 16-bit Analog Input Change Object	2 - 16-bit without time	0 - All variation 1 - 32-bit without time 3 - 32-bit with time 4 - 16-bit with time 5 - Single-precision, floating point without time 6 - Double-precision, floating point without time 7 - Single-precision, floating point with time 8 - Double-precision, floating point with time
	Frozen 32-bit Analog Input Change Object	1 - 32-bit without time	0 - All variation 2 - 16-bit without time 3 - 32-bit with time 4 - 16-bit with time 5 - Single-precision, floating point without time 6 - Double-precision, floating point without time 7 - Single-precision, floating point with time 8 - Double-precision, floating point with time
	Frozen Short Float Analog Input Change Object	5 - Single-precision, floating point without time	0 - All variation 1 - 32-bit without time 2 - 16-bit without time 3 - 32-bit with time 4 - 16-bit with time 6 - Double-precision, floating point without time 7 - Single-precision, floating point with time 8 - Double-precision, floating point with time
34	16-bit Analog Input Reporting Dead Band	1 - 16-bit	0 - All variation 2 - 32-bit 3 - Single-precision, floating point
	32-bit Analog Input Reporting Dead Band	2 - 32-bit	0 - All variation 1 - 16-bit 3 - Single-precision, floating point
	Short Float Analog Input Reporting Dead Band	3 - Single-precision, floating point	0 - All variation 1 - 16-bit 2 - 32-bit
40	16-bit Analog Output Static Object	2 - 16-bit with flag	0 - All variation 1 - 32-bit with flag 3 - Single-precision, floating point with flag 4 - Double-precision, floating point with flag
	32-bit Analog Output Static Object	1 - 32-bit with flag	0 - All variation 2 - 16-bit with flag 3 - Single-precision, floating point with flag 4 - Double-precision, floating point with flag
	Short Float Analog Output Status	3 - Single-precision, floating point with flag	0 - All variation 1 - 32-bit with flag 2 - 16-bit with flag 4 - Double-precision, floating point with flag

Table 21 - DNP3 Default Variation Configuration File (Continued)

Group	Default Variation for the Following Objects	Standard Default Variation	Alternate Default Variations
42	16-bit Analog Output Change Object	2 - 16-bit without time	0 - All variation 1 - 32-bit without time 3 - 32-bit with time 4 - 16-bit with time 5 - Single-precision, floating point without time 6 - Double-precision, floating point without time 7 - Single-precision, floating point with time 8 - Double-precision, floating point with time
	32-bit Analog Output Change Object	1 - 32-bit without time	0 - All variation 2 - 16-bit without time 3 - 32-bit with time 4 - 16-bit with time 5 - Single-precision, floating point without time 6 - Double-precision, floating point without time 7 - Single-precision, floating point with time 8 - Double-precision, floating point with time
	Short Float Analog Output Change Object	5 - Single-precision, floating point without time	0 - All variation 1 - 32-bit without time 2 - 16-bit without time 3 - 32-bit with time 4 - 16-bit with time 6 - Double-precision, floating point without time 7 - Single-precision, floating point with time 8 - Double-precision, floating point with time
43	16-bit Analog Output Command Change Object	2 - 16-bit without time	0 - All variation 1 - 32-bit without time 3 - 32-bit with time 4 - 16-bit with time 5 - Single-precision, floating point without time 6 - Double-precision, floating point without time 7 - Single-precision, floating point with time 8 - Double-precision, floating point with time
	32-bit Analog Output Command Change Object	1 - 32-bit without time	0 - All variation 2 - 16-bit without time 3 - 32-bit with time 4 - 16-bit with time 5 - Single-precision, floating point without time 6 - Double-precision, floating point without time 7 - Single-precision, floating point with time 8 - Double-precision, floating point with time
	Short Float Analog Output Command Change Object	5 - Single-precision, floating point without time	0 - All variation 1 - 32-bit without time 2 - 16-bit without time 3 - 32-bit with time 4 - 16-bit with time 6 - Double-precision, floating point without time 7 - Single-precision, floating point with time 8 - Double-precision, floating point with time

DNP3 Slave Application Layer

This section covers DNP3 Slave Application Layer Function Codes and Internal Indications. All Function Codes that are supported in the controller are summarized in Function Codes for DNP3 in Micro870 Controllers on page 118.

For details of Packet Formats for the request and response, see the DNP3 Protocol specifications.

Function Codes

CONFIRM (FC Byte = 0x00)

00 – Confirm

A DNP3 master sends a message with this function code to confirm receipt of a response fragment. In a general environment, the controller receives a response with this function code. But the controller may generate a response with this function code when a DNP3 Master sends a request with the CON bit set in the application control header.

READ (FC Byte = 0x01)

01 – Read

The READ function code is used by a DNP3 master to request data from the controller.

WRITE (FC Byte = 0x02)

02 – Write

The WRITE function code is used to write the contents of DNP3 objects from the DNP3 master to the controller. This function code is used for clearing bit IIN1.7 [DEVICE_RESTART], setting time in the controller and downloading user programs to the controller.

SELECT (FC Byte = 0x03)

03 – Select

The SELECT function code is used with the OPERATE function code as part of the select-before-operate method for issuing control requests. This procedure is used for controlling binary output (CROB) or analog output (AOB) objects.

OPERATE (FC Byte = 0x04)

04 – Operate

See SELECT (FC Byte = 0x03).

DIRECT_OPERATE (FC Byte = 0x05)

05 – Direct Operate

This direct operate function is similar to the FC_OPERATE function code except that no preceding select command is required.

DIRECT_OPERATE_NR (FC Byte = 0x06)

06 – Direct Operate No Resp

See DIRECT_OPERATE (FC Byte = 0x05). No response message is returned when this request is issued from a DNP3 master.

IMMED_FREEZE (FC Byte = 0x07)

07 – Immediate Freeze

Upon receiving a request with this function, the controller copies the current value of a counter point to a separate memory location associated with the same point. The copied value remains constant until the next freeze operation to the same point.

IMMED_FREEZE_NR (FC Byte = 0x08)

08 – Immediate Freeze No Resp

See IMMED_FREEZE (FC Byte = 0x07). No response message is returned when this request is issued from a DNP3 master.

FREEZE_CLEAR (FC Byte = 0x09)

09 – Freeze and Clear

Upon receiving a request with this function, the controller copies the current value to the frozen value, then clears the current value to 0 immediately.

FREEZE_CLEAR_NR (FC Byte = 0x0A)

10 – Freeze and Clear No Resp

See FREEZE_CLEAR (FC Byte = 0x09). No response message is returned when this request is issued from a DNP3 master.

COLD_RESTART (FC Byte = 0x0D)

13 – Cold Restart

This function code forces the controller to perform a complete restart upon powering up.

WARM_RESTART (FC Byte = 0x0E)

14 – Warm Restart

This function code forces the controller to perform a partial reset.

INITIALIZE_APPL (FC Byte = 0x10)

16 – Initialize Application

This function code is used to initialize the user program that was downloaded by Connected Components Workbench software.

START_APPL (FC Byte = 0x11)

17 – Start Application

This function code is used to start the user program that was downloaded by Connected Components Workbench software.

STOP_APPL (FC Byte = 0x12)

18 – Stop Application

This function code is used to stop the user program that was downloaded by Connected Components Workbench software.

ENABLE_UNSOLICITED (FC Byte = 0x14)

20 – Enable Unsolicited Message

This function is used to enable dynamically unsolicited messages generated in the controller.

DISABLE_UNSOLICITED (FC Byte = 0x15)

21 – Disable Unsolicited Message

This function is used to disable dynamically unsolicited messages generated in the controller.

DELAY_MEASURE (FC Byte = 0x17)

23 – Delay Measurement, used for Non-LAN Procedure

This function code is used to measure the communication channel delay time.

RECORD_CURRENT_TIME (FC Byte = 0x18)

24 – Record Current Time, used for LAN Procedure

This function code is used in the procedure for time synchronizing controllers that communicate over a LAN.

OPEN_FILE (FC Byte = 0x19)

25 – Open File

This function code is used to make a file available for reading or writing.

CLOSE_FILE (FC Byte = 0x1A)

26 – Close File

After the file read or write operation, this function code is used to unlock the file.

DELETE_FILE (FC Byte = 0x1B)

27 – Delete File

A DNP3 master uses this function code to delete a file.

GET_FILE_INFO (FC Byte = 0x1C)

28 – Get File Information

This function code is for the master to retrieve information about a file in the controller.

AUTHENTICATE_FILE (FC Byte = 0x1D)

29 – Authenticate File

This function code is used to obtain an authentication key that is required to open or delete a file.

ABORT_FILE (FC Byte = 0x1E)

30 – Abort File

This function code is used to immediately request termination of the current read/write operation and close the file, without saving.

ACTIVATE_CONFIG (FC Byte = 0x1F)

31 – Activate Config

This function code is used to begin using the configuration or executable code that is specified by the objects included in the request.

AUTHENTICATE_REQ (FC Byte = 0x20)

32 – Authentication Request

The master uses this function code when sending authentication messages to the controller that require a response

AUTHENTICATE_ERR (FC Byte = 0x21)

33 – Authentication Request No Resp

This function code is used by the master to send authentication messages when no return response is required.

RESPONSE (FC Byte = 0x22)

129 – Response

All responses except for Unsolicited Response messages use this function code.

UNSOLICITED_RESPONSE (FC Byte = 0x23)

130 – Unsolicited Response

Unsolicited Responses always use this function code regardless of which DNP3 objects are included.

AUTHENTICATE_RESPONSE (FC Byte = 0x24)

131 – Authentication Response

This function code is used to issue authentication messages to the master.

Internal Indications

Internal Indication bits are set under the following conditions of the controllers:

- IIN1.0: ALL_STATIONS. This bit is set when an all-stations message is received.
- IIN1.1: CLASS_1_EVENTS. This bit is set when Class 1 event data is available.
- IIN1.2: CLASS_2_EVENTS. This bit is set when Class 2 event data is available.
- IIN1.3: CLASS_3_EVENTS. This bit is set when Class 3 event data is available.
- IIN1.4: NEED_TIME. This bit is set when time synchronization is required.
- IIN1.5: LOCAL_CONTROL. This bit is set when the controller is in Non-executing mode.
- IIN1.6: DEVICE_TROUBLE. This bit is set when the controller is in Fault mode.
- IIN1.7: DEVICE_RESTART. This bit is set when the DNP3 driver is configured, in channel configuration, or when the controller has been restarted.
  To set this bit during the driver configuration and channel configuration, you must select the "Send Init. Unsol. Null Resp. on Restart" setting and set Status Bit S:36/13 to 1 before downloading to the controller.
- IIN2.0: NO_FUNC_CODE_SUPPORT. This bit is set when a request that has an unknown function code is received.
- IIN2.1: OBJECT_UNKNOWN. This bit is set when a request that has an unknown object is received.
- IIN2.2: PARAMETER_ERROR. This bit is set when a request with a qualifier/range field that cannot be processed is received.
- IIN2.3: EVENT_BUFFER_OVERFLOW. This bit is set when an event buffer overflow condition exists in the controller and at least one unconfirmed event is lost.
- IIN2.4: ALREADY_EXECUTING. Not supported.
- IIN2.5: CONFIG_CORRUPT. This bit is set when a bad file type and bad file number are detected.
- IIN2.6: Reserved.
- IIN2.7: Reserved.

DNP3 Objects and Controller Variables

All DNP3 Objects that are supported in the controller are summarized in .

Variables that are used in DNP3 Objects are not the same as that used in the controller, but are similar. Mapping is required between variables in DNP3 Objects and controller variables.

Overview

DNP3 Data objects that are implemented in the controller are listed as follows:

- DNP3 Binary Input Object
- DNP3 Double Bit Binary Input Object
- DNP3 Binary Output Object
- DNP3 Counter Object
- DNP3 Frozen Counter Object
- DNP3 Analog Input Object
- DNP3 Analog Output Object
- DNP3 BCD Object
- DNP3 Data-Set™ Object

Some of objects are divided into several Object files to map data in the controller.

- Counter Object — 16-bit and 32-bit Counter Object
- Analog Input Object — 16-bit and 32-bit Analog Input Object, and Short Floating Point Analog Input Object.
- Analog Output Object — 16-bit and 32-bit Analog Output Object, and Short Floating Point Analog Output Object.

DNP3 Mapping for Micro870 controllers
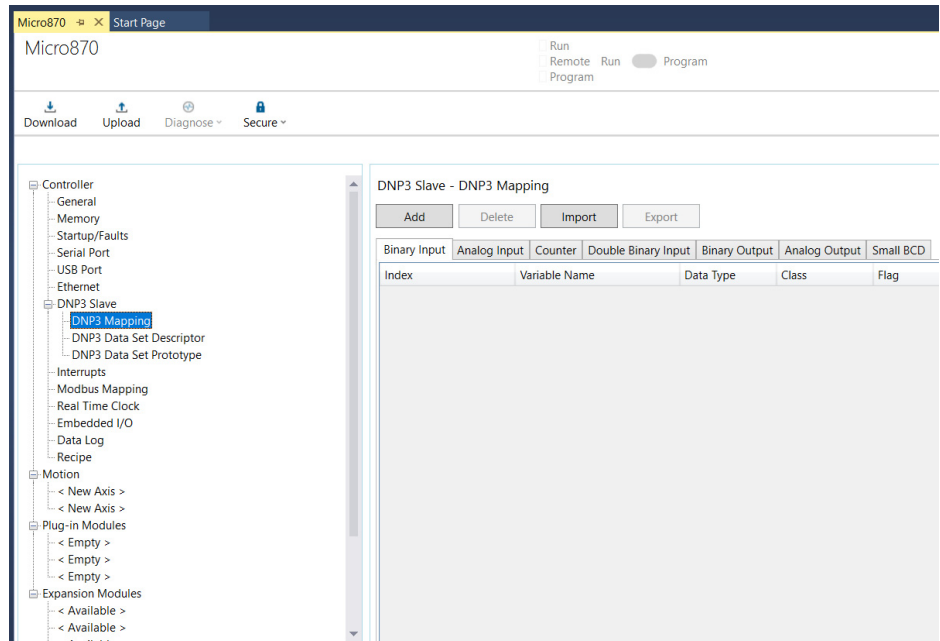
You can create the different data objects by mapping them to the variables created in the controller. You can configure the Data object for each DNP3 Object in the DNP3 Slave configuration page. Variables can be BOOL, INT, DINT, or REAL data types.

DNP3 Object Data

Table 22 - Relationship between DNP3 Object Database and Micro800 Variables

DNP Objects			Micro800 Variables	
Object Name	**Related Groups**	**Maximum Configurable Index**	**Data Name**	**Maximum Configurable Elements**
Binary Input Object	1, 2	4096	Binary Input Object	256
Double Bit Binary Input Object	3, 4	2048	Double Bit Binary Input Object	256
Binary Output Object	10, 12	4096	Binary Output Object	256
Counter Object	20, 22	256	16-bit Counter Object	256
			32 bit Counter Object	
Frozen Counter Object	21, 23	Reflection of Counter Object that was configured	Reflection of 16-bit Counter Object	—
			Reflection of 32-bit Counter Object	
Analog Input Object	30, 32	256	16-bit Analog Input Object	256
			32-bit Analog Input Object	
			Short Floating Point Analog Input Object	
Analog Output Object	40, 41	256	16-bit Analog Output Object	256
			32-bit Analog Output Object	
			Short Floating Point Analog Output Object	
BCD Object	101	256	Small BCD Object	256
Data-Set Object	85, 87, 88	10	Data-Set Prototypes Object	10
	86, 87, 88		Data-Set Descriptors Object	

DNP3 Configuration

You can configure parameters such as Class level and Object Flag bit information for each element. This information is defined during object creation in the DNP3 Mapping page under DNP3 Slave.

DNP3 Data Set Object

To create a Data Set Object from the DNP3 Subsystem in the controller, configure Data Set Prototypes/Descriptors Object in the DNP3 Data Set Descriptor/Prototype page under DNP3 Slave.
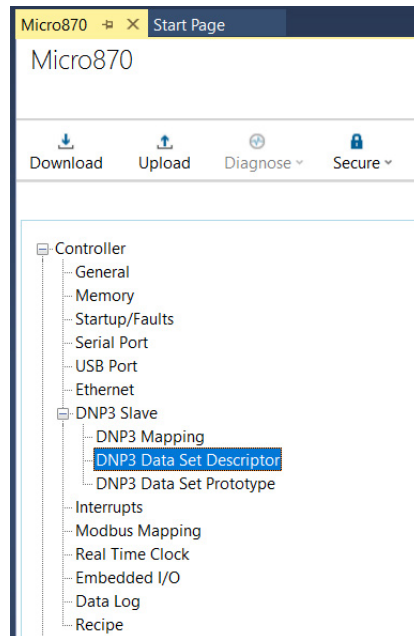
Each Data Set Prototypes Object can have up to 10 elements of Data Set Prototypes, and each Data Set Descriptors Object can have up to 10 elements of Data-set Descriptors.

As an example, with Data Set Prototypes entry, you can create any number of Data Set Prototype objects in the DNP3 Data Set Prototype configuration screen, up to a maximum of 10 entries.

Figure 26 - DNP3 Data Set Prototype Page
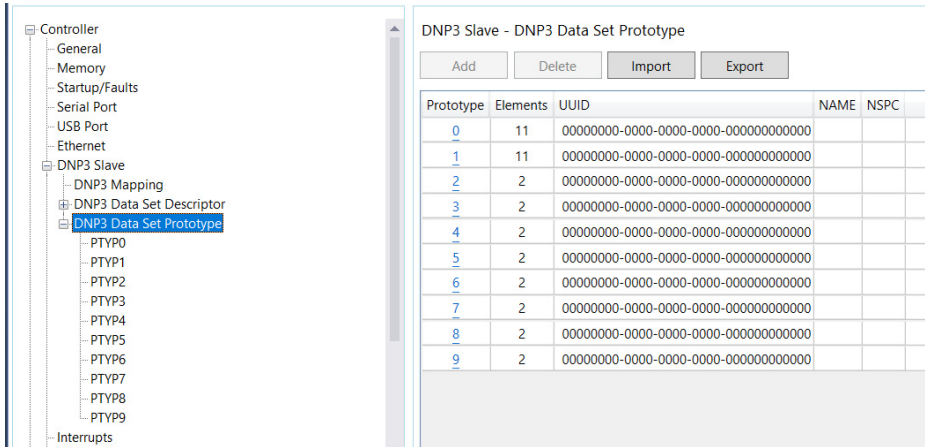
As an example, with Data Set Descriptors entry, you can create any number of Data Set Descriptor objects in the DNP3 Data Set Descriptor page, up to a maximum of 10 entries

Figure 27 - DNP3 Data Set Descriptor Page
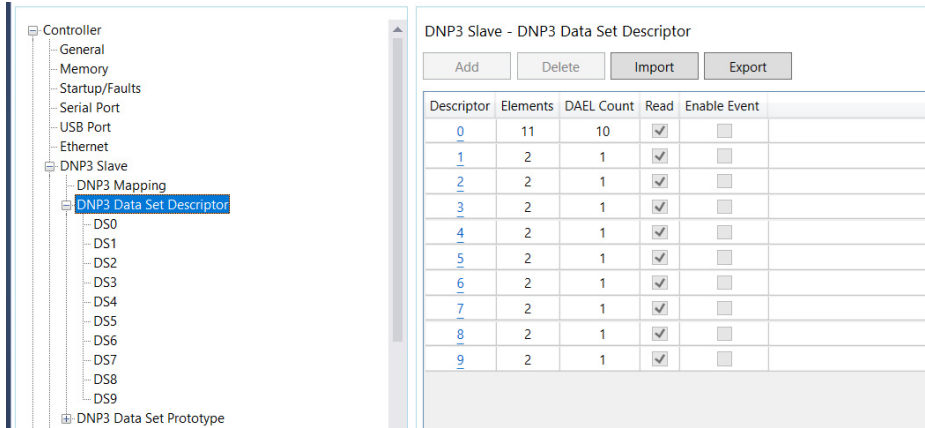
Once the Data Set Prototypes and Descriptors are configured in the DNP3 Slave setting page of Connected Components Workbench software version 20.01.00 or later, you can see the DNP3 Descriptor DSX and Prototype PTYPX under the respective DNP3 Data Set branch, where X is the element numbers of each Prototype or Descriptor.
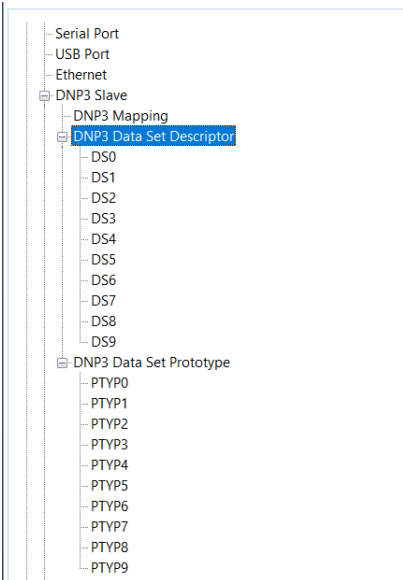
For DNP3 PTYPX, you can configure the controller to construct the Data Set Prototype objects.

For DNP3 DSX, you can configure the controller to construct the Data Set Descriptor objects.

Data Set Prototypes Configuration Parameters

These parameters are used to construct Data Set Prototypes object.

You can add or delete the index in each PTYPx entry.

- Descriptor Code: UUID for index 2. NSPC/NAME/DAEL for index 3 or higher.
- Data Type Code: NONE for index 2. VSTR/UINT/INT/FLT/OSTR/BSTR/TIME for index 3 or higher.
- Max Data Length (bytes): 0 for element 1. 0...255 for index 3 or higher.
- Ancillary Value: Binary Array in hexadecimal for element 1. ASCII strings for index 3 or higher. Maximum 32 bytes.

Data-Set Descriptors Configuration Parameters

These parameters are used to construct Data-Set Descriptors objects.

The Read and Enable Event check boxes allow assignment of characteristics to this Descriptor object.

- Read: Set if the Data-Set is readable.
- Enable Event: Set if the outstation generates a Data-Set event.

With Enable Event checked, you can assign an Event Class to this Descriptor object.

- 1: Class 1
- 2: Class 2
- 3: Class 3

Additional Event parameters:

- Trigger Event: Define and set this parameter to generate an event by the ladder logic to generate timed events. Once the ladder logic or communications sets this parameter, the controller clears it automatically after generating an event at the end of a scan.
- Disable Change Event: Define and set this parameter to suppress the events generated by any Event Occurrence Condition.
- Event Occurrence Condition: The conditions of Data-Set Event for each Data-Set Descriptor can be configured by Data-Set Event Occurrence Condition 1/2/3/4 in the DNP3 Data-Set Descriptors Object. When one of the values that are pointing to the Event Occurrence Condition 1/2/3/4 are changed or the criteria are met, the controller generates a Data-Set Event, retrievable using the object Group 88, Variation 1.

[Table 23 on page 104](#) shows the supported conditions for Point Addressing under Event Occurrence Conditions.

Table 23 - Event Occurrence Condition for Supported Point Addressing

Point Address Type	Point Type	Point Index	Event Occurrence Condition
Empty	NONE: No point type is associated	0	No event is generated.
Standard DNP3 Point	BI: Binary input	0...4095	When the Point Type and Point Index point to a specific point, if the value of the point changes, an event is generated.
	B2I: Double-bit input	0...2047	
	CI: Counter	0...511	
	AI: Analog input	0...767	
	BCD: BCD point	0...255	
Variable	Select the tag from the controller	0	An event is generated based on the condition of the variable tag selected.

A Data-set event can consume any number of event buffers, depending on the Data-set configuration. This is only applicable for Data-set events. The event for other objects consumes one event buffer. When using Data-set events, increase the number of events in the DNP3 Slave configuration.

Descriptor Element Configuration: Each Descriptors element is configured by clicking on the individual DSX under the DNP3 Data-set Descriptor. Add new index by clicking Add in each DSX.
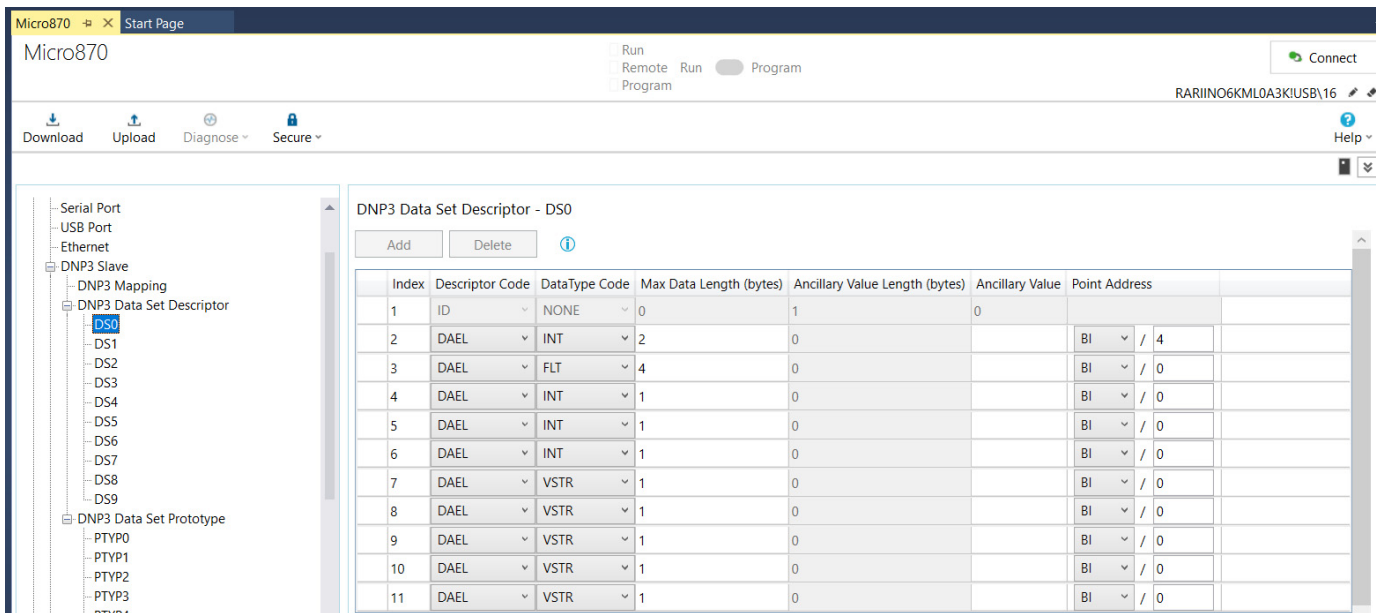
Descriptor Code: NAME, DAEL, PTYP

Data Type Code: VSTR, UINT, INT, FLT, OSTR, BSTR, TIME

Max Data Length (bytes): 0...255

Ancillary Value: Any string. This can be a binary array or ASCII string, up to 32 bytes.

Point Addressing under Descriptor Element Configuration: Data-Set value for each Data-Set element is configured by:
- Point Address Type
- Point Index

When these values are configured properly according to the supported data files, the controller responds with a Group 87, Variation 1 object filled with the value in the data file. shows the supported data files for the Point Addressing.

Table 24 - Point Address Type — Standard DNP3 Point

Point Address Type	Data Type Code	Maximum Data Length (bytes)	Point Type	Point Index Low Byte	Point Index High Byte
Standard DNP3 Point	NONE = 0	0	NONE = 0: No point type is associated.	0	
	NONE = 0 UINT = 2 INT = 3 OSTR = 5 BSTR = 6 TIME = 7	0 0, 1, 2, or 4 0, 1, 2, or 4 0…255 0…255 0…6	BI = 1: Binary input	0…4095 max When using Data Types other than OSTR and BSTR, the Point Index must be set to a point offset that is divisible by 16.	
			B2I = 3: Double-bit input	0…2047 max When using Data Types other than OSTR and BSTR, the Point Index must be set to a point offset that is divisible by 8.	
			CI = 20: Counter	0…511 max	
			AI = 30: Analog input	0…767 max	
			BCD = 101: BCD point	0…255 max	

When the Descriptor Code is selected as PTYP, the Point Addressing parameters for the Descriptor element are replaced by 10 Point Addressing parameters. These should be configured in the same order of the DAEL elements in the relevant Prototypes.

For instance, if Prototype 0 includes a Namespace at Index 3 and Name at Index 4, then the first DAEL in the Prototype 0 is at Index 5.

Object Quality Flags

The object flag is composed of an 8-bit string for some DNP3 objects. The tables below show Flag Descriptions for each object. The ONLINE, RESTART, COMM_LOST, REMOTE_FORCED and LOCAL_FORCED flags are common to all object group types that contain flags.

There are some rules for the Object flag set or clear for each bit by the controller. The rules below are also applied to Event data.

- When the controller is in Non-executing mode, the object flag is always all 0.
- When the controller is in Executing mode and there is no configuration file, only the Online flag in the object flag is set.
- When the controller is in Executing mode and there is a configuration file, the flags in the object flag are set according to the upper byte of the configuration files.

Table 25 - Object Flags for Binary Input

Bit Offset	Name	Description
0	ONLINE	0 when the controller is or was in Non-executing mode. 1 when the controller is or was in Executing mode and the configuration file does not exist. May be 1 when the controller is or was in Executing mode and the configuration file exists.
1	RESTART	Always 0. Not used.
2	COMM_LOST	Always 0. Not used.
3	REMOTE_FORCED	Always 0. Not used.
4	LOCAL_FORCED	Always 0. Not used.
5	CHATTER_FILTER	Always 0. Not used.
6	Reserved	Always 0. Not used.
7	STATE	Reflects point state of Binary Input point.

Table 26 - Object Flags for Double Binary Input

Bit Offset	Name	Description
0	ONLINE	0 when the controller is or was in Non-executing mode. 1 when the controller is or was in Executing mode and the configuration file does not exist. May be 1 when the controller is or was in Executing mode and the configuration file exists.
1	RESTART	Always 0. Not used.
2	COMM_LOST	Always 0. Not used.
3	REMOTE_FORCED	Always 0. Not used.
4	LOCAL_FORCED	Always 0. Not used.
5	CHATTER_FILTER	Always 0. Not used.
6	STATE	Reflects point state of Double-bit Binary Input point. Double-bit LSB.
7	STATE	Reflects point state of Double-bit Binary Input point. Double-bit MSB

Table 27 - Object Flags for Binary Output

Bit Offset	Name	Description
0	ONLINE	0 when the controller is or was in Non-executing mode. 1 when the controller is or was in Executing mode and the configuration file does not exist. May be 1 when the controller is in Executing mode and the configuration file exists.
1	RESTART	Always 0. Not used.
2	COMM_LOST	Always 0. Not used.
3	REMOTE_FORCED	Always 0. Not used.
4	LOCAL_FORCED	Always 0. Not used.
5	Reserved	Always 0. Not used.
6	Reserved	Always 0. Not used.
7	STATE	Reflects point state of Binary Output point.

Table 28 - Object Flags for Counter

Bit Offset	Name	Description
0	ONLINE	0 when the controller is or was in Non-executing mode. 1 when the controller is or was in Executing mode and the configuration file does not exist. May be 1 when the controller is in Executing mode and the configuration file exists.
1	RESTART	0 when the controller is or was in Non-executing mode. 0 when the controller is or was in Executing mode and the configuration file does not exist. May be 1 when the controller is in Executing mode and the configuration file exists.
2	COMM_LOST	
3	REMOTE_FORCED	
4	LOCAL_FORCED	
5	ROLLOVER	
6	DISCONTINUITY	
7	Reserved	

Table 29 - Object Flags for Analog Input

Bit Offset	Name	Description
0	ONLINE	0 when the controller is or was in Non-executing mode. 1 when the controller is or was in Executing mode and the configuration file does not exist. May be 1 when the controller is in Executing mode and the configuration file exists.
1	RESTART	0 when the controller is or was in Non-executing mode. 0 when the controller is or was in Executing mode and the configuration file does not exist. May be 1 when the controller is in Executing mode and the configuration file exists.
2	COMM_LOST	
3	REMOTE_FORCED	
4	LOCAL_FORCED	
5	OVER_RANGE	
6	REFERENCE_ERR	
7	Reserved	

Table 30 - Object Flags for Analog Output

Bit Offset	Name	Description
0	ONLINE	0 when the controller is or was in Non-executing mode. 1 when the controller is or was in Executing mode and the configuration file does not exist. May be 1 when the controller is in Executing mode and the configuration file exists.

Table 30 - Object Flags for Analog Output (Continued)

Bit Offset	Name	Description
1	RESTART	0 when the controller is or was in Non-executing mode. 0 when the controller is or was in Executing mode and the configuration file does not exist. May be 1 when the controller is in Executing mode and the configuration file exists.
2	COMM_LOST	
3	REMOTE_FORCED	
4	LOCAL_FORCED	
5	Reserved	
6	Reserved	
7	Reserved	

DNP3 Device Attribute Object

The Device Attribute object can be used to identify DNP3 Slave devices. With the controller, some of the variations are written so that you can read or write your own strings in your application.

The R/W property shows if the object is Read-only, Read, or Write. If the R/W property is writable, the value that was written by the DNP3 master device is stored to nonvolatile memory.

The object group of the Device Attribute is 0. The supported range of the variation is 209...255.

Table 31 - Object Group 0, Variations for Attribute Set 0

Variation	Read/Write	Attribute Data Type	Max Length in Bytes	Description	Value
209	Read-only	UINT	1	Secure authentication version supported in out station	5
210	Read-only	UINT	1	Number of secure statistic available in out station	18
212	Read-only	UINT	1	Number of master-defined Data-Set prototypes	1
213	Read-only	UINT	1	Number of outstation-defined Data-Set prototypes	1
214	Read-only	UINT	1	Number of master-defined Data-Sets	1
215	Read-only	UINT	1	Number of outstation-defined Data-Sets	1
216	Read-only	UINT	1	Max number of binary outputs per request	1
217	Read-only	UINT	4	Local timing accuracy	100 in microseconds
218	Read-only	UINT	4	Duration of timing accuracy	100 in seconds
219	Read-only	INT	1	Support for analog output events	1
220	Read-only	UINT	4	Max analog output index	0
221	Read-only	UINT	4	Number of analog outputs	0
222	Read-only	INT	1	Support for binary output events	1
223	Read-only	UINT	4	Max binary output index	0
224	Read-only	UINT	4	Number of binary outputs	0
225	Read-only	INT	1	Support for frozen counter events	1
226	Read-only	INT	1	Support for frozen counters	1
227	Read-only	INT	1	Support for counter events	1
228	Read-only	UINT	4	Max counter index	0
229	Read-only	UINT	4	Number of counter points	0
230	Read-only	INT	1	Support for frozen analog inputs	1
231	Read-only	INT	1	Support for analog input events	1
232	Read-only	UINT	4	Maximum analog input index	0
233	Read-only	UINT	4	Number of analog input points	0
234	Read-only	INT	1	Support for double-bit binary input events	1
235	Read-only	UINT	4	Maximum double-bit binary input index	0
236	Read-only	UINT	4	Number of double-bit binary input points	0
237	Read-only	INT	1	Support for binary input events	1
238	Read-only	UINT	4	Max binary input index	0
239	Read-only	UINT	4	Number of binary input points	0
240	Read-only	UINT	4	Max transmit fragment size	Response size

Table 31 - Object Group 0, Variations for Attribute Set 0 (Continued)

Variation	Read/Write	Attribute Data Type	Max Length in Bytes	Description	Value
241	Read-only	UINT	4	Max receive fragment size	Response size
242	Read-only	VSTR	5	Device manufacturer's software version	This variation returns firmware FRN. FRN 1.00. Supported ranges: FRN x.yy, FRN x.yyy, FRN xx.yy, or FRN xx.yyy Where x or xx is 0...99, and yy or yyy is 00...999. For example, FRN 1.00, FRN 1.05, FRN 12.05, FRN 102.27, or FRN 103.117.
243	Read-only	VSTR	6	Device manufacturer's hardware version	Supported ranges: HW SER x/REV yy Where x is A...F and yy is 00...31. For example, HW SER A/REV 01, HW SER B/REV 03, or HW SER C/REV 31.
244	–	–	–	Reserved for future assignment	–
245	Read /Write	VSTR	length of the string value, max 255 bytes	User-assigned location name	Default
246	Read/Write	VSTR	length of the string value, max 255 bytes	User-assigned ID code/number	Default
247	–	–	–	Reserved for future assignment	–
248	–	–	–	Reserved for future assignment	–
249	Read-only	VSTR	6	DNP subset and conformance	This variation returns Subset level and Test procedure version 3:2004.
250	Read-only	VSTR	length of the string value	Device manufacturer's product name and model	Supported ranges: 2080-L70E-24xxxN SER A Where xxx is QWB or QBB. For example, 2080-L70E-QWBN SER A or 2080-L70E-24QBBN SER A.
251	–	–	–	Reserved for future assignment	–
252	Read-only	VSTR	19	Device manufacturer's name	This variation returns the Company name, Rockwell Automation.
253	Read-only	–	–	Reserved for future assignment	–
254	Read-only	–	–	Non-specific all attributes request	This variable returns all variations in this group except this variation.
255	Read-only	–	–	List of attribute variations	This variation returns the R/W property for each variation. 0 for Read-only 1 for Read or Write

Event Reporting

This section covers how to generate DNP3 events from DNP3 Data Objects and how to report the generated events by polled response or unsolicited response.

Generate Events

The controller has a separate buffer area that you can use to log DNP3 events internally.

The maximum number of events that can be logged is 10000 (see DNP3 10K Event Logging on page 111), regardless of the Event data type. In 2080-L70E-24Q*x*BN controllers, a Data-Set event can consume multiple numbers of the event buffer.

If the number of the generated events reaches this value, the controller sets IIN2.3 [EVENT_BUFFER_OVERFLOW]. Further events are not logged until the logged events are reported to the DNP3 Master and the buffer is available.

The logged events are not removed until they are reported to the DNP3 Master successfully. Logged events can also be cleared when one of the following events occur:

• New OS firmware update
• New user program download

Figure 28 on page 110 shows how to generate events for a Binary Input Object and a 16-bit Analog Input Object. In the DNP3 Slave configuration, Binary Input Object Data and 16-bit Analog Input Object Data is configured in the DNP3 mapping table.

Figure 28 - Generate Events for Binary Input Object
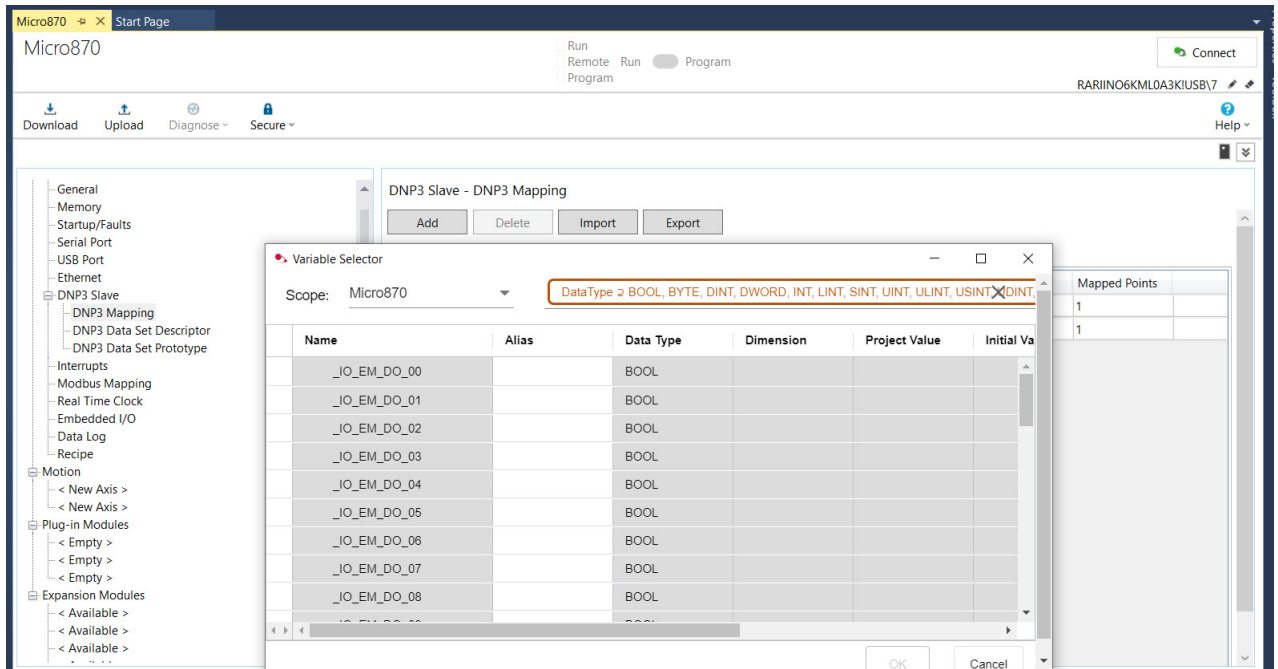
Select Add, then select a variable from the Variable Selector dialog for the Binary Input Object.

Figure 29 - Data Types for Binary Input Object Example

There are two methods to create Binary Input Object. You can create the object as a BOOL data type or as a UINT data type. The UINT data type represents similar to a BOOL array of 16 bits and this representation is similar to the BOOL array used in MicroLogix™ 1400 controllers.

Figure 30 - Class Events for Binary Input Object Example

Figure 30 shows the different class events that can be triggered when the BOOL variable is triggered. When using the UINT variable, any of the bits in the UINT trigger the respective class event.

In the same manner, create this 16-bit Analog Input Object in the Analog Input tab and define Index 0 as Class 1, Index 1 as Class 2, and Index 2 as Class 3.

Figure 31 - Analog Input Object Example

DNP3 10K Event Logging

Up to 10,000 events can be logged. To configure the number of events, change the Unsolicited Responses for the Class 1/2/3 to Enabled and define 10,000 into the Number of Events field. Download the project into the controller for the setting to take effect.

Figure 32 - Configure Number of Events to Log

When there is a change in the configuration for the number of events, you must download the project into the controller for the setting to take effect.

Control Generating Event

The controller checks all elements in the Object Data for changes at the end of a scan and generates events where needed.

The key method to turn on and off event generating by ladder logic is to assign the variable to be used for controlling into the Trigger Event and Disable Change Events field.

shows how to control the event generation condition by using the Deadband for Analog Input Objects.

In this example, for 16-bit Analog Input point 0, if the absolute value of the difference between the present value of the variable AnalogValue1 and the value that was most recently queued as an event for that point exceeds the deadband value in variable AnalogV1Deadband, then an event is generated for that point.

Figure 33 - Control Event Generation Example

Report Event By Polled Response

When a DNP3 Master sends a poll to read Class events, any events that are logged to the event buffer are reported in the polled response.

Define the Number of Events for the respective Class event in the DNP3 Slave configuration page as shown in .

Figure 34 - Define Number of Events for each Class Event

Report Event By Unsolicited Response

To initiate and send Unsolicited Responses to a DNP3 Master, configure the following parameters correctly. For more information, see .

- Master Node 0
- Channel for Unsolicited Response
- Enable Unsolicited On Start Up
- Enable Unsolicited For Class1
- Enable Unsolicited For Class2
- Enable Unsolicited For Class3
- Send Initial Unsolicited On Start Up
- Number of Class1 Events
- Hold Time after Class1 Events (x1s)
- Number of Class2 Events
- Hold Time after Class2 Events (x1s)
- Number of Class3 Events
- Hold Time after Class3 Events (x1s)

- DNP3 Object Data
- DNP3 Object Config
- Content of the Config File

In some cases, the controller may not send an Unsolicited Response even though the parameters are configured properly.

- Normally, when the parameter Enable Unsolicited On Start Up is checked, the controller initiates an Unsolicited Response with the function code ENABLE_UNSOLICITED(20), if there are any events that are logged into the event buffer. However, when a request with the function code DISABLE_UNSOLICITED(21) is received, an Unsolicited Response is not sent.
- When the parameter Enable Unsolicited On Start Up is unchecked, the controller does not trigger the Unsolicited Response until a request with the function code ENABLE_UNSOLICITED(20) from the DNP3 Master is received.

Figure 35 shows how to initiate and send the Unsolicited Response. Master Node 0 in the DNP3 Slave configuration page indicates that the Unsolicited Response is reported to the Master with the node address 3.

Figure 35 - Initiate and Send Unsolicited Response

The parameter for Unsolicited Response in the DNP3 Slave configuration indicates that the Unsolicited Response is reported for all communication ports (Serial and Ethernet) that are defined. In this example, an Initial Unsolicited Response is sent on startup and all events of Class 1, 2, and 3 are reported. Since Hold Times are configured to 5 seconds, generated events are reported after 5 seconds.

Collision Avoidance

The controller supports the following two methods for collision avoidance.

- Detecting transmitted data (TX/RX line on RS-485 communication)
- Detecting out-of-band carrier (DCD on RS-232C communication)

When the controller is connected to a RS-485 network, it monitors all data on the link. If the controller is preparing to transmit a packet and finds the link busy, it waits for an interval that is defined by the Backoff_Time until it is no longer busy.

Backoff_Time = Pre Transmit Delay (x1 ms) + Max Random Delay (x1 ms)

The Pre Transmit Delay (x1 ms) in the Link Layer Channel Configuration file is a fixed delay and the Max Random Delay (x1 ms) in the Channel Configuration file is a maximum random delay for Channel 0 and Channel 2. You must specify those parameters to get the collision avoidance mechanism.

After the Backoff_Time, the controller tries again, either indefinitely, or up to a configurable maximum number of retries. If a maximum is used, the protocol considers this as a link failure.

Time Synchronization

The time value in the real-time clock (RTC) of the controller (firmware real-time clock) or plug-in module (2080-MEMBAK-RTC2 or 2080-SDMEMRTC-SC) is updated every second.

The DNP3 subsystem and the RTC (firmware or plug-in module) are synchronized by the following conditions:

- At power-up
- A request for time synchronization from a DNP3 Master

At power-up, the DNP3 subsystem gets the time from the controller. For the controller to acquire the correct time, a plug-in module RTC should be used and enabled before a power cycle to acquire the correct time from the controller. The firmware RTC always reverts to a default time and date after a power cycle, so it is not able to provide an accurate time to the DNP3 subsystem.

If a plug-in module RTC is not used, then the firmware RTC must be enabled. To enable firmware RTC, select Enable firmware real-time clock.
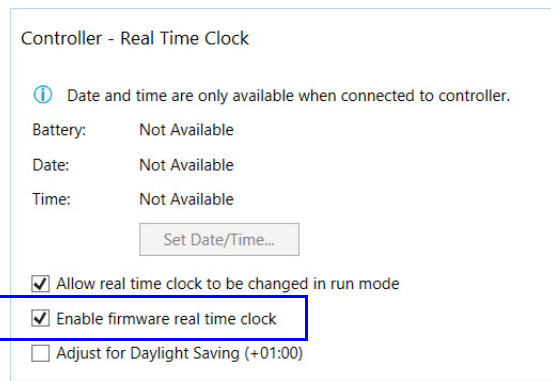
When there is a write request for time synchronization from a DNP3 Master, the time in the RTC (firmware or plug-in module) is synchronized with the time from the DNP3 Master.

Table 32 shows the accuracy of the 2080-MEMBAK-RTC2 plug-in module. Use these values to configure the Time Synchronization Interval in the Micro870 DNP3 Slave configuration, so that a DNP3 Master can send the time synchronization request periodically for more accurate times in the controller.

Table 32 - 2080-MEMBAK-RTC2 Accuracy

RTC Accuracy[1]	Ambient Temperature
±5 sec/month	25 °C (77 °F)
±9 sec/month	-20…+65 °C (-4…+149 °F)

(1)   These numbers are maximum worst case values over a 31-day month.

Adjust for Daylight Saving

Use this feature to shift the controller clock by one hour during the summer time. To enable this feature, select Adjust for Daylight Saving and download the project to the controller. You can also change this setting during runtime.

This feature is available for Micro850 (2080-L50E) and Micro870 (2080-L70E) controllers with Connected Components Workbench software version 22.00 or later.
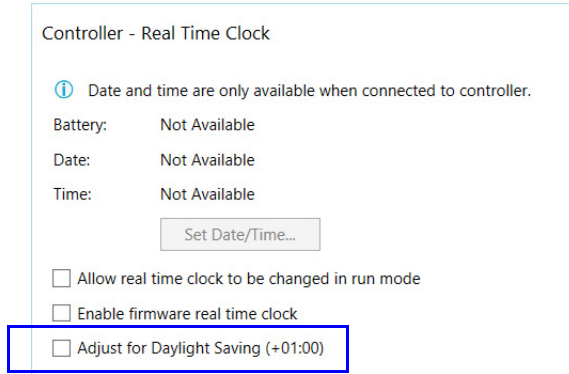
Diagnostics

Errors in a DNP3 Slave subsystem are logged in the Communication Diagnostics page in Connected Components Workbench software.

Figure 36 - Communication Diagnostics for Embedded Serial Port

Figure 37 - Communication Diagnostics for 2080-SERIALISOL Plug-in Module

Micro870 - Communication Diagnostics	Micro870		
Communication:	Serial Port		Reset Counters
Channel:	Slot 2 | 2080-SERIALISOL at port 6		
Driver:	DNP3 Slave		

Diagnostic Counters

Channel Status:	NAK Messages Received:
Messages Received:	Messages Sent:
Messages Retried:	Messages Undelivered:
Bytes Received on Channel:	Bytes Sent on Channel:
Frames Received on Channel:	Frames Sent on Channel:
Fragments Received on Channel:	Fragments Sent on Channel:
Link Layer Error Count:	Last Link Layer Error Code:
Application Layer Error Count:	Last Application Layer Error Code:
Transport Layer Error Count:	Last Transport Layer Error Code:
Physical Layer Errors:	Last Physical Layer Error Code:
Current Session Status:	Session Events Overflowed:
Session Events Confirmed:	Session Events Sent:

Diagnostics for Ethernet Port

Diagnostic Counters and Errors in the DNP3 Slave subsystem for the Ethernet port are shown in the Connected Components Workbench software. Select Diagnose to display the Diagnostic page.

Figure 38 - Communication Diagnostics for Ethernet Port

Micro870 - Communication Diagnostics	Micro870		
Communication:	Ethernet		Reset Counters
Protocols:	DNP3 Slave		

Diagnostic Counters

Channel Status:	0	NAK Messages Received:	0
Messages Received:	0	Messages Sent:	0
Messages Retried:	0	Messages Undelivered:	0
Bytes Received on Channel:	0	Bytes Sent on Channel:	0
Frames Received on Channel:	0	Frames Sent on Channel:	0
Fragments Received on Channel:	0	Fragments Sent on Channel:	0
Link Layer Error Count:	0	Last Link Layer Error Code:	0
Application Layer Error Count:	0	Last Application Layer Error Code:	0
Transport Layer Error Count:	0	Last Transport Layer Error Code:	0
Physical Layer Errors:	0	Last Physical Layer Error Code:	0
Current Session Status:	0	Session Events Overflowed:	0
Session Events Confirmed:	0	Session Events Sent:	0
Session Request Timeouts:	0	Session Requests Failed:	0
Session Link Status Requests Received:	0	Session Link Status Frames Received:	0

Table 33 - Error Codes

Value (Dec)	Mnemonic	Description
0	ERROR_PHYS_TRANSMIT	Error returned from target transmit routine
1	ERROR_PHYS_CHAR_TIMEOUT	Intercharacter timeout occurred
2	ERROR_PHYS_REMOTE_CLOSE	Remote side of channel closed connection
3	ERROR_LINK_FRAME_LENGTH	Incoming frame too short or exceeded buffer size
4	ERROR_LINK_ADDRESS_UNKNOWN	Received frame was for an unknown link address
5	ERROR_LINK_ILLEGAL_FUNCTION	Illegal link function code in received frame
6	ERROR_LINK_INVALID_CHECKSUM	Invalid checksum or CRC
7	ERROR_LINK_NOT_RESET	Link has not been reset, frame rejected
8	ERROR_LINK_FCB	Received invalid frame count bit
9	ERROR_LINK_INVALID_START_CHAR	Did not receive correct starting sync char
10	ERROR_LINK_FRAME_TIMEOUT	Entire frame was not received in specified time
11	ERROR_LINK_CNFM_TIMEOUT	Link Confirm was not received in specified time
12	ERROR_LINK_STATUS_TIMEOUT	Link status response not received in specified time
The following link errors are used by 101/103		
13	ERROR_LINK_WRONG_SESN	Response was not from expected session
14	ERROR_LINK_WRONG_REPLY	Received unexpected reply, frame rejected
15	ERROR_LINK_INVALID_2ND_CHAR	Did not receive correct second sync char
16	ERROR_LINK_INVALID_END_CHAR	Did not receive correct ending sync character
17	ERROR_LINK_MISMATCHING_LENGTH	Variable length bytes in FT1.2 frame did not match
18	ERROR_LINK_INV_DIR	Received invalid dir bit in control octet
The following link errors are used by 104		
19	ERROR_LINK_NO_CNFM_RECEIVED	Confirmation of 104 U-format APDU not received
20	ERROR_LINK_NO_ACK_RECEIVED	Acknowledgment of 104 I-format APDU not received
21	ERROR_LINK_SEQUENCE_UNKNOWN	Unknown confirming sequence number in received APDU
22	ERROR_LINK_OUT_OF_SEQUENCE	Received APDU not in sequence with previous APDU
Transport layer errors		
23	ERROR_TPRT_SEQUENCE_ERROR	Sequence number error

Diagnostics for Secure Authentication

Diagnostic information for Secure Authentication is shown in Connected Components Workbench software. Select Diagnose to display the Diagnostic page.

Figure 39 – Communication Diagnostics for Secure Authentication

Micro870 - Communication Diagnostics		Micro870	
Diagnostic Counters			
Channel Status:	1	NAK Messages Received:	0
Messages Received:	0	Messages Sent:	0
Messages Retried:	0	Messages Undelivered:	0
Bytes Received on Channel:	0	Bytes Sent on Channel:	0
Frames Received on Channel:	0	Frames Sent on Channel:	0
Fragments Received on Channel:	0	Fragments Sent on Channel:	0
Link Layer Error Count:	0	Last Link Layer Error Code:	0
Application Layer Error Count:	0	Last Application Layer Error Code:	0
Transport Layer Error Count:	0	Last Transport Layer Error Code:	0
Physical Layer Errors:	0	Last Physical Layer Error Code:	0
Current Session Status:	0	Session Events Overflowed:	0
Session Events Confirmed:	0	Session Events Sent:	0
Session Request Timeouts:	0	Session Requests Failed:	0
Session Link Status Requests Received:	0	Session Link Status Frames Received:	0
Session Secure Authentication Messages Received:	0	Session Secure Authentication Messages Sent:	0
Session Secure Authentication Response Timeouts:	0	Session Authentication Key Exchanges:	0

Function Codes

Table 34 shows the Application Layer Function codes that are implemented in the controller.

Table 34 – Function Codes for DNP3 in Micro870 Controllers

Message Type	Function Code	Name	Micro870 Support	Description
Confirmation	0 (0x00)	CONFIRM	Yes	Controller parses/sends
Request	1 (0x01)	READ	Yes	Controller parses
Request	2 (0x02)	WRITE	Yes	Controller parses
Request	3 (0x03)	SELECT	Yes	Controller parses
Request	4 (0x04)	OPERATE	Yes	Controller parses
Request	5 (0x05)	DIRECT_OPERATE	Yes	Controller parses
Request	6 (0x06)	DIRECT_OPERATE_NR	Yes	Controller parses
Request	7 (0x07)	IMMED_FREEZE	Yes	Controller parses
Request	8 (0x08)	IMMED_FREEZE_NR	Yes	Controller parses
Request	9 (0x09)	FREEZE_CLEAR	Yes	Controller parses
Request	10 (0x0A)	FREEZE_CLEAR_NR	Yes	Controller parses
Request	11 (0x0B)	FREEZE_AT_TIME	No	—
Request	12 (0x0C)	FREEZE_AT_TIME_NR	No	—
Request	13 (0x0D)	COLD_RESTART	Yes	Controller parses. Controller should not be in the executing mode and any program and files should not be in open state.
Request	14 (0x0E)	WARM_RESTART	No	Controller parses
Request	15 (0x0F)	INITIALIZE_DATA	No	Obsolete
Request	16 (0x10)	INITIALIZE_APPL	Yes	Controller parses. Clears the fault and changes the controller mode to Remote Program.
Request	17 (0x11)	START_APPL	Yes	Controller parses. Clears the fault and changes the controller mode to Remote Run.
Request	18 (0x12)	STOP_APPL	Yes	Controller parses. Changes the controller mode to Remote Program.

Table 34 - Function Codes for DNP3 in Micro870 Controllers (Continued)

Message Type	Function Code	Name	Micro870 Support	Description
Request	19 (0x13)	SAVE_CONFIG	No	Deprecated
Request	20 (0x14)	ENABLE_UNSOLICITED	Yes	Controller parses
Request	21 (0x15)	DISABLE_UNSOLICITED	Yes	Controller parses
Request	22 (0x16)	ASSIGN_CLASS	No	—
Request	23 (0x17)	DELAY_MEASURE	Yes	Controller parses. Used for non-LAN
Request	24 (0x18)	RECORD_CURRENT_TIME	No	Controller parses. Used for LAN
Request	25 (0x19)	OPEN_FILE	Yes	Controller parses
Request	26 (0x1A)	CLOSE_FILE	Yes	Controller parses
Request	27 (0x1B)	DELETE_FILE	Yes	Controller parses
Request	28 (0x1C)	GET_FILE_INFO	No	Controller parses
Request	29 (0x1D)	AUTHENTICATE_FILE	Yes	Controller parses
Request	30 (0x1E)	ABORT_FILE	No	Controller parses
Request	31 (0x1F)	ACTIVATE_CONFIG	No	Controller parses
Request	32 (0x20)	AUTHENTICATE_REQ	No	Controller parses
Request	33 (0x21)	AUTHENTICATE_ERR	No	Controller parses
Response	34 (0x22)	RESPONSE	Yes	Controller sends
Response	35 (0x23)	UNSOLICITED_RESPONSE	Yes	Controller sends
Response	36 (0x24)	AUTHENTICATE_RESPONSE	No	Controller sends
	37 (0x25)...47 (0x2F)		No	Reserved

Implementation Table

Micro870 (2080-L70E-24QxBN) controllers support DNP3 Certification Subset Level 2.

Table 35 identifies which object groups and variations, function codes, and qualifiers the device supports in both requests and responses. The Request and Response columns identify all requests and responses that may be sent/parsed by a DNP3 Master, or must be parsed/sent by the controller.

The implementation table lists all functionality that is required by either a DNP3 Master or controller as defined within the DNP3 IED Conformance Test Procedures. Any functionality beyond the highest subset level that is supported is indicated by the gray table cells.

Table 35 - Implementation Table for Micro870 controllers

DNP Object Group and Variation			Request DNP3 Master may issue, controller must parse		Response DNP3 Master must parse, controller may issue	
Group Num	Var Num	Description	Function Codes (Dec)	Qualifier Codes (Hex)	Function Codes (Dec)	Qualifier Codes (Hex)
0	211...239 241...243 248...250 252	Device Attribute	1 (read)	00, 01 (start-stop) 06 (no range, or all)	129 (response)	00 (start-stop)
0	240 245...247	Device Attribute	1 (read)	00, 01 (start-stop) 06 (no range, or all)	129 (response)	00 (start-stop)
			2 (write)	00, 01 (start-stop)		
0	254	Device Attribute – Non-specific all attributes request	1 (read)	00, 01 (start-stop) 06 (no range, or all)		
0	255	Device Attributes – List of attribute variations	1 (read)	00, 01 (start-stop) 06 (no range, or all)	129 (response)	00 (start-stop)
1	0	Binary Input – Any Variation	1 (read)	00, 01 (start-stop)		
				06 (no range, or all)		
1	1	Binary Input – Packed format	1 (read)	00, 01 (start-stop) 06 (no range, or all)	129 (response)	00, 01 (start-stop)
1	2	Binary Input – With flags	1 (read)	00, 01 (start-stop) 06 (no range, or all)	129 (response)	00, 01 (start-stop)
2	0	Binary Input Event – Any Variation	1 (read)	06 (no range, or all) 07, 08 (limited qty)		
2	1	Binary Input Event – Without time	1 (read)	06 (no range, or all) 07, 08 (limited qty)	129 (response) 130 (unsol. resp)	17, 28 (index)

Table 35 - Implementation Table for Micro870 controllers (Continued)

Group Num	Var Num	Description	Function Codes (Dec)	Qualifier Codes (Hex)	Function Codes (Dec)	Qualifier Codes (Hex)
DNP Object Group and Variation			**Request** DNP3 Master may issue, controller must parse		**Response** DNP3 Master must parse, controller may issue	
2	2	Binary Input Event – With absolute time	1 (read)	06 (no range, or all) 07, 08 (limited qty)	129 (response) 130 (unsol. resp)	17, 28 (index)
2	3	Binary Input Event – With relative time	1 (read)	06 (no range, or all) 07, 08 (limited qty)	129 (response) 130 (unsol. resp)	17, 28 (index)
3	0	Double-bit Binary Input – Any Variation	1 (read)	00, 01 (start-stop) 06 (no range, or all)		
3	1	Double-bit Binary Input – Packed format	1 (read)	00, 01 (start-stop) 06 (no range, or all)	129 (response)	00, 01 (start-stop)
3	2	Double-bit Binary Input – With flags	1 (read)	00, 01 (start-stop) 06 (no range, or all)	129 (response)	00, 01 (start-stop)
4	0	Double-bit Binary Input Event – Any Variation	1 (read)	06 (no range, or all) 07, 08 (limited qty)		
4	1	Double-bit Binary Input Event – Without time	1 (read)	06 (no range, or all) 07, 08 (limited qty)	129 (response) 130 (unsol. resp)	17, 28 (index)
4	2	Double-bit Binary Input Event – With absolute time	1 (read)	06 (no range, or all) 07, 08 (limited qty)	129 (response) 130 (unsol. resp)	17, 28 (index)
4	3	Double-bit Binary Input Event – With relative time	1 (read)	06 (no range, or all) 07, 08 (limited qty)	129 (response) 130 (unsol. resp)	17, 28 (index)
10	0	Binary Output – Any Variation	1 (read)	00, 01 (start-stop)		
				06 (no range, or all)		
10	2	Binary Output – Output status with flags	1 (read)	00, 01 (start-stop) 06 (no range, or all)	129 (response)	00, 01 (start-stop)
12	1	Binary Command – Control relay output block (CROB)	3 (select) 4 (operate) 5 (direct op) 6 (dir. op, no ack)	17, 28 (index)	129 (response)	echo of request
20	0	Counter – Any Variation	1 (read)	00, 01 (start-stop)		
				06 (no range, or all)		
20	1	Counter – 32-bit with flag	1 (read)	00, 01 (start-stop) 06 (no range, or all)	129 (response)	00, 01 (start-stop)
20	1	Counter – 32-bit with flag	7 (freeze) 8 (freeze noack) 9 (freeze clear) 10 (frz. cl. noack)	06 (no range, or all)		
20	2	Counter – 16-bit with flag	1 (read)	00, 01 (start-stop) 06 (no range, or all)	129 (response)	00, 01 (start-stop)
20	2	Counter – 16-bit with flag	7 (freeze) 8 (freeze noack) 9 (freeze clear) 10 (frz. cl. noack)	06 (no range, or all)		
20	5	Counter – 32-bit without flag	1 (read)	00, 01 (start-stop) 06 (no range, or all)	129 (response)	00, 01 (start-stop)
20	5	Counter – 32-bit without flag	7 (freeze) 8 (freeze noack) 9 (freeze clear) 10 (frz. cl. noack)	06 (no range, or all)		
20	6	Counter – 16-bit without flag	1 (read)	00, 01 (start-stop) 06 (no range, or all)	129 (response)	00, 01 (start-stop)
20	6	Counter – 16-bit without flag	7 (freeze) 8 (freeze noack) 9 (freeze clear) 10 (frz. cl. noack)	06 (no range, or all)		
21	0	Frozen Counter – Any Variation	1 (read)	00, 01 (start-stop)		
				06 (no range, or all)		
21	1	Frozen Counter – 32-bit with flag	1 (read)	06 (no range, or all)	129 (response)	00, 01 (start-stop)
21	2	Frozen Counter – 16-bit with flag	1 (read)	00, 01 (start-stop) 06 (no range, or all)	129 (response)	00, 01 (start-stop)
21	5	Frozen Counter – 32-bit with flag and time	1 (read)	00, 01 (start-stop) 06 (no range, or all)	129 (response)	00, 01 (start-stop)

Table 35 - Implementation Table for Micro870 controllers (Continued)

Group Num	Var Num	Description	Function Codes (Dec)	Qualifier Codes (Hex)	Function Codes (Dec)	Qualifier Codes (Hex)
DNP Object Group and Variation			**Request** DNP3 Master may issue, controller must parse		**Response** DNP3 Master must parse, controller may issue	
21	6	Frozen Counter – 16-bit with flag and time	1 (read)	00, 01 (start-stop) 06 (no range, or all)	129 (response)	00, 01 (start-stop)
21	9	Frozen Counter – 32-bit without flag	1 (read)	00, 01 (start-stop) 06 (no range, or all)	129 (response)	00, 01 (start-stop)
21	10	Frozen Counter – 16-bit without flag	1 (read)	00, 01 (start-stop) 06 (no range, or all)	129 (response)	00, 01 (start-stop)
22	0	Counter Event – Any Variation	1 (read)	06 (no range, or all) 07, 08 (limited qty)		
22	1	Counter Event – 32-bit with flag	1 (read)	06 (no range, or all) 07, 08 (limited qty)	129 (response) 130 (unsol. resp)	17, 28 (index)
22	2	Counter Event – 16-bit with flag	1 (read)	06 (no range, or all) 07, 08 (limited qty)	129 (response) 130 (unsol. resp)	17, 28 (index)
22	5	Counter Event – 32-bit with flag and time	1 (read)	06 (no range, or all) 07, 08 (limited qty)	129 (response) 130 (unsol. resp)	17, 28 (index)
22	6	Counter Event – 16-bit with flag and time	1 (read)	06 (no range, or all) 07, 08 (limited qty)	129 (response) 130 (unsol. resp)	17, 28 (index)
23	0	Frozen Counter Event – Any Variation	1 (read)	06 (no range, or all) 07, 08 (limited qty)		
23	1	Frozen Counter Event – 32-bit with flag	1 (read)	06 (no range, or all) 07, 08 (limited qty)	129 (response) 130 (unsol. resp)	17, 28 (index)
23	2	Frozen Counter Event – 16-bit with flag	1 (read)	06 (no range, or all) 07, 08 (limited qty)	129 (response) 130 (unsol. resp)	17, 28 (index)
23	5	Frozen Counter Event – 32-bit with flag and time	1 (read)	06 (no range, or all) 07, 08 (limited qty)	129 (response) 130 (unsol. resp)	17, 28 (index)
23	6	Frozen Counter Event – 16-bit with flag and time	1 (read)	06 (no range, or all) 07, 08 (limited qty)	129 (response) 130 (unsol. resp)	17, 28 (index)
30	0	Analog Input – Any Variation	1 (read)	00, 01 (start-stop) 06 (no range, or all)		
30	1	Analog Input – 32-bit with flag	1 (read)	00, 01 (start-stop) 06 (no range, or all)	129 (response)	00, 01 (start-stop)
30	2	Analog Input – 16-bit with flag	1 (read)	00, 01 (start-stop) 06 (no range, or all)	129 (response)	00, 01 (start-stop)
30	3	Analog Input – 32-bit without flag	1 (read)	00, 01 (start-stop) 06 (no range, or all)	129 (response)	00, 01 (start-stop)
30	4	Analog Input – 16-bit without flag	1 (read)	00, 01 (start-stop) 06 (no range, or all)	129 (response)	00, 01 (start-stop)
30	5	Analog Input – Single-prec flt-pt with flag	1 (read)	00, 01 (start-stop) 06 (no range, or all)	129 (response)	00, 01 (start-stop)
32	0	Analog Input Event – Any Variation	1 (read)	06 (no range, or all) 07, 08 (limited qty)		
32	1	Analog Input Event – 32-bit without time	1 (read)	06 (no range, or all) 07, 08 (limited qty)	129 (response) 130 (unsol. resp)	17, 28 (index)
32	2	Analog Input Event – 16-bit without time	1 (read)	06 (no range, or all) 07, 08 (limited qty)	129 (response) 130 (unsol. resp)	17, 28 (index)
32	3	Analog Input Event – 32-bit with time	1 (read)	06 (no range, or all) 07, 08 (limited qty)	129 (response) 130 (unsol. resp)	17, 28 (index)
32	4	Analog Input Event – 16-bit with time	1 (read)	06 (no range, or all) 07, 08 (limited qty)	129 (response) 130 (unsol. resp)	17, 28 (index)
32	5	Analog Input Event – Single-prec flt-pt without time	1 (read)	06 (no range, or all) 07, 08 (limited qty)	129 (response) 130 (unsol. resp)	17, 28 (index)
32	7	Analog Input Event – Single-prec flt-pt with time	1 (read)	06 (no range, or all) 07, 08 (limited qty)	129 (response) 130 (unsol. resp)	17, 28 (index)
40	0	Analog Output Status – Any Variation	1 (read)	00, 01 (start-stop) 06 (no range, or all)		
40	1	Analog Output Status – 32-bit with flag	1 (read)	00, 01 (start-stop) 06 (no range, or all)	129 (response)	00, 01 (start-stop)
40	2	Analog Output Status – 16-bit with flag	1 (read)	00, 01 (start-stop) 06 (no range, or all)	129 (response)	00, 01 (start-stop)

Table 35 - Implementation Table for Micro870 controllers (Continued)

DNP Object Group and Variation			Request DNP3 Master may issue, controller must parse		Response DNP3 Master must parse, controller may issue	
Group Num	Var Num	Description	Function Codes (Dec)	Qualifier Codes (Hex)	Function Codes (Dec)	Qualifier Codes (Hex)
40	3	Analog Output Status – Single-prec flt-pt with flag	1 (read)	00, 01 (start-stop) 06 (no range, or all)	129 (response)	00, 01 (start-stop)
41	1	Analog Output – 32-bit	3 (select) 4 (operate) 5 (direct op) 6 (dir. op, no ack)	17, 27, 28 (index)	129 (response)	echo of request
41	2	Analog Output – 16-bit	3 (select) 4 (operate) 5 (direct op) 6 (dir. op, no ack)	17, 27, 28 (index)	129 (response)	echo of request
41	3	Analog Output – Single-prec flt-pt	3 (select) 4 (operate) 5 (direct op) 6 (dir. op, no ack)	17, 27, 28 (index)	129 (response)	echo of request
50	1	Time and Date – Absolute time	1 (read)	07 (limited qty = 1)	129 (response)	07 (limited qty) (qty = 1)
			2 (write)	07 (limited qty = 1)		
50	3	Time and Date – Absolute time at last recorded time	2 (write)	07 (limited qty = 1)		
51	1	Time and Date CTO – Absolute time, synchronized			129 (response) 130 (unsol. resp)	07 (limited qty) (qty = 1)
51	2	Time and Date CTO – Absolute time, unsynchronized			129 (response) 130 (unsol. resp)	07 (limited qty) (qty = 1)
52	2	Time Delay – Fine			129 (response)	07 (limited qty) (qty = 1)
60	1	Class Objects – Class 0 data	1 (read)	06 (no range, or all)		
60	2	Class Objects – Class 1 data	1 (read)	06 (no range, or all) 07, 08 (limited qty)		
			20 (enbl. unsol.) 21 (dab. unsol.)	06 (no range, or all)		
60	3	Class Objects – Class 2 data	1 (read)	06 (no range, or all) 07, 08 (limited qty)		
60	4	Class Objects – Class 3 data	1 (read)	06 (no range, or all) 07, 08 (limited qty)		
			20 (enbl. unsol.) 21 (dab. unsol.)	06 (no range, or all)		
80	1	Internal Indications – Packed format	1 (read)	00, 01 (start-stop)	129 (response)	00, 01 (start-stop)
			2 (write)	00 (start-stop) index=7		
85	0	Data-Set Prototype	1 (read)	06 (no range, or all)		
85	1	Data-Set Prototype	1 (read)	00, 01 (start-stop) 06 (no range, or all) 17, 28 (index)	129 (response)	5B (free format)
86	1	Data-Set Descriptor – Contents	1 (read)	00, 01 (start-stop) 06 (no range, or all) 17, 28 (index)	129 (response)	5B (free format)
86	2	Data-Set Descriptor – Characteristics	1 (read)	00, 01 (start-stop) 06 (no range, or all) 17, 28 (index)	129 (response)	5B (free format)
87	0	Data-Set – Present Value	1 (read)	00, 01 (start-stop) 06 (no range, or all) 17, 28 (index)		
87	1	Data-Set – Present Value	1 (read)	00, 01 (start-stop) 06 (no range, or all) 17, 28 (index)	129 (response)	5B (free format)
88	0	Data-Set Event	1 (read)	06 (no range, or all) 07, 08 (limited qty)		
88	1	Data-Set Event – Snapshot	1 (read)	06 (no range, or all) 07, 08 (limited qty)	129 (response) 130 (unsol. resp)	5B (free format)

Table 35 - Implementation Table for Micro870 controllers (Continued)

DNP Object Group and Variation			Request DNP3 Master may issue, controller must parse		Response DNP3 Master must parse, controller may issue	
Group Num	Var Num	Description	Function Codes (Dec)	Qualifier Codes (Hex)	Function Codes (Dec)	Qualifier Codes (Hex)
90	1	Application – Identifier	16 (init. appl.) 17 (start appl.) 18 (stop appl.)	06 (no range, or all) 5B (free format)		
91	1	Status of Requested Operation			129 (response)	07 (limited qty) (qty = 1)
120	1	Authentication – Challenge	32 (auth request)	5B (free format)	131 (Auth. resp)	5B (free format)
120	2	Authentication – Reply	32 (auth request)	5B (free format)	131 (Auth. resp)	5B (free format)
120	3	Authentication – Aggressive Mode Request	Any requests	07 (limited qty)	129 (response)	07 (limited qty)
120	3	Authentication – Aggressive Mode Request			130 (unsol. resp)	07 (limited qty)
120	4	Authentication – Session Key Status Request	32 (auth request)	07 (limited qty)		
120	5	Authentication – Session Key Status			131 (Auth. resp)	5B (free format)
120	6	Authentication – Session Key Change	32 (auth request)	5B (free format)		
120	7	Authentication – Error	33 (auth request, no ack)	5B (free format)	131 (Auth. resp)	5B (free format)
120	7	Authentication – Error	1 (read)	06 (no range, or all)	129 (response)	5B (free format)
120	9	Authentication – HMAC	Any requests	5B (free format)	129 (response)	5B (free format)
120	9	Authentication – HMAC			130 (unsol. resp)	5B (free format)
No Object (function code only)			13 (cold restart)			
No Object (function code only)			14 (warm restart)			
No Object (function code only)			23 (delay meas.)			
No Object (function code only)			24 (record current time)			

Notes:

Program Execution in Micro800 Controllers

This chapter provides a brief overview of running or executing programs in a Micro800 controller.

IMPORTANT	This chapter generally describes program execution in Micro800 controllers. Certain elements may not be applicable or true for certain models (for example, Micro820 does not support PTO motion control).

For detailed information regarding ladder diagrams, instructions, function blocks, and so on, see the Micro800 Programmable Controllers Instruction Manual, publication 2080-RM001.

Overview of Program Execution

A Micro800 cycle or scan consists of reading inputs, executing programs in sequential order, updating outputs, and performing housekeeping (data log, recipe, communications).

Program names must begin with a letter or underscore, followed by up to 127 letters, digits, or single underscores. Use programming languages such as ladder logic, function block diagrams, and structured text.

Up to 256 programs may be included in a project, depending on available controller memory. By default, the programs are cyclic (executed once per cycle or scan). As each new program is added to a project, it is assigned the next consecutive order number. When you start up the Project Organizer in Connected Components Workbench, it displays the program icons based on this order. You can view and modify an order number for a program from the program's properties. However, the Project Organizer does not show the new order until the next time the project is opened.

The Micro800 controller supports jumps within a program. Call a subroutine of code within a program by encapsulating that code as a user-defined function (UDF) or user-defined function block (UDFB). A UDF is similar to a traditional subroutine and uses less memory than a UDFB, while a UDFB can have multiple instances. Although a UDFB can be executed within another UDFB, a maximum nesting depth of five is supported. A compilation error occurs if this is exceeded. This also applies to UDFs.

Alternatively, you can assign a program to an available interrupt and have it executed only when the interrupt is triggered. A program assigned to the User Fault Routine runs once prior to the controller going into Fault mode.

In addition to the User Fault Routine, Micro800 controllers also support two Selectable Timed Interrupts (STI). STIs execute assigned programs once every setpoint interval (1…65535 ms).

The Global System Variables that are associated with cycles/scans are:
- __SYSVA_CYCLECNT – Cycle counter
- __SYSVA_TCYCURRENT – Current cycle time
- __SYSVA_TCYMAXIMUM – Maximum cycle time since last start.

Execution Rules

The execution follows four main steps within a loop, as shown in [Figure 40](). The loop duration is a cycle time for a program.

Figure 40 - Program Execution Steps
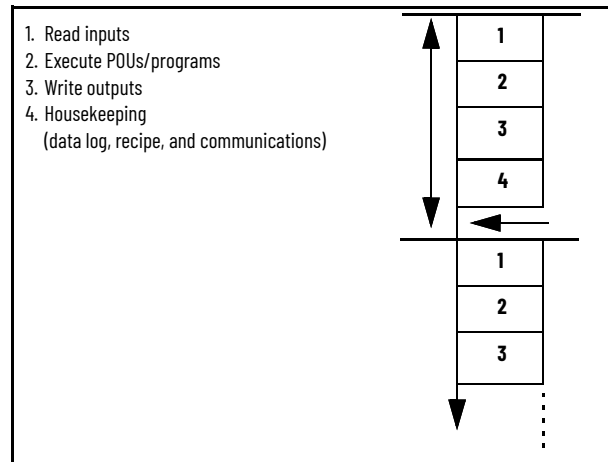
When a cycle time is specified, a resource waits until this time has elapsed before starting the execution of a new cycle. The Program Organizational Unit (POU) execution time varies depending on the number of active instructions. When a cycle exceeds the specified time, the loop continues to execute the cycle but sets an overrun flag. In such a case, the application no longer runs in real time.

When a cycle time is not specified, a resource performs all steps in the loop then restarts a new cycle without waiting.

Optional Module Configuration

Normally, before the read inputs step, the controller verifies the presence of any configured plug-in and expansion I/O modules. If a plug-in or expansion I/O module is missing, the controller faults. In Connected Components Workbench software release 10 or later, an Optional Module configuration option is added to prevent a missing plug-in I/O or expansion I/O module from faulting the controller if enabled. You can enable this option for each plug-in I/O or expansion I/O module separately.

> ⚠️ **ATTENTION:** If the optional module feature is enabled, use the MODULE_INFO instruction to verify that the module is present because the controller will not fault if the module is missing.

Controller Load and Performance Considerations

Within one program scan cycle, the execution of the main steps (as indicated in [Figure 40]()) could be interrupted by other controller activities that have higher priority than the main steps. Such activities include:

1. User Interrupt events, including STI, EII, and HSC interrupts (when applicable);

2. Communication data packet receiving and transmitting;

3. PTO Motion engine periodical execution (if supported by the controller).

When one or several of these activities occupy a significant percentage of the Micro800 controller execution time, the program scan cycle time is prolonged. The Watchdog timeout fault (0xD011) could be reported if the impact of these activities is underestimated, and the Watchdog timeout is set marginally. The Watchdog setting defaults to 2 s and generally never needs to be changed.

In firmware revision 21.011, significant usage of interrupts in the project can affect the Ethernet Implicit I/O messaging performance. As the interrupt has a higher priority than Implicit messaging, use the Ethernet Diagnostics in Connected Components Workbench software to check if there are dropped or missed packets with the desired setup.

Periodic Execution of Programs

For applications where periodic execution of programs with precise timing is required, such as for PID, it is recommended that STI (Selectable Timed Interrupt) be used to execute the program. STI provides precise time intervals.

It is not recommended that the system variable __SYSVA_TCYCYCTIME be used to periodically execute all programs as this also causes all communication to execute at this rate.

> **WARNING:** Communication timeouts may occur if programmed cycle time is set too slow (for example, 200 ms) to maintain communications.

Table 36 - System Variable for Programmed Cycle Time

Variable	Type	Description
__SYSVA_TCYCYCTIME	TIME	Programmed cycle time **Note**: Programmed cycle time only accepts values in multiples of 10 ms. If the entered value is not a multiple of 10, it is rounded up to the next multiple of 10.

Power-up and First Scan

In Program mode, all analog and digital input variables hold their last state, and the LEDs are always updated. Also all analog and digital output variables hold their last state, but only the analog outputs hold their last state while the digital outputs are off.

When transitioning from Program mode to Run mode, all analog output variables hold their last state but all digital output variables are cleared.

Two system variables are also available from revision 2 and later.

Table 37 - System Variables for Scan and Power-up on Firmware Revision 2 and later

Variable	Type	Description
_SYSVA_FIRST_SCAN	BOOL	First scan bit Can be used to initialize or reset variables immediately after every transition from Program to Run mode. **Note**: True only on first scan. After that, it is false.
_SYSVA_POWER_UP_BIT	BOOL	Power-up bit Can be used to initialize or reset variables immediately after download from the Connected Components Workbench software or immediately after being loaded from memory backup module (for example, microSD™ card). **Note**: True only on the first scan after a power-up, or running a new ladder for the first time.

Variable Retention

After a power cycle, all variables inside instances of instructions are cleared. Micro830 and Micro850 controllers retain all user-created variables. Micro870 controllers can only retain a maximum of 128 kilobytes of user-created variable values.

For example: A user-created variable that is called My_Timer of Time data type is retained after a power cycle but the elapsed time (ET) within a user-created timer TON instruction is cleared. This means that after a power cycle, global variables are cleared or set to initial value, and depending on the controller, some or all user-created variable values are retained. You can choose which variables to retain by selecting them on the global variable page.

Figure 41 - Retain Variable Example
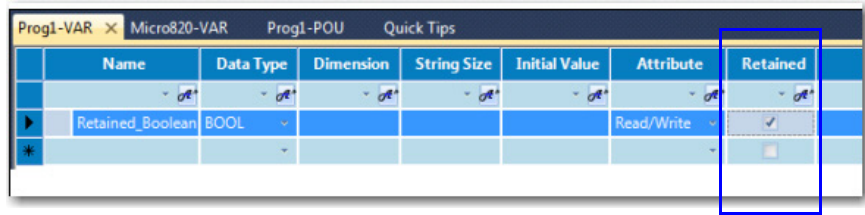
Memory Allocation

Depending on base size, the available memory on Micro800 controllers is shown in Table 38.

Table 38 - Memory Allocation for Micro800 Controllers

Attribute	10-point and 16-point (Micro830)	20-point (Micro820)	24-point and 48-point (Mico830, Micro850)	24-point (Micro870)
Program steps[1]	4 K	10 K	10 K	20 K
Data bytes	8 KB	20 KB	20 KB	40 KB

(1)   Estimated Program and Data size are "typical" – program steps and variables are created dynamically.
       1 Program Step = 12 data bytes.

These specifications for instruction and data size are typical numbers. When a Micro800 project is created, memory is dynamically allocated as either program or data memory at build time. This means that program size can exceed the published specifications if data size is sacrificed and vice versa. This flexibility allows maximum usage of execution memory. In addition to the user-defined variables, data memory also includes any constants and temporary variables that are generated by the compiler at build time.

If your project is larger, it affects the power-up time. Typical power up time is 10...15 seconds for all controllers. However, if your project has many initial and project values, it may cause the power-up time to exceed 30 seconds. After start-up, EtherNet/IP connections may take up to 60 seconds to establish.

The Micro800 controllers also have project memory, which stores a copy of the entire downloaded project (including comments), and configuration memory for storing plug-in setup information, and so on.

For Micro850 (2080-L50E) controllers, the memory allocation in the Connected Components Workbench software Controller - Memory page (Figure 42) shows a larger memory footprint, which is much larger than the previous equivalent Micro850 (2080-LC50) controller catalog. The reason for this difference is because the internal memory footprint has change with the release of the 2080-L50E controllers and the memory consumption on each element have increased, so the memory space is increased to cater for that. This does not imply that the 2080-L50E controllers have additional memory. The same Program and Data memory limit still applies as each element usage is increased to match the same amount of consumption.

Figure 42 - Memory Allocation Comparison Between 2080-L50E and 2080-LC50 Controllers
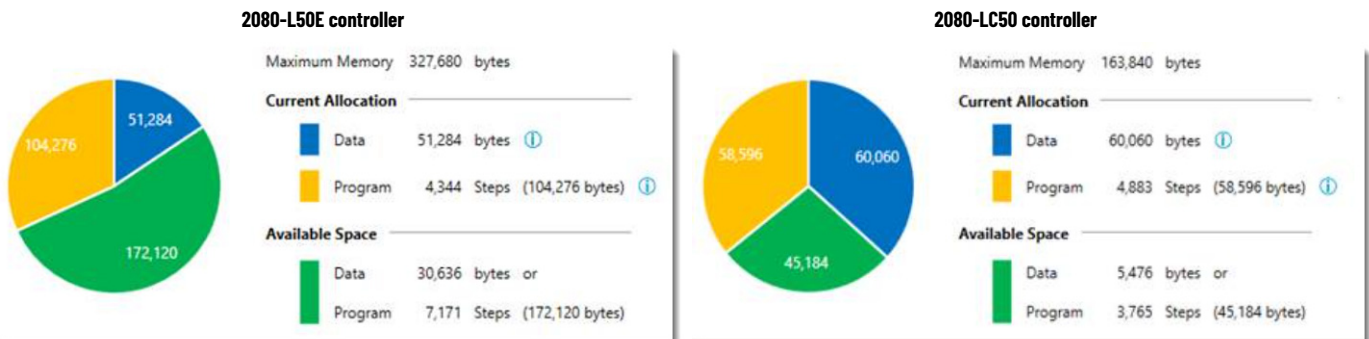
Guidelines and Limitations for Advanced Users

Here are some guidelines and limitations to consider when programming a Micro800 controller using Connected Components Workbench software:

- Each program/POU can use up to 64 Kb of internal address space. For all controllers except Micro870, it is recommended that you split large programs into smaller programs to improve code readability, simplify debugging and maintenance tasks.

- A user-defined function (UDF) uses less memory than a user-defined function block (UDFB). For example, 30% less for a typical sized program compared to a UDFB with one instance. The savings increases as the number of UDFB instances increases.

- A user-defined function block (UDFB) can be executed within another UDFB, with a limit of five nested UDFBs. Avoid creating UDFBs with references to other UDFBs, as executing these UDFBs too many times can result in a compile error. This also applies to UDFs.

Figure 43 - Example of Five Nested UDFBs

```
 ────────► UDFB1
            └────────► UDFB2
                       └────────► UDFB3
                                  └────────► UDFB4
                                             └────────► UDFB5
```
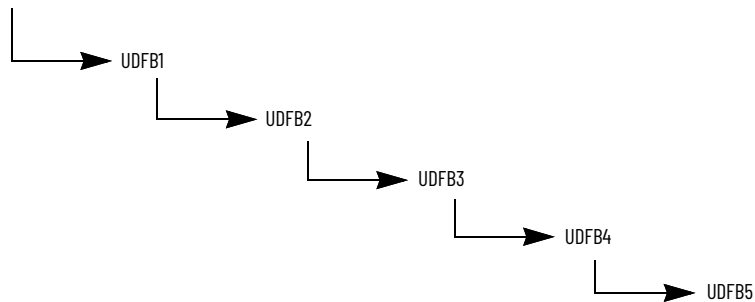
- Structured Text (ST) is much more efficient and easier to use than Ladder Logic, when used for equations. If you are used to using the RSLogix 500® CPT Compute instruction, a great alternative is to use ST combined with either UDF or UDFB.
  As an example, for an Astronomical Clock Calculation, Structured Text uses 40% fewer Instructions.

  Display_Output LD:
  Memory Usage (Code): 3148 steps
  Memory Usage (Data): 3456 bytes

  Display_Output ST:
  Memory Usage (Code): 1824 steps
  Memory Usage (Data): 3456 bytes

- You may encounter an Insufficient Reserved Memory error while downloading and compiling a program over a certain size. One workaround is to use arrays, especially if there are many variables.

Notes:

EtherNet/IP Network

Overview

The EtherNet/IP network offers a full suite of control, configuration, and data collection services by layering the Common Industrial Protocol (CIP™) over the standard Internet protocols, such as TCP/IP and UDP. This combination of well-accepted standards provides the capability that is required to support information data exchange and control applications.

Micro850 and Micro870 controllers with firmware revision 20.011 or earlier support Explicit messaging over the EtherNet/IP network. Micro850 (2080-L50E) and Micro870 (2080-L70E) controllers with firmware revision 21.011 or later add support for Implicit messaging.

For information on Implicit and Explicit messaging and how to configure EtherNet/IP or USB drivers, see the EtherNet/IP Network Devices User Manual, publication ENET-UM006.

The controllers use socket interface transactions and conventional communication over the EtherNet/IP network to communicate with Ethernet devices that do not support the EtherNet/IP application protocol.

EtherNet/IP Network Functionality

Micro850 and Micro870 controllers support the following EtherNet/IP network functionality:
- Support for these EtherNet/IP network topologies:
  - Star
- Support for these EtherNet/IP network communication rates:
  - 10 Mbps
  - 100 Mbps
- Duplicate IP address detection

For more information about network design, see the Ethernet Design Considerations Reference Manual, publication ENET-RM002.

Star Network Topology

A star network topology is a traditional EtherNet/IP network that includes multiple devices that are connected to each other through an Ethernet switch.

Figure 44 - Example of Micro850 Controller in a Star Network Topology

Micro850 controller

PanelView 800 terminal

Stratix® 2500 switch

PowerFlex 525 drive
with predefined tags in
the controller

1734-AENTR adapter
1734 POINT I/O modules
as a generic profile in the project

Kinetix 5100 drive
with predefined tags in
the controller
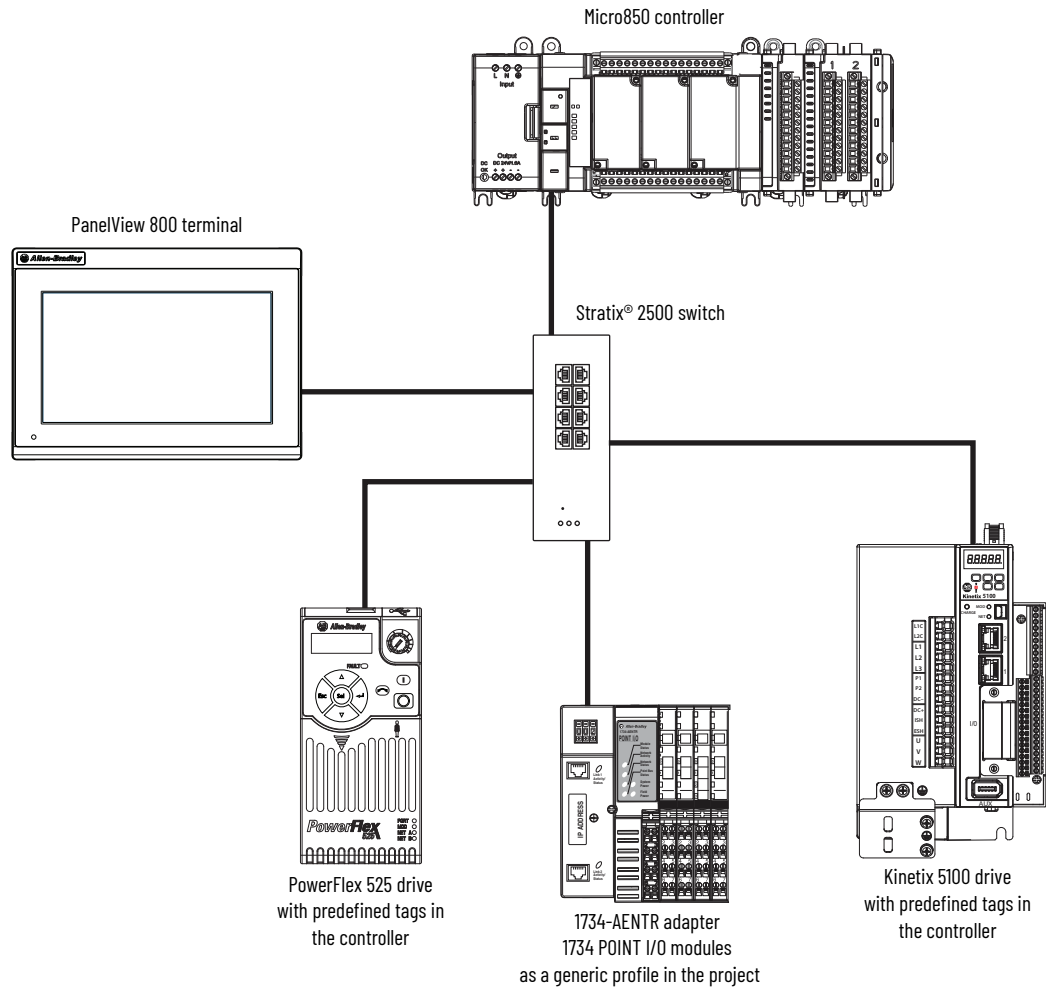
Implicit Messaging I/O Nodes on an EtherNet/IP Network

Applies to these controllers:
Micro850 (2080-L50E only) with firmware revision 21.011 or later
Micro870 (2080-L70E only) with firmware revision 21.011 or later

Micro850 (2080-L50E) and Micro870 (2080-L70E) controllers with firmware revision 21.011 or later support Implicit messaging to EtherNet/IP devices in either a generic tag or as a predefined tag in Connected Components Workbench software version 21 or later. PowerFlex® 520-series drives and Kinetix® 5100 motion drives are supported with predefined tags. Other EtherNet/IP devices are supported with generic profile tags. These controllers support up to eight EtherNet/IP devices in the module configuration section of the project.

Table 39 - Micro850 (2080-L50E) and Micro870 (2080-L70E) Controller EtherNet/IP Nodes

Micro850 (2080-L50E) Controllers	Micro870 (2080-L70E) Controllers	Nodes Supported, Max
2080-L50E-24AWB, 2080-L50E-24QWB, 2080-L50E-24QVB, 2080-L50E-24QBB	2080-L70E-24AWB, 2080-L70E-24QWB, 2080-L70E-24QWBK, 2080-L70E-24QWBN, 2080-L70E-24QWBNK, 2080-L70E-24QBB, 2080-L70E-24QBBK, 2080-L70E-24QBBN	8
2080-L50E-48AWB, 2080-L50E-48QWB, 2080-L50E-48QWBK, 2080-L50E-48QVB, 2080-L50E-48QBB	—	8

Devices Included in the Node Count

Any EtherNet/IP devices that you add to the module configuration section are counted toward the controller node limit. The following are examples of devices that must be counted:

- Kinetix 5100 Motion Drives
- PowerFlex 520-series Adjustable Frequency AC Drives
- Third-party devices that are configured as generic devices

Devices Excluded from the Node Count

When you calculate the EtherNet/IP node limitation of a controller, do not count devices that exist on the EtherNet/IP network but are not added to the Module configuration section.

The following devices are **not added** to the module configuration section and are **not counted** among the number of nodes:

- Computer
- Devices that are the target of MSG Instructions but were not added to the module configuration section
- Standard Ethernet devices with which the controller communicates through a socket interface

Figure 45 shows nodes in the module configuration.

Figure 45 - Example EtherNet/IP Nodes

How to Add PowerFlex 520-series and Kinetix 5100 Drives over EtherNet/IP

Applies to these controllers:

Micro850 (2080-L50E only)
with firmware revision 21.011 or later

Micro870 (2080-L70E only)
with firmware revision 21.011 or later

Micro850 (2080-L50E) and Micro870 (2080-L70E) controllers support predefined tags in both PowerFlex 520-series drives and Kinetix 5100 drives. All other EtherNet/IP devices use the generic profile.

When a PowerFlex 520-series drive or Kinetix 5100 drive is configured for I/O Mode operation and is used with Connected Components Workbench software, the use of predefined user-defined function block (UDFB) motion instructions provides an easy way to program your simple motion control application on Micro800 controllers with the Class 1 connectivity feature.

For instructions on where to download the UDFB files, see Download the User-defined Function Block Instruction Files on page 146.

> **IMPORTANT**   Induction and linear motors are not supported with the UDFB.

Add a PowerFlex 523 or PowerFlex 525 Drive

To add a PowerFlex 520-series drive into the Micro800 controller project, do the following.

1. Choose Modules under the Ethernet branch in the device configuration tree.
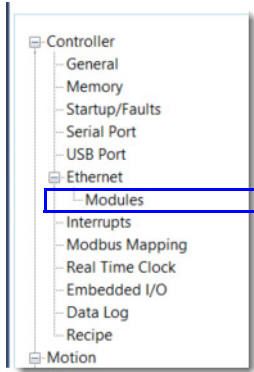
2. Select Add to create a device.
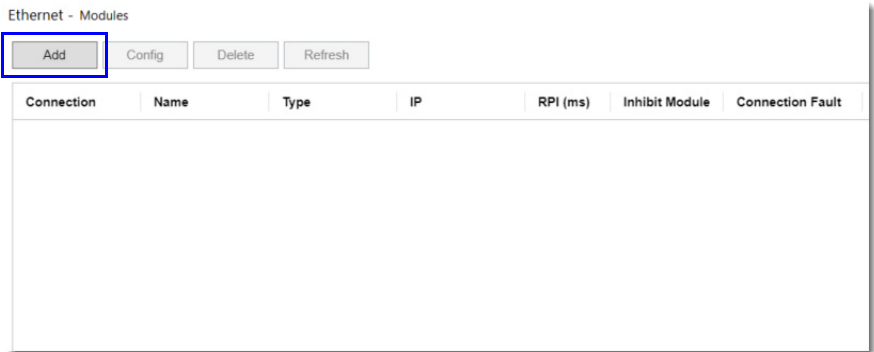
3. In the New Module window, select the Type and configure the module properties, and select OK.
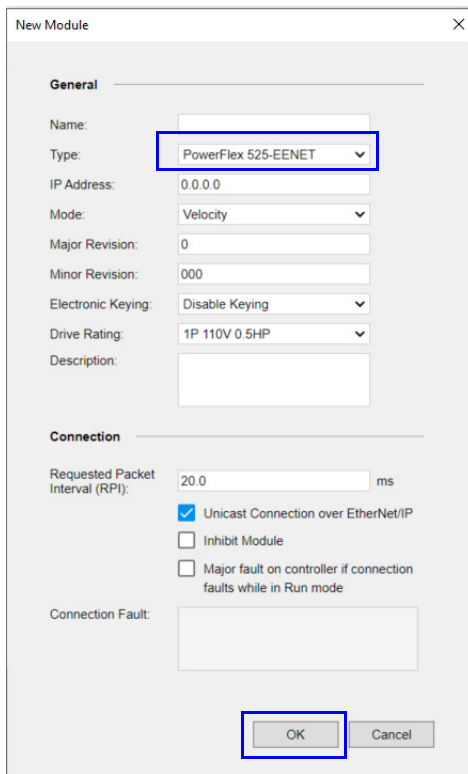
The module is added to the table.

Add a Kinetix 5100 Drive

To configure the Kinetix 5100 drive in Connected Components Workbench software, do the following.

1. Add the Kinetix 5100 drive in the Controller > Ethernet > Modules dialog. Select the correct K5100 catalog number then enter the name and IP address. Verify that both the Micro800 controller and Kinetix 5100 drive are in the same subnet, and select the "Data with Camming" connection.

2. The Kinetix 5100 drive is configured as shown in the Ethernet – Modules dialog. Repeat step 1 to add more K5100 drives in the same subnet of the Micro800 controller.

The Kinetix 5100 drive UDFBs are designed to follow the user experience of the K5100 Opr Add-On Instruction Library from Logix as close as possible. This also includes a Device Object handler (Figure 73) designed to provide a robust method to control the read/write functions of the drive assemblies.

Table 40 shows the use of the UDFB and Device Object handler for Kinetix 5100 drives.

Table 40 - Device Object Handler and UDFB Availability

Kinetix 5100 Drive Firmware Major Revision	Device Object Handler	UDFB
1.xx	raC_Dvc_K5100	raC_Opr_K5100_xxx

> **IMPORTANT**   The UDFBs can be downloaded from the Rockwell Automation Product Compatibility Download Center (PCDC) website at rok.auto/pcdc; enter the keywords "Kinetix 5100 UDFB Library" in the Search field.

Modify an Existing Module

To modify the existing module, do the following.

1. Select the module to modify and select Config.
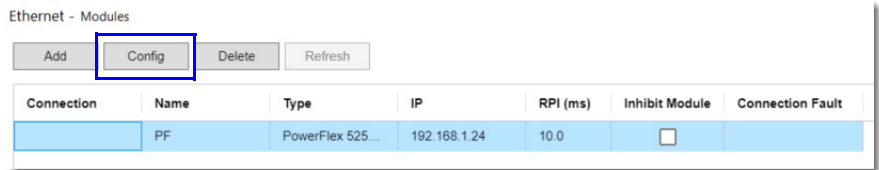
2. In the Configure Module window, adjust the module properties, and select OK.
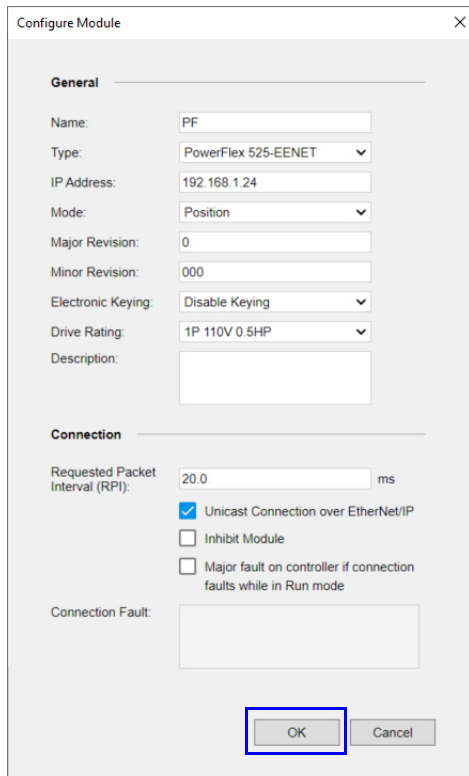
Module Properties

The New Module/Configure Module window provides the properties for the module.

Table 41 - Module Properties Parameters

Parameter	Description	Value
General		
Name	Unique name of the device.	User defined
Type	Select the type of profile to use with the device. See Type Definition in Module Dialog on page 138 for more information.	• Generic Device • PowerFlex 523-E2P • PowerFlex 525-E2P • PowerFlex 525-EENET • Kinetix 5100
IP Address	Unique IP address of the device.	User defined
Mode [(1)]	Define the mode of operation for the PowerFlex drive.	• Position • Velocity
Catalog [(2)]	Select the right Kinetix 5100 catalog to be used.	List of Kinetix 5100 drive catalogs (2198-E*xxxx*-ERS)
Major Revision	Module firmware major revision.	Module specific
Minor Revision	Module firmware minor revision.	Module specific
Electronic Keying	Software method to help reduce the possibility of using a mismatched device in a control system. Carefully consider the implications of each keying option when selecting one. See Table 42 for detailed descriptions of each option.	• Exact Match • Compatible Module • Disable Keying
Drive Rating [(1)]	Define the PowerFlex drive rating.	List of PowerFlex drive ratings (*x*P *xx*V *xxx*HP)
Description	Optional description for the device.	User defined
Connection		
Requested Packet Interval (RPI)	Set the RPI rate. See Requested Packet Interval on page 138 for more information.	• Generic Device - 5.0…9999.9 ms • PowerFlex Device - 5.0…9999.9 ms • Kinetix Device - 5.0…3200.0 ms
Unicast Connection over EtherNet/IP	Set the type of connection to use over the EtherNet/IP network.	• Unchecked = Multicast • Checked = Unicast
Inhibit Module	Inhibit the module. See Module Inhibiting on page 139 for more information.	• Unchecked = Module not inhibited • Checked = Module inhibited
Major fault on controller if connection faults while in Run mode	Select whether the controller triggers a major fault if a connection fault occurs while in Run mode.	• Unchecked = Module does not fault • Checked = Module faults
Connection Fault	Displays the major or minor fault code when it occurs, and can be used to help troubleshoot the module. See Connection Fault Codes on page 149 for a list of possible fault codes.	Fault code 0xYYYY (for example, 0x0001)
Comm Config [(3)]		
Comm Format	Defines the data structure data type in the generic device.	• Data - DIN • Data - INT • Data - REAL • Data - SINT • Input Data - DINT • Input Data - INT • Input Data - REAL • Input Data - SINT
Input	Define the Assembly Instance and size of the input assembly in the generic device.	User defined
Output	Define the Assembly Instance and size of the output assembly in the generic device.	User defined
Configuration	Define the Assembly Instance and size of the configuration assembly in the generic device.	User defined

(1)   Only displayed when PowerFlex is selected as the Type.
(2)   Only displayed when Kinetix is selected as the Type.

(3)   These parameters are displayed only when Generic Device is selected as the Type, and is used to define a generic device data structure.

Table 42 - Module Keying Options

Keying Option	Description

Compatible Module	Lets the installed device accept the key of the device that is defined in the project when the installed device can emulate the defined device. With Compatible Module, you can typically replace a device with another device that has the following characteristics:
	• Same catalog number
	• Same or higher Major Revision
	• Minor Revision as follows:
	  – If the Major Revision is the same, the Minor Revision must be the same or higher.
	  – If the Major Revision is higher, the Minor Revision can be any number.

Disable Keying	Indicates that the keying attributes are not considered when attempting to communicate with a device. With Disable Keying, communication can occur with a device other than the type specified in the project. **ATTENTION:** Be cautious when using Disable Keying; if used incorrectly, this option can lead to personal injury or death, property damage, or economic loss. We **strongly recommend** that you **do not use** Disable Keying. If you use Disable Keying, you must take full responsibility for understanding whether the device being used can fulfill the functional requirements of the application. **IMPORTANT:** Do not use this option in Safety applications.
Exact Match	Indicates that all keying attributes must match to establish communication. If any attribute does not match precisely, communication with the device does not occur.

Requested Packet Interval

The Requested Packet Interval (RPI) is a configurable parameter that defines a rate at which the owner-controller and the module exchange data. You set the RPI value during initial module configuration and can adjust it as necessary after module operation has begun.

IMPORTANT	You can change the RPI while the project is online. If you change the RPI while the project is online, however, the connection to the module is closed and reopened in one of the following ways:
	• You inhibit the connection to the module, change the RPI value, and uninhibit the connection. • You change the RPI value. In this case, the connection is closed and reopened immediately after you apply the change to the module configuration. • During Run Mode Change (RMC), if the RPI is too fast, it can cause a momentary drop in connection on the Ethernet device and fault the device if the drop is detected as a communication loss. To recover, reset the fault and resume operations.

Type Definition in Module Dialog

The Type in the module creation defines the type of devices to be connected. There are five types to choose from:

• Generic Device is for all EtherNet/IP devices that do not have supported predefined data structures in the Micro800 controller.

• PowerFlex 523-E2P or PowerFlex 525-E2P are for PowerFlex 520-series drives that use the dual-port Ethernet plug-in communication card.

• PowerFlex 525-EENET is for PowerFlex 525 drives that use the embedded Ethernet port.

• Kinetix 5100 is for all Kinetix 5100 servo drives. When this option is selected, a new field that is called Catalog is shown where you must select the correct Kinetix 5100 catalog number.

Module Inhibiting

Module inhibiting lets you indefinitely suspend a connection between an owner-controller and an I/O module without removing the module from the configuration. This process lets you temporarily disable a module, such as to perform maintenance.

To inhibit the module, do the following.
1. Go to the Ethernet - Modules dialog.

2. Select the Inhibit Module checkbox on the Ethernet device that you want to inhibit.
3. Select Save to Controller to accept the inhibit change to the Ethernet device.

4. Device connection is updated as Inhibited.
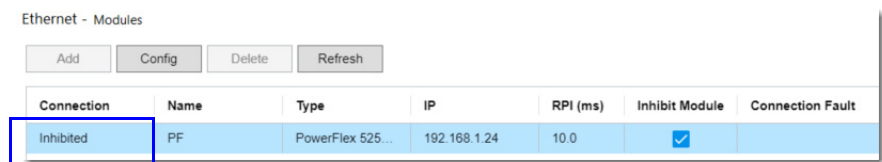
To uninhibit the module, do the following.
1. Go to the Ethernet - Modules dialog.

2. Clear the Inhibit Module checkbox on the Ethernet device that you want to uninhibit.
3. Select Save to Controller to accept the uninhibit change to the Ethernet device.
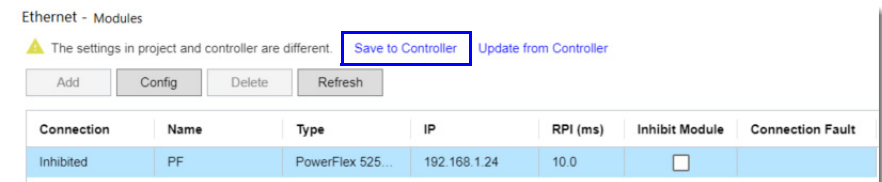
4. Device connection is updated as Running.
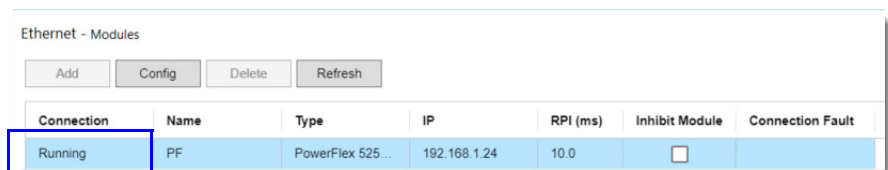
Predefined Tags in PowerFlex 520-series and Kinetix 5100 Drives

With Connected Components Workbench software version 21 or later, when Kinetix 5100 and PowerFlex 520 series products are added into the Micro800 controller (with firmware revision 21 or later) project under the Ethernet-Module tree, predefined tags are made available in the Connected Components Workbench software to help you program the logic easily.

The Input and Output assembly are shown here for your convenience. While you can directly manipulate these assemblies, they rely on your logic to operate correctly, including timing, pre-existing drive conditions, and so on. It is typical to use the predefined Motion Operation UDFBs to perform motion operations.

Table 43 - Kinetix 5100 Motion Drive Input Assembly Data

Name	Data Type	Description	Value	Comment
InfoBits	SINT			
InfoBits.0	BIT	Indicates whether the drive is in run mode.	0 = Drive is idle 1 = Drive is in run mode	RunMode
InfoBits.1	BIT	Indicates whether the connection is faulted.	0 = Connection is not faulted 1 = Connection is faulted	ConnectionFaulted
InfoBits.2	BIT	Indicates whether the diagnostic is active.	0 = Diagnostic is not active 1 = Diagnostic is active	DiagnosticActive
DiagnosticSequenceCount	SINT	The sequence count for the diagnostic.		
DataBits	SINT			
DataBits.1	Bit	Indicates whether the drive is in a faulted state.	0 = No Fault 1 = Faulted	Fault
DataBits.2	BIT	Indicates whether the data validity is questionable.	0 = Data is valid 1 = Date validity is questionable	Uncertain
StatusBits	SINT	Indicates whether the drive is in a warning state.	0 = No warnings 1 = Drive is in a warning state	
StatusBits.1	BIT	Indicates whether the motor is enabled.	0 = Motor is not enabled 1 = Motor is enabled	WarningPresent
StatusBits.2	BIT	Indicates whether the motor is ready to be enabled.	0 = Motor is not ready 1 = Motor is ready	Active
StatusBits.3	BIT	Indicates whether the drive received the command from the controller.	Indicates that the new command has been received by the K5100 drive. It toggles between 0 and 1 after a new command has been received by the K5100 drive. When this bit toggles, it stays at the toggled state until a new command is received.	Ready
StatusBits.4	BIT	Indicates whether the drive completed the home operation.	1 = Drive completed the home operation	CommandinProgress
StatusBits.5	BIT	Indicates whether the motor is stopped.	1 = Motor is stopped	HomedStatus
StatusBits.6	BIT	Motor actual at reference (position, speed, torque) based on mode.	1 = Motor actual at reference (position, speed, torque) based on mode	Stopped
StatusBits.7	BIT			AtReference
OperatingMode	SINT	Indicates the drive operation mode.	-128...-1 = Reserved 0 = Mode not specified 1 = Position mode 2 = Speed mode 3 = Home mode 4 = Torque mode 5 = Gear mode 6 = Index mode 7...127 = Reserved	
ActiveIndex	SINT	Indicates the currently executing index (PR).	-128...-1 = Reserved 0 = PR 0: Homing 1...99 = PR 1...PR 99 100...127 = Reserved	
MotorType	SINT	Indicates which type of motor is connected to the drive.	0 = No motor connected 1 = Rotary motor connected 2 = Linear motor connected	
ActualSpeed	DINT	Motor actual velocity.	A valid value in RPM.	
FaultCode	INT	Fault code.	For more information, see the Kinetix 5100 EtherNet/IP Indexing Servo Drives User Manual, publication [2198-UM004](#).	
WarningCode	INT	Warning code.		

Table 43 - Kinetix 5100 Motion Drive Input Assembly Data (Continued)

Name	Data Type	Description	Value	Comment
ActualPosition	DINT	Actual position of the motor.	PUU (counts or user units)	
AcutalTorque	DINT	Actual torque of the motor.	% motor rated torque	
ParamterMonitor1Value	DINT	Parameter monitor selection 1.	0 = No parameter is selected 0x0001...0xFFFF = returned value that is mapped from KNX5100C Function List > Parameter Editor > StatusMonitor ID060	
ParamterMonitor2Value	DINT	Parameter monitor selection 2.	1 = No parameter is selected 0x0001...0xFFFF = returned value that is mapped from KNX5100C Function List > Parameter Editor > StatusMonitor ID061	
ParamterMonitor3Value	DINT	Parameter monitor selection 3.	2 = No parameter is selected 0x0001...0xFFFF = returned value that is mapped from KNX5100C Function List > Parameter Editor > StatusMonitor ID062	
ParamterMonitor4Value	DINT	Parameter monitor selection 4.	3 = No parameter is selected 0x0001...0xFFFF = returned value that is mapped from KNX5100C Function List > Parameter Editor > StatusMonitor ID063	
ParamterMonitor5Value	DINT	Parameter monitor selection 5.	4 = No parameter is selected 0x0001...0xFFFF = returned value that is mapped from KNX5100C Function List > Parameter Editor > StatusMonitor ID064	

Table 44 - Kinetix 5100 Motion Drive Output Assembly Data

Name	Data Type	Description	Value	Comment
OperatingMode	SINT	This enumerated value indicates the drive's internal mode setting. The drive can operate in different submodes while in IO Mode.	-128...-1 = Reserved 0 = Mode not specified 1 = Position mode 2 = Speed mode 3 = Home mode 4 = Torque mode 5 = Gear mode 6 = Index mode 7 = ECAM mode 8...127 = Reserved	
ServoControl	SINT			
ServoControl.0	BIT	A transition from 0 to 1 enables the motor.		ServoOn
ServoControl.1	BIT	A transition from 0 to 1 disables the motor.		ServoOff
ServoControl.2	BIT	A transition from 0 to 1 stops motion on the motor.		StopMotion
ServoControl.3	BIT	A transition from 0 to 1 clears an active drive fault.		FaultReset
ServoControl.4	BIT	A transition from 0 to 1 means that the motion command is issued from the external controller.		StartMotion
HomingMethod	SINT	The method of Homing.	For more information, see the Kinetix 5100 EtherNet/IP Indexing Servo Drives User Manual, publication 2198-UM004.	
SpeedReference	DINT	The commanded speed for the motor.	Units are in 0.1 RPM -80000...+80000 1...20000 (home mode)	
AccelReference	DINT	The commanded acceleration rate for the motor.	Units are in 0.1 RPM/sec	
DecelReference	DINT	The commanded deceleration rate for the motor.	Units are in 0.1 RPM/sec	
PositionReference	DINT	The commanded position used for indexing.	The scaling relationship from the E-gear ratio in KNX5100C software defines the User units.	
HomeReturnSpeed	DINT	The return speed when home mode is the operating mode.	Units are in 0.1 RPM (rotary motors) 1...5000	
NonCyclicMoveType	SINT	Enumerated value used to determine the noncyclic move type.	-128...-1 = Reserved 0 = Absolute 1 = Relative 2 = Incremental 3 = High-speed capture 4...127 = Reserved	

Table 44 - Kinetix 5100 Motion Drive Output Assembly Data (Continued)

Name	Data Type	Description	Value	Comment
CylicMoveType	SINT	Enumerated value used to determine the cyclic move type.	-128...-1 = Reserved 0 = Rotary positive 1 = Rotary negative 2 = Rotary shortest path 3...127 = Reserved	
TravelMode	DINT	Enumerated value used to determine the travel constraints of the axis.	-128...+1 = Reserved 2 = Non-cyclic move 3...9 = Reserved 10 = Cyclic move 11...127 = Reserved	
PositionControl	SINT			
PositionControl.0	BIT	When executing a motion command, the next movement can override the previous movement.	0 = Does not override the previous movement 1 = Can override the previous movement	PositionCommandOverride
PositionControl.1	BIT	The next movement can overlap the end of the current movement.	0 = Does not overlap the next movement 1 = Overlaps the next movement	PositionCommandOverlap
PositionControl.2	BIT	Selects between the high-speed digital inputs that are used to capture position feedback.	Vendor specific 0 = DI9 is selected 1 = DI10 is selected	CapturedPositionSelect
TorqueReference	DINT	Represents the output torque level when the operation mode is Torque Mode (3). This value is in percent of motor rated torque.	-4000...+4000 (enumeration is 0.1x)	
TorqueRampTime	DINT	Represents the time to reach the torque reference.	1...65500 ms	
StartingIndex	SINT	The first index (position register) that the drive should execute.	-128...-1 = Reserved 0 = PR 0: Homing 1...99 = PR 1...PR 99 100...127 = Reserved	
CamMasterReference	SINT			
CamExecutionSchedule	SINT	Future		
CamExecutionMode	SINT	Future		
CamSetting	SINT	Future		
CamSetting3	BIT	Future		CamStopMode
CamSlaveScaling	DINT	Future		
CamLockPosition	DINT	Future		
CamMasterLockPosition	DINT	Future		
CamMasterLeadingCounts	DINT	Future		
CamMasterUnlockCounts	DINT	Future		
CamMasterCyclicLeadingCounts	DINT	Future		
GearRatioSlaveCounts	DINT	Integer value that represents slave counts. This value is P1.044 Gear Ratio Follower Counts from the E-gear ratio in Kinetix 5100 software.		
GearRatioMasterCounts	DINT	Integer value that represents master counts. This value is P1.045 Gear Ratio Master Counts from the E-gear ratio in Kinetix 5100 software.		

Table 45 - PowerFlex 520-series Drive Input Assembly Data (Position Mode)

Name	Data Type	Description	Value	Comment
DriveStatus	INT			
DriveStatus.0	BIT	Indicates whether the drive is ready for operation.	0 = Not ready 1 = Ready	Ready
DriveStatus.1	BIT	Indicates whether the drive is operating.	0 = Not active 1 = Active (Running)	Active
DriveStatus.2	BIT	Indicates the command direction.	0 = Cmd reverse 1 = Cmd forward	CommandDir
DriveStatus.3	BIT	Indicates the rotating direction.	0 = Rotating reverse 1 = Rotating forward	ActualDir
DriveStatus.4	BIT	Indicates the acceleration state.	0 = Not accelerating 1 = Accelerating	Accelerating

Table 45 - PowerFlex 520-series Drive Input Assembly Data (Position Mode) (Continued)

Name	Data Type	Description	Value	Comment
DriveStatus.5	BIT	Indicates the deceleration state.	0 = Not decelerating 1 = Decelerating	Decelerating
DriveStatus.6	BIT	Indicates the Travel Position direction.	0 = Reverse travel position 1 = Forward travel position	ForwardTravel
DriveStatus.7	BIT	Indicates the fault state.	0 = Not faulted 1 = Faulted	Faulted
DriveStatus.8	BIT	Indicates that the drive is at reference speed.	0 = Not at reference 1 = At reference	AtReference
DriveStatus.9	BIT	Indicates that the drive is at commanded position.	0 = Not at position 1 = At position	AtPos
DriveStatus.10	BIT	Indicates that the drive is at the reference home.	0 = Not at home 1 = At home	AtHome
DriveStatus.11	BIT	Indicates whether the drive has been homed since power-up.	0 = Drive not homed 1 = Drive homed	DriveHomed
DriveStatus.12	BIT	Indicates if the frequency is holding.	0 = Not sync hold 1 = Sync hold	SyncHold
DriveStatus.13	BIT	Indicates if the frequency accelerating to the new commanded frequency in drive parameter A571 [Sync Time].	0 = Not sync ramp 1 = Sync ramp	SyncRamp
DriveStatus.14	BIT	Indicates if Traverse is enabled.	0 = Traverse off 1 = Traverse on	TraverseOn
DriveStatus.15	BIT	Indicates if the drive is decelerating in Traverse mode.	0 = Not Traverse decel 1 = Traverse decel	TraverseDecel
OutputFreq	INT	Display the reference speed of the drive.	In units of 0.01 Hz.	

Table 46 - PowerFlex 520-series Drive Output Assembly Data (Position Mode)

Name	Data Type	Description	Value	Comment
LogicCommand	INT			
LogicCommand.0	BIT	Perform a normal stop.	0 = Not normal stop 1 = Normal stop	Stop
LogicCommand.1	BIT	Command the drive the start.	0 = Not start 1 = Start	Start
LogicCommand.2	BIT	Command the drive to jog.	0 = Not jog 1 = Jog	Jog
LogicCommand.3	BIT	Clear drive fault.	0 = Not clear fault 1 = Clear fault	ClearFaults
LogicCommand.4	BIT	Command the direction of the drive.	00 = No command 01 = Forward command 10 = Reverse command 11 = No command	Forward
LogicCommand.5	BIT			Reverse
LogicCommand.6	BIT	This provides an identical function as the "Logic In1" Digital Input option.	1 = Logic In 1	LogicIn1
LogicCommand.7	BIT	This provides an identical function as the "Logic In2" Digital Input option.	1 = Logic In 2	LogicIIn2
LogicCommand.8	BIT	Select the pre-programed frequency and position step.	000 = Frequency and position step 0 001 = Frequency and position step 1 010 = Frequency and position step 2 011 = Frequency and position step 3 100 = Frequency and position step 4 101 = Frequency and position step 5 110 = Frequency and position step 6 111 = Frequency and position step 7	Freq_PosSel01
LogicCommand.9	BIT			Freq_PosSel02
LogicCommand.10	BIT			Freq_PosSel03
LogicCommand.11	BIT	Next start command causes the drive to find home.	1 = Find home	FindHome
LogicCommand.12	BIT	Overrides other inputs and causes the drive to remain at its current step (running at zero speed once it reaches its position) until released.	1 = Hold step	HoldStep
LogicCommand.13	BIT	Resets the home position to the current position of the machine. Set this bit to 0 after completing the homing routine.	1 = Pos redefine	PosRedefine

Table 46 - PowerFlex 520-series Drive Output Assembly Data (Position Mode) (Continued)

Name	Data Type	Description	Value	Comment
LogicCommand.14	BIT	Hold the existing frequency when sync time is set to enable speed synchronization.	1 = Sync enable	SyncEnabled
LogicCommand.15	BIT	Disable the traverse function	1 = Traverse disable	TravDisable
FreqCommand	INT	Control the reference speed of the drive.	In units of 0.01 Hz	

Table 47 - PowerFlex 520-series Drive Input Assembly Data (Velocity Mode)

Name	Data Type	Description	Value	Comment
DriveStatus	INT			
DriveStatus.0	BIT	Indicates whether the drive is ready for operation.	0 = Not ready 1 = Ready	Ready
DriveStatus.1	BIT	Indicates whether the drive is operating.	0 = Not active 1 = Active (Running)	Active
DriveStatus.2	BIT	Indicates the command direction.	0 = Cmd reverse 1 = Cmd forward	CommandDir
DriveStatus.3	BIT	Indicates the rotating direction.	0 = Rotating reverse 1 = Rotating forward	ActualDir
DriveStatus.4	BIT	Indicates the acceleration state.	0 = Not accelerating 1 = Accelerating	Accelerating
DriveStatus.5	BIT	Indicates the deceleration state.	0 = Not decelerating 1 = Decelerating	Decelerating
DriveStatus.6	BIT	Reserved.		
DriveStatus.7	BIT	Indicates the fault state.	0 = Not faulted 1 = Faulted	Faulted
DriveStatus.8	BIT	Indicates that the drive is at reference speed.	0 = Not at reference 1 = At reference	AtReference
DriveStatus.9	BIT	Indicates that the main frequency is controlled by the active communication.	1 = Main frequency controlled by active comm	CommFreqCnt
DriveStatus.10	BIT	Indicates that the operation command is controlled by the active communication.	1 = Operation command controlled by active comm	CommLogicCnt
DriveStatus.11	BIT	Indicates that the parameters are locked.	1 = Parameters are locked	ParmsLocked
DriveStatus.12	BIT	Indicates the Digital Input 1 status.		DigIn1Active
DriveStatus.13	BIT	Indicates the Digital Input 2 status.		DigIn2Active
DriveStatus.14	BIT	Indicates the Digital Input 3 status.		DigIn3Active
DriveStatus.15	BIT	Indicates the Digital Input 4 status.		DigIn4Active
OutputFreq	INT	Display the reference speed of the drive.	In units of 0.01 Hz.	

Table 48 - PowerFlex 520-series Drive Output Assembly Data (Velocity Mode)

Name	Data Type	Description	Value	Comment
LogicCommand	INT			
LogicCommand.0	BIT	Perform a normal stop.	0 = Not normal stop 1 = Normal stop	Stop
LogicCommand.1	BIT	Command the drive the start.	0 = Not start 1 = Start	Start
LogicCommand.2	BIT	Command the drive to jog.	0 = Not jog 1 = Jog	Jog
LogicCommand.3	BIT	Clear drive fault.	0 = Not clear fault 1 = Clear fault	ClearFaults
LogicCommand.4	BIT	Command the direction of the drive.	00 = No command 01 = Forward command 10 = Reverse command 11 = No command	Forward
LogicCommand.5	BIT			Reverse
LogicCommand.6	BIT	Force keypad control.	0 = Not keypad control 1 = Forced keypad control	ForceKeypadCtrl
LogicCommand.7	BIT	Increases the value of drive parameter A427 [MOP Freq] at the rate set in A430 [MOP Time].	0 = Not increment 1 = MOP increment	MOPIncrement

Table 48 - PowerFlex 520-series Drive Output Assembly Data (Velocity Mode) (Continued)

Name	Data Type	Description	Value	Comment
LogicCommand.8	BIT	Select the Accel Rate.	00 = No command 01 = Accel Rate 1 enable 10 = Accel Rate 2 enable 11 = Hold Accel Rate selected	AccelRate1
LogicCommand.9	BIT			AccelRate2
LogicCommand.10	BIT	Select the Decel Rate.	00 = No command 01 = Decel Rate 1 enable 10 = Decel Rate 2 enable 11 = Hold Decel Rate selected	DecelRate1
LogicCommand.11	BIT			DecelRate2
LogicCommand.12	BIT	Frequency selection of drive parameters Speed Reference 1…3 and Preset Freq 0…3.	000 = No command 001 = Freq source = P047 (Speed Reference 1) 010 = Freq source = P049 (Speed Reference 2) 011 = Freq source = P051 (Speed Reference 3) 100 = Freq source = A410 (Preset Freq 0) 101 = Freq source = A411 (Preset Freq 1) 110 = Freq source = A412 (Preset Freq 2) 111 = Freq source = A413 (Preset Freq 3)	FreqSel01
LogicCommand.13	BIT			FreqSel02
LogicCommand.14	BIT			FreqSel03
LogicCommand.15	BIT	Decreases the value of drive parameter A427 [MOP Freq] at the rate set in A430 [MOP Time].	0 = Not decrement 1 = MOP decrement	MOPDecrement
FreqCommand	INT	Control the reference speed of the drive.	In units of 0.01 Hz	

Use of the User-defined Function Block Library

When a PowerFlex 520-series or Kinetix 5100 drive is used with Connected Components Workbench software, the use of the predefined user-defined function block (UDFB) provides an easy way to program your simple drive control application.

Table 49 - UDFB List for PowerFlex520-series Drives

Name	Description
RA_PF523_VEL	PowerFlex 523 Velocity Mode control. This instruction allows simple PowerFlex 523 drive control in Velocity Mode.
RA_PF525_VEL	PowerFlex 525 Velocity Mode control. This instruction allows simple PowerFlex 525 drive control in Velocity Mode.
RA_PF525_POS	PowerFlex 525 Position Mode control. This instruction allows simple PowerFlex 525 drive control in Position Mode.

Table 50 - UDFB Motion Instruction List for Kinetix 5100 Drives

Name	Description
raC_Opr_K5100_MSO	Motion Servo On. Use the Motion Servo On instruction to activate the drive output and to activate the drive servo loops.
raC_Opr_K5100_MSF	Motion Servo Off. Use the Motion Servo Off instruction to deactivate the drive output and to deactivate the drive servo loops.
raC_Opr_K5100_MAJ	Motion Axis Jog Use the Motion Axis Jog instruction to accelerate or decelerate the motor at a constant speed without termination.
raC_Opr_K5100_MAT	Motion Axis Torque Use the Motion Axis Torque instruction to use torque limiting while a predefined speed is used to move the motor.
raC_Opr_K5100_MAM	Motion Axis Move Use the Motion Axis Move instruction to move the motor to a specified position.
raC_Opr_K5100_MAH	Motion Axis Home Use the Motion Axis Home instruction to home the motor.
raC_Opr_K5100_MAG	Motion Axis Gear Use the Motion Axis Gear instruction to set the gear ratio between a pulse-source and follower drive. **IMPORTANT:** This UDFB changes the drive E-gear ratio; Slave/Follower ID151 (P1.044) and Master ID152 (P1.045) Counts. If your drive is positioning, be aware that the units are impacted because the E-gear ratio controls the counts/motor rotation value.

Table 50 - UDFB Motion Instruction List for Kinetix 5100 Drives (Continued)

Name	Description
raC_Opr_K5100_MAS	Motion Axis Stop Use the Motion Axis Stop instruction to stop a specific motion process on the motor or to stop the motor completely.
raC_Opr_K5100_MAFR	Motion Axis Fault Reset. Use the Motion Axis Fault Reset instruction to clear many motion faults for the drive. Some faults cannot be cleared until you cycle power to the drive. The faults, which can be cleared by raC_xxx_K5100_MAFR, are listed in the fault list section.
raC_Opr_K5100_MAI	Motion Axis Index Use the Motion Axis Index instruction to execute the specified PR (index) function of the drive. Use K5100C configuration software or explicit messaging to set the PR (index) parameters. The raC_Opr_K5100_MAI instruction specifies the PR (index) number to be executed.

For details on each UDFB instruction, see .

Download the User-defined Function Block Instruction Files

You can download the user-defined function blocks for Kinetix 5100 and PowerFlex 520-series drives from the Product Compatibility Download Center (PCDC) website at rok.auto/pcdc.

In the Search PCDC text field (Figure 46), enter "Kinetix 5100 UDFB Library" or "PowerFlex 520-series UDFB Library".

Figure 46 - Search PCDC
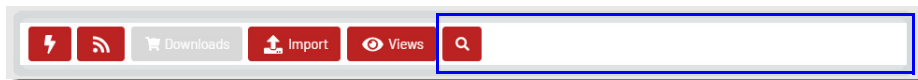
Select the files that you want to download from the search results.

Import the User-defined Function Block Instruction Files

To import the user-defined function blocks into your Connected Components Workbench project, do the following.

1. In the Controller Organizer, right-click the controller and select Import > Import Exchange File.
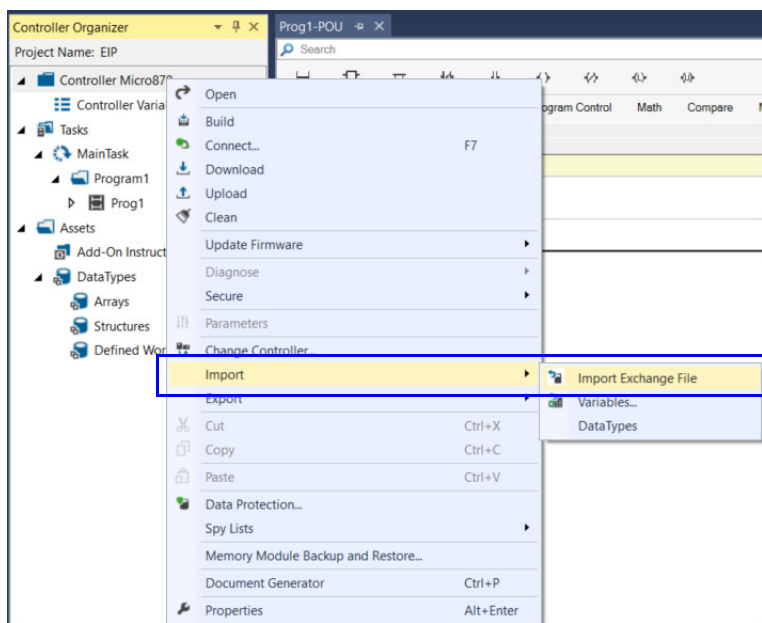
Alternatively, from the Device menu, select Import > Import Exchange File.
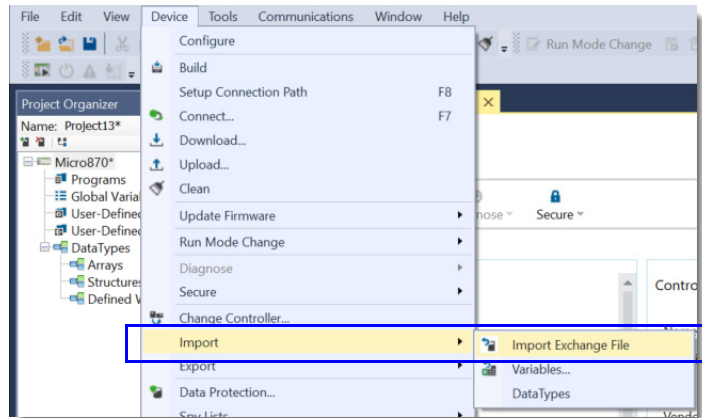
The Import Export dialog box appears.
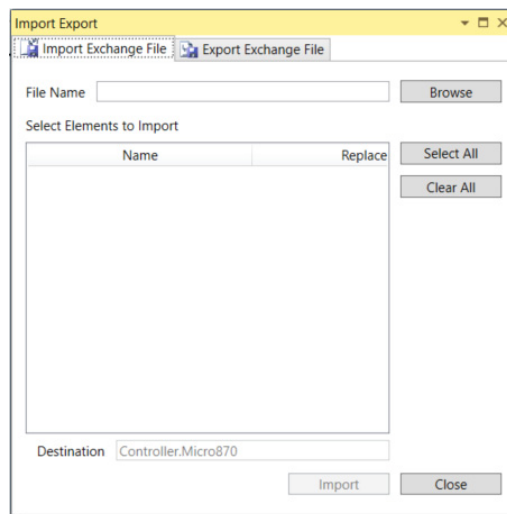
2.   Select Browse and navigate to the folder that contains the UDFB files.

3.   Select the UDFB file to add and select Open.

UDFBs for Kinetix 5100 Drives
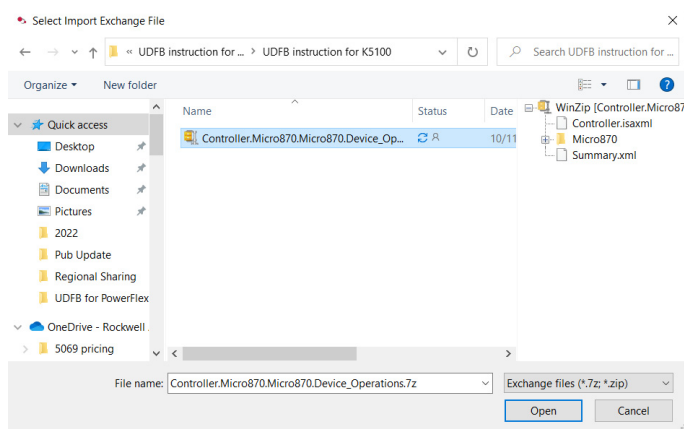
UDFBs for PowerFlex 520-series Drives
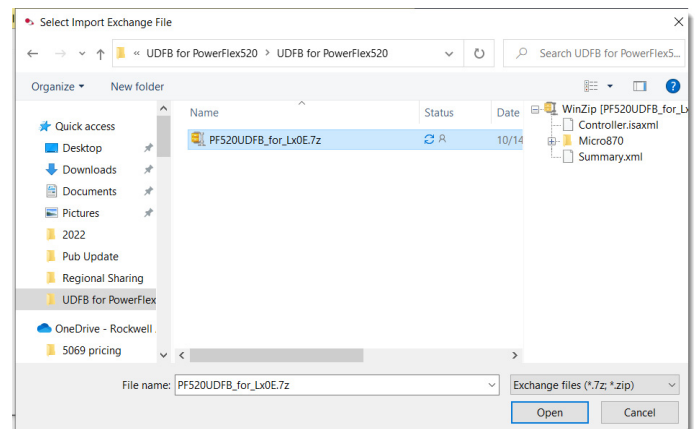
The selected UDFB file is shown in the dialog box.

UDFBs for Kinetix 5100 Drives
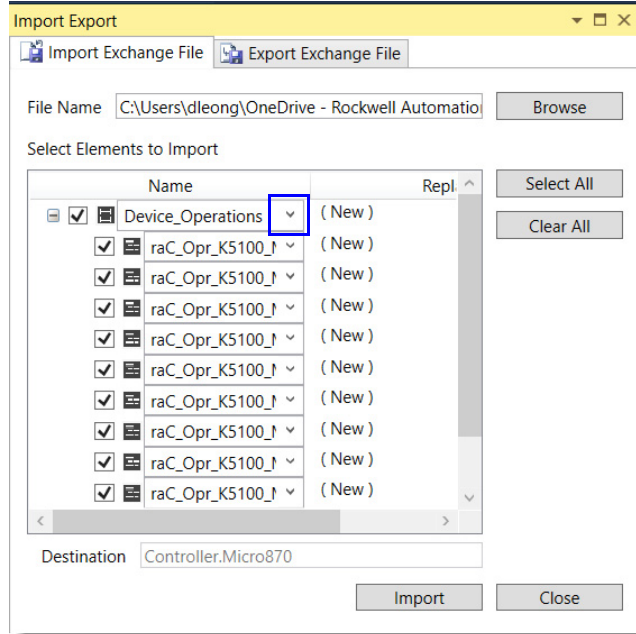
UDFBs for PowerFlex 520-series Drives
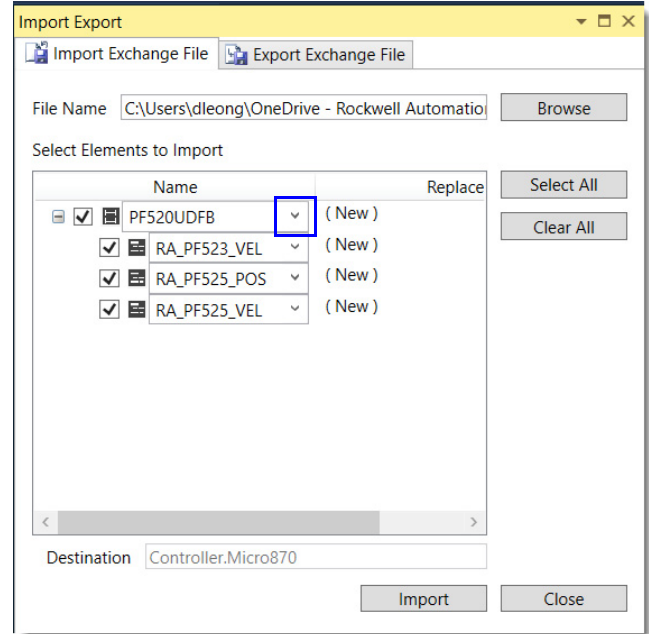
4. Select the pull-down arrow to expand the list and see all UDFBs within the UDFB file.

5. Verify that the checkboxes next to the UDFBs that are required for your application are selected. You can clear the checkboxes for UDFBs that are not required.

6. Select Import to add the selected UDFBs into the project.

The imported UDFBs appear in the Controller Organizer under the Add-On Instructions folder, along with the Add-On defined data types, which appear in the Controller Organizer under the Data Types > Structures folder.

UDFBs for Kinetix 5100 Drives   **UDFBs for PowerFlex 520-series Drives**

There are three UDFBs for PowerFlex 520-series drives to provide a necessary function block with Assembly Instance.

There are 10 UDFBs (raC_Opr_K5100_xxx) and one Device Object instruction (raC_Dvc_K5100) for Kinetix 5100 drives firmware revision 2 or later to provide a necessary function block with output Assembly Instance 106 (or "Connection" is "Data with Camming") for the K5100 module configuration. These UDFBs may not work if output Assembly Instance 104 is configured (or "Connection" is "Data") for the K5100 module configuration.

The UDFBs also appear in the (User defined) tab in the ladder logic toolbox.

UDFBs for Kinetix 5100 Drives
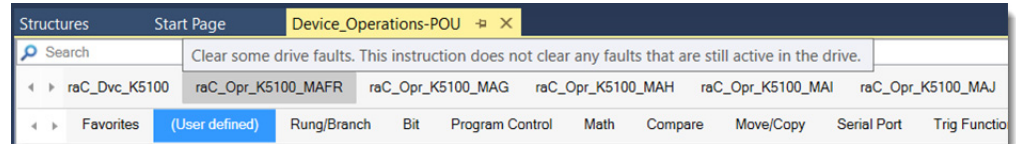
UDFBs for PowerFlex 520-series Drives
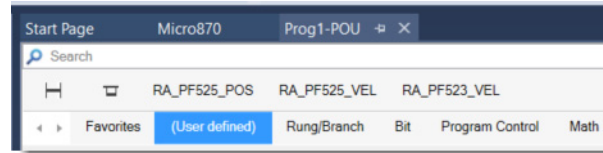
Connection Fault Codes

The following table lists the possible connection faults that can occur and the corrective actions that you can take to resolve the fault.

Table 51 - List of Connection Fault Codes

Fault Code	Display Text	Fault	Corrective Action
0x0001	Connection failure.	A connection to a module failed.	Extended status codes in these ranges provide more detail as to the connection failure.
0x0002	Resource unavailable.	Possible causes include the following: • There are not enough connections available either for the controller or for the communication module being used to connect through. • The I/O module that is targeted does not have enough connections available.	• Check the connection use of the controller or communication module. If all connections are used, try to free some of the used connections or add another module to route the errant connection through. • Check the number of controllers making a connection to this I/O module and verify that the number of connections is within the limits of the I/O module.
0x0003	Invalid value in an object specific data parameter of a service request.	A parameter that is associated with the request was invalid. Possible causes include the following: • The module in use (that is, the physical module) is different than the module specified in the I/O configuration tree and is therefore causing the connection or service to fail. • The fault may occur even when the module passes the electronic keying test. This may result when Disable Keying or Compatible Keying options were used in the module configuration instead of the Exact Match option. • Despite passing the electronic keying test, the module being connected to does not have the same features or settings as the module specified in the I/O configuration tree and does not support the connection or service being attempted.	Check the module in use and verify that it exactly matches the module that is specified in the application. For more information about electronic keying, see the user manual for the module you are using.
0x0004	Connection Request Error: Bad Segment.	IOI segment error on connection request.	The target module does not like one of the parameters in the connection request. This may occur if improper module keying is being used or if the wrong map type attribute is set in the map entry.

Table 51 - List of Connection Fault Codes (Continued)

Fault Code	Display Text	Fault	Corrective Action
0x0005	Connection Request Error: Bad Class.	The controller is attempting to make a connection to the module and has received an error. Possible causes include the following: • The configured address for the connection to the module is incorrect. • The module in use (that is, the physical module) is different than the module specified in the I/O configuration tree and is therefore causing the connection or service to fail. • The fault may occur even when the module passes the electronic keying test. This may result when Disable Keying or Compatible Keying options were used in the module configuration instead of the Exact Match option. • Despite passing the electronic keying test, the module being connected to does not have the same features or settings as the module specified in the I/O configuration tree and does not support the connection or service being attempted.	• Check the module in use and verify that it exactly matches the module that is specified in the application. For more information about electronic keying, see the user manual for the module you are using. • If you are using a 1756-DHRIO module, verify that the Channel type selected in the software (DH+™ or remote I/O network) matches the module's rotary switch settings.
0x0006	Partial data transferred.	Only part of the expected data was transferred. Possible causes include the following: • The response buffer is too small to handle the response data. • The module in use (that is, the physical module) is different than the module specified in the I/O configuration tree and is therefore causing the connection or service to fail. • The fault may occur even when the module passes the electronic keying test. This may result when Disable Keying or Compatible Keying options were used in the module configuration instead of the Exact Match option. • Despite passing the electronic keying test, the module being connected to does not have the same features or settings as the module specified in the I/O configuration tree and does not support the connection or service being attempted.	Check the module in use and verify that it exactly matches the module that is specified in the application. For more information about electronic keying, see the user manual for the module you are using.
0x0007	Messaging connection lost.	The messaging connection was lost.	A service request is unconnected, but should be connected.
0x0008	Service Request Error: Unsupported Service.	The controller is attempting to request a service from the module that is not supported by the module. Possible causes include the following: • The module in use (that is, the physical module) is different than the module specified in the I/O configuration tree and is therefore causing the connection or service to fail. • The fault may occur even when the module passes the electronic keying test. This may result when Disable Keying or Compatible Keying options were used in the module configuration instead of the Exact Match option. • Despite passing the electronic keying test, the module being connected to does not have the same features or settings as the module specified in the I/O configuration tree and does not support the connection or service being attempted.	Check the module in use and verify that it exactly matches the module that is specified in the application. For more information about electronic keying, see the user manual for the module you are using.
0x0009	Module Configuration Rejected: parameter error.	The configuration for the module is invalid. The module configuration may have been changed in the Data Monitor or programmatically.	• Additional fault information for this fault will be displayed as a hex code on the Connections tab. If available for the module, open the Connections tab of the Module Properties dialog box for the additional fault code. The additional fault code indicates the configuration parameter that is causing the fault. You may have to correct multiple parameters before this fault is cleared and the module is connected. • Verify that the configuration is valid by using the module configuration software to validate your configuration. Consult the module user manual for a list of fault codes to determine the configuration parameter that is in error.
0x000B	Object already in requested mode or state.	The object is already in the mode/state being requested by the service.	You do not have to request a mode/state because the object is already in the requested mode/state.

Table 51 - List of Connection Fault Codes (Continued)

Fault Code	Display Text	Fault	Corrective Action
0x000C	Service Request Error: Invalid mode or state for service request.	The controller is attempting to request a service from the module and has received an error.	First, verify that the module is not faulted. For an I/O module, this may indicate that the module has one of the following conditions: • Limited communication is possible, but the module has a Major Fault. • A firmware update needs to be completed or is currently being completed. See the Module Info tab to determine the exact cause.
0x000D	Object Already Exists.	An object instance is created where the instance exists.	Use another object instance number.
0x000E	Attribute value not settable.	A MSG instruction is configured to change an attribute that cannot be changed.	See the device user manual for available programming options.
0x000F	Access permission denied for requested service.	A MSG instruction has been configured to delete a map object that cannot be deleted.	See the device user manual for available programming options.
0x0010	Mode or state of module does not allow object to perform requested service.	The state of the device prevents a service request from being handled.	See the device user manual for available programming options.
0x0011	Reply data too large.	The reply to a message has a data size that is too large for the destination.	Change the destination to a tag that can handle the data size and type being returned.
0x0012	Requested service specifies an operation that is going to fragment a primitive data value.	The service specified an operation that is going to fragment a primitive data value (for example, half a REAL data type).	• The size of the data being read or written must be rounded to the next primitive data type size. • The service is operating on data that is too large for the underlying communication size limits.
0x0013	Module Configuration Rejected: Data size too small.	The configuration for the module is invalid. Not enough configuration data was sent.	Verify that the correct module is being targeted.
0x0014	Undefined or unsupported attribute.	A MSG instruction is configured to change an attribute that does not exist.	See the device user manual for available programming options.
0x0015	Module Configuration Rejected: Data size too large.	The configuration for the module is invalid. Too much configuration data was sent.	Verify that the correct module is being targeted.
0x0100	Connection Request Error: Module in Use.	The connection being accessed is already in use.	Check for these conditions: • The controller is attempting to make a specific connection to a module and the module cannot support multiple connections. • The target of a connection recognizes that the owner is attempting to remake a connection that is already running.
0x0103	Service Request Error: CIP transport class not supported.	This indicates that the controller is requesting a transport class that is not supported by the module. Possible causes include the following: • The controller is requesting services that are not supported by the module. • The module in use (that is, the physical module) is different than the module specified in the I/O configuration tree and is therefore causing the connection or service to fail. • The fault may occur even when the module passes the electronic keying test. This may result when Disable Keying or Compatible Keying options were used in the module configuration instead of the Exact Match option. • Despite passing the electronic keying test, the module being connected to does not have the same features or settings as the module specified in the I/O configuration tree and does not support the connection or service being attempted.	Check the module in use and verify that it exactly matches the module that is specified in the application. For more information about electronic keying, see the user manual for the module you are using.

Table 51 - List of Connection Fault Codes (Continued)

Fault Code	Display Text	Fault	Corrective Action
0x0106	Connection Request Error: Module owned and configured by another controller. Module may accept only one connection if Unicast is used.	An ownership conflict occurred for the connection.	Check for these conditions: • The Connection Request to this module has been rejected due to an ownership conflict with another owner (for example, another controller). This may occur with modules, such as output modules that allow only one owner to configure and control its outputs. This fault may also occur if the module is configured as Listen Only and supports only one connection. • If the owner is connected to the module with a unicast connection over EtherNet/IP network, other connections to the module may fail because the owner controls the one connection. • If the owner is connected to the module with a multicast connection over EtherNet/IP network, unicast connections to the module may fail because the owner controls the one connection. Configure both the owner and the Listen Only connection as multicast.
0x0107	Connection Request Error: Target Connection Not Found.	A connection being accessed was not found. Possible causes include the following: • You have configured module to use a unicast connection over EtherNet/IP network, but the producer does not support unicast connections. • The module in use (that is, the physical module) is different than the module specified in the I/O configuration tree and is therefore causing the connection or service to fail. • The fault may occur even when the module passes the electronic keying test. This may result when Disable Keying or Compatible Keying options were used in the module configuration instead of the Exact Match option. • Despite passing the electronic keying test, the module being connected to does not have the same features or settings as the module specified in the I/O configuration tree and does not support the connection or service being attempted.	Check the module in use and verify that it exactly matches the module that is specified in the application. For more information about electronic keying, see the user manual for the module you are using.
0x0108	Connection Request Error: Connection type (Multicast/Unicast) not supported.	The controller is requesting a connection type that is not supported by the module. Possible causes include the following: • You have configured the module to use a unicast connection over EtherNet/IP network, but the producer does not support unicast connections. • The module in use (that is, the physical module) is different than the module specified in the I/O configuration tree and is therefore causing the connection or service to fail. • The fault may occur even when the module passes the electronic keying test. This may result when Disable Keying or Compatible Keying options were used in the module configuration instead of the Exact Match option. • Despite passing the electronic keying test, the module being connected to does not have the same features or settings as the module specified in the I/O configuration tree and does not support the connection or service being attempted.	Check the module in use and verify that it exactly matches the module that is specified in the application. For more information about electronic keying, see the user manual for the module you are using.

Table 51 - List of Connection Fault Codes (Continued)

Fault Code	Display Text	Fault	Corrective Action
0x0109	Connection Request Error: Invalid connection size.	The connection size is inconsistent with the expected size. Possible causes include the following: • The controller is attempting to set up a connection with the module and cannot because the size of the connection is invalid. • The module in use (that is, the physical module) is different than the module specified in the I/O configuration tree and is therefore causing the connection or service to fail. • The fault may occur even when the module passes the electronic keying test. This may result when Disable Keying or Compatible Keying options were used in the module configuration instead of the Exact Match option. • Despite passing the electronic keying test, the module being connected to does not have the same features or settings as the module specified in the I/O configuration tree and does not support the connection or service being attempted.	• Check the module in use and verify that it exactly matches the module that is specified in the application. For more information about electronic keying, see the user manual for the module you are using. • If the module is a 1756 ControlNet® module, verify that the chassis size is correct. For remote I/O adapters, verify that the rack size and rack density are correct.
0x0110	Connection Request Error: Module not configured.	The controller is attempting to set up a Listen Only connection with the module and cannot because the module has not been configured and connected to by an owner (for example, another controller).	This controller is not an owner of this module because it is attempting to establish a Listen Only connection, which requires no module configuration. The controller cannot connect until an owner configures and connects to the module first.
0x0111	Requested Packet Interval (RPI) out of range.	RPI not supported. Possible causes include the following: • The Requested Packet Interval (RPI) specified is invalid for this module or for a module in the path to this module. • The module in use (that is, the physical module) is different than the module specified in the I/O configuration tree and is therefore causing the connection or service to fail. • The fault may occur even when the module passes the electronic keying test. This may result when Disable Keying or Compatible Keying options were used in the module configuration instead of the Exact Match option. • Despite passing the electronic keying test, the module being connected to does not have the same features or settings as the module specified in the I/O configuration tree and does not support the connection or service being attempted.	• Check the module in use and verify that it exactly matches the module that is specified in the application. For more information about electronic keying, see the user manual for the module you are using. • For Listen Only connections, the RPI set by the owner of this module is slower than the one requested. Either increase the requested RPI or decrease the RPI the owner controller is using. See the Connection tab for valid RPI values.
0x0113	Connection Request Error: Module connection limit exceeded.	The number of connections is greater than what is available on the module. The number of connections must be reduced or the hardware must be upgraded.	To reduce the number of connections: • Change the FLEX™ I/O communication adapter Communications Format from Input or Output configuration to Rack Optimization. When the Communications Format changes, the adapter must be removed and recreated in the I/O configuration tree. • If the configuration uses messaging over ControlNet, sequence the messages to reduce the number that are executing simultaneously, or reduce the number of messages. Messages (MSG instructions) also use connections.
0x0114	Electronic Keying Mismatch: Electronic keying product code and/or vendor ID mismatch.	The Product Code of the actual module hardware does not match the Product Code of the module that is created in the software.	Electronic Keying failed for this module. You may have a mismatch between the module that is created in the software and the actual module hardware.
0x0115	Electronic Keying Mismatch: Electronic Keying product type mismatch.	The Product Type of the actual module hardware does not match the Product Type of the module that is created in the software.	Electronic Keying failed for this module. You may have a mismatch between the module that is created in the software and the actual module hardware.
0x0116	Electronic Keying Mismatch: Major and/or Minor revision invalid or incorrect.	The Major or Minor revisions of the module do not match the Major or Minor revisions of the module that is created in the software.	Electronic Keying failed for this module. • Verify that you have specified the correct Major and Minor Revision if you have chosen Compatible Module or Exact Match keying. • You may have a mismatch between the module that is created in the software and the actual module hardware.

Table 51 - List of Connection Fault Codes (Continued)

Fault Code	Display Text	Fault	Corrective Action
0x0117	Connection Request Error: Invalid Connection Point.	The connection is to an invalid port or port that is already in use. Possible causes include the following: • Another controller owns this module and has connected with a Communications Format different than the one chosen by this controller. Verify that the Communications Format chosen is identical to that chosen by the first owner controller of the module. • The controller may be attempting to connect to a nonexistent tag in a producing controller. • The module in use (that is, the physical module) is different than the module specified in the I/O configuration tree and is therefore causing the connection or service to fail. • The fault may occur even when the module passes the electronic keying test. This may result when Disable Keying or Compatible Keying options were used in the module configuration instead of the Exact Match option. • Despite passing the electronic keying test, the module being connected to does not have the same features or settings as the module specified in the I/O configuration tree and does not support the connection or service being attempted.	Additional Error Information for this fault appears as the tag name associated with the controller to controller (C2C) that has the fault. Check the module in use and verify that it exactly matches the module that is specified in the application. For more information about electronic keying, see the user manual for the module you are using.
0x0118	Module Configuration Rejected: Format error.	An invalid configuration format was used. Possible causes include the following: • The configuration class that is specified does not match the class that is supported by the module. • The connection instance is not recognized by the module. • The path that is specified for the connection is inconsistent. • The module in use (that is, the physical module) is different than the module specified in the I/O configuration tree and is therefore causing the connection or service to fail. • The fault may occur even when the module passes the electronic keying test. This may result when Disable Keying or Compatible Keying options were used in the module configuration instead of the Exact Match option. • Despite passing the electronic keying test, the module being connected to does not have the same features or settings as the module specified in the I/O configuration tree and does not support the connection or service being attempted.	Check the module in use and verify that it exactly matches the module that is specified in the application. For more information about electronic keying, see the user manual for the module you are using.
0x0119	Connection Request Error: Module not owned.	The controlling connection is not open.	Where a Listen Only connection is requested, the controlling connection is not open.
0x011A	Connection Request Error: Out of Connection Resources.	The controller is attempting to set up a connection with the module and cannot because required resources are unavailable.	• If the module is a 1756 ControlNet module, up to five controllers can make Rack Optimization connections to the module. Verify that this number has not been exceeded. • If the module is a 1794-ACN15, 1794-ACNR15, or 1797-ACNR15 adapter, only one controller can make a Rack Optimization connection to the module. Verify that this number has not been exceeded.
0x0127	Connection Request Error: Invalid output size.	The controller is attempting to set up a connection with the module and cannot because the output packet size is invalid.	If generic Ethernet module is being used, check if the output size is correct and expected by the target device.
0x0128	Connection Request Error: Invalid input size.	The controller is attempting to set up a connection with the module and cannot because the input packet size is invalid.	If generic Ethernet module is being used, check if the input size is correct and expected by the target device.
0x0203	Connection timed out.	The owner or originator recognizes that the target device is on the network or backplane. However, I/O data and messages are not being responded to.	The target can be reached, but its response is not as expected. For example, this fault may be indicated where multicast Ethernet packets are not returned. When this fault occurs, the controller usually attempts to continuously remove and remake the connection. If you use FLEX I/O modules, verify that you are using the correct terminal device.

Table 51 - List of Connection Fault Codes (Continued)

Fault Code	Display Text	Fault	Corrective Action
0x0204	Connection Request Error: Connection request timed out.	The controller is attempting to make a connection; however, the target module is not responding. The device also appears to be missing from the backplane or network.	To recover, take the following steps. • Verify that the module has not been removed and is still functioning and receiving power. • Verify that the correct slot number has been specified. • Verify that the module is properly connected to the network. • If you are using FLEX I/O modules, verify that the correct terminal block is used.
0x0205	Connection Request Error: Invalid parameter.	The controller is attempting to set up a connection with the module and has received an error (a parameter is in error). Possible causes include the following: • The module in use (that is, the physical module) is different than the module specified in the I/O configuration tree and is therefore causing the connection or service to fail. • The fault may occur even when the module passes the electronic keying test. This may result when Disable Keying or Compatible Keying options were used in the module configuration instead of the Exact Match option. • Despite passing the electronic keying test, the module being connected to does not have the same features or settings as the module specified in the I/O configuration tree and does not support the connection or service being attempted.	Check the module in use and verify that it exactly matches the module that is specified in the application. For more information about electronic keying, see the user manual for the module you are using.
0x0206	Connection Request Error: Requested size too large.	The controller is attempting to set up a connection with the module and has received an error – the request size is too large. Possible causes include the following: • The module in use (that is, the physical module) is different than the module specified in the I/O configuration tree and is therefore causing the connection or service to fail. • The fault may occur even when the module passes the electronic keying test. This may result when Disable Keying or Compatible Keying options were used in the module configuration instead of the Exact Match option. • Despite passing the electronic keying test, the module being connected to does not have the same features or settings as the module specified in the I/O configuration tree and does not support the connection or service being attempted.	• Verify that the path to this module is sufficiently close to the controller. • Check the module in use and verify that it exactly matches the module that is specified in the application. For more information about electronic keying, see the user manual for the module you are using.
0x0301	Connection Request Error: Out of buffer memory.	The controller is attempting to set up a connection with the module and has received an error: a module in the path is out of memory. Possible causes include the following: • The controller may be attempting to connect to a tag in a producing controller that is not marked as being produced. • The controller may be attempting to connect to a tag in a producing controller. That tag may not be configured to allow enough consumers. • The size or number of connections through this module must be reduced. • One of the network modules between the module and the controller may be out of memory. Check network configuration of the system. • The module may be out of memory. Check the system configuration and capabilities of the module. • The module in use (that is, the physical module) is different than the module specified in the I/O configuration tree and is therefore causing the connection or service to fail. • The fault may occur even when the module passes the electronic keying test. This may result when Disable Keying or Compatible Keying options were used in the module configuration instead of the Exact Match option. • Despite passing the electronic keying test, the module being connected to does not have the same features or settings as the module specified in the I/O configuration tree and does not support the connection or service being attempted.	Check the module in use and verify that it exactly matches the module that is specified in the application. For more information about electronic keying, see the user manual for the module you are using.

Table 51 - List of Connection Fault Codes (Continued)

Fault Code	Display Text	Fault	Corrective Action
0x0302	Connection Request Error: Out of communication bandwidth.	The controller is attempting to set up a connection with the module and has received an error: a module in the path has exceeded its communication bandwidth capacity.	• Increase the Requested Packet Interval (RPI) and reconfigure your network with RSNetWorx™ software. • Distribute the load on another bridge module.
0x0303	Connection Request Error: No bridge available.	The controller is attempting to set up a connection with the module and has received an error: a module in the path has exceeded its communication bandwidth capacity.	Distribute the load on another bridge module.
0x0304	Not configured to send scheduled data.	The ControlNet module is not scheduled to send data.	Use RSNetWorx for ControlNet software to schedule or reschedule the ControlNet network.
0x0305	Connection Request Error: ControlNet configuration in controller does not match configuration in bridge.	The ControlNet configuration in the controller does not match the configuration in the bridge module. This may occur because a ControlNet module was changed after the network was scheduled, or because a new control program has been loaded into the controller.	Use RSNetWorx for ControlNet software to reschedule the connections.
0x0306	ControlNet Keeper not available.	The ControlNet Configuration Master (CCM) cannot be found. The 1756-CNB module and PLC-5® ControlNet processor are the only devices capable of being a CCM and the CCM must be node 1. This fault may temporarily occur when the system is powered up and is being cleared when the CCM is located.	Verify that a 1756-CNB modules or PLC-5 ControlNet processor is at node 1 and is functioning properly.
0x0311	Connection Request Error: Invalid port.	The controller is attempting to set up a connection with the module and has received an error.	Verify that all modules in the I/O Configuration tree are the correct modules.
0x0312	Connection Request Error: Invalid link address.	The controller is attempting to set up a connection with the module and has received an error: an invalid link address has been specified. A link address can be a slot number, a network address, or the remote I/O chassis number and starting group.	• Verify that the chosen slot number for this module is not greater than the size of the rack. • Verify that the ControlNet node number is not greater than the maximum node number configured for the network in RSNetWorx for ControlNet software.
0x0315	Connection Request Error: Invalid segment type.	The segment type or route is invalid. Possible causes include the following: • The controller is attempting to set up a connection with the module and has received an error: the connection request is invalid. • The module in use (that is, the physical module) is different than the module specified in the I/O configuration tree and is therefore causing the connection or service to fail. • The fault may occur even when the module passes the electronic keying test. This may result when Disable Keying or Compatible Keying options were used in the module configuration instead of the Exact Match option. • Despite passing the electronic keying test, the module being connected to does not have the same features or settings as the module specified in the I/O configuration tree and does not support the connection or service being attempted.	Check the module in use and verify that it exactly matches the module that is specified in the application. For more information about electronic keying, see the user manual for the module you are using.
0x0317	Connection Request Error: Connection not scheduled.	The controller is attempting to set up a ControlNet connection with the module and has received an error.	Use RSNetWorx for ControlNet software to schedule or reschedule the connection to this module.
0x0318	Connection Request Error: Invalid link address - cannot route to self.	The controller is attempting to set up a connection with the module and has received an error: the link address is invalid.	Verify that the associated ControlNet module has the correct slot or node number selected.
0x0319	Connection Request Error: No secondary resources available in redundant chassis.	The controller is attempting to set up a connection with the module and has received an error: the redundant module does not have the necessary resources to support the connection.	Reduce the size or number of connections through this module or add another controller or ControlNet module to the system.
0x031B	Connection Request Error: Rack Connection Refused.	The controller is attempting to set up a Direct connection with the module and has received an error.	A rack-optimized connection has already been established to this module through the 1756-CNB/R in the same chassis. • Connect to this module through the 1756-CNB/R in the same chassis. • Connect to this module through another 1756-CNB/R to use a Direct connection. • Change the first connection from rack-optimized to Direct, and then re-establish the second direct connection. • Connect to this module from a controller in the same chassis as the module (do not connect through a 1756-CNB/R).

Table 51 - List of Connection Fault Codes (Continued)

Fault Code	Display Text	Fault	Corrective Action
0x031E	Connection Request Error: Cannot consume tag.	The controller is attempting to connect to a tag in a producing controller and has received an error.	• The controller is attempting to connect to a tag in a producing controller and that tag has already been used by too many consumers. • Increase the maximum number of consumers on the tag.
0x031F	Connection Request Error: Tag not published.	The Consume tag was not configured to be produced in the target module.	Make sure the name of the tag being consumed is spelled correctly in both the consumer and producer.
0x0322	Connection Request Error: Data format for requested connection does not match data format of connection already established.	A new connection that is requested does not match the existing connection.	Check the controllers that are using the connection and verify that all configurations are identical.
0x0800	Network link in path to module is offline.	A connection has already been established to the target module at a different RPI and it cannot be changed to accommodate the RPI specified.	It is either not physically connected to the network (media disconnected) or otherwise unable to forward the request onto the destination network.
0x0801	Incompatible Multicast RPI.	The controlling application has not initialized the data to be produced by the target device. This may be caused when "Send Data" connections are configured in a target device and the controlling application for that target device has not initialized the data to be produced.	• Configure the tag as unicast. • Adjust the RPI so that all multicast consumers of the tag use the same RPI. • Configure the RPI so that is in the range that is allowed by the producer.
0x0810	No target application data available.	Given data type does not match Produce tag data type in the target module.	For the target device associated with the "Send Data" connection reporting this connection error, start the controlling application and perform at least one write of data. See the user manual for the target device and its controlling application for information on how to do this.
0x0814	Connection Request Error: Data Type Mismatch.	Given data type does not match Produce tag data type in the target module.	Verify that both data types match.
0xFD02	Connection Request Error: No error code is supplied by an I/O module to describe an I/O fault.	No error code is supplied by an I/O module to describe an I/O fault.	See the user manual for your device.

Notes:

Motion Control

Generally two types of motion control are used in Micro800 controller motion applications that have Kinetix 3 servo drives.

- Indexed Motion – The Micro800 controller issues position indexes to the servo drive using Modbus RTU communications or digital I/O. Used for simple positioning. See publication CC-QS025 for building block example.
- PTO Motion – The Micro800 controller uses pulse and direction outputs to the servo drive for precise control of position and velocity with Modbus RTU communications or digital I/O for feedback. Micro800 motion configuration and instructions make programming easy. See publication CC-QS033 for building block example.

PTO Motion Control

Certain Micro830, Micro850, and Micro870 controllers that are shown in Table 52, support motion control through high-speed pulse-train outputs (PTO). PTO functionality refers to the ability of a controller to generate a specific number of pulses at a specified frequency accurately. These pulses are sent to a motion device, such as a servo drive, which in turn controls the number of rotations (position) of a servo motor. Each PTO is exactly mapped to one axis, to allow for control of simple positioning in stepper motors and servo drives with pulse/direction input.

As the duty cycle of the PTO can be changed dynamically, the PTO can also be used as a pulse-width modulation (PWM) output.

PTO/PWM and motion axes support on the Micro830, Micro850, and Micro870 controllers are summarized as follows.

Table 52 - PTO/PWM and Motion Axis Support on Micro830, Micro850, and Micro870

Controller	PTO (built-in)	Number of Axes Supported
10/16-point [1] 2080-LC30-10QVB 2080-LC30-16QVB	1	1
24-point 2080-LC30-24QVB [1] 2080-LC30-24QBB [1] 2080-LC50-24QVB 2080-L50E-24QVB 2080-LC50-24QBB 2080-L50E-24QBB 2080-LC70-24QBB 2080-L70E-24QBB 2080-LC70-24QBBK 2080-L70E-24QBBK 2080-L70E-24QBBN	2	2
48-point 2080-LC30-48QVB [1] 2080-LC30-48QBB [1] 2080-LC50-48QVB 2080-L50E-48QVB 2080-LC50-48QBB 2080-L50E-48QBB	3	3

(1)   For Micro830 catalogs, Pulse Train Output functionality is only supported from firmware revision 2 and later.

> ⚠ **ATTENTION:** To use the Micro800 Motion feature effectively, you must have a basic understanding of the following:
> • PTO components and parameters
>   See Use the Micro800 Motion Control Feature on page 160 for a general overview of Motion components and their relationships.
> • Programming and working with elements in the Connected Components Workbench software
>   You must have a working knowledge of ladder diagram, structured text, or function block diagram programming to be able to work with motion function blocks, variables, and axis configuration parameters.

> ⚠ **ATTENTION:** To learn more about Connected Components Workbench software and detailed descriptions of the variables for the Motion Function Blocks, you can refer to Connected Components Workbench software Online Help that comes with your Connected Components Workbench software installation.

> **IMPORTANT**   The PTO function can only be used with the controller's embedded I/O. It cannot be used with expansion I/O modules.

Use the Micro800 Motion Control Feature

The Micro800 motion control feature has the following elements. You must have a basic understanding of the function of each element to use the feature effectively.

Table 53 - Components of Motion Control

Element	Description	Page
Pulse-train Outputs	Consists of one pulse output and one direction output. A standard interface to control a servo or stepper drive.	• Input and Output Signals on page 161
Axis	From a system point of view, an axis is a mechanical apparatus that is driven by a motor and drive combination. The drive receives position commands through the Micro800 pulse train outputs interface based on the PLC execution of motion function blocks. On the Micro800 controller, it is a pulse train output and a set of inputs, outputs, and configuration.	• Motion Axis and Parameters on page 172 • Motion Axis Configuration in Connected Components Workbench on page 182
Motion Function Blocks	A set of instructions that configure or act upon an axis of motion.	• Connected Components Workbench Online Help • Motion Control Function Blocks on page 164 • Axis_Ref Data Type on page 177 • Function Block and Axis Status Error Codes on page 179 • Homing Function Block on page 191
Jerk	Rate of change of acceleration. The Jerk component is mainly of interest at the start and end of the motion. Too high of a Jerk may induce vibrations.	• Acceleration, Deceleration, and Jerk Inputs on page 165

To use the Micro800 motion feature, you must:

1. Configure the Axis Properties

   For instructions, see Motion Axis Configuration in Connected Components Workbench on page 182.

2. Write your motion program through the Connected Components Workbench software

   For instructions on how to use the Micro800 motion control feature, see the Micro800 Programmable Controllers: Getting Started with Motion Control Using a Simulated Axis Quick Start, publication 2080-QS001.

3. Wire the Controller

For fixed and configurable inputs/outputs, see <u>Input and Output Signals on page 161</u>.
For reference, see <u>Sample Motion Wiring Configuration on 2080-LC30-xxQVB / 2080-LC50-xxQVB / 2080-LC70-xxQVB on page 163</u>.

The next sections provide a more detailed description of the motion components. You can also see the Connected Components Workbench Online Help for more information about each motion function block and their variable inputs and outputs.

Input and Output Signals

Multiple input/output control signals are required for each motion axis, as described <u>Table 54</u> and <u>Table 55</u>. PTO Pulse and PTO Direction are required for an axis. The rest of the input/outputs can be disabled and reused as regular I/O.

Table 54 - Fixed PTO Input/Output

Motion Signals	PT00 (EM_00)		PT01 (EM_01)		PT02 (EM_02)	
	Logical Name in Software	Name on Terminal Block	Logical Name in Software	Name on Terminal Block	Logical Name in Software	Name on Terminal Block
PTO pulse	_IO_EM_DO_00	O-00	_IO_EM_DO_01	O-01	IO_EM_DO_02	O-02
PTO direction	_IO_EM_DO_03	O-03	_IO_EM_DO_04	O-04	IO_EM_DO_05	O-05
Lower (Negative) Limit switch	_IO_EM_DI_00	I-00	_IO_EM_DI_04	I-04	IO_EM_DI_08	I-08
Upper (Positive) Limit switch	_IO_EM_DI_01	I-01	_IO_EM_DI_05	I-05	IO_EM_DI_09	I-09
Absolute Home switch	_IO_EM_DI_02	I-02	_IO_EM_DI_06	I-06	IO_EM_DI_10	I-10
Touch Probe Input switch	_IO_EM_DI_03	I-03	_IO_EM_DI_07	I-07	IO_EM_DI_11	I-11

Table 55 - Configurable Input/Output

Motion Signals	Input/Output	Notes
Servo/Drive On	OUTPUT	Can be configured as any embedded output.
Servo/Drive Ready	INPUT	Can be configured as any embedded input.
In-Position signal (from servo/motor)	INPUT	Can be configured as any embedded input.
Home Marker	INPUT	Can be configured as any embedded input, from input 0...15.

You can configure the I/O through the axis configuration feature in the Connected Components Workbench software. You should not control any outputs that are assigned for motion in your program. See <u>Motion Axis Configuration in Connected Components Workbench on page 182</u>.

IMPORTANT	If an output is configured for motion, then your program can no longer control or monitor that output and the output cannot be forced. For example, when a PTO Pulse output is generating pulses, the corresponding logical variable IO_EM_DO_xx does not toggle its value and does not display the pulses in the Variable Monitor, but the physical LED gives an indication.
	If an input is configured for motion, then forcing the input only affects your program logic and not motion. For example, if the input Drive Ready is false, then you cannot force Drive Ready to true by forcing the corresponding logical variable IO_EM_DI_xx to be true.

Table 56 - Motion Wiring Input/Output Description

Motion Signals	Input/Output	Description	Uniqueness
PTO pulse	OUTPUT	PTO pulse from the embedded fast output, to be connected to Drive PTO input.	Not Shared
PTO direction	OUTPUT	PTO pulse direction indication, to be connected to Drive Direction input.	Not Shared
Servo/Drive On	OUTPUT	The control signal used to activate/deactivate Servo/Drive. This signal becomes Active when MC_Power (on) is commanded.	Can be shared with multiple drives
Lower (Negative) Limit switch	INPUT	The input for the hardware negative limit switch, to be connected to the mechanical/electrical negative limit sensor.	Not Shared
Upper (Positive) Limit switch	INPUT	The input for the hardware positive limit switch, to be connected to the mechanical/electrical positive limit sensor.	Not Shared
Absolute Home switch	INPUT	The input for the hardware home switch (sensor), to be connected to the mechanical/electrical home sensor.	Not Shared
Touch Probe Input switch	INPUT	The input for the hardware touch probe signal, to be used with the Motion MC_TouchProbe and MC_AbortTrigger function blocks to capture the axis commanded position during the motion path.	Not Shared
Servo/Drive Ready	INPUT	The input signal that indicates Servo/Drive is ready to receive the PTO pulse and direction signal from the controller. No moving function blocks can be issued to an axis before the axis has this signal ready if this signal is Enabled in the motion axis configuration or axis properties page.	Can be shared with multiple drives
In-Position signal (from servo/motor)	INPUT	The input signal that indicates the moving part is in the commanded position. This signal has to be Active after the moving part reaches the commanded position for the MoveAbsolute and MoveRelative function blocks. For MoveAbsolute and MoveRelative function blocks, when In_Position is enabled, the controller reports an error (EP_MC_MECHAN_ERR) if the signal is not active within 5 seconds when the last PTO pulse sent out.	Not Shared
Home Marker	INPUT	This signal is the zero pulse signal from the motor encoder. This signal can be used for fine homing sequence to improve the homing accuracy.	Not Shared

Sample Motion Wiring Configuration on 2080-LC30-xxQVB / 2080-LC50-xxQVB / 2080-LC70-xxQVB
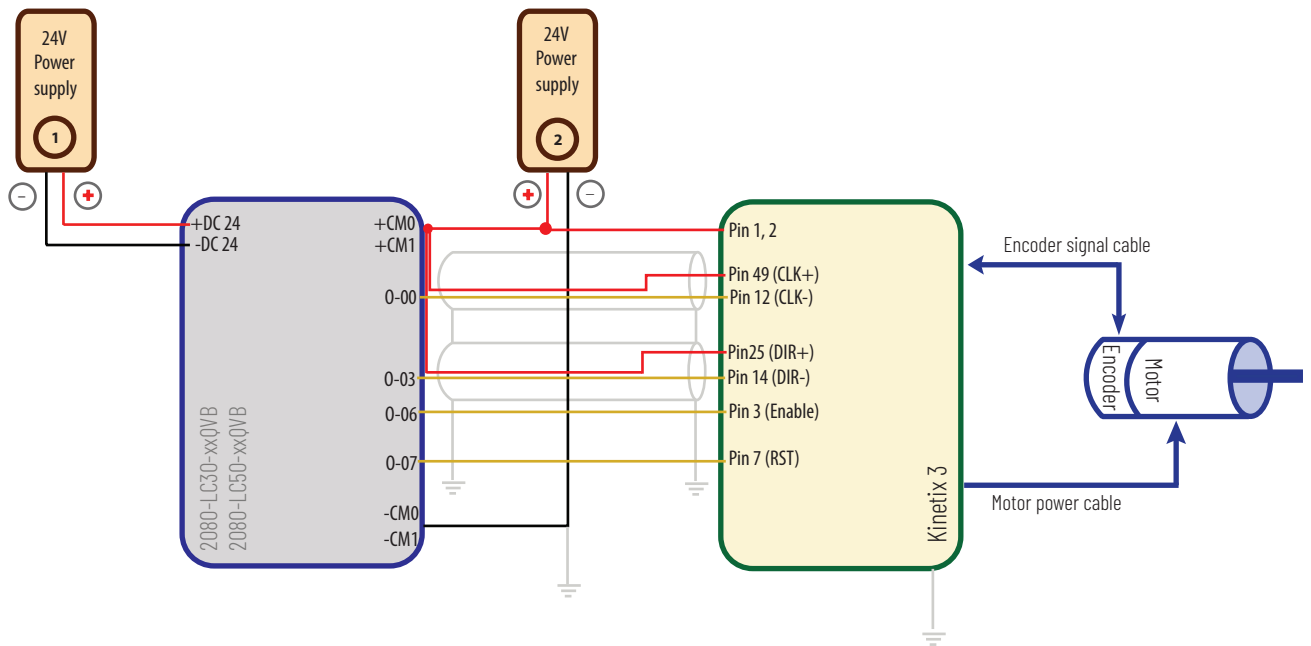
Notes:

1. Drive Enable (Pin 3) and Reset Drive (Pin 7) are operating as sourcing inputs when (Pin 1, 2) connected to ⊖ of Power Supply 2.
2. To help you configure Kinetix 3 drive parameters so that a Micro830/Micro850/Micro870 controller can communicate and control the drive, see publication CC-QS033. The parameter Command Type must be set to "Step/Direction.Positive Logic", and the parameter Controller Output Type must be set to "Open Collector Input".

Sample Motion Wiring Configuration on 2080-LC30-xxQBB / 2080-LC50-xxQBB / 2080-LC70-xxQBB
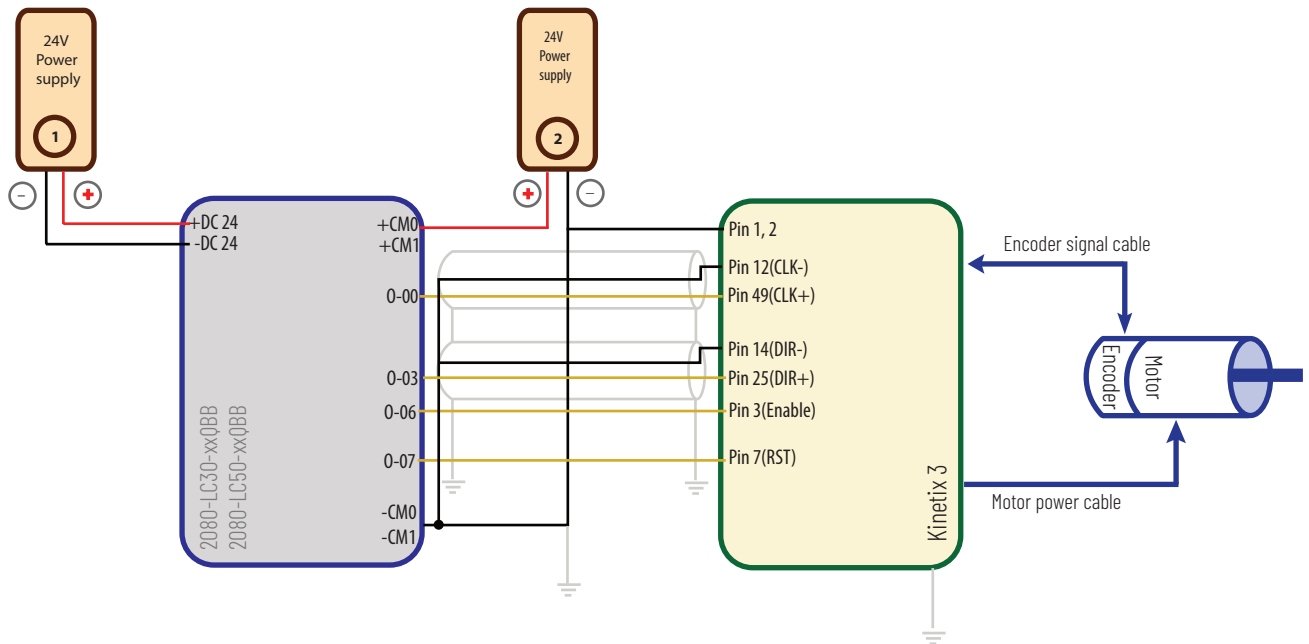
Notes:

1. Drive Enable (Pin 3) and Reset Drive (Pin 7) are operating as sinking inputs when (Pin 1, 2) connected to ⊕ of the Power Supply 2.
2. To help you configure Kinetix 3 drive parameters, so that a Micro830/Micro850/Micro870 controller can communicate and control the drive, see publication CC-QS033. The parameter Command Type must be set to "Step/Direction.Positive Logic", and the parameter Controller Output Type must be set to "Open Collector Input".

Motion Control Function Blocks

Motion control function blocks instruct an axis to a specified position, distance, velocity, and state. Function blocks are categorized as Administrative (Table 57) and Movement (driving motion, Table 58).

Table 57 - Administrative Function Blocks

Function Block Name	Function Block Name
MC_Power	MC_ReadAxisError
MC_Reset	MC_ReadParameter
MC_TouchProbe	MC_ReadBoolParameter
MC_AbortTrigger	MC_WriteParameter
MC_ReadStatus	MC_WriteBoolParameter
MC_SetPosition	

⚠️ **WARNING:** During Run Mode Change (RMC), the MC_Power function block should be disabled, which powers down the axis. Otherwise the axis remains powered even if the function block is deleted.
Take note of the following:

- If a new instance of MC_Power accesses the axis, the axis enters the error stop state.
- If MC_Power is inside a UDFB and any edit is made to the UDFB that changes the UDFB template (for example, adding a local variable), the axis enters the error stop state.

Table 58 - Movement Function Blocks

Function Block Name	Description	Correct Axis State for issuing Function Block
MC_MoveAbsolute	This function block commands an axis to a specified absolute position.	Standstill, Discrete Motion, Continuous Motion
MC_MoveRelative	This function block commands an axis of a specified distance relative to the actual position at the time of execution.	Standstill, Discrete Motion, Continuous Motion
MC_MoveVelocity	This function block commands a never-ending axis move at a specified velocity.	Standstill, Discrete Motion, Continuous Motion
MC_Home	This function block commands the axis to perform the "search home" sequence. The "Position" input is used to set the absolute position when the reference signal is detected, and the configured Home offset is reached. This function block completes at "StandStill" if the homing sequence is successful.	Standstill
MC_Stop	This function block commands an axis stop and transfers the axis to the state "Stopping". It closes any ongoing function block execution. While the axis is in state Stopping, no other function block can perform any motion on the same axis. After the axis has reached velocity zero, the Done output is set to TRUE immediately. The axis remains in the state "Stopping" as long as Execute is still TRUE or velocity zero is not yet reached. Once "Done" is SET and "Execute" is FALSE the axis goes to the state "StandStill".	Standstill, Discrete Motion, Continuous Motion, Homing
MC_Halt	This function block commands an axis to a controlled motion stop. The axis is moved to the state "Discrete Motion", until the velocity is zero. With the Done output set, the state is transferred to "StandStill".	Standstill, Discrete Motion, Continuous Motion

⚠️ **ATTENTION:** During Run Mode Change, the Movement Function Blocks can only be deleted when that Function Block has been done or aborted. Otherwise unintended axis and Function Block behavior may occur.

> **ATTENTION:** Each motion function block has a set of variable inputs and outputs that allows you to control a specific motion instruction. See the Connected Components Workbench Online Help for a description of these variable inputs and outputs.

General Rules for the Motion Control Function Blocks

To work with motion control function blocks, you must be familiar with the following general rules.

Table 59 - General Rules for the Motion Function Block

Parameter	General Rules
Input parameters	**When Execute is True**: The parameters are used with the rising edge of the Execute input. To modify any parameter, it is necessary to change the input parameters and to trigger the motion again. **When Enable is True**: The parameters are used with the rising edge of the Enable input and can be modified continuously.
Inputs exceeding application limits	If a function block is configured with parameters that result in a violation of application limits, the instance of the function block generates an error. The Error output is flagged On, and error information is indicated by the output ErrorID. The controller, in most cases, remains in Run mode, and no motion error is reported as a major controller fault.
Position/Distance Input	For MC_MoveAbsolute function block, the position input is the absolute location that is commanded to the axis. For MC_MoveRelative, the distance input is the relative location (consider the current axis position is 0) from the current position.
Velocity Input	Velocity can be a signed value. Users are advised to use positive velocity. Direction input for the MC_MoveVelocity function block can be used to define the direction of the move (that is, negative velocity x negative direction = positive velocity). For MC_MoveRelative and MC_MoveAbsolute function blocks the absolute value of the velocity is used. Velocity input does not need to be reached if Jerk input is equal to 0.
Direction Input	For MC_MoveAbsolute, the direction input is ignored. (This parameter is reserved for future use.) For MC_MoveVelocity, the direction input value can be 1 (positive direction), 0 (current direction) or -1 (negative direction). For any other value, only the sign is considered. For example, -3 denotes negative direction, +2 denotes positive direction, and so on. For MC_MoveVelocity, the resulting sign of the product value that is derived from velocity x direction decides the motion direction, if the value is not 0. For example, if velocity x direction = +300, then the direction is positive.
Acceleration, Deceleration, and Jerk Inputs	• Deceleration or Acceleration inputs should have a positive value. If Deceleration or Acceleration is set to be a non-positive value, an error is reported (ErrorID: MC_FB_ERR_RANGE). • The Jerk input should have a non-negative value. If Jerk is set to be a negative value, an error is reported. (ErrorID: MC_FB_ERR_RANGE). • If maximum Jerk is configured as zero in the Connected Components Workbench motion configuration, all jerk parameters for the motion function block have to be configured as zero. Otherwise, the function block reports an error (ErrorID: MC_FB_ERR_RANGE). • If Jerk is set as a nonzero value, an S-curve profile is generated. If Jerk is set as zero, a trapezoidal profile is generated. • If the motion engine fails to generate the motion profile prescribed by the dynamic input parameters, the function block reports an error (ErrorID: MC_FB_ERR_PROFILE). See for more information about error codes.

Table 59 - General Rules for the Motion Function Block (Continued)

Parameter	General Rules
Output Exclusivity	**With Execute**: The outputs Busy, Done, Error, and CommandAborted indicate the state of the function block and are mutually exclusive – only one of them can be true on one function block. If execute is true, one of these outputs has to be true. The outputs Done, Busy, Error, ErrorID, and CommandAborted are reset with the falling edge of Execute. However, the falling edge of Execute does not stop or even influence the execution of the actual function block. Even if Execute is reset before the function block completes, the corresponding outputs are set for at least one cycle. If an instance of a function block receives a new Execute command before it completes (as a series of commands on the same instance), the new Execute command is ignored, and the previously issued instruction continues with execution.
Output Exclusivity	**With Enable**: The outputs Valid and Error indicate whether a read function block executes successfully. They are mutually exclusive: only one of them can be true on one function block for MC_ReadBool, MC_ReadParameter, MC_ReadStatus. The Valid, Enabled, Busy, Error, and ErrorID outputs are reset with the falling edge of Enable as soon as possible.
Axis Output	When used in a Function Block Diagram, you can connect the axis output parameter to the Axis input parameter of another motion function block for convenience (for example, MC_POWER to MC_HOME). When used in a Ladder Diagram, you cannot assign a variable to the Axis output parameter of another motion function block because it is read-only.

Table 59 – General Rules for the Motion Function Block (Continued)
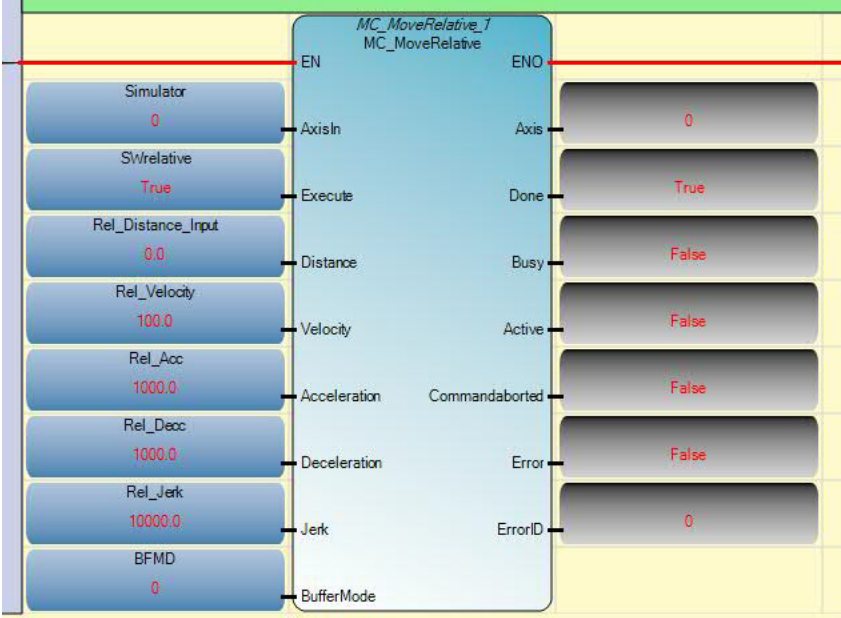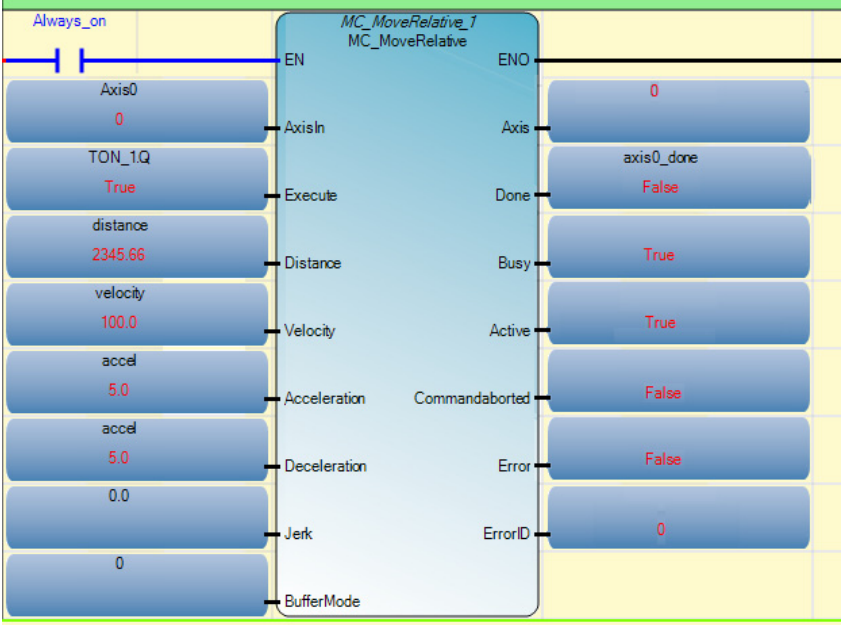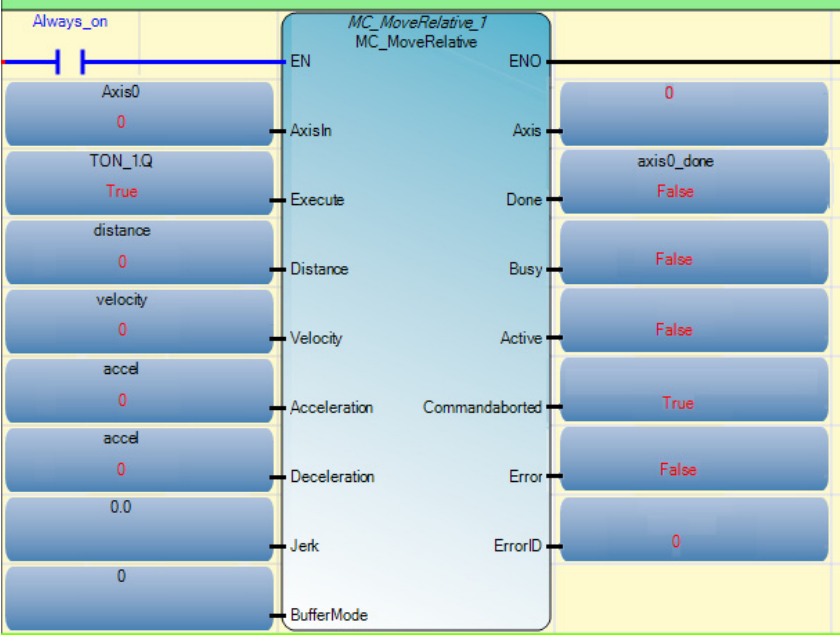
Parameter	General Rules
Behavior of Done Output	The output Done is set when the commanded action has completed successfully. With multiple function blocks working on the same axis in a sequence, the following rule applies: When one movement on an axis is aborted with another movement on the same axis without having reached the final goal, output Done is not set on the first function block.
Behavior of Busy Output	Every function block has a Busy output, which indicates that the function block is not yet finished (for function blocks with an Execute input), and new output values are pending (for function blocks with Enable input). Busy is set at the rising edge of Execute and reset when one of the outputs Done, Aborted, or Error is set, or it is set at the rising edge of Enable and reset when one of the outputs Valid or Error is set. It is recommended that the function block continue executing in the program scan for as long as Busy is true, because the outputs are only updated when the instruction is executing. For example, in the ladder diagram, if the rung becomes false before the instruction finishes executing, the Busy output stays true forever even though the function block has finished executing.
Output Active	In current implementation, buffered moves are not supported. Consequently, Busy and Active outputs have the same behavior.

Table 59 – General Rules for the Motion Function Block (Continued)

| Parameter	General Rules
Behavior of CommandAborted Output	CommandAborted is set when another motion command aborts the current commanded motion. When CommandAborted occurs, other output signals such as InVelocity are reset.
Enable and Valid Status	The Enable input for read function blocks is level-sensitive. On every program scan with the Enable input as true, the function block performs a read and update its outputs. The Valid output parameter shows that a valid set of outputs is available. The Valid output is true as long as valid output values are available and the Enable input is true. The relevant output values are refreshed as long as the input Enable is true. If there is a function block error, and the relevant output values are not valid, then the valid output is set to false. When the error condition no longer exists, the values are updated and the Valid output is set again.
Relative Move versus Absolute Move	Relative move does not require the axis to be homed. It simply refers to a move in a specified direction and distance. Absolute move requires that the axis be homed. It is a move to a known position within the coordinate system, regardless of distance and direction. Position can be negative or positive value.
Buffered Mode	For all motion control function blocks, BufferMode input parameter is ignored. Only aborted moves are supported for this release.

Error Handling	All blocks have two outputs, which deal with errors that can occur during execution. These outputs are defined as follows:
	• **Error** – The rising edge of "Error" informs that an error occurred during the execution of the function block, where the function block cannot successfully complete.
	• **ErrorID** – Error number.
	• **Types of errors**:
	  – Function block logic (such as parameters out of range, state machine violation attempted)
	  – Hard limits or soft limits reached
	  – Drive failure (Drive Ready is false)
	For more information about function block error, see Motion Function Block and Axis Status ErrorID on page 180.

Simultaneous Execution of Two Movement Function Blocks (Busy Output = True)

The general rule is that when a movement function block is busy, then a function block **with the same instance (**for example, MC_MoveRelative2) cannot be executed again until the function block status is not busy.

> MC_MoveRelative and MC_MoveAbsolute are busy until the final position is reached. MC_MoveVelocity, MC_Halt, and MC_Stop are busy until the final velocity is reached.
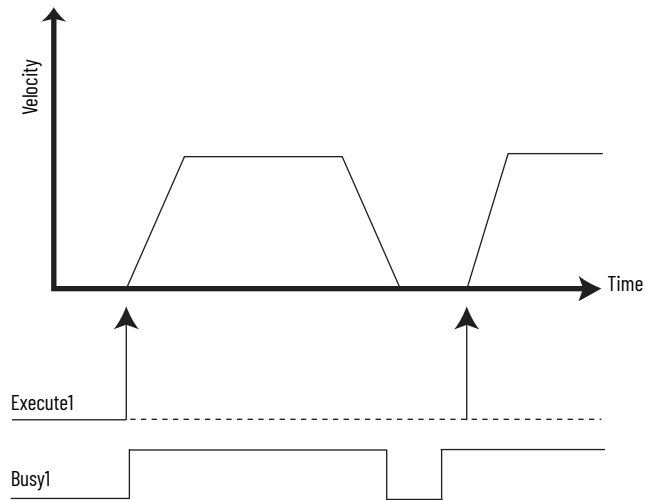
When a movement function block is busy, a function block **with another instance** (for example, MC_MoveRelative1 and MC_MoveAbsolute1 on the same axis) can abort the currently executing function block. This is mostly useful for on-the-fly adjustments to position, velocity, or to halt after a specific distance.

Example: Move to Position Ignored Due to Busy
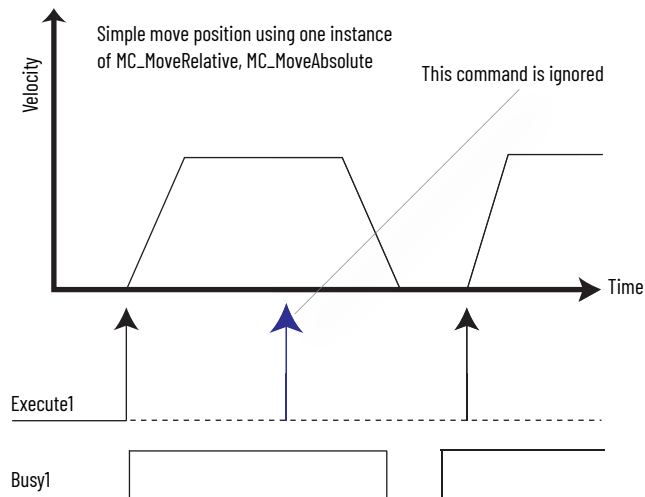
For simple moves, the movement function block finishes. A busy output indicates that the function block is executing and must be allowed to finish before Execute input is toggled again.

If Execute is toggled again before Busy is false, the new command is ignored. No error is generated.

Example: Successful Aborted Move
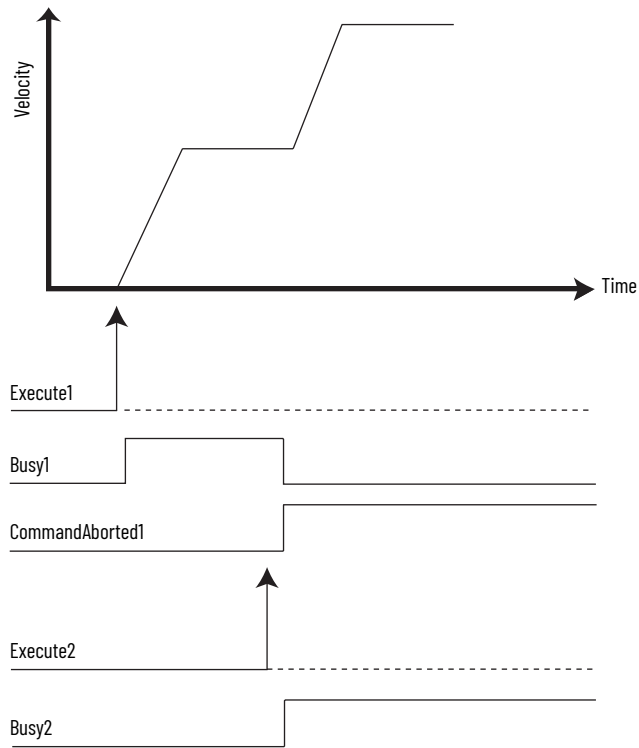
Aborted move is possible if using two instances of MC_MoveRelative, MC_MoveAbsolute. The second instance can immediately abort the first instance (and vice versa) for applications where on-the-fly corrections are needed.

Example: Changing Velocity With No Abort

When changing velocity, generally, an aborted move is not necessary since the function block is only Busy during acceleration (or deceleration). Only one instance of the function block is required.

To bring the axis to a standstill, use MC_Halt.
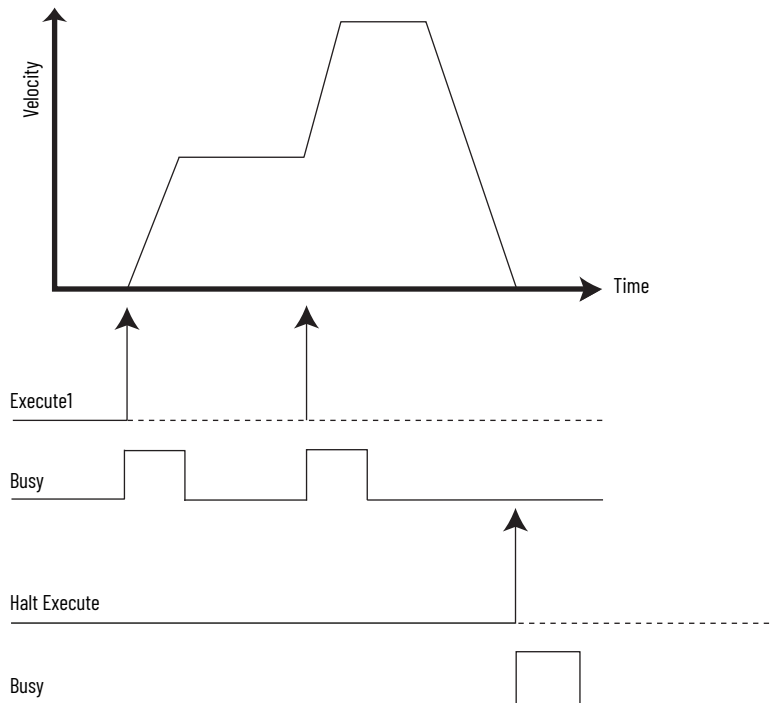
It is possible for the movement function blocks and MC_Halt to abort another motion function block during acceleration/deceleration. This is not recommended as the resulting motion profile may not be consistent.

> ⚠️ **ATTENTION:** If MC_Halt aborts another motion function block during acceleration and the MC_Halt Jerk input parameter is less than the Jerk of the currently executing function block, the Jerk of the currently executing function block is used to prevent an excessively long deceleration.

Example: Aborted Movement Function Block During Acceleration/Deceleration
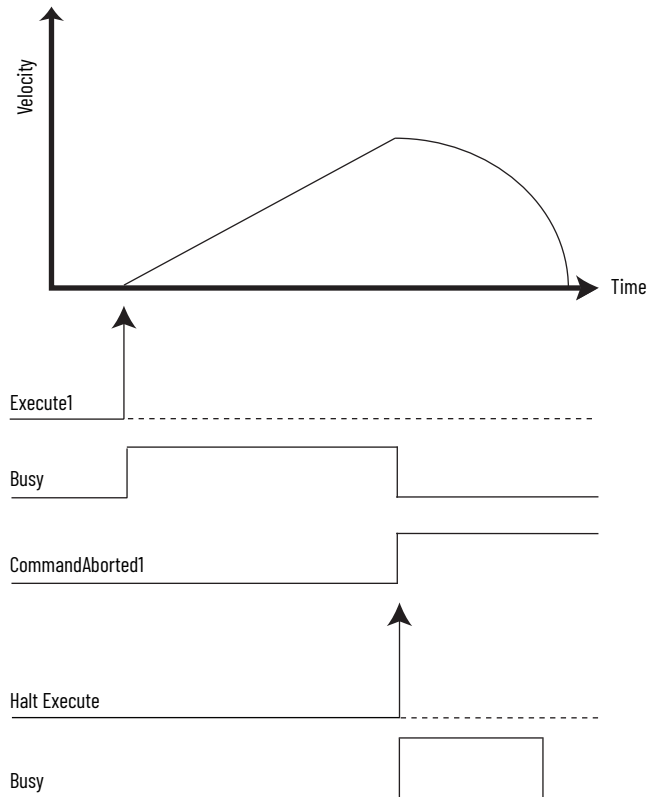
> **IMPORTANT**   If MC_Halt aborts another movement function block during acceleration and the MC_Halt Jerk input parameter is less than the Jerk of the currently executing FB, the Jerk of the currently executing function block is used to prevent excessively long deceleration.

Example: Error Stop using MC_Stop cannot be Aborted
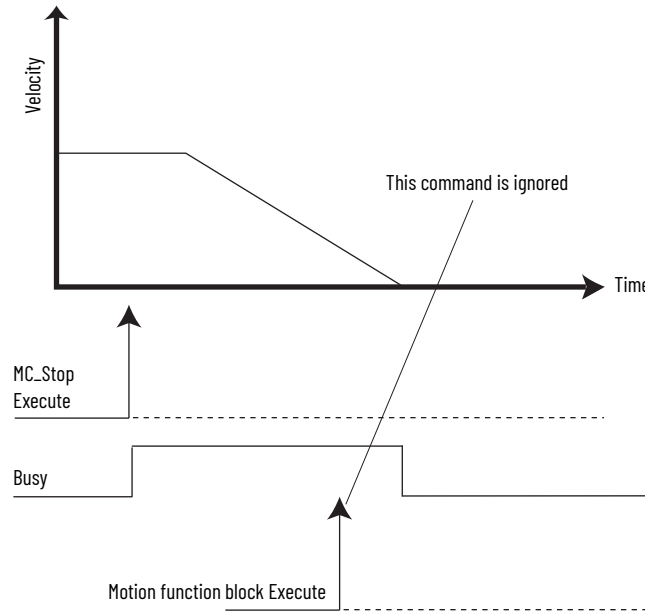
MC_Halt and MC_Stop are both used to bring an axis to a StandStill but MC_Stop is used when an abnormal situation occurs.

MC_Stop can abort other motion function blocks but can never be aborted itself.

MC_Stop goes to the Stopping state and normal operation cannot resume.

Motion Axis and Parameters

The following state diagram illustrates the behavior of the axis at a high level when multiple motion control function blocks are activated. The basic rule is that motion commands are always taken sequentially, even if the controller can perform real parallel processing. These commands act on the axis' state diagram.

The axis is always in one of the defined states, see . Any motion command is a transition that changes the state of the axis and, as a consequence, modifies the way that the current motion is computed.

Figure 47 - Motion Axis State Diagram
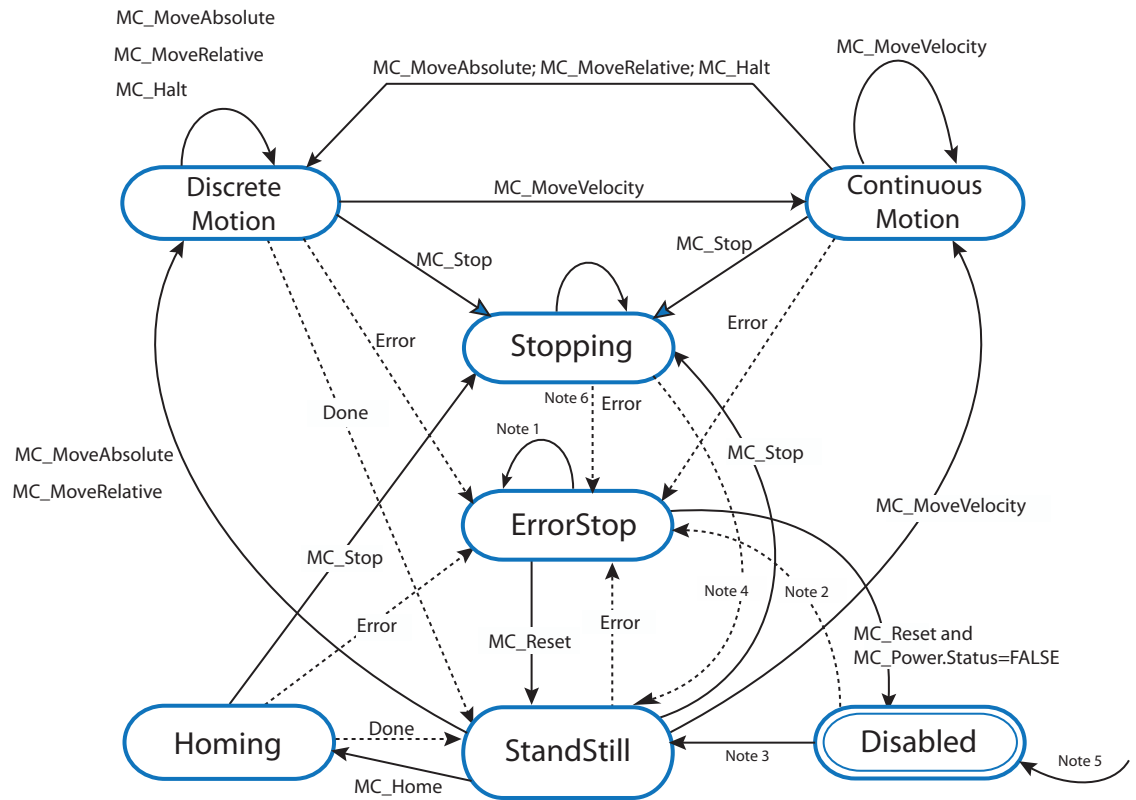
NOTES:

1. In the ErrorStop and Stopping states, all function blocks (except MC_Reset), can be called although they are not executed.

   MC_Reset generates a transition to the StandStill state. If an error occurs while the state machine is in the Stopping state, a transition to the ErrorStop state is generated.

   Axis position still updates even if the Axis state is ErrorStop. Also, the MC_TouchProbe function block is still active if it was executed before the ErrorStop state.

2. Power.Enable = TRUE and there is an error in the Axis.
3. Power.Enable = TRUE and there is no error in the Axis.
4. MC_Stop.Done AND NOT MC_Stop.Execute.
5. When MC_Power is called with Enable = False, the axis goes to the Disabled state for every state including ErrorStop.
6. If an error occurs while the state machine is in the Stopping state, a transition to the ErrorStop state is generated.

Axis States

The axis state can be determined from one of the following predefined states. Axis state can be monitored through the Axis Monitor feature of the Connected Components Workbench software when in debug mode.

Motion States

State Value	State Name
0x00	Disabled
0x01	Standstill
0x02	Discrete Motion
0x03	Continuous Motion
0x04	Homing
0x06	Stopping
0x07	Stop Error

Axis State Update

On motion execution, although the Motion Engine controls the motion profile as a background task, which is independent from the POU scan, the axis state update is still dependent on when the relevant motion function block is called by the POU scan.

For example, on a moving axis on a Ladder POU (state of a rung=true), an MC_MoveRelative function block in the rung is scanned and the axis starts to move. Before MC_MoveRelative completes, the state of the rung becomes False, and MC_MoveRelative is no longer scanned. In this case, the state of this axis cannot switch from Discrete Motion to StandStill, even after the axis fully stops, and the velocity comes to 0.

Limits

The Limits parameter sets a boundary point for the axis, and works with the Stop parameter to define a boundary condition for the axis on the type of stop to apply when certain configured limits are reached.

There are three types of motion position limits.

- Hard Limits
- Soft Limits
- PTO Pulse Limits

See Motion Axis Configuration in Connected Components Workbench on page 182 for information on how to configure limits and stop profiles and the acceptable value range for each.

If any one of these limits is reached on a moving axis (except on homing), an over travel limit error will be reported and the axis will be stopped based on configured behavior.

Figure 48 - Sample Limits configuration in Connected Components Workbench
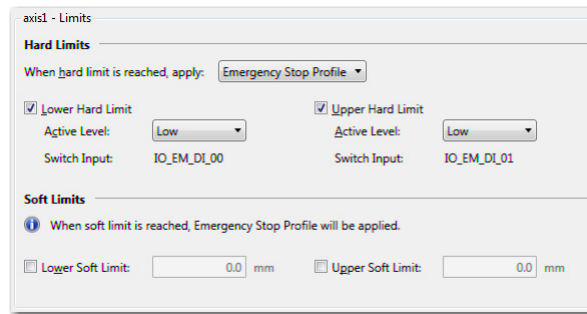
Hard Limits

Hard limits refer to the input signals received from physical hardware devices such as limit switches and proximity sensors. These input signals detect the presence of the load at the maximum upper and minimum lower extents of allowable motion of the load or movable structure that carries the load, such as a load tray on a transfer shuttle.

Hardware limits are mapped to discrete inputs that are associated with data tags/variables.

When a hard limit switch is enabled, the axis comes to a stop when the limit switch is detected during motion. If a hard stop on hard limit switch is configured as ON and the limit is detected, motion is stopped immediately (that is, the hardware immediately stops the PTO pulse). Alternatively, if a hard stop on hard limit switch is configured as OFF, motion is stopped using the Emergency Stop parameters.

When any hard limit switch is enabled, the input variable connecting to this physical input can still be used in User Application.

When a hard limit switch is enabled, it is used automatically for the MC_Home function block, if the switch is in the Homing direction that is configured in the Connected Components Workbench software (Mode: MC_HOME_ABS_SWITCH or MC_HOME_REF_WITH_ABS). See .

Soft Limits

Soft limits refer to data values that the motion controller manages. Unlike hardware limits that detect the presence of the physical load at specific points in the allowable motion of the load, soft limits are based on the stepper commands and the motor and load parameters.

Soft limits are displayed in user-defined units. You can enable individual soft limits. For non-enabled soft limits (whether upper or lower), an infinite value is assumed.

Soft Limits are activated only when the corresponding axis is homed. You can enable or disable soft limits, and configure an upper and lower limit setting through the Connected Components Workbench software.

Table 60 - Soft Limits Checking on the Function Blocks

Function Block	Limits Checking
MC_MoveAbsolute	The target position is checked against the soft limits before motion starts.
MC_MoveRelative	
MC_MoveVelocity	The soft limits are checked dynamically during motion.

When a soft limit is enabled, the axis comes to a stop when the limit is detected during motion. The motion is stopped using emergency stop parameters.

If both hard and soft limits are configured as enabled, for two limits in the same direction (upper or lower), the limits should be configured such that the soft limit is triggered before the hard limit.

PTO Pulse Limits

You cannot configure this limit parameter because it is the physical limitation of the embedded PTO. The limits are set at 0x7FFF0000 and -0x7FFF0000 pulses, for upper and lower limits, respectively.

The controller checks the PTO pulse limits unconditionally — that is, the checking is always ON.

On a non-continuous motion, to prevent a moving axis going to ErrorStop status with Motion PTO Pulse limits detected, you must prevent the current position value from going beyond the PTO Pulse limit.

On a continuous motion (driven by the MC_MoveVelocity function block), when the current position value goes beyond the PTO pulse limit, the PTO pulse current position automatically rolls over to 0 (or the opposite soft limit, if it is activated), and the continuous motion continues.

For a continuous motion, if the axis is homed, and the soft limit in the motion direction is enabled, the soft limit is detected before the PTO pulse limit being detected.

Motion Stop

There are three types of stops that can be configured for an axis.

Immediate Hardware Stop

The hardware controls this type of Immediate Stop. If a Hard Stop on a Hard Limit switch is enabled, and the Hard Limit has been reached, the controller cuts off the PTO pulse for the axis immediately. The stop response has no delay (less than 1 µs).
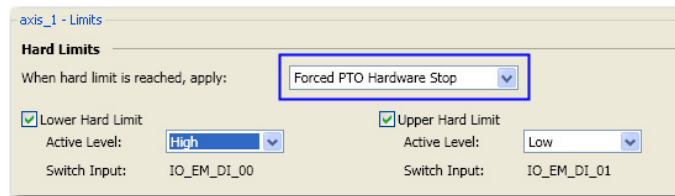
Immediate Soft Stop

The maximum possible response delay for this type of stop could be as much as the Motion Engine Execution time interval. This type of stop is applicable in the following scenarios:

- During motion, when axis PTO Pulse Limit is reached;
- One Hard Limit is enabled for an axis, but a hard stop on hard limit switch is configured as Off. If the Emergency Stop is configured as Immediate Software Stop, during motion, when the Hard Limit switch is detected;
- One Soft Limit is enabled for an axis and the axis has been homed. If the emergency stop is configured as Immediate Soft Stop, during motion, when the Soft Limit reach is detected;
- The Emergency Stop is configured as Immediate Soft Stop. During motion, the MC_Stop function block is issued with Deceleration parameter equal to 0.

Decelerating Soft Stop

Decelerating soft stop could be delayed as much as the Motion Engine Execution Time interval. This type of stop applies in the following scenarios:

- One Hard Limit is enabled for an axis, but the Hard Stop on Hard Limit switch is configured as Off. If the emergency stop is configured as decelerating stop, during motion, when the Hard Limit switch is detected;

- One Soft Limit is enabled for an axis and the axis has been homed. If the emergency stop is configured as decelerating stop, during motion, when the soft limit reach is detected by firmware;

- The Emergency Stop is configured as Decelerating Stop. During motion, the MC_Stop function block is issued with the deceleration parameter set to 0.

- During motion, the MC_Stop function block is issued with the deceleration parameter not set to 0.

Motion Direction

For distance (position) motion, with the target position defined (absolute or relative), the direction input is ignored.

For velocity motion, the direction input value can be positive (1), current (0) or negative (-1). For any other value, only the sign (whether positive or negative) is considered and defines whether the direction is positive or negative. This means that if the product of velocity and direction is -3, then the direction type is negative.

Table 61 - MC_MoveVelocity Supported Direction Types

Direction Type	Value Used[1]	Direction Description
Positive direction	1	Specific for motion/rotation direction. Also called clockwise direction for rotation motion.
Current direction	0	Current direction instructs the axis to continue its motion with new input parameters, without direction change. The direction type is valid only when the axis is moving and the MC_MoveVelocity is called.
Negative direction	-1	Specific for motion/rotation direction. Also referred to as counter-clockwise direction for rotation motion.

(1)   Data type: short integer.

Axis Elements and Data Types

Axis_Ref Data Type

Axis_Ref is a data structure that contains information on a motion axis. It is used as an input and output variable in all motion function blocks. One axis_ref instance is created automatically in the Connected Components Workbench software when you add one motion axis to the configuration.

You can monitor this variable in controller debug mode through the software when the motion engine is active, or as part of your application logic. It can also be monitored remotely through various communication channels.

Table 62 – Data Elements for Axis_Ref

Element Name	Data Type	Description
Axis_ID	UINT8	The logic axis ID automatically assigned by the Connected Components Workbench software. You cannot edit or view this parameter.
ErrorFlag	UINT8	Indicates whether an error is present in the axis.
AxisHomed	UINT8	Indicates whether the homing operation is successfully executed for the axis or not. When you try to redo homing for an axis with AxisHomed already set (homing is performed successfully), and the result is not successful, the AxisHomed status is cleared.
ConsVelFlag	UINT8	Indicates whether the axis is in constant velocity movement or not. Stationary axis is not considered to be in constant velocity.
AccFlag	UINT8	Indicates whether the axis is in an accelerating movement or not.
DecFlag	UINT8	Indicates whether the axis is in a decelerating movement or not.
AxisState	UINT8	Indicates the current state of the axis. For more information, see Axis States on page 173.
ErrorID	UINT16	Indicates the cause for axis error when error is indicated by ErrorFlag. This error usually results from motion function block execution failure. See Motion Function Block and Axis Status ErrorID on page 180.
ExtraData	UINT16	Reserved.
TargetPos	REAL (float)[1]	Indicates the final target position of the axis for the MoveAbsolute and MoveRelative function blocks. For the MoveVelocity, Stop, and Halt function blocks, TargetPos is 0 except when the TargetPos set by the previous position function blocks is not cleared.
CommandPos	REAL (float)[1]	On a moving axis, this is the current position the controller commands the axis to go to.
TargetVel	REAL (float)[1]	The maximum target velocity that is issued to the axis by a move function block. The value of TargetVel is the same as the velocity setting in the current function block, or smaller, depending on other parameters in the same function block. This element is a signed value indicating direction information. See PTO Pulse Accuracy on page 190 for more information.
CommandVel	REAL (float)[1]	During motion, this element refers to the velocity the controller commands the axis to use. This element is a signed value indicating direction information.

(1) See Real Data Resolution on page 188 for more information on REAL data conversion and rounding.

IMPORTANT	• Once an axis is flagged with error, and the ErrorID is not zero, you must reset the axis (with MC_Reset) before issuing any other movement function block. • The update for axis status is performed at the end of one program scan cycle, and the update is aligned with the update of Motion Axis status.

Axis Error Scenarios

In most cases, when a movement function block instruction issued to an axis results in a function block error, the axis is also flagged as being in the Error state. The corresponding ErrorID element is set on the axis_ref data for the axis. However, there are exception scenarios where an axis error is not flagged. The exception can be, but not limited to, the following scenarios:

• A movement function block instructs an axis, but the axis is in a state where the function block could not be executed properly. For example, the axis has no power, or is in the homing sequence, or in the Error Stop state.

• A movement function block instructs an axis, but another movement function block still controls the axis. The axis cannot allow the new function block to control the motion without going to a full stop. For example, the new function block commands the axis to change motion direction.

- When one movement function block tries to control an axis, but another movement function block still controls the axis, and the controller cannot achieve the newly defined motion profile. For example, your application issues an S-curve MC_MoveAbsolute function block to an axis with too short a distance that is given when the axis is moving.

- When one movement function block is issued to an axis, and the axis is in the Stopping or Error Stopping sequence.

For the above exceptions, it is still possible for your application to issue a successful movement function block to the axis after the axis state changes.

MC_Engine_Diag Data Type

The MC_Engine_Diag data type contains diagnostic information on the embedded motion engine. It can be monitored in debug mode through the Connected Components Workbench software when the motion engine is active, or through the user application as part of user logic. It can also be monitored remotely through various communication channels.

One MC_Engine_Diag instance is created automatically in the Connected Components Workbench software when you add the first motion axis in the motion configuration. All user-configured motion axes share this instance.

Table 63 - Data Elements for MC_Engine_Diag

Element Name	Data Type
MCEngState	UINT16
CurrScantime[1]	UINT16
MaxScantime[1]	UINT16
CurrEngineInterval[1]	UINT16
MaxEngineInterval[1]	UINT16
ExtraData	UINT16

(1)   The time unit for this element is microsecond. This diagnostic information can be used to optimize motion configuration and user application logic adjustment.

Table 64 - MCEngstate States

State Name	State	Description
MCEng_Idle	0x01	MC engine exists (at least one axis defined), but the engine is idle as there is no axis is moving. The Engine diagnostic data is not being updated.
MCEng_Running	0x02	MC engine exists (at least one axis defined) and the engine is running. The diagnostic data is being updated.
MCEng_Faulted	0x03	MC engine exists, but the engine is faulted.

Function Block and Axis Status Error Codes

All motion control function blocks share the ErrorID definition.

Axis error and function block error share the ErrorID, but error descriptions are different, as described in .

> Error code 128 is warning information to indicate that the motion profile has been changed and velocity has been adjusted to a lower value but the function block can execute successfully.

Table 65 - Motion Function Block and Axis Status ErrorID

ErrorID	ErrorID MACRO	Error Description for Function Block	Error Description for Axis Status[1]
00	MC_FB_ERR_NO	Function block execution is successful.	The axis is in operational state.
01	MC_FB_ERR_WRONG_STATE	The function block cannot execute because the axis is not in the correct state. Check the axis state.	The axis is not operational due to an incorrect axis state detected during a function block execution. Reset the state of the axis using the MC_Reset function block.
02	MC_FB_ERR_RANGE	The function block cannot execute because there are invalid axis dynamic parameters (velocity, acceleration, deceleration, or jerk) set in the function block. Correct the setting for the dynamic parameters in the function block against the Axis Dynamics configuration page.	The axis is not operational due to invalid axis dynamic parameters (velocity, acceleration, deceleration, or jerk) set in a function block. Reset the state of the axis using the MC_Reset function block. Correct the setting for the dynamic parameters in the function block against the Axis Dynamics configuration page.
03	MC_FB_ERR_PARAM	The function block cannot execute because there are invalid parameters other than velocity, acceleration, deceleration, or jerk, set in the function block. Correct the setting for the parameters (for example, mode or position) for the function block.	The axis is not operational due to invalid parameters other than velocity, acceleration, deceleration, or jerk, set in a function block. Reset the state of the axis using the MC_Reset function block. Correct the setting for the parameters (for example, mode or position) for the function block.
04	MC_FB_ERR_AXISNUM	The function block cannot execute because the axis does not exist, the axis configuration data is corrupted, or the axis is not correctly configured.	Motion internal Fault, ErrorID = 0x04. Call Tech support.
05	MC_FB_ERR_MECHAN	The function block cannot execute because the axis is faulty due to drive or mechanical issues. Check the connection between the drive and the controller (Drive Ready and In-Position signals), and verify that the drive is operating normally.	The axis is not operational due to drive or mechanical issues. Check the connection between the drive and the controller (Drive Ready and In-Position signals), and verify that the drive is operating normally. Reset the state of the axis using the MC_Reset function block.
06	MC_FB_ERR_NOPOWER	The function block cannot execute because the axis is not powered on. Power on the axis using the MC_Power function block.	The axis is not powered on. Power on the axis using the MC_Power function block. Reset the state of the axis using the MC_Reset function block.
07	MC_FB_ERR_RESOURCE	The function block cannot execute because the resource that is required by the function block is controlled by some other function block or not available. Verify that the resource that is required by the function block is available for use. Some examples: • MC_power function block attempts to control the same axis. • MC_Stop function block is executed against the same axis simultaneously. • Two or more MC_TouchProbe function blocks are executed against the same axis simultaneously.	The axis is not operational due to the resource that is required by a function block is under the control of other function block, or not available. Verify that the resource that is required by the function block is available for use. Reset the state of the axis using the MC_Reset function block.
08	MC_FB_ERR_PROFILE	The function block cannot execute because the motion profile that is defined in the function block cannot be achieved. Correct the profile in the function block.	The axis is not operational due to the motion profile that is defined in a function block cannot be achieved. Reset the state of the axis using the MC_Reset function block. Correct the profile in the function block.
09	MC_FB_ERR_VELOCITY	The function block cannot execute because the motion profile that is requested in the function block cannot be achieved due to current axis velocity. Some examples: • The function block requests the axis to reverse the direction while the axis is moving. • The required motion profile cannot be achieved due to current velocity too low or too high. Check the motion profile setting in the function block, and correct the profile, or re-execute the function block when the axis velocity is compatible with the requested motion profile.	The axis is not operational. The motion profile that is requested in the function block cannot be achieved because of current axis velocity. Some examples: • The function block requests the axis to reverse the direction while the axis is moving. • The required motion profile cannot be achieved due to current velocity too low or too high. Reset the state of the axis using the MC_Reset function block. Correct the motion profile in the function block, or re-execute the function block when the axis velocity is compatible with the requested motion profile.
10	MC_FB_ERR_SOFT_LIMIT	This function block cannot execute as it ends up moving beyond the soft limit, or the function block is aborted as the soft limit has been reached. Check the velocity or target position settings in the function block, or adjust the soft limit setting.	The axis is not operational due to soft limit error detected, or due to expected soft limit error in a function block. Reset the state of the axis using the MC_Reset function block. Check the velocity or target position settings for the function block, or adjust the Soft Limit setting.
11	MC_FB_ERR_HARD_LIMIT	This function block is aborted as the Hard Limit switch active state has been detected during axis movement, or aborted as the Hard Limit switch active state has been detected before axis movement starts. Move the axis away from the hard limit switch in the opposite direction.	The axis is not operational due to hard limit error detected. Reset the state of the axis using the MC_Reset function block, and then move the axis away from the hard limit switch in the opposite direction.

Table 65 - Motion Function Block and Axis Status ErrorID (Continued)

ErrorID	ErrorID MACRO	Error Description for Function Block	Error Description for Axis Status[1]
12	MC_FB_ERR_LOG_LIMIT	This function block cannot execute as it ends up moving beyond the PTO Accumulator logic limit, or the function block is aborted as the PTO Accumulator logic limit has been reached. Check the velocity or target position settings for the function block. Or, use the MC_SetPosition function block to adjust the axis coordinate system.	The axis is not operational due to PTO Accumulator logic limit error detected, or due to expected PTO accumulator logic limit error in a function block. Reset the state of the axis using the MC_Reset function block. Check the velocity or target position settings for the function block. Or, use the MC_SetPosition function block to adjust the axis coordinate system.
13	MC_FB_ERR_ENGINE	A motion engine execution error is detected during the execution of this function block. Cycle power to the entire motion setup, including controller, drives and actuators, and then download the User Application again. If the fault is persistent, call Tech support.	The axis is not operational due to a motion engine execution error. Cycle power to the entire motion setup, including controller, drives and actuators, and then download the User Application again. If the fault is persistent, contact your local Rockwell Automation technical support representative. For contact information, see rok.auto/support.
16	MC_FB_ERR_NOT_HOMED	The Function Block cannot execute because the axis must be homed first. Execute homing against the axis using the MC_Home Function Block.	The axis is not operational because the axis is not homed. Reset the state of the axis using the MC_Reset Function Block.
128	MC_FB_PARAM_MODIFIED	**Warning**: The requested motion parameter for the axis has been adjusted. The function block executes successfully.	Motion internal Fault, ErrorID = 0x80. Contact your local Rockwell Automation technical support representative. For contact information, see rok.auto/support.

(1)   You can view axis status through the Axis Monitor feature of the Connected Components Workbench software.

When a motion control function block ends with an error, and the axis is in ErrorStop state, in most cases, the MC_Reset function block (or MC_Power Off/On and MC_Reset) can be used to recover the axis. With this, the axis can get back to normal motion operation without stopping the controller operation.

Major Fault Handling

In case the controller encounters issues where recovery is not possible through the Stop, Reset, or Power function blocks, controller operation is stopped and a major fault is reported.

Table 66 defines the motion-related major fault codes for Micro830, Micro850, and Micro870 controllers.

Table 66 - Major Fault Error Codes and Description

Major Fault Value	Fault ID MACRO	Major Fault Description
0xF100	EP_MC_CONFIG_GEN_ERR	There is general configuration error, which is detected in the motion configuration that is downloaded from Connected Components Workbench software, such as Num of Axis, or the Motion execution interval being configured out of range. When this major fault is reported, there could be no axis in the ErrorStop state.
0xF110	EP_MC_RESOURCE_MISSING	Motion configuration has mismatch issues with the motion resource that is downloaded to the controller. There are some motion resources missing. When this major fault is reported, there could be no axis in the ErrorStop state.
0xF12x	EP_MC_CONFIG_AXS_ERR	This catalog cannot support the motion configuration for the axis, or the configuration has some resource conflict with some other motion axis, which has been configured earlier. The possible reason could be maximum velocity, max acceleration is configured out of supported range. x = The logic Axis ID (0...3)
0xF15x	EP_MC_ENGINE_ERR	There is a motion engine logic error (firmware logic issue or memory crash) for one axis that is detected during motion engine cyclic operation. One possible reason can be motion engine data/memory crash. (This is a motion engine operation error, and should not happen in normal condition.) x = The logic Axis ID (0...3)

Motion Axis Configuration in Connected Components Workbench

A maximum of three motion axes can be configured through the Connected Components Workbench software. To add, configure, update, delete, and monitor an axis in the Connected Components Workbench software, see the sections that follow.

💡 Configuration changes must be compiled and downloaded to the controller to take effect.

💡 Values for the different motion axis parameters are validated based on a set of relationships and pre-determined absolute range. See Motion Axis Parameter Validation on page 191 for a description of the relationships between parameters.

Add New Axis

IMPORTANT   Motion Engine Execution Time
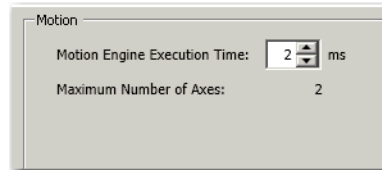
When an axis is added to the configuration, the Motion Engine Execution Time can be configured from 1…10 ms (default: 1 ms). This global parameter applies to all motion axis configurations.

---

1. On the Device Configuration tree, right-click <New Axis> and select Add.
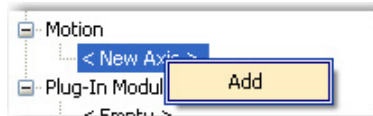
2. Provide an axis name. Select Enter.

    💡 • Name must begin with a letter or underscore character, followed by a letter or single underscore characters.
    • You can also press the F2 key to edit the axis name.

3. Expand the newly created Axis to see the configuration categories.
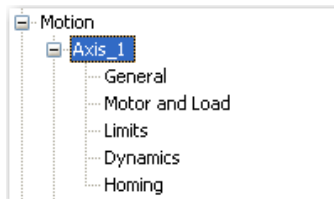
    💡 To help you edit these motion properties, see Edit Axis Configuration on page 182. You can also learn more about axis configuration parameters.

Edit Axis Configuration

Select each axis configuration category to view and edit the propertie:

- General
- Motor and Load
- Limits
- Dynamics
- Homing

General

Edit General parameters. See <u>Table 67 on page 183</u> for a description of the general configuration parameters for a motion axis.

> **IMPORTANT** To edit these general parameters, see <u>Input and Output Signals on page 161</u> for more information about fixed and configurable outputs.

Table 67 - General Parameters

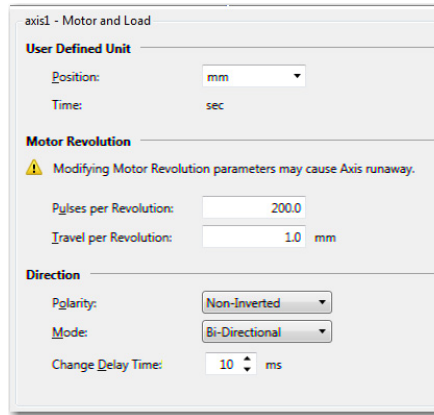| Parameter	Description and Values
Axis Name	User defined. Provides a name for the motion axis.
PTO Channel	Shows the list of available PTO channels.
Pulse Output	Presents the logical variable name of the Direction Output channel based on the PTO channel value that has been assigned.
Direction Output	Presents the logical variable name of the Direction Output channel based on the PTO channel value that has been assigned.
Drive Enable Output	Servo On Output Enable flag. Check the option box to enable.
- Output	The list of available digital output variables that can be assigned as servo/drive output.
- Active Level	Set as High (default) or Low.
In-position Input	Check the option box to enable in-position input monitoring.
- Input	List of digital input variables for in-position input monitoring. Select an input.
- Active Level	Set as High (default) or Low.
Drive Ready Input	Servo Ready Input Enable flag. Check the option box to enable the input.
- Input	The list of digital input variables. Select an input.
- Active Level	Set as High (default) or Low.
Touch Probe Input	Configure whether an input for touch probe is used. Check the option box to enable touch probe input.
- Input	List of digital input variables. Select an input.
- Active Level	Set the active level for touch probe input as High (default) or Low.

PTO Channel Naming

Names of embedded PTO channels have the prefix EM (embedded) and each available PTO channel is enumerated starting from 0. For example, a controller that supports three axes have the following PTO channels available:

- EM_0
- EM_1
- EM_2

Motor and Load

Edit the Motor Load properties as defined in .

IMPORTANT	Certain parameters for Motor and Load are Real values. For more information, see	

Table 68 – Motor and Load Parameters

Parameter	Description and Values
User-defined Unit	Defines user unit scaling that matches your mechanical system values. These units are carried forward into all command and monitor axis in user unit values throughout programming, configuration, and monitoring functions.
Position	Select from any of the following options: – mm – cm – inches – revs – custom unit (ASCII format of up to 7 characters long)
Time	Read-only. Predefined in seconds.
Motor Revolution	Defines pulse per revolution and travel per revolution values.
Pulse per Revolution[1]	Defines the number of pulses that are needed to obtain one revolution of the drive motor. *Range*: 0.0001…8388607 *Default*: 200.0
Travel per Revolution[1]	Travel per revolution defines the distance, either linear or rotational, that the load moves per revolution of the motor. *Range*: 0.0001…8388607. *Default*: 1.0 user unit.
Direction	Defines polarity, mode, and change of delay time values.
Polarity	Direction polarity determines whether the direction signal received by the controller as a discrete input should be interpreted on the input as received by the motion controller (that, is the non-inverted case), or whether the signal should be inverted before interpretation by the motion control logic. Set as Inverted or Non-inverted (default).
Mode	Set as bidirectional (default), Positive (clockwise), or Negative (counter-clockwise) direction.
Change Delay Time	Configure from 0…100 ms. Default value is 10 ms.

(1) The parameter is set as the REAL (float) value in the Connected Components Workbench software. To learn more about conversions and rounding of REAL values, see .

A red border on an input field indicates that an invalid value has been entered. Scroll over the field to see the tooltip message that lets you know the valid value range for the parameter. Supply the valid value.

ATTENTION: Modifying Motor Revolution parameters may cause axis runaway.

Limits

Edit the Limits parameters based .

⚠️ **ATTENTION:** To learn more about the different types of Limits, see .

Table 69 - Limits Parameters

Parameter[1]	Value
Hard Limits	Defines upper and lower hard limits for the axis.
When hard limit is reached, apply	Configure whether to perform a forced PTO hardware stop (immediately turn off pulse output) or whether to decelerate (leave pulse output on and use deceleration values as defined on the Emergency Stop profile). Set as any of the following: • Forced PTO Hardware Stop • Emergency Stop Profile
Lower Hard Limit	Click checkbox to enable a lower hard limit.
Active Level (for Lower Hard Limit)	High or Low.
Upper Hard Limit	Click checkbox to enable.
Active Level (for Upper Hard Limit)	High or Low.
Soft Limits	Defines upper and lower soft limits values.
Lower Soft Limit[2]	Lower soft limit should be less than upper soft limit. 1. Select the checkbox to enable an lower/upper soft limit. 2. Specify a value (in mm).
Upper Soft Limit[2]	

(1) To convert from user units to pulse:

Value in user unit = Value in pulse x $\dfrac{\text{Travel per revolution}}{\text{Pulse per revolution}}$

(2) The parameter is set as the REAL (float) value in the Connected Components Workbench software. To learn more about conversions and rounding of REAL values, see Real Data Resolution on page 188.

💡 A red border on an input field indicates that an invalid value has been entered. Scroll over the field to see the tooltip message that lets you know the valid value range for the parameter. Supply the valid value.

Dynamics

Select Dynamics. The <Axis Name> - Dynamics tab appears. Edit the Dynamics parameters based on the values in Table 70 on page 186.

Table 70 - Dynamics Parameters

Parameter	Values
Start/Stop Velocity[1][2] Start/Stop Velocity in rpm[1][2]	The range is based on Motor and Load parameters (See Motor and Load Parameters on page 184) using: *Range* 1…100,000 pulse/sec *Default*: 300 rpm For example, you can configure the value from 0.005…500 mm/s for 200 pulses per revolution and units of 1 mm per revolution.[3] Rpm value is automatically populated when a value in user units is specified, but you can also initially enter an rpm value. Start/stop velocity should not be greater than maximum velocity.
Max Velocity[1][2]	The range is based on Motor and Load parameters (See Motor and Load Parameters on page 184) using: Range: 1…10,000,000 pulse/sec. Default: 100,000.0 pulse/sec
Max Acceleration[1]	The range is based on Motor and Load parameters (See Motor and Load Parameters on page 184) using: *Range*: 1…10,000,000 pulse/sec^2 *Default*: 10,000,000 pulse/sec^2
Max Deceleration[1]	The range is based on Motor and Load parameters (See Motor and Load Parameters on page 184) using: *Range*: 1…100,000 pulse/sec^2 *Default*: 10,000,000 pulse/sec^2
Max Jerk[1]	The range is based on Motor and Load parameters (See Motor and Load Parameters on page 184) using: *Range*: 0…10,000,000 pulse/sec^3 *Default*: 10,000,000 pulse/sec^3
Emergency Stop Profile	Defines stop type, velocity, deceleration, and jerk values.
Stop Type	Set as Deceleration Stop (default) or Immediate Stop.

Table 70 - Dynamics Parameters (Continued)

Parameter	Values
Stop Velocity[1]	The range is based on Motor and Load parameters (See Motor and Load Parameters on page 184) using: *Range*: 1...100,000 pulse/sec *Default*: 300 rpm
Stop Deceleration[1]	The range is based on Motor and Load parameters (See Motor and Load Parameters on page 184) using: *Range*: 1...10,000,000 pulse/sec *Default*: 300.0 rpm^2
Stop Jerk[1]	The range is based on Motor and Load parameters (See Motor and Load Parameters on page 184) using: *Range*: 0...10,000,000 pulse/sec^3 *Default*: 0.0 rpm^3 (Disabled)

(1)   The parameter is set as the REAL (float) value in the Connected Components Workbench software. To learn more about conversions and rounding of REAL values, see Real Data Resolution on page 188.

(2)   The formula for deriving rpm to user unit, and vice versa:

$$v \text{ (in rpm)} = \frac{v \text{ (in user unit/sec)} \times 60 \text{ s}}{\text{travel per revolution (in user unit)}}$$

(3)   To convert from parameter value from pulse to user units:

$$\text{Value in user unit} = \text{Value in pulse} \times \frac{\text{Travel per revolution}}{\text{Pulse per revolution}}$$

> A red border on an input field indicates that an invalid value has been entered. Scroll over the field to see the tooltip message that lets you know the valid value range for the parameter. Supply the valid value.

Homing

Set the Homing parameters based on the description in Table 71.
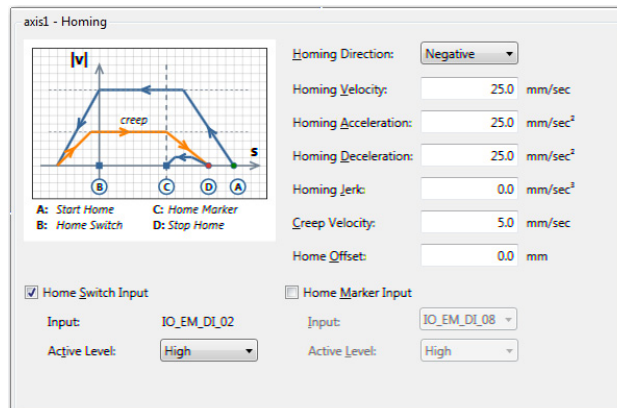
Table 71 - Homing Parameters

Parameter	Value Range
Homing Direction	Positive (clockwise) or negative (counterclockwise).
Homing Velocity[1]	*Range*: 1...100,000 pulse/sec *Default*: 5,000.0 pulse/sec (25.0 mm/sec) **NOTE**: Homing Velocity should not be greater than the maximum velocity.
Homing Acceleration[1]	*Range*: 1...10,000,000 pulse/sec^2 *Default*: 5000.0 pulse/sec^2 (25.0 mm/sec^2) **NOTE**: Homing Acceleration should not be greater than the maximum acceleration.
Homing Deceleration[1]	*Range*: 1...10,000,000 pulse/sec^2 *Default*: 5000.0 pulse/sec^2 (25.0 mm/sec^2) **NOTE**: Homing Deceleration should not be greater than the maximum deceleration.
Homing Jerk[1]	*Range*: 0...10,000,000 pulse/sec^3 *Default*: 0.0 pulse/sec^3 (0.0 mm/sec^3) **NOTE**: Homing Jerk should not be greater than the maximum jerk.

Table 71 - Homing Parameters (Continued)

Parameter	Value Range
Creep Velocity[1]	*Range*: 1...5,000 pulse/sec *Default*: 1000.0 pulse/sec (5.0 mm/sec) **NOTE**: Homing Creep Velocity should not be greater than the maximum velocity.
Homing Offset[1]	*Range*: −1073741824...+1073741824 pulse *Default*: 0.0 pulse (0.0 mm)
Home Switch Input	Enable home switch input by selecting the checkbox.
- Input	Read-only value specifying the input variable for home switch input.
- Active Level	High (default) or Low.
Home Marker Input	Enable the setting of a digital input variable by clicking the checkbox.
- Input	Specify digital input variable for home marker input.
- Active Level	Set the active level for the home switch input as High (default) or Low.

(1) The parameter is set as the REAL (float) value in the Connected Components Workbench software. To learn more about conversions and rounding of REAL values, see <u>Real Data Resolution on page 188</u>.

Axis Start/Stop Velocity

Start/Stop velocity is the initial velocity when an axis starts to move, and the last velocity before the axis stops moving. Generally, Start/Stop velocity is configured at some low value, so that it is smaller than most velocity used in the motion function block.

- When the target velocity is smaller than the Start/Stop velocity, move the axis immediately at the target velocity.
- When the target velocity is NOT smaller than Start/Stop velocity, move the axis immediately at Start/Stop velocity.

Real Data Resolution

Certain data elements and axis properties use REAL data format (single-precision floating point format). Real data has seven-digit resolution and when you enter digit values that are longer than seven digits, the digits are converted. See the examples in <u>Table 72</u>.

Table 72 - REAL Data Conversion Examples

User Value	Converted To
0.12345678	0.1234568
1234.1234567	1234.123
12345678	1.234568E+07 (exponential format)
0.000012345678	1.234568E−05 (exponential format)
2147418166	2.147418+E09
−0.12345678	−0.1234568

If the number of digits is greater than seven and the eighth digit is greater than or equal to 5, then the seventh digit is rounded up.

For example:

21474185 rounded to 2.147419E+07
21474186 rounded to 2.147419E+07

If the eighth digit is less than 5, no rounding is done and the seventh digit remains the same.

For example:

21474181 rounded to 2.147418E+07

Table 73 - Examples for Motion Configuration

Parameter	Actual Value Entered by User	Converted Value in Connected Components Workbench	Tooltip Error Value[1]
Pulses per Revolution	8388608	8388608 (no conversion)	Pulse per Revolution must be in the range of 0.0001…8388607 user unit.
Upper Soft Limit	10730175	1.073018E+7	Upper Soft Limit must be greater than Lower Soft Limit. The range is from 0 (exclusive) to 1.073217E+07 user unit.
Lower Soft Limit	-10730175	-1.073018E+7	Lower Soft Limit must be smaller than Upper Soft Limit. The range is from -1.073217E+07…0 (exclusive) user unit.

(1) On the axis configuration page in the Connected Components Workbench software, an input field with a red border indicates that the value that has been entered is invalid. A tooltip message should let you know the expected range of values for the parameter. The range of values that are presented in the tooltip messages are also presented in REAL data format.
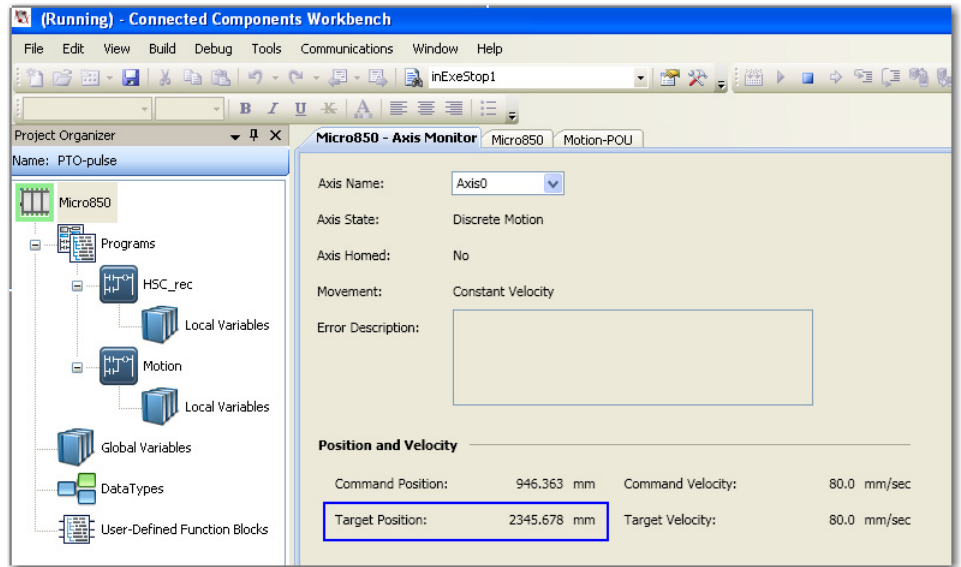
Variable Monitor Example

The Variable Monitor displays six significant digits with rounding, although the real data type still contains seven significant digits.

In this example, you have entered the Target Position value of 2345.678.
This value is rounded up to six digits (2345.68) in the Variable Monitoring screen.

Axis Monitor Example

The Axis Monitor displays seven significant digits with rounding.

> ⚠ **ATTENTION:** See Motion Axis Configuration in Connected Components Workbench on page 182 to learn more about the different axis configuration parameters.

PTO Pulse Accuracy

Micro800 motion feature is pulse-based and the value of distance and velocity are designed in such a way that all PTO-related values are integers at the hardware level, when converting to PTO pulse.

For example, if you configure Motor Pulses per Revolution as 1,000 and Travel per Revolution as 10 cm and you want to drive velocity at 4.504 cm/sec. The target velocity is 4.504 cm/sec (that is, 450.4 pulse/sec). In this case, the actual commanded velocity is 4.5 cm/sec (that is, 450 pulse/sec), and the 0.4 pulse/sec is rounded off.
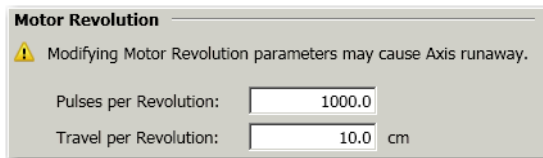
This rounding scheme also applies to other input parameters such as Position, Distance, Acceleration, Deceleration, and Jerk. For instance, with the above motor revolution configuration, setting Jerk as 4.504 cm/sec^3 is the same as setting Jerk as 4.501 cm/sec^3, as both are rounded off to 4.5 cm/sec^3. This rounding applies to both axis configuration input in the Connected Components Workbench software and function block input.

Motion Axis Parameter Validation

Besides falling within the predetermined absolute range, motion axis parameters are validated based on relationships with other parameters. These relationships or rules are listed as follows. Error is flagged whenever there is violation to these relationships.

- Lower Soft Limit should be less than the Upper Soft Limit.
- Start/Stop velocity should not be greater than the maximum velocity.
- Emergency Stop velocity should not be greater than the maximum velocity.
- Homing velocity should not be greater than the maximum velocity.
- Homing acceleration should not be greater than the maximum acceleration.
- Homing deceleration should not be greater than the maximum deceleration.
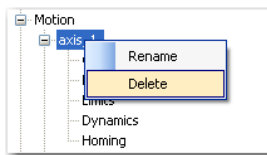- Homing jerk should not be greater than the maximum jerk.
- Homing creep velocity should not be greater than the maximum velocity.

Delete an Axis

1. On the device configuration tree, and under Motion, right-click the axis name and select Delete.

2. A message box appears asking to confirm deletion. Select Yes.

Monitor an Axis

To monitor an axis, the Connected Components Workbench software should be connected to the controller and in DEBUG mode.

1. On the device configuration page, click Axis Monitor.
2. The Axis Monitor window appears with the following characteristics available for viewing:
   - Axis state
   - Axis homed
   - Movement
   - Error description
   - Command position in user unit
   - Command velocity in user unit per second
   - Target position in user unit
   - Target velocity in user unit per second

Homing Function Block

The homing function block MC_Home commands the axis to perform the "search home" sequence. The Position input is used to set the absolute position when the reference signal is detected, and the configured home offset is reached. This function block completes at StandStill if the homing sequence is successful.

MC_Home can be aborted only by the function blocks MC_Stop or MC_Power. Any abort attempt from other moving function blocks result in function block failure with ErrorID = MC_FB_ERR_STATE. However, the homing operation is not interrupted, and can be executed as usual.

If MC_Home is aborted before it completes, the previously searched home position is considered as invalid, and the axis Homed status is cleared.

After axis power-on is done, the axis Homed status is reset to 0 (not homed). On most scenarios, the MC_Home function block must be executed to calibrate the axis position against the axis home that is configured after MC_Power (On) is done.

The five homing modes that Micro830, Micro850, and Micro870 controllers support are described in Table 74.

Table 74 - Homing Modes

Homing Mode Value	Homing Mode Name	Description
0x00	MC_HOME_ABS_SWITCH	Homing process searches for Home Absolute switch.
0x01	MC_HOME_LIMIT_SWITCH	Homing process searches for limit switch.
0x02	MC_HOME_REF_WITH_ABS	Homing process searches for Home Absolute switch plus using encoder reference pulse.
0x03	MC_HOME_REF_PULSE	Homing process searches for limit switch plus using encoder reference pulse.
0x04	MC_HOME_DIRECT	Static homing process with direct forcing a home position from user reference. The function block sets the current position that the mechanism is in as the home position, with its position determined by the input parameter, "Position".

> **IMPORTANT**   If the axis is powered On with only one direction that is enabled, the MC_Home function block (in modes 0, 1, 2, 3) generates an error and only the MC_Home function block (mode 4) can be executed. See the MC_Power function block for more details.

Conditions for Successful Homing

For homing operation to be successful, all configured switches (or sensors) must be properly positioned and wired. The correct position order from the most negative position to the most positive position—that is, from the leftmost to the rightmost in the homing setup diagrams in this section—for the switches are:

1. Lower Limit switch
2. ABS Home switch
3. Upper Limit switch

During the MC_Home function block execution, the home position is reset, and the soft limits mechanical position is recalculated. During the homing sequence, the motion configuration for the soft limits is ignored.

The homing motion sequence that is discussed in this section has the following configuration assumptions:

1. Homing direction is configured as negative direction;
2. The Lower Limit switch is configured as enabled and wired;

The different homing modes, as defined in Table 74, can have different, but still similar motion sequence. The concept that is discussed below applies to various homing configurations.

MC_HOME_ABS_SWITCH

> **IMPORTANT**   If home switch is not configured as enabled, MC_HOME_ABS_SWITCH (0) homing fails with MC_FB_ERR_PARAM.

MC_HOME_ABS_SWITCH (0) homing procedure performs a homing operation against the home switch. The actual motion sequence is dependent on the home switch, limit switch configuration, and the actual status for the switches before homing starts—that is, when the MC_Home function block is issued.

Scenario 1: Moving part at right (positive) side of home switch before Homing starts

The homing motion sequence for this scenario is as follows:

1. Moving part moves to the left side (negative direction);
2. When the home switch is detected, the moving part decelerates to stop;
3. Moving part moves back (positive direction) in creep velocity to detect the home switch On > Off edge;
4. Once the home switch On > Off is detected, record the position as mechanical home position, and decelerate to stop;
5. Move to the configured home position. The mechanical home position that is recorded during the moving back sequence, plus the home offset configured for the axis in the Connected Components Workbench software.

Scenario 2: Moving part is in between Lower Limit and Home switch before Homing starts

The homing motion sequence for this scenario is as follows:

1. Moving part moves to its left side (negative direction);
2. When the lower limit switch is detected, the moving part decelerates to stop, or stop immediately, according to the limit switch hard stop configuration;
3. Moving part moves back (in positive direction) in creep velocity to detect the home switch On > Off edge;
4. Once the home switch On > Off edge is detected, record the position as the mechanical home position, and decelerate to stop;
5. Move to the configured home position. The mechanical home position that is recorded during the moving back sequence, plus the home offset configured for the axis in the Connected Components Workbench software.

> If the Lower Limit switch is not configured, or not wired, the homing motion fails, and moves continuously to the left until the drive or moving part fails to move.

Scenario 3: Moving part on Lower Limit or Home switch before Homing starts

The homing motion sequence for this scenario is as follows:

1. Moving part moves to its right side (in positive direction) in creep velocity to detect the home switch On > Off edge;
2. Once the home switch On > Off edge is detected, record the position as the mechanical home position, and decelerate to stop;
3. Move to the configured home position. The mechanical home position that is recorded during moving right sequence, plus the home offset configured for the axis in the Connected Components Workbench software.

Scenario 4: Moving part at left (negative) side of Lower Limit switch before Homing starts

In this case, the homing motion fails and moves continuously to the left until the drive or moving part fails to move. You must make sure that the moving part is at the proper location before homing starts.

MC_HOME_LIMIT_SWITCH

IMPORTANT	If the Lower Limit switch is not configured as Enabled, MC_HOME_LIMIT_SWITCH (1) homing fails (ErrorID: MC_FB_ERR_PARAM).

For homing against the Lower Limit switch, one positive home offset can be configured; for homing against the Upper Limit switch, one negative home offset can be configured.

MC_HOME_LIMIT_SWITCH (1) Homing procedure performs a homing operation against the Limit switch. The actual motion sequence is dependent on the limit switch configuration and the actual status for the switch before homing starts — that is, when the MC_Home function block is issued.

Scenario 1: Moving part at right (positive) side of Lower Limit switch before Homing starts

The homing motion sequence for this scenario is as follows:
1. Moving part moves to its left side (in negative direction);
2. When the Lower Limit switch is detected, the moving part decelerates to stop, or stops immediately, according to the Limit Switch Hard Stop configuration;
3. Moving part moves back (in positive direction) in creep velocity to detect the Lower Limit switch On > Off edge;
4. Once the Lower Limit switch On > Off edge is detected, record the position as the mechanical home position, and decelerate to stop;
5. Move to the configured home position. The mechanical home position that is recorded during the moving back sequence, plus the home offset configured for the axis through the Connected Components Workbench software.

Scenario 2: Moving part on Lower Limit switch before Homing starts

The homing motion sequence for this scenario is as follows:
1. Moving part moves to its right side (in positive direction) in creep velocity to detect the Lower Limit switch On > Off edge;
2. Once the Lower Limit switch On > Off edge is detected, record the position as the mechanical home position, and decelerate to stop;
3. Move to the configured home position. The mechanical home position that is recorded during moving right sequence, plus the home offset configured for the axis through the software.

Scenario 3: Moving part at left (negative) side of Lower Limit switch before Homing starts

In this case, the homing motion fails and moves continuously to the left until the drive or moving part fails to move. You must make sure that the moving part is at the proper location before homing starts.

MC_HOME_REF_WITH_ABS

IMPORTANT	If the Home switch or Ref Pulse is not configured as Enabled, MC_HOME_REF_WITH_ABS (2) homing fails with ErrorID: MC_FB_ERR_PARAM.

MC_HOME_REF_WITH_ABS (2) homing procedure performs a homing operation against the Home switch, plus fine Ref Pulse signal. The actual motion sequence is dependent on the home switch, limit switch configuration, and the actual status for the switches before homing starts — that is, when the MC_Home function block is issued.

Scenario 1: Moving part at right (positive) side of Home switch before Homing starts

The homing motion sequence for this scenario is as follows:
1. Moving part moves to its left side (in negative direction);
2. When the Home Abs switch is detected, the moving part decelerates to stop;
3. Moving part moves back (in positive direction) in creep velocity to detect the Home Abs On > Off edge;

4. Once the Home Abs switch On > Off is detected, start to detect the first Ref Pulse signal coming in;

5. Once the first Ref Pulse signal comes, record the position as mechanical home position, and decelerate to stop;

6. Move to the configured home position. The mechanical home position that is recorded during the moving back sequence, plus the home offset configured for the axis through the Connected Components Workbench software.

Scenario 2: Moving part between Lower Limit and Home switch before Homing starts

The homing motion sequence for this scenario is as follows:

1. Moving part moves to its left side (in negative direction);

2. When the Lower Limit switch is detected, the moving part decelerates to stop, or stops immediately, according to the Limit Switch Hard Stop configuration;

3. Moving part moves back (in positive direction) in creep velocity to detect the Home switch On > Off edge;

4. Once the Home Abs switch On > Off is detected, start to detect first Ref Pulse signal;

5. Once the first Ref Pulse signal comes, record the position as mechanical home position, and decelerate to stop.

6. Move to the configured home position. The mechanical home position that is recorded during the moving back sequence, plus the home offset configured for the axis through the Connected Components Workbench software.

IMPORTANT	In this case, if the Lower Limit switch is not configured, or not wired, the homing motion fails and moves continuously to the left until the drive or moving part fails to move.

Scenario 3: Moving part on Lower Limit or Home switch before Homing starts

The homing motion sequence for this scenario is as follows:

1. Moving part moves to its right side (in positive direction) in creep velocity to detect the Home switch On > Off edge;

2. Once the Home Abs switch On > Off is detected, start to detect the first Ref Pulse signal;

3. Once the first Ref Pulse signal comes, record the position as mechanical home position, and decelerate to stop;

4. Move to the configured home position. The mechanical home position that is recorded during moving right sequence, plus the home offset configured for the axis in the Connected Components Workbench software.

Scenario 4: Moving part at left (negative) side of Lower Limit switch before Homing starts

In this case, the homing motion fails and moves continuously to the left until the drive or moving part fails to move. You must make sure that the moving part is at the proper location before homing starts.

MC_HOME_REF_PULSE

IMPORTANT	If the Lower Limit switch or Ref Pulse is not configured as Enabled, MC_HOME_REF_PULSE (3) homing fails (ErrorID: MC_FB_ERR_PARAM).

For homing against the Lower Limit switch, one positive home offset can be configured; for homing against the Upper Limit switch, one negative home offset can be configured.

MC_HOME_REF_PULSE (3) homing procedure performs a homing operation against the Limit switch, plus the fine Ref Pulse signal. The actual motion sequence is dependent on the limit switch configuration, and the actual status for the switches before homing starts — that is, when the MC_Home function block is issued.

Scenario 1: Moving part at right (positive) side of Lower Limit switch before Homing starts

The homing motion sequence for this scenario is as follows:

1. Moving part moves to its left side (in negative direction);
2. When the Lower Limit switch is detected, the moving part decelerates to stop, or stops immediately, according to the Limit Switch Hard Stop configuration;
3. Moving part moves back (in positive direction) in creep velocity to detect the Lower Limit switch On > Off edge;
4. Once the Lower Limit switch On > Off edge is detected, start to detect the first Ref Pulse signal;
5. Once the first Ref Pulse signal comes, record the position as the mechanical home position, and decelerate to stop;
6. Move to the configured home position. The mechanical home position that is recorded during the moving back sequence, plus the home offset configured for the axis through the Connected Components Workbench software.

Scenario 2: Moving part on Lower Limit switch before Homing starts

The homing motion sequence for this scenario is as follows:

1. Moving part moves to its right side (in Positive direction) in creep velocity to detect the Lower Limit switch On > Off edge;
2. Once the Lower Limit switch On > Off edge is detected, start to detect the first Ref Pulse signal;
3. Once the first Ref Pulse signal comes, record the position as the mechanical home position, and decelerate to stop;
4. Move to the configured home position. The mechanical home position that is recorded during the moving back sequence, plus the home offset configured for the axis through the Connected Components Workbench software.

Scenario 3: Moving part at left (negative) side of Lower Limit switch before Homing starts

In this case, the homing motion fails and moves continuously to the left until the drive or moving part fails to move. You must make sure that the moving part is at the proper location before homing starts.

MC_HOME_DIRECT

MC_HOME_DIRECT (4) homing procedure performs a static homing by directly forcing an actual position. No physical motion is performed in this mode. This is equivalent to a MC_SetPosition action, except that Axis Homed status is on once MC_Home (mode = 4) is performed successfully.

Use PTO for PWM Control

The example in this section shows you how to use a PTO axis as a PWM. Create the ladder program in Connected Components Workbench software, as shown in Figure 49, Figure 50, and Figure 51.
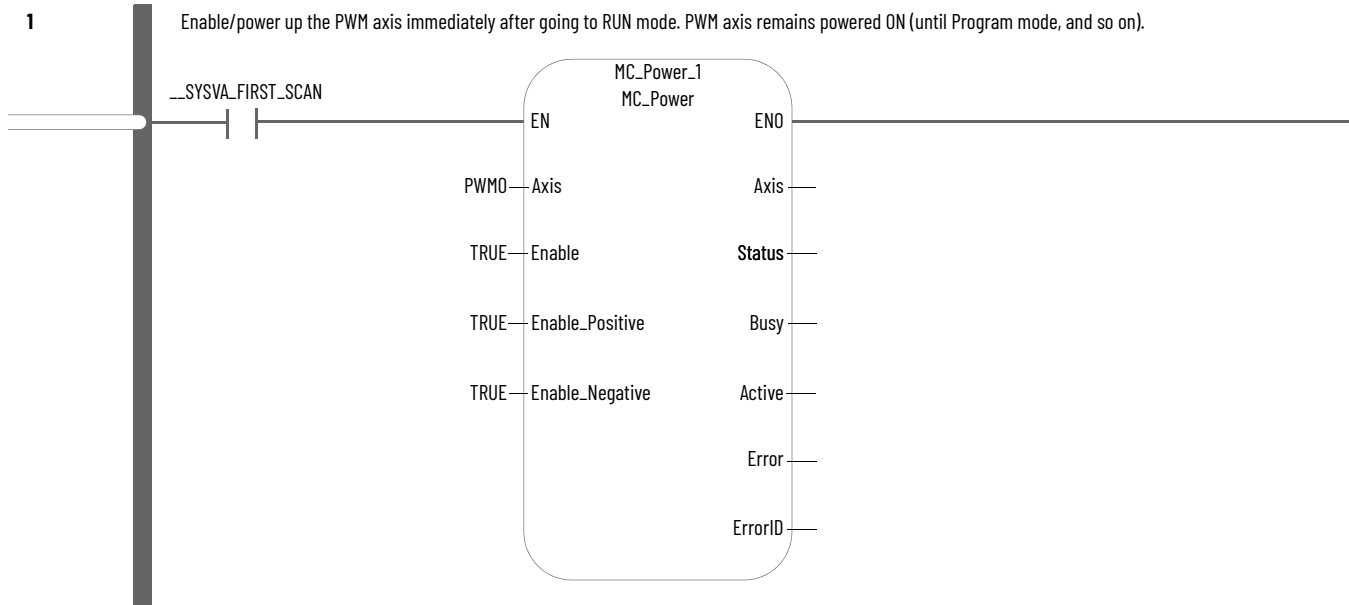
Figure 49 - Example: PTO Axis as a PWM – Rung 1

1   Enable/power up the PWM axis immediately after going to RUN mode. PWM axis remains powered ON (until Program mode, and so on).

```
                __SYSVA_FIRST_SCAN                           MC_Power_1
                                                             MC_Power
                ───┤ ├───                        EN                        ENO

                                          PWM0 ─── Axis                    Axis ───

                                          TRUE ─── Enable                Status ───

                                          TRUE ─── Enable_Positive        Busy ───

                                          TRUE ─── Enable_Negative      Active ───

                                                                         Error ───

                                                                       ErrorID ───
```

Figure 50 - Example: PTO Axis as a PWM – Rung 2

2   Continually use the MC_WriteParameter (Parameter 1005) to change the Duty Cycle from the global variable G_PWM_Duty_Cycle (example: 0.5 => 50%)

```
                        Update_PWM_Duty_Cycle        Update_PWM_Duty_Cycle
                        ───┤/├───                         ──( )──

                                                             MC_WriteParameter_1
                                                             MC_WriteParameter
                                                  EN                        ENO

                                          PWM0 ─── AxisIn                   Axis ───

                         Update_PWM_Duty_Cycle ─── Execute                  Done ───

                                          1005 ─── Parameter_Number         Busy ───

                             G_PWM_Duty_Cycle ─── Value                    Error ───

                                             0 ─── MC_ExecutionMode       ErrorID ───
```
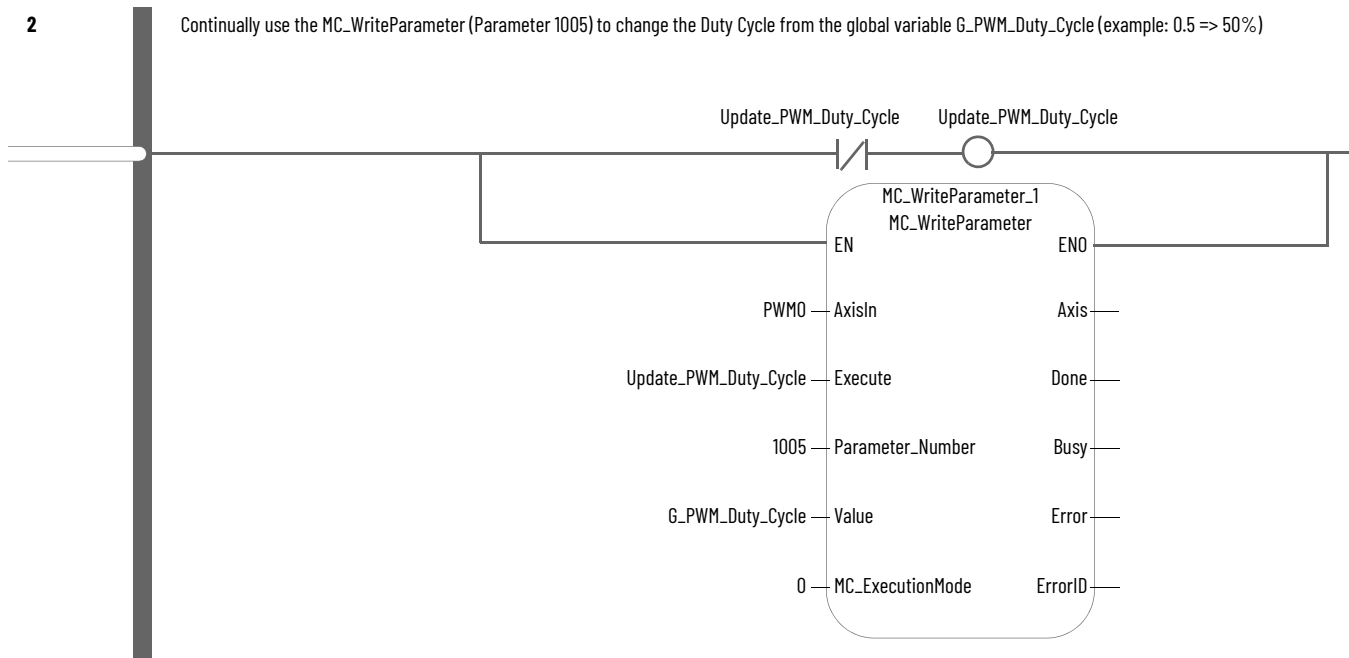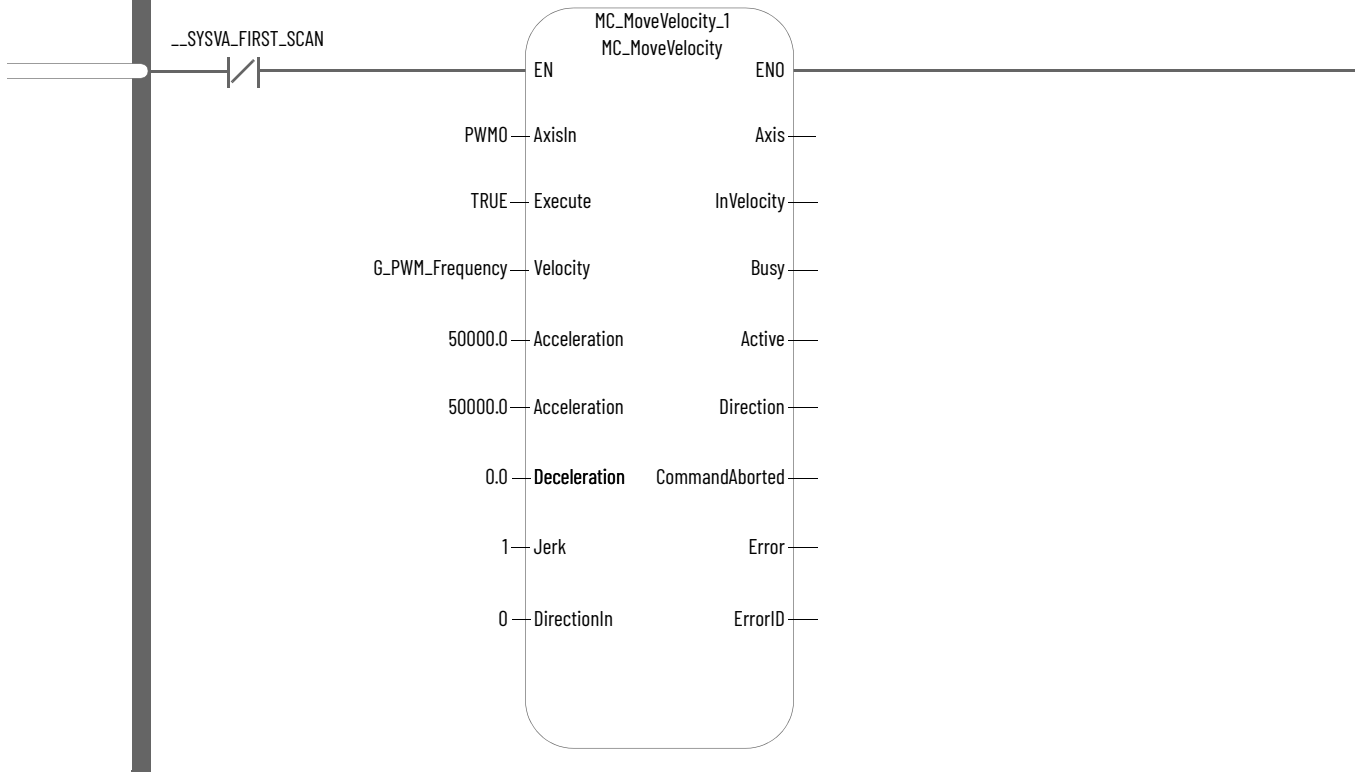
Figure 51 - Example: PTO Axis as a PWM – Rung 3

3

After the first scan, use the MC_MoveVelocity to set the PWM frequency (for example: 50,000 => 50 KHz) continually from the global variable G_PWM_Frequency. PWM axis runs forever (until Program Mode, MC_Halt, and so on).

```
__SYSVA_FIRST_SCAN                        MC_MoveVelocity_1
                                          MC_MoveVelocity
      ─┤/├─                          EN                    ENO

                         PWM0 ─── AxisIn              Axis ───

                         TRUE ─── Execute         InVelocity ───

              G_PWM_Frequency ─── Velocity              Busy ───

                      50000.0 ─── Acceleration        Active ───

                      50000.0 ─── Acceleration     Direction ───

                          0.0 ─── Deceleration  CommandAborted ───

                            1 ─── Jerk               Error ───

                            0 ─── DirectionIn       ErrorID ───
```

POU PWM_Program

The POU defines four variables.

Variable MC_Power_1 (* *) Direction: VAR Data Type: MC_Power Attribute: ReadWrite Direct variable (Channel):	**Variable MC_MoveVelocity_1** (* *) Direction: VAR Data Type: MC_MoveVelocity Attribute: ReadWrite Direct variable (Channel):
Variable Update_PWM_Duty_Cycle (* *) Direction: Var Data type: BOOL Attribute: ReadWrite Direct variable (Channel):	**Variable MC_Power_1** (* *) Direction: VAR Data Type: MC_Power Attribute: ReadWrite Direct variable (Channel):

HSC Feedback Axis

From Connected Components Workbench software version 8.0 onwards, support has been added for a High-speed Counter (HSC) Feedback Axis that uses the same instructions as the PTO Motion Axis. UDFBs are still supported. You can use either one but you cannot select both for the same plug-in.

Figure 52 - Example of Selecting Feedback Axis or UDFB with 2080-MOT-HSC Plug-in Module
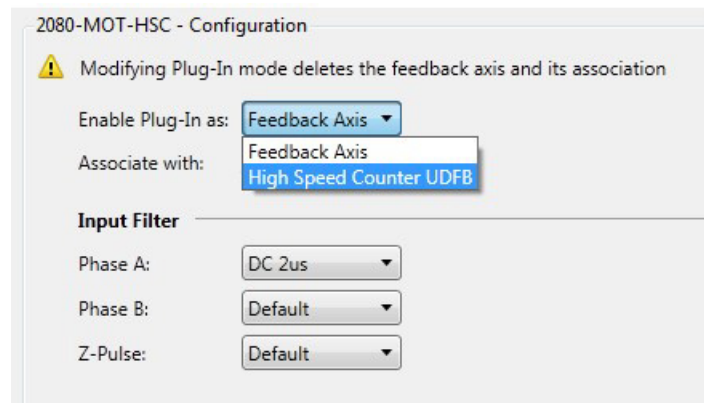
The HSC Feedback Axis provides ease-of-use as you no longer need to program the function blocks, and it uses up less memory on the controller. The HSC Feedback Axis uses only the administrative function blocks from the PTO Motion Axis and they share the same Axis Monitor.

IMPORTANT	The counters are not reset to zero for program download. For example, if using the feedback axis, use the MC_ResetPosition function block to reset the position to zero.

IMPORTANT	If the feedback axis is in the error state because the configured position limits have been exceeded, using the MC_Reset function block to reset the axis may not clear the error as there may still be pulse that is detected from the encoder.

Notes:

Use the High-speed Counter and Programmable Limit Switch

High-Speed Counter Overview

All Micro830, Micro850, and Micro870 controllers, except for 2080-LCxx-AWB, support up to six high-speed counters (HSC). The HSC feature in the Micro800 controllers consists of two main components: the high-speed counter hardware (embedded inputs in the controller), and high-speed counter instructions in the application program. High-speed counter instructions apply configuration to the high-speed counter hardware and updates the accumulator.

> ⚠️ **ATTENTION:** To use the Micro800 HSC feature effectively, you must have a basic understanding of the following:
> - HSC components and data elements
>   The first sections of the chapter provide a detailed description of these components. Quick start instructions (see page 267) are also available to guide you through building a sample HSC project.
> - Programming and working with elements in Connected Components Workbench software
>   You must have a working knowledge of programming through ladder diagram, structured text, or function block diagram to be able to work with the HSC function block and variables.

> ⚠️ **ATTENTION:** Additional information is available on the HSC function block and its elements in the Connected Components Workbench software Online Help that comes with your Connected Components Workbench software installation.

This chapter describes how to use the HSC function and also contains sections on the HSC and HSC_SET_STS function blocks, as follows:

- High-speed Counter (HSC) Data Structures
- High-speed Counter (HSC) Function Block
- HSC_SET_STS Function Block
- Programmable Limit Switch (PLS) Function
- HSC Interrupts

Programmable Limit Switch Overview

The Programmable Limit Switch function allows you to configure the High-Speed Counter to operate as a PLS (Programmable Limit Switch) or rotary cam switch. For more information, see Programmable Limit Switch (PLS) Function on page 219.

What is High-Speed Counter?

High-Speed Counter is used to detect narrow (fast) pulses, and its specialized instructions to initiate other control operations based on counts reaching preset values. These control operations include the automatic and immediate execution of the high-speed counter interrupt routine and the immediate update of outputs based on a source and mask pattern that you set.

The HSC functions are different than most other controller instructions. Their operation is performed by custom circuitry that runs in parallel with the main system processor. This is necessary because of the high-performance requirements of these functions.

Features and Operation

The HSC is versatile; you can select or configure the master HSC for any one of 10 modes and the sub HSC for any one of 5 modes of operation. See HSC Mode (HSCAPP.HSCMode) on page 206 for more information.

Some of the enhanced capabilities of the High-speed Counters are:
- 100 kHz operation
- Direct control of outputs
- 32-bit signed integer data (count range of ± 2,147,483,647)
- Programmable High and Low presets, and Overflow and Underflow setpoints
- Automatic Interrupt processing that is based on accumulated count
- Change parameters on-the-fly (from the user control program)

The High-speed Counter function operates as described in the following diagram.

Figure 53 - High-speed Counter Operation
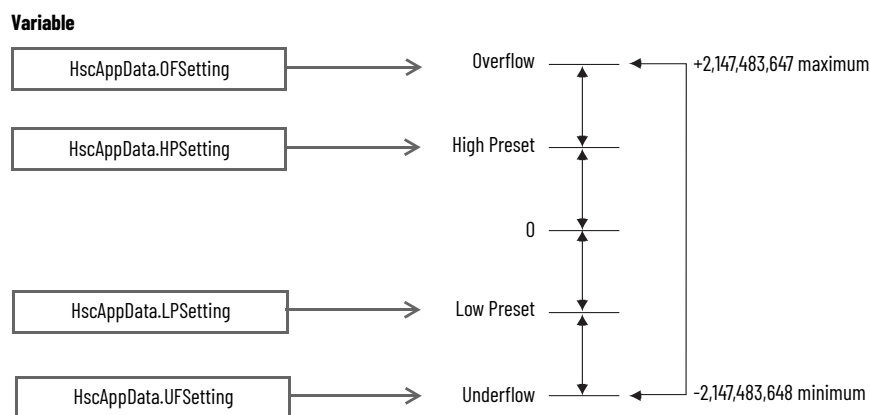
> You must set a proper value for the variables OFSetting, HPSetting, and UFSetting before triggering Start/Run HSC. Otherwise, the controller is faulted. (Setting a value for LPSetting is optional for certain counting modes.)
>
> To learn more about HscAppData variable input, see HSC APP Data Structure on page 204.

When using HSC function blocks, it is recommended that you:
- Set HSCAppData underflow setting (UFSetting) and low preset setting (LPSetting) to a value less than 0 to avoid possible HSC malfunction when the HSC accumulator is reset to 0.
- Set HSCAppData overflow setting (OFSetting) and high preset setting (HPSetting) to a value greater than 0 to avoid possible HSC malfunction when the HSC accumulator is reset to 0.

In some cases, a master counter mode can disable a sub counter. For more information, see HSC Mode (HSCAPP.HSCMode) on page 206.

> HSC0 is used in this document to define how any HSC works.

> **IMPORTANT**   The HSC function can only be used with the controller's embedded I/O. It cannot be used with expansion I/O modules.

HSC Inputs and Wiring Mapping

All Micro830, Micro850, and Micro870 controllers, except 2080-LCxx-xxAWB, have 100 kHz high-speed counters. Each main high-speed counter has four dedicated inputs and each sub high-speed counter has two dedicated inputs.

Table 75 - Micro830, Micro850, and Micro870 High Speed Counters

	10/16-point	24-point	48-point
Number of HSC	2	4	6
Main high-speed counters	1 (counter 0)	2 (counters 0 and 2)	3 (counters 0, 2, and 4)
Sub high-speed counters	1 (counter 1)	2 (counters 1 and 3)	3 (counters 1, 3, and 5)

High-speed Counter	Inputs Used
HSC0	0, 1, 2, 3
HSC1	2, 3
HSC2	4, 5, 6, 7
HSC3	6, 7
HSC4	8, 9, 10, 11
HSC5	10, 11

HSC0's sub counter is HSC1, HSC2's sub counter is HSC3, and HSC4's sub counter is HSC5. Each set of counters shares the input. Table 76 shows the dedicated inputs for the HSCs depending on the mode.

Table 76 - HSC Input Wiring Mapping

	Embedded Input											
	0	01	02	03	04	05	06	07	08	09	10	11
HSC0	A/C	B/D	Reset	Hold								
HSC1			A/C	B/D								
HSC2					A/C	B/D	Reset	Hold				
HSC3							A/C	B/D				
HSC4									A/C	B/D	Reset	Hold
HSC5											A/C	B/D

The following tables show the input wiring mapping for the different Micro830, Micro850, and Micro870 controllers.

Table 77 - Micro830 10-point and 16-point Controller HSC Input Wiring Mapping

Modes of Operation	Input 0 (HSC0) Input 2 (HSC1)	Input 1 (HSC0) Input 3 (HSC1)	Input 2 (HSC0)	Input 3 (HSC0)	Mode Value in User Program (HSCAppData.HSCMode)
Counter with Internal Direction (mode 1a)	Count Up	Not Used			0
Counter with Internal Direction, External Reset and Hold (mode 1b)	Count Up	Not Used	Reset	Hold	1
Counter with External Direction (mode 2a)	Count Up/Down	Direction	Not Used		2
Counter with External Direction, Reset, and Hold (mode 2b)	Count	Direction	Reset	Hold	3
Two Input Counter (mode 3a)	Count Up	Count Down	Not Used		4
Two Input Counter with External Reset and Hold (mode 3b)	Count Up	Count Down	Reset	Hold	5
Quadrature Counter (mode 4a)	A Type input	B Type input	Not Used		6
Quadrature Counter with External Reset and Hold (mode 4b)	A Type input	B Type input	Z Type Reset	Hold	7
Quadrature X4 Counter (mode 5a)	A Type input	B Type input	Not Used		8
Quadrature X4 Counter with External Reset and Hold	A Type input	B Type input	Z Type Reset	Hold	9

Table 78 - Micro830/Micro850/Micro870 24-point Controller HSC Input Wiring Mapping

Modes of Operation	Input 0 (HSC0) Input 2 (HSC1) Input 4 (HSC2) Input 6 (HSC3)	Input 1 (HSC0) Input 3 (HSC1) Input 5 (HSC2) Input 7 (HSC3)	Input 2 (HSC0) Input 6 (HSC2)	Input 3 (HSC0) Input 7 (HSC2)	Mode Value in User Program
Counter with Internal Direction (mode 1a)	Count Up	Not Used			0
Counter with Internal Direction, External Reset and Hold (mode 1b)	Count Up	Not Used	Reset	Hold	1
Counter with External Direction (mode 2a)	Count Up/Down	Direction	Not Used		2
Counter with External Direction, Reset, and Hold (mode 2b)	Count Up/Down	Direction	Reset	Hold	3
Two Input Counter (mode 3a)	Count Up	Count Down	Not Used		4
Two Input Counter with External Reset and Hold (mode 3b)	Count Up	Count Down	Reset	Hold	5
Quadrature Counter (mode 4a)	A Type input	B Type input	Not Used		6
Quadrature Counter with External Reset and Hold (mode 4b)	A Type input	B Type input	Z Type Reset	Hold	7
Quadrature X4 Counter (mode 5a)	A Type input	B Type input	Not Used		8
Quadrature X4 Counter with External Reset and Hold	A Type input	B Type input	Z Type Reset	Hold	9

Table 79 - Micro830/Micro850 48-point Controller HSC Input Wiring Mapping

Modes of Operation	Input 0 (HSC0) Input 2 (HSC1) Input 4 (HSC2) Input 6 (HSC3) Input 8 (HSC4) Input 10 (HSC5)	Input 1 (HSC0) Input 3 (HSC1) Input 5 (HSC2) Input 7 (HSC3) Input 9 (HSC4) Input 11 (HSC5)	Input 2 (HSC0) Input 6 (HSC2) Input 10 (HSC4)	Input 3 (HSC0) Input 7 (HSC2) Input 11 (HSC4)	Mode Value in User Program
Counter with Internal Direction (mode 1a)	Count Up	Not Used			0
Counter with Internal Direction, External Reset and Hold (mode 1b)	Count Up	Not Used	Reset	Hold	1
Counter with External Direction (mode 2a)	Count Up/Down	Direction	Not Used		2
Counter with External Direction, Reset, and Hold (mode 2b)	Count Up/Down	Direction	Reset	Hold	3
Two Input Counter (mode 3a)	Count Up	Count Down	Not Used		4
Two Input Counter with External Reset and Hold (mode 3b)	Count Up	Count Down	Reset	Hold	5
Quadrature Counter (mode 4a)	A Type input	B Type input	Not Used		6
Quadrature Counter with External Reset and Hold (mode 4b)	A Type input	B Type input	Z Type Reset	Hold	7
Quadrature X4 Counter (mode 5a)	A Type input	B Type input	Not Used		8
Quadrature X4 Counter with External Reset and Hold	A Type input	B Type input	Z Type Reset	Hold	9

High-speed Counter (HSC) Data Structures

The following section describes HSC data structures.

HSC APP Data Structure

Define an HSC App Data (configuration data, data type HSCAPP) when programming an HSC. During HSC counting, you should not change the data, except if you must reload the configuration.

To reload HSC configuration, change the HSC APP Data, then call HSC function block with command 0x03 (set/reload). Otherwise, the change to HSC App Data during HSC counting is ignored.

HSC1, HSC3, and HSC5 support mode 0, 2, 4, 6, and 8 only, and HSC0, HSC2, and HSC4 support all counting modes.

PLS Enable (HSCAPP.PLSEnable)

Description	Data Format	User Program Access
PLSEnable	bit	read/write

This bit enables and disables the HSC Programmable Limit Switch (PLS) function.

When the PLS function is enabled, the following settings are superseded by corresponding data values from PLS data.

- HSCAPP.HpSetting
- HSCAPP.LpSetting
- HSCAPP.HPOutput
- HSCAPP.LPOutput

See for more information.

HSCID (HSCAPP.HSCID)

Description	Data Format	User Program Access
HSCID	Word (UINT)	read/write

lists the definition for HSCID.

Table 80 - HSCID Definition

Bits	Description
15...13	HSC Module Type: 0x00: Embedded 0x01: Expansion (not yet implemented) 0x02: Plug-in module
12...8	Module Slot ID: 0x00: Embedded 0x01...0x1F: Expansion (not yet implemented) 0x01...0x05: Plug-in module
7...0	Module internal HSC ID: 0x00-0x0F: Embedded 0x00-0x07: Expansion (not yet implemented) 0x00-0x07: Plug-in module

For Embedded HSC, the valid HSCID value is only 0...5.

HSC Mode (HSCAPP.HSCMode)

Description	Data Format	User Program Access
HSC Mode	word (UINT)	read/write

The HSCMode variable sets the high-speed counter to one of 10 types of operation. This integer value is configured through the programming device and is accessible in the control program.

Table 81 - HSC Operating Modes

Mode Number	Type
0	Up Counter – The accumulator is immediately cleared (0) when it reaches the high preset. A low preset cannot be defined in this mode.
1	Up Counter with external reset and hold – The accumulator is immediately cleared (0) when it reaches the high preset. A low preset cannot be defined in this mode.
2	Counter with external direction
3	Counter with external direction, reset, and hold.
4	Two input counter (up and down)
5	Two input counter (up and down) with external reset and hold
6	Quadrature counter (phased inputs A and B)
7	Quadrature counter (phased inputs A and B) with external reset and hold
8	Quadrature X4 counter (phased inputs A and B)
9	Quadrature X4 counter (phased inputs A and B) with external reset and hold

The main high-speed counters support 10 types of operation mode and the sub high-speed counters support 5 types (mode 0, 2, 4, 6, and 8). If the main high-speed counter is set to mode 1, 3, 5, 7, or 9, then the resub high-speed counter is disabled.

For more information on HSC Function Operating Modes and Input Assignments, see .

HSC Mode 0 – Up Counter

Table 82 - HSC Mode 0 Examples

Input Terminals	Embedded Input 0				Embedded Input 1				Embedded Input 2				Embedded Input 3				CE Bit	Comments
Function	Count				Not Used				Not Used				Not Used					
Example 1	⇑																on (1)	HSC Accumulator + 1 count
Example 2	⇑	on (1)	⇓	off (0)													off (0)	Hold accumulator value

Blank cells = do not care, ⇑= rising edge, ⇓ = falling edge

💡 Inputs 0…11 are available for use as inputs to other functions regardless of the HSC being used.

HSC Mode 1 – Up Counter with External Reset and Hold

Table 83 - HSC Mode 1 Examples

Input Terminals	Embedded Input 0				Embedded Input 1				Embedded Input 2				Embedded Input 3				CE Bit	Comments
Function	Count				Not Used				Reset				Hold					
Example 1	⇑									on (1)	⇓	off (0)				off (0)	on (1)	HSC Accumulator + 1 count
Example 2										on (1)	⇓	off (0)		on (1)				Hold accumulator value
Example3										on (1)	⇓	off (0)				off (0)		Hold accumulator value
Example 4		on (1)	⇓	off (0)						on (1)	⇓	off (0)						Hold accumulator value
Example 5									⇑									Clear accumulator (=0)

Blank cells = do not care, ⇑= rising edge, ⇓ = falling edge

💡 Inputs 0…11 are available for use as inputs to other functions regardless of the HSC being used.

HSC Mode 2 – Counter with External Direction

Table 84 - HSC Mode 2 Examples

Input Terminals	Embedded Input 0			Embedded Input 1			Embedded Input 2			Embedded Input 3			CE Bit	Comments
Function	Count			Direction			Not Used			Not Used				
Example 1	⇑					off (0)							on (1)	HSC Accumulator + 1 count
Example 2	⇑			on (1)									on (1)	HSC Accumulator - 1 count
Example3													off (0)	Hold accumulator value

Blank cells = do not care, ⇑ = rising edge, ⇓ = falling edge

Inputs 0…11 are available for use as inputs to other functions regardless of the HSC being used.

HSC Mode 3 – Counter with External Direction, Reset, and Hold

Table 85 - HSC Mode 3 Examples

Input Terminals	Embedded Input 0			Embedded Input 1			Embedded Input 2			Embedded Input 3			CE Bit	Comments
Function	Count			Direction			Reset			Hold				
Example 1	⇑					off (0)	on (1)	⇓	off (0)			off (0)	on (1)	HSC Accumulator + 1 count
Example 2	⇑			on (1)			on (1)	⇓	off (0)			off (0)	on (1)	HSC Accumulator - 1 count
Example3							on (1)	⇓	off (0)	on (1)				Hold accumulator value
Example 4							on (1)	⇓	off (0)				off (0)	Hold accumulator value
Example 5	on (1)	⇓	off (0)				on (1)	⇓	off (0)					Hold accumulator value
Example 6							⇑							Clear accumulator (=0)

Blank cells = do not care, ⇑ = rising edge, ⇓ = falling edge

Inputs 0…11 are available for use as inputs to other functions regardless of the HSC being used.

HSC Mode 4 – Two Input Counter (up and down)

Table 86 - HSC Mode 4 Examples

Input Terminals	Embedded Input 0			Embedded Input 1			Embedded Input 2			Embedded Input 3			CE Bit	Comments
Function	Count Up			Count Down			Not Used			Not Used				
Example 1	⇑			on (1)	⇓	off (0)							on (1)	HSC Accumulator + 1 count
Example 2	on (1)	⇓	off (0)	⇑									on (1)	HSC Accumulator - 1 count
Example3													off (0)	Hold accumulator value

Blank cells = do not care, ⇑ = rising edge, ⇓ = falling edge

Inputs 0…11 are available for use as inputs to other functions regardless of the HSC being used.

HSC Mode 5 – Two Input Counter (up and down) with External Reset and Hold

Table 87 - HSC Mode 5 Examples

Input Terminals	Embedded Input 0			Embedded Input 1			Embedded Input 2			Embedded Input 3			CE Bit	Comments
Function	Count			Direction			Reset			Hold				
Example 1	⇑			on (1)	⇓	off (0)	on (1)	⇓	off (0)			off (0)	on (1)	HSC Accumulator + 1 count
Example 2	on (1)	⇓	off (0)	⇑			on (1)	⇓	off (0)			off (0)	on (1)	HSC Accumulator - 1 count
Example3							on (1)	⇓	off (0)	on (1)				Hold accumulator value
Example 4							on (1)	⇓	off (0)				off (0)	Hold accumulator value
Example 5	on (1)	⇓	off (0)				on (1)	⇓	off (0)					Hold accumulator value
Example 6							⇑							Clear accumulator (=0)

Blank cells = do not care, ⇑ = rising edge, ⇓ = falling edge

Inputs 0…11 are available for use as inputs to other functions regardless of the HSC being used.

Using the Quadrature Encoder

The Quadrature Encoder is used for determining direction of rotation and position for rotating, such as a lathe. The Bidirectional Counter counts the rotation of the Quadrature Encoder.

Figure 54 shows a quadrature encoder that is connected to inputs 0, 1, and 2. The phase angle between A and B determines the count direction. If A leads B, the counter increments. If B leads A, the counter decrements.

The counter can be reset using the Z input. The Z outputs from the encoders typically provide one pulse per revolution.

Figure 54 - Quadrature Encoder Connected to Inputs
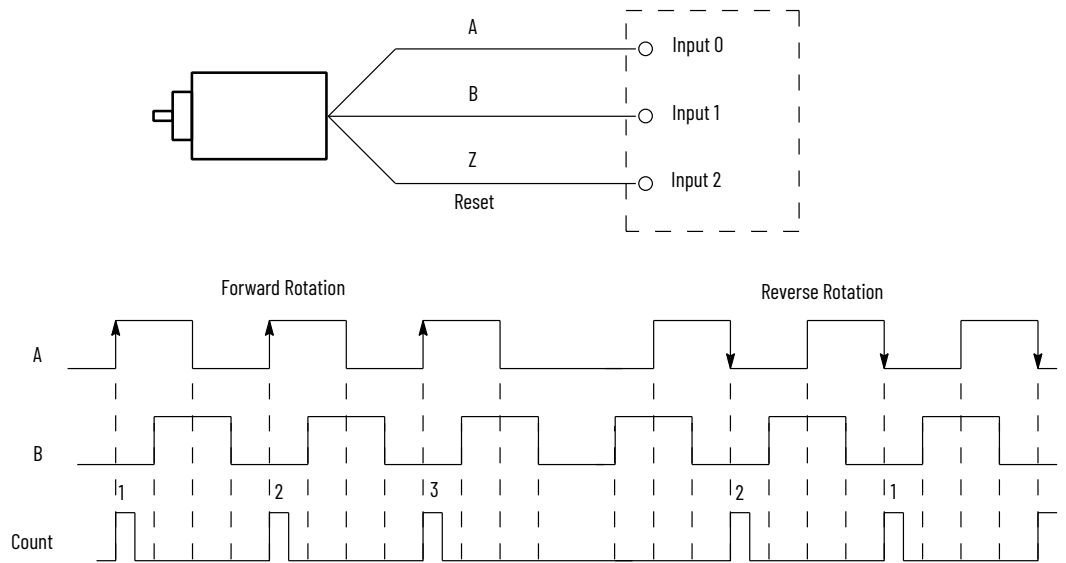
HSC Mode 6 – Quadrature Counter – Phased Inputs A and B

Table 88 - HSC Mode 6 Examples

Input Terminals	Embedded Input 0				Embedded Input 1				Embedded Input 2				Embedded Input 3				CE Bit	Comments
Function	Count A				Count B				Not Used				Not Used					
Example 1[1]	⇑							off (0)									on (1)	HSC Accumulator + 1 count
Example 2[2]			⇓					off (0)									on (1)	HSC Accumulator - 1 count
Example 3				off (0)														Hold accumulator value
Example 4		on (1)																Hold accumulator value
Example 5						on (1)												Hold accumulator value
Example 6																	off (0)	Hold accumulator value

(1)   Count input A leads count input B.
(2)   Count input B leads count input A.

Blank cells = do not care, ⇑ = rising edge, ⇓ = falling edge

Inputs 0…11 are available for use as inputs to other functions regardless of the HSC being used.

HSC Mode 7 – Quadrature Counter – Phased Inputs A and B with External Reset and Hold

Table 89 - HSC Mode 7 Examples

Input Terminals	Embedded Input 0	Embedded Input 1	Embedded Input 2	Embedded Input 3	CE Bit	Comments
Function	Count A	Count B	Z Reset	Hold		
Example 1[1]	⇑	off (0)		off (0)	on (1)	HSC Accumulator + 1 count
Example 2[2]	⇓	off (0)	off (0)	off (0)	on (1)	HSC Accumulator - 1 count
Example3	⇓ off (0)	off (0)	on (1)			Reset accumulator to zero
Example 4	on (1)					Hold accumulator value
Example 5		on (1)				Hold accumulator value
Example 6			off (0) on (1)			Hold accumulator value
Example 7			off (0)		off (0)	Hold accumulator value

(1)   Count input A leads count input B.
(2)   Count input B leads count input A.

Blank cells = do not care, ⇑ = rising edge, ⇓ = falling edge

Inputs 0…11 are available for use as inputs to other functions regardless of the HSC being used.

HSC Mode 8 – Quadrature X4 Counter

Table 90 - HSC Mode 8 Examples

Embedded Input 1 (HSC0) (A)	Embedded Input 1 (HSC0) (B)	Value of CE Bit	Accumulator and Counter Action
⇑	OFF	TRUE	Count Up Acc. Value
⇑	ON	TRUE	Count Down Acc. Value
⇓	OFF	TRUE	Count Down Acc. Value
⇓	ON	TRUE	Count Up Acc. Value
OFF	⇑	TRUE	Count Down Acc. Value
ON	⇑	TRUE	Count Up Acc. Value
OFF	⇓	TRUE	Count Up Acc. Value
ON	⇓	TRUE	Count Down Acc. Value
OFF or ON	OFF or ON	X	Hold Acc. Value
X	X	FALSE	Hold Acc. Value

HSC Mode 9 – Quadrature X4 Counter with External Reset and Hold

Table 91 - HSC Mode 9 Examples

Embedded Input 0 (HSC0) (A)	Embedded Input 1 (HSC0) (B)	Embedded Input 2 (HSC0) (Reset)	Embedded Input 3 (HSC0) (Hold)	Value of CE Bit	Accumulator and Counter Action
⇑	OFF	X	-	TRUE	Count Up Acc. Value
⇑	ON	X	-	TRUE	Count Down Acc. Value
⇓	OFF	X	-	TRUE	Count Down Acc. Value
⇓	ON	X	-	TRUE	Count Up Acc. Value
OFF	⇑	X	-	TRUE	Count Down Acc. Value
ON	⇑	X	-	TRUE	Count Up Acc. Value
OFF	⇓	X	-	TRUE	Count Up Acc. Value
ON	⇓	X	-	TRUE	Count Down Acc. Value
OFF or ON	OFF or ON	OFF	X	X	Hold Acc. Value
OFF	OFF	ON	X	X	Reset Acc. to Zero
X	X	OFF	ON	X	Hold Acc. Value
X	X	OFF	X	FALSE	Hold Acc. Value

Accumulator (HSCAPP. Accumulator)

Description	Data Format	User Program Access
HSCAPP.Accumulator	long word (32-bit INT)	read/write

This parameter is the initial HSC Accumulator value that must be set when starting the HSC. The HSC subsystem updates this parameter automatically when the HSC is in Counting mode to reflect the actual HSC accumulator value.

High Preset (HSCAPP.HPSetting)

Description	Data Format	User Program Access
HSCAPP.HPSetting	long word (32-bit INT)	read/write

The HSCAPP.HPSetting is the upper setpoint (in counts) that defines when the HSC subsystem generates an interrupt.

The data that is loaded into the high preset must be less than to the data resident in the overflow (HSCAPP.OFSetting) parameter or an HSC error is generated.

Low Preset (HSCAPP.LPSetting)

Description	Data Format	User Program Access
HSCAPP.LPSetting	long word (32-bit INT)	read/write

The HSCAPP.LPSetting is the lower setpoint (in counts) that defines when the HSC subsystem generates an interrupt.

The data that is loaded into the low preset must be:
1. Less than or equal to 0 for HSC Mode (HSCAPP.HSCMode) parameter values 0 and 1 or an HSC error is generated.
2. Greater than or equal to the data resident in the underflow (HSCAPP.UFSetting) parameter for all HSC Mode (HSCAPP.HSCMode) or an HSC error is generated.

If the underflow and low preset values are negative numbers, the low preset must be a number with a smaller absolute value.

Overflow Setting (HSCAPP.OFSetting)

Description	Data Format	Type	User Program Access
HSCAPP.OFSetting	long word (32-bit INT)	control	read/write

The HSCAPP.OFSetting defines the upper count limit for the counter. If the counter's accumulated value increments past the value that is specified in this variable, an overflow interrupt is generated. When the overflow interrupt is generated, the HSC subsystem rolls the accumulator over to the underflow value and the counter continues counting from the underflow value (counts are not lost in this transition). You can specify any value for the overflow position, provided it is greater than the underflow value and falls between -2,147,483,648 and 2,147,483,647.

> Data that is loaded into the overflow variable must be greater than the data resident in the high preset (HSCAPP.HPSetting) or an HSC error is generated.

Underflow Setting (HSCAPP.UFSetting)

Description	Data Format	User Program Access
HSCAPP.UFSetting	long word (32-bit INT)	read/write

The HSCAPP.UFSetting defines the lower count limit for the counter. If the counter's accumulated value decrements past the value specified in this variable, an underflow interrupt is generated. When the underflow interrupt is generated, the HSC subsystem resets the accumulated value to the overflow value and the counter then begins counting from the overflow value (counts are not lost in this transition). you can specify any value for the underflow position, provided it is less than the overflow value and falls between -2,147,483,648 and 2,147,483,647.

> Data that is loaded into the underflow variable must be less than or equal to the data resident in the low preset (HSCAPP.LPSetting) or an HSC error is generated.

Output Mask Bits (HSCAPP.OutputMask)

Description	Data Format	User Program Access
HSCAPP.OutputMask	word (32 bit binary)	read/write

The HSCAPP.OutputMask defines which embedded outputs on the controller that the high-speed counter can control directly. The HSC subsystem can directly (without control program interaction) turn outputs ON or OFF based on the HSC accumulator reaching the High or Low presets. The bit pattern that is stored in the HSCAPP.OutputMask variable defines which outputs the HSC controls and which outputs the HSC does not control.

For example, if you want to control outputs 0, 1, 3, using HSC then you must assign, HscAppData.OutputMask = 2#1011 (Hex) or HscAppData.OutputMask = 11 (Decimal)

The bit pattern of the HSCAPP.OutputMask variable directly corresponds to the output bits on the controller. Bits that are set (1) are enabled and can be turned on or off by the HSC subsystem. Bits that are clear (0) cannot be turned on or off by the HSC sub-system. The mask bit pattern can be configured only during initial setup.

Table 92 shows an example of how HPOutput and OutputMask control embedded outputs.

Table 92 - Effect of HSC Output Mask on Embedded Outputs

Output Variable	32-Bit Signed Integer Data Word																				
	32...20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HSCAPP.HPOutput (high preset output)		0	1	0	1	0	1	0	1	0	0	1	1	0	0	0	1	1	0	0	1
HSCAPP.OutputMask (output mask)		1	1	0	0	0	0	0	0	0	1	1	0	0	0	1	1	0	0	1	1
Embedded output (10-point)																					
Embedded output (16-point)																					
Embedded output (24-point)																					
Embedded output (48-point)																					

The outputs that are shown in the black boxes are the outputs under the control of the HSC subsystem. The mask defines which outputs can be controlled. The high preset output or low preset output values (HSCAPP.HPOutput or HSCAPP.LPOutput) define if each output is either ON (1) or OFF (0). Another way to view this is that the high or low preset output is written through the output mask, with the output mask acting like a filter.

The bits in the gray boxes are unused. For the 10-point controller, the first 4 bits of the mask word are used and the remaining mask bits are not functional because they do not correlate to any physical outputs on the base unit. For the 16, 24 and 48-point controllers, the first 6, 10, and 20 bits of the mask word are used, respectively.

The mask bit pattern can be configured only during initial setup.

High Preset Output (HSCAPP.HPOutput)

Description	Data Format	User Program Access
HSCAPP.HPOutput	long word (32 bit binary)	read/write

The High Preset Output defines the state (1 = ON or 0 = OFF) of the outputs on the controller when the high preset is reached. For more information on how to turn outputs on or off directly based on the high preset being reached, see Output Mask Bits (HSCAPP.OutputMask) on page 211.

The high output bit pattern can be configured during initial setup, or while the controller is operating. Use the HSC function block to load the new parameters while the controller is operating.

Low Preset Output (HSCAPP.LPOutput)

Description	Data Format	User Program Access
HSCAPP.LPOutput	long word (32 bit binary)	read/write

The Low Preset Output defines the state (1 = "on", 0 = "off") of the outputs on the controller when the low preset is reached. See Output Mask Bits (HSCAPP.OutputMask) on page 211 for more information on how to directly turn outputs on or off based on the low preset being reached.

The low output bit pattern can be configured during initial setup, or while the controller is operating. Use the HSC function block to load the new parameters while the controller is operating.

HSC STS (HSC Status) Data Structure

Define an HSC STS data (HSC status information data, data type HSCSTS) when programming an HSC.
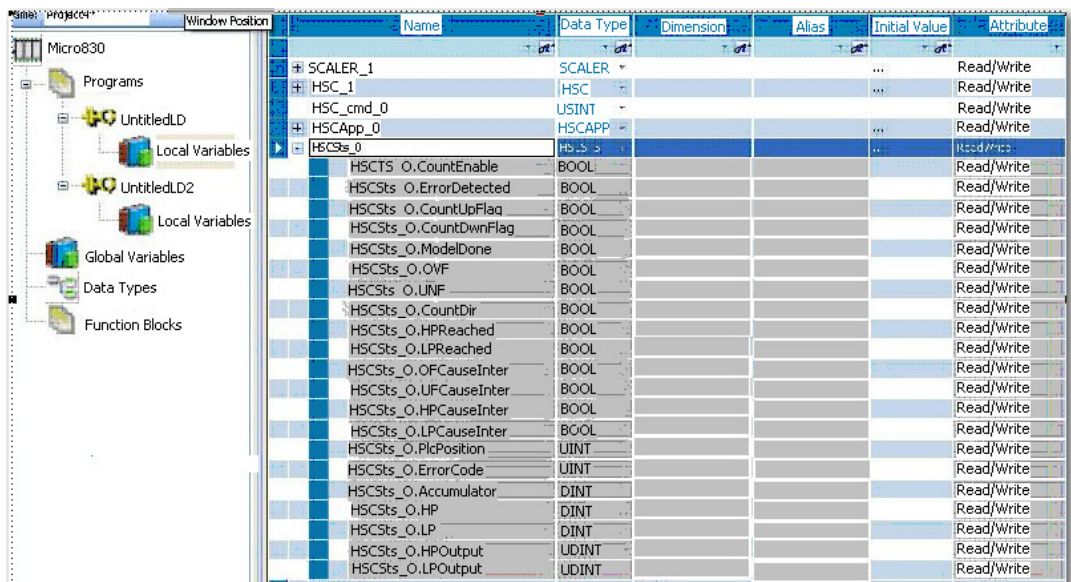
Counting Enabled (HSCSTS.CountEnable)

Description	Data Format	HSC Modes[1]	User Program Access
HSCSTS.CountEnable	bit	0...9	read-only

[1]   For Mode descriptions, see HSC Mode (HSCAPP.HSCMode) on page 206.

The Counting Enabled control bit is used to indicate the status of the High-speed Counter, whether counting is enabled (1) or disabled (0, default).

Error Detected (HSCSTS.ErrorDetected)

Description	Data Format	HSC Modes[1]	User Program Access
HSCSTS.ErrorDetected	bit	0…9	read/write

(1) For Mode descriptions, see HSC Mode (HSCAPP.HSCMode) on page 206.

The Error Detected flag is a status bit that can be used in the control program to detect if an error is present in the HSC subsystem. The most common type of error that this bit represents is a configuration error. When this bit is set (1), you should examine the specific error code in the parameter HSCSTS.ErrorCode. The controller maintains this bit and sets the bit when there is an HSC error. You can clear this bit if required.

Count Up (HSCSTS.CountUpFlag)

Description	Data Format	HSC Modes[1]	User Program Access
HSCSTS.CountUpFlag	bit	0…9	read-only

(1) For Mode descriptions, see HSC Mode (HSCAPP.HSCMode) on page 206.

The Count Up bit is used with all HSCs (modes 0…9). If the HSCSTS.CountEnable bit is set, the Count Up bit is set (1). If the HSCSTS.CountEnable is cleared, the Count Up bit is cleared (0).

Count Down (HSCSTS.CountDownFlag)

Description	Data Format	HSC Modes[1]	User Program Access
SCSTS.CountDownFlag	bit	2…9	read-only

(1) For Mode descriptions, see HSC Mode (HSCAPP.HSCMode) on page 206.

The Count Down bit is used with the bidirectional counters (modes 2…9). If the HSCSTS.CountEnable bit is set, the Count Down bit is set (1). If the HSCSTS.CountEnable bit is clear, the Count Down bit is cleared (0).

Mode Done (HSCSTS.Mode1Done)

Description	Data Format	HSC Modes[1]	User Program Access
HSCSTS.Mode1Done	bit	0 or 1	read/write

(1) For Mode descriptions, see HSC Mode (HSCAPP.HSCMode) on page 206.

The Mode Done status flag is set (1) by the HSC subsystem when the HSC is configured for Mode 0 or Mode 1 behavior, and the accumulator counts up to the High Preset.

Overflow (HSCSTS.OVF)

Description	Data Format	HSC Modes[1]	User Program Access
HSCSTS.OVF	bit	0…9	read/write

(1) For Mode descriptions, see HSC Mode (HSCAPP.HSCMode) on page 206.

The HSCSTS.OVF status flag is set (1) by the HSC subsystem whenever the accumulated value (HSCSTS.Accumulator) has counted through the overflow variable (HSCAPP.OFSetting).

This bit is transitional the HSC subsystem sets this bit. It is up to the control program to use, track if necessary, and clear (0) the overflow condition.

Overflow conditions do not generate a controller fault.

Underflow (HSCSTS.UNF)

Description	Data Format	HSC Modes[1]	User Program Access
HSCSTS.UNF	bit	0…9	read/write

(1) For Mode descriptions, see HSC Mode (HSCAPP.HSCMode) on page 206.

The Underflow status flag is set (1) by the HSC subsystem whenever the accumulated value (HSCSTS.Accumulator) has counted through the underflow variable (HSCAPP.UFSetting).

This bit is transitional and the HSC subsystem sets this bit. It is up to the control program to use, track if necessary, and clear (0) the underflow condition.

Underflow conditions do not generate a controller fault.

Count Direction (HSCSTS.CountDir)

Description	Data Format	HSC Modes[1]	User Program Access
HSCSTS.CountDir	bit	0...9	read-only

(1)   For Mode descriptions, see <u>HSC Mode (HSCAPP.HSCMode) on page 206</u>.

The HSC subsystem controls the Count Direction status flag. When the HSC accumulator counts up, the direction flag is set (1). Whenever the HSC accumulator counts down, the direction flag is cleared (0).

If the accumulated value stops, the direction bit retains its value. The only time the direction flag changes is when the accumulated count reverses.

The HSC subsystem updates this bit continuously whenever the controller is in a run mode.

High Preset Reached (HSCSTS.HPReached)

Description	Data Format	HSC Modes[1]	User Program Access
HSCSTS.HPReached	bit	2...9	read/write

(1)   For Mode descriptions, see <u>Count Down (HSCSTS.CountDownFlag) on page 213</u>.

The High Preset Reached status flag is set (1) by the HSC subsystem whenever the accumulated value (HSCSTS.Accumulator) is greater than or equal to the high preset variable (HSCAPP.HPSetting).

The HSC subsystem updates this bit continuously whenever the controller is in an executing mode. Writing to this element is not recommended.

Low Preset Reached (HSCSTS.LPReached)

Description	Data Format	HSC Modes[1]	User Program Access
HSCSTS.LPReached)	bit	2...9	read only

(1)   For Mode descriptions, see <u>HSC Mode (HSCAPP.HSCMode) on page 206</u>.

The Low Preset Reached status flag is set (1) by the HSC subsystem whenever the accumulated value (HSCSTS.Accumulator is less than or equal to the low preset variable HSCAPP.LPSetting).

The HSC subsystem updates this bit continuously whenever the controller is in an executing mode. Writing to this element is not recommended.

Overflow Interrupt (HSCSTS.OFCauseInter)

Description	Data Format	HSC Modes[1]	User Program Access
HSCSTS.OFCauseInter	bit	0...9	read/write

(1)   For Mode descriptions, see <u>HSC Mode (HSCAPP.HSCMode) on page 206</u>.

The Overflow Interrupt status bit is set (1) when the HSC accumulator counts through the overflow value and the HSC interrupt is triggered. This bit can be used in the control program to identify that the overflow variable caused the HSC interrupt. If the control program must perform any specific control action based on the overflow, this bit is used as conditional logic.

This bit can be cleared (0) by the control program and is also cleared by the HSC subsystem whenever these conditions are detected:

- Low Preset Interrupt executes
- High Preset Interrupt executes
- Underflow Interrupt executes

Underflow Interrupt (HSCSTS.UFCauseInter)

Description	Data Format	HSC Modes[1]	User Program Access
HSCSTS.UFCauseInter	bit	2...9	read/write

(1)   For Mode descriptions, see HSC Mode (HSCAPP.HSCMode) on page 206.

The Underflow Interrupt status bit is set (1) when the HSC accumulator counts through the underflow value and the HSC interrupt is triggered. This bit can be used in the control program to identify that the underflow condition caused the HSC interrupt. If the control program must perform any specific control action based on the underflow, this bit is used as conditional logic.

This bit can be cleared (0) by the control program and is also cleared by the HSC subsystem whenever these conditions are detected:

- Low Preset Interrupt occurs
- High Preset Interrupt occurs
- Overflow Interrupt occurs

High Preset Interrupt (HSCSTS.HPCauseInter)

Description	Data Format	HSC Modes[1]	User Program Access
HSCSTS.HPCauseInter	bit	0...9	read/write

(1)   For Mode descriptions, see HSC Mode (HSCAPP.HSCMode) on page 206.

The High Preset Interrupt status bit is set (1) when the HSC accumulator reaches the high preset value and the HSC interrupt is triggered. This bit can be used in the control program to identify that the high preset condition caused the HSC interrupt. If the control program must perform any specific control action based on the high preset, this bit is used as conditional logic.

This bit can be cleared (0) by the control program and is also cleared by the HSC subsystem whenever these conditions are detected:

- Low Preset Interrupt occurs
- Underflow Interrupt occurs
- Overflow Interrupt occurs

Low Preset Interrupt (HSCSTS.LPCauseInter)

Description	Data Format	HSC Modes[1]	User Program Access
HSCSTS.LPCauseInter	bit	2...9	read/write

(1)   For Mode descriptions, see HSC Mode (HSCAPP.HSCMode) on page 206.

The Low Preset Interrupt status bit is set (1) when the HSC accumulator reaches the low preset value and the HSC interrupt has been triggered. This bit can be used in the control program to identify that the low preset condition caused the HSC interrupt. If the control program must perform any specific control action based on the low preset, this bit would be used as conditional logic.

This bit can be cleared (0) by the control program and is also cleared by the HSC sub-system whenever these conditions are detected:

- High Preset Interrupt occurs
- Underflow Interrupt occurs
- Overflow Interrupt occurs

Programmable Limit Switch Position (HSCSTS.PLSPosition)

Description	Data Format	HSC Modes[1]	User Program Access
HSCSTS.PLSPosition	Word (INT)	0...9	read-only

(1)   For Mode descriptions, see HSC Mode (HSCAPP.HSCMode) on page 206.

When the HSC is in Counting mode, and PLS is enabled, this parameter indicates which PLS element is used for the current HSC configuration.

Error Code (HSCSTS.ErrorCode)

Description	Data Format	HSC Modes[1]	User Program Access
HSCSTS.ErrorCode	Word (INT)	0...9	read-only

(1)   For Mode descriptions, see HSC Mode (HSCAPP.HSCMode) on page 206.

Table 93 shows the Error Codes that are detected by the HSC subsystem and are displayed in this word.

Table 93 - HSC Error Codes

Error Code Sub-element	HSC Counting Error Codes	Error Description
Bit 15...8 (high byte)	0...255	The nonzero value for the high byte indicates that the HSC error is due to PLS data setting. The value of high byte indicates which element of PLS data triggers the error.
Bit 7...0 (low byte)	0x00	No error
	0x01	Invalid HSC counting mode
	0x02	Invalid High preset
	0x03	Invalid overflow
	0x04	Invalid underflow
	0x05	No PLS data

Writing to this element is not recommended except for clearing existing errors and to capture new HSC errors.

Accumulator (HSCSTS.Accumulator)

Description	Data Format	User Program Access
HSCSTS.Accumulator	long word (32-bit INT)	read-only

HSCSTS.Accumulator contains the number of counts that are detected by the HSC subsystem. If either mode 0 or mode 1 is configured, the accumulator is reset to 0 when a high preset is reached or when an overflow condition is detected.

High Preset (HSCSTS.HP)

Description	Data Format	User Program Access
HSCSTS.HP	long word (32-bit INT)	read-only

The HSCSTS.HP is the upper setpoint (in counts) that defines when the HSC subsystem generates an interrupt.

The data that is loaded into the high preset must be less than or equal to the data resident in the overflow (HSCAPP.OFSetting) parameter or an HSC error is generated.

This is the latest high preset setting, which may be updated by PLS function from the PLS data block.

Low Preset (HSCSTS.LP)

Description	Data Format	User Program Access
HSCSTS.LP	long word (32-bit INT)	read-only

The HSCSTS.LP is the lower setpoint (in counts) that defines when the HSC subsystem generates an interrupt.

The data that is loaded into the low preset must be greater than or equal to the data resident in the underflow (HSCAPP.UFSetting) parameter, or an HSC error is generated. If the underflow and low preset values are negative numbers, the low preset must be a number with a smaller absolute value.

This is the latest low preset setting, which may be updated by PLS function from the PLS data block.

High Preset Output (HSCSTS.HPOutput)

Description	Data Format	User Program Access
HSCSTS.HPOutput	long word (32 bit binary)	read-only

The High Preset Output defines the state (1 = ON or 0 = OFF) of the outputs on the controller when the high preset is reached. See Output Mask Bits (HSCAPP.OutputMask) on page 211 for more information on how to turn outputs on or off directly based on the high preset being reached.

This is the latest high preset output setting, which may be updated by PLS function from the PLS data block.

Low Preset Output (HSCSTS.LPOutput)

Description	Data Format	User Program Access
HSCSTS.LPOutput	long word (32 bit binary))	read-only

The Low Preset Output defines the state (1 = "on", 0 = "off") of the outputs on the controller when the low preset is reached. See Output Mask Bits (HSCAPP.OutputMask) on page 211 for more information on how to turn outputs on or off directly based on the low preset being reached.

This is the latest low preset output setting, which may be updated by the PLS function from the PLS data block.

High-speed Counter (HSC) Function Block

The HSC function block can be used to start/stop HSC counting, to refresh HSC status, to reload HSC setting, and to reset the HSC accumulator.
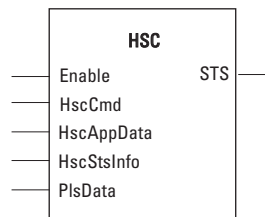
Table 94 - HSC Parameters

Parameter	Parameter Type	Data Type	Parameter Description
Enable	Input	BOOL	Enable function block When Enable = TRUE, perform the HSC operation that is specified in "HSC command" parameter. When Enable = FALSE, there is no HSC operation and no HSC status update.
HscCmd	Input	USINT	See HSC Commands on page 218.
HscAppData	Input	See HSC APP Data Structure on page 204	HSC application configuration Only the initial configuration is needed usually.
PlsData	Input	See array of Programmable Limit Switch (PLS) Function on page 219	Programmable Limit Switch (PLS) Data
HscStsInfo	Output	See HSC STS (HSC Status) Data Structure on page 212	HSC dynamic status Status info is usually continuously updated during HSC counting.
Sts	Output	UINT	HSC function block execution status

HSC Commands (HScCmd)

HscCmd is an input parameter with data type USINT. All HSC commands (1...4) are Level commands. Users are advised to disable the instruction before updating the command.

HscCmd = 1 starts the HSC mechanism. Once the HSC is in running mode, the **HscCmd = 2** must be issued to stop counting. Setting the Enable input parameter to False does not stop counting while in running mode.

HscCmd = 3 reloads the following parameter values: HighPreset, LowPreset, OverFlow, UnderFlow, HighPreset Output, and LowPreset Output.

The parameter values shown in the Variable Monitor may not match the values in the Hardware. Command 3 must be executed to load the values from the variables to the hardware without stopping the HSC.

If the HSC Enable is True, HscCmd = 3 will continuously load the parameters. Trigger HscCmd = 3 only once.

HscCmd = 4 (reset) sets the Acc value to the HSC AppData.Accumalator value. The HscCmd =4 does not stop HSC counting. If the HSC is counting when the HscCmd =4 is issued, some counting may be lost.

To reset the Acc value and then continue the counting, trigger the HscCmd =4 only once. If the command is enabled continuously, it may cause errors.

The HSC mechanism updates the HSC AppData.Accumalator value automatically with the same value as the HSC Sts.Accumulator. To set one specific value to HSC Acc while counting, write the value to HSC AppData.Accumalator immediately before HscCmd =4 is issued.

Table 95 - HSC Commands

HSC Command	Description
0x00	Reserved
0x01	HSC RUN • Start HSC (if HSC in Idle mode and Rung is Enabled) • Update HSC Status Info only (if HSC already in RUN mode and Rung is Enabled) • Update HSC status Info only (if Rung is disabled)
0x02	HSC Stop: Stop an HSC counting (if HSC is in RUN mode and Rung is Enabled.)
0x03	HSC Load: reload HSC Configuration (if Rung is Enabled) for 6 input elements: HPSetting, LPSetting, HPOutput, LPOutput, OFSetting, and UFSetting. HSC accumulator is NOT reloaded by cmd = 0x03.
0x04	HSC Reset: set Accumulator to assigned value, and reset HSC status information (if Rung is Enabled)

Table 96 - HSC Function Block Status Codes

HSC Status Code	Description
0x00	No action from controller because the function block is not enabled
0x01	HSC function block successfully executed
0x02	HSC command invalid
0x03	HSC ID out of range
0x04	HSC configuration error

HSC_SET_STS Function Block
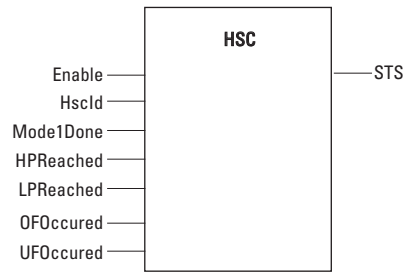
The HSC Set Status function block can be used to change the HSC counting status. This function block is called when the HSC is not counting (stopped).

Table 97 - HSC Parameters

Parameter	Parameter Type	Data Type	Parameter Description
Enable	Input	BOOL	Enable function block When Enable = TRUE, set/reset the HSC status. When Enable = FALSE, there is no HSC status change.
HscId	Input	See HSC APP Data Structure on page 204	Describes which HSC status to set.
Mode1Done	Input	BOOL	Mode 1A or 1B counting is done.
HPReached	Input	BOOL	High Preset reached This bit can be reset to FALSE when HSC is not counting.
LPReached	Input	BOOL	Low Preset reached This bit can be reset to FALSE when HSC is not counting.
OFOccurred	Input	BOOL	Overflow occurred This bit can be reset to FALSE when necessary.
UFOccurred	Input	BOOL	Underflow occurred This bit can be reset to FALSE when necessary.
Sts	Output	UINT	HSC function block execution status See HSC Function Block Status Codes on page 218 for HSC status code description (except 0x02 and 0x04).

Programmable Limit Switch (PLS) Function

The Programmable Limit Switch function allows you to configure the High-Speed Counter to operate as a PLS (programmable limit switch) or rotary cam switch.

When PLS operation is enabled (HSCAPP.PLSEnable = True), the HSC (High-Speed Counter) uses PLS data for limit/cam positions. Each limit/cam position has corresponding data parameters that are used to set or clear physical outputs on the controller's base unit. The PLS data block is illustrated in Figure 55 on page 220.

> **IMPORTANT**   The PLS Function only operates in tandem with the HSC of a Micro830 controller. To use the PLS function, an HSC must first be configured.

PLS Data Structure

The Programmable Limit Switch function is an additional set of operating modes for the high-speed counter. When operating in these modes, the preset and output data values are updated using user supplied data each time one of the presets is reached. These modes are programmed by providing a PLS data block that contains the data sets to be used.

PLS data structure is a flexible array, with each element defined as follows.

Element Order	Data Type	Element Description
Word 0...1	DINT	High preset setting

Element Order	Data Type	Element Description
Word 2…3	DINT	Low preset setting
Word 4…5	UDINT	High preset Output data
Word 6…7	UDINT	Low preset Output data

The total number of elements for one PLS data cannot be larger than 255.

When PLS is not enabled, PLS data is still required to be defined, but can be not initialized.

Figure 55 - PLS Data Block
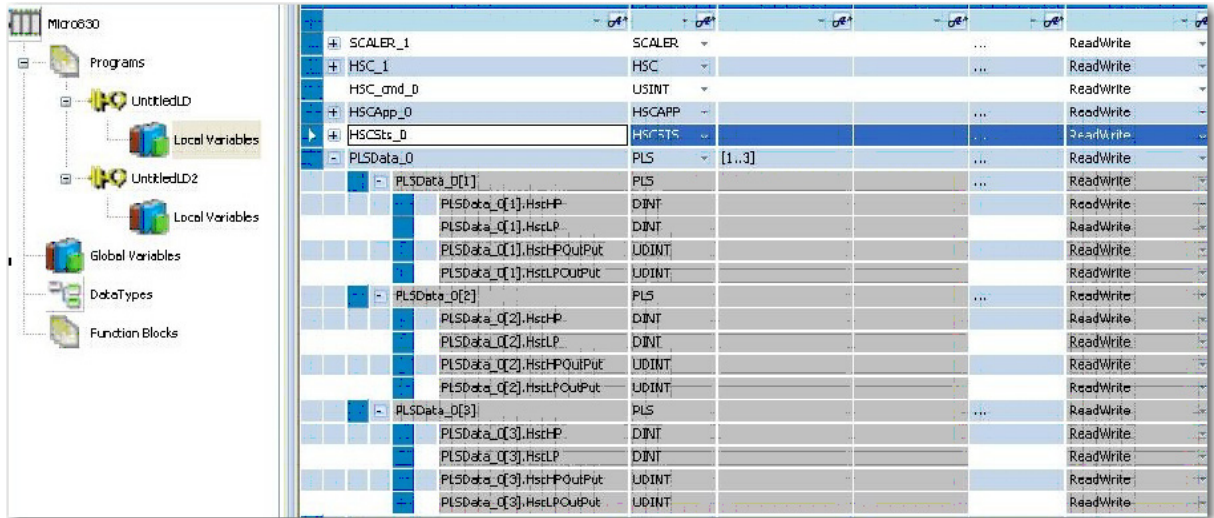
PLS Operation

When the PLS function is enabled, and the controller is in the run mode, the HSC counts incoming pulses. When the count reaches the first preset (HSCHP or HSCLP) defined in the PLS data, the output source data (HSCHPOutput or HSCLPOutput) is written through the HSC mask (HSCAPP.OutputMask).

At that point, the next presets (HSCHP and HSCLP) defined in the PLS data become active.

When the HSC counts to that new preset, the new output data is written through the HSC mask. This process continues until the last element within the PLS data block is loaded. At that point, the active element within the PLS data block is reset to zero. This behavior is referred to as circular operation.

> The HSCHPOutput is only written when HSCHP is reached. The HSCLPOutput is written when HSCLP is reached.

> Output High Data is only operational when the counter is counting up. Output Low Data is only operational when the counter is counting down.

If invalid data is loaded during operation, an HSC error is generated and causes a controller fault.

You can use the PLS in Up (high), Down (low), or both directions. If your application only counts in one direction, ignore the other parameters.

The PLS function can operate with all other HSC capabilities. The ability to select which HSC events generate a user interrupt are not limited.

PLS Example

Setting Up the PLS Data

Using Connected Components Workbench software, define the PLS data HSC_PLS's dimension as [1...4].

PLS Data Definition

Data	Description	Data Format
HSCHP	High Preset	32 bit signed integer
HSCLP	Low Preset	
HSCHPOutput	Output High Data	32 bit binary
HSCLPOutput	Output Low Data	(bit 31--> 0000 0000 0000 0000 0000 0000 0000 0000 <--bit 0)

Once the values above for all 4 PLS data elements have been entered, the PLS is configured.

Assume that HSCAPP.OutputMask = 31 (HSC mechanism controls Embedded Output 0...4 only), and HSCAPP.HSCMode = 0.

PLS Operation for This Example

When the ladder logic first runs, HSCSTS.Accumulator = 1, therefore all outputs are turned off. The value of HSCSTS.HP = 250

When HSCSTS.Accumulator = 250, the HSC_PLS[1].HscHPOutput is sent through the HSCAPP.OutputMask and energizes the outputs 0 and 1.

This repeats as the HSCSTS.Accumulator reaches 500, 750, and 1000. The controller energizes outputs 0...2, 0...3, and 0...4 respectively. Once completed, the cycle resets and repeats from HSCSTS.HP = 250.

HSC Interrupts

An interrupt is an event that causes the controller to suspend the task it is performing, perform another task, and then return to the suspended task at the point where it suspended. Micro800 controllers support up to six HSC interrupts.

An HSC interrupt is a mechanism that Micro830, Micro850, and Micro870 controllers provide to execute selected user logic at a pre-configured event.
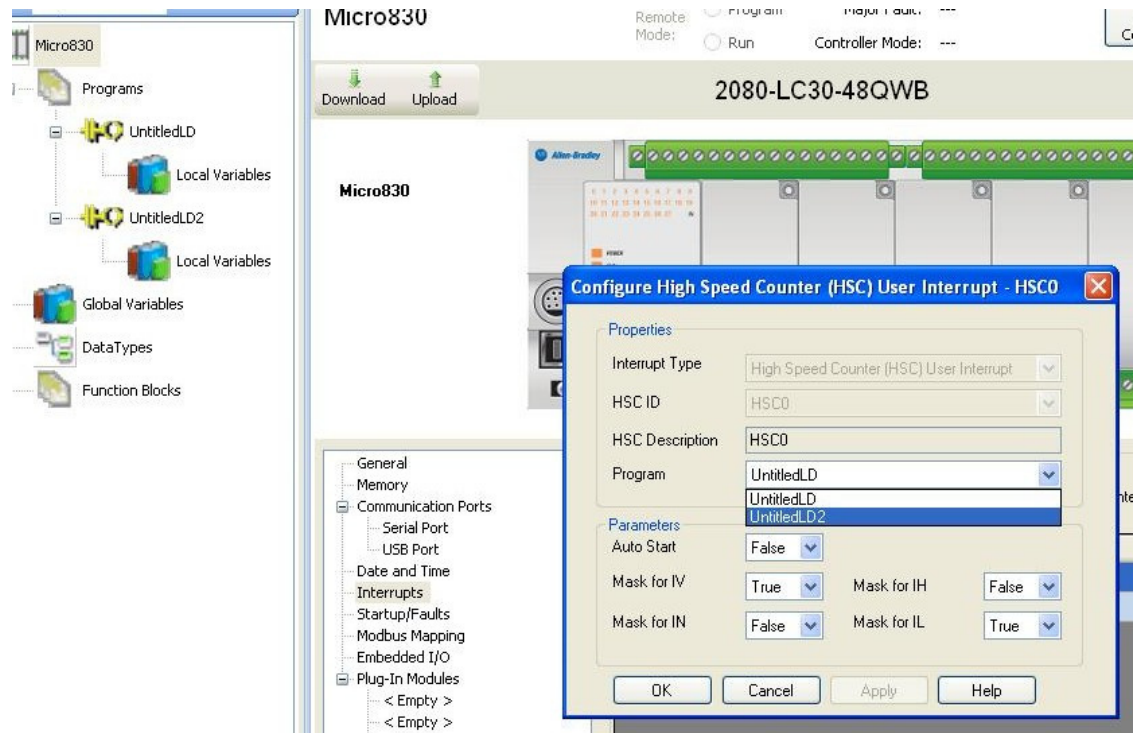
HSC0 is used in this document to define how HSC interrupts work.

HSC Interrupt Configuration

In the User Interrupt configuration window, select HSC and HSC ID, which is the interrupt that triggers the User Interrupt.

Figure 56 shows the selectable fields in the Interrupt configuration window.

Figure 56 - Interrupt Configuration Window

HSC Interrupt POU

This is the name of the Program Organizational Unit (POU) which is executed immediately when this HSC Interrupt occurs. You can choose any pre-programmed POU from the dropdown list.

Auto Start (HSC0.AS)

Description	Data Format	HSC Modes[1]	User Program Access
AS - Auto Start	bit	0...9	read-only

(1)   For Mode descriptions, see Count Down (HSCSTS.CountDownFlag) on page 213.

The Auto Start is configured with the programming device and stored as part of the user program. The auto start bit defines if the HSC interrupt function automatically starts whenever the controller enters any run or test mode.

Mask for IV (HSC0.MV)

Description	Data Format	HSC Modes[1]	User Program Access
MV - Overflow Mask	bit	0...9	read-only

(1)   For Mode descriptions, see Count Down (HSCSTS.CountDownFlag) on page 213.

The MV (Overflow Mask) control bit is used to enable (allow) or disable (not allow) an overflow interrupt from occurring. If this bit is clear (0), and an overflow reached condition is detected by the HSC, the HSC user interrupt is not executed.

This bit is controlled by the user program and retains its value through a power cycle. It is up to the user program to set and clear this bit.

Mask for IN (HSC0.MN)

Description	Data Format	HSC Modes[1]	User Program Access
MN - Underflow Mask	bit	2...9	read-only

(1)   For Mode descriptions, see Count Down (HSCSTS.CountDownFlag) on page 213.

The MN (Underflow Mask) control bit is used to enable (allow) or disable (not allow) an underflow interrupt from occurring. If this bit is clear (0), and an Underflow Reached condition is detected by the HSC, the HSC user interrupt is not executed.

This bit is controlled by the user program and retains its value through a power cycle. It is up to the user program to set and clear this bit.

Mask for IH (HSC0.MH)

Description	Data Format	HSC Modes[1]	User Program Access
MH - High Preset Mask	bit	0...9	read-only

(1)   For Mode descriptions, see Count Down (HSCSTS.CountDownFlag) on page 213.

The MH (High Preset Mask) control bit is used to enable (allow) or disable (not allow) a high preset interrupt from occurring. If this bit is clear (0), and a High Preset Reached condition is detected by the HSC, the HSC user interrupt is not executed.

This bit is controlled by the user program and retains its value through a power cycle. It is up to the user program to set and clear this bit.

Mask for IL (HSC0.ML)

Description	Data Format	HSC Modes[1]	User Program Access
ML - Low Preset Mask	bit	2...9	read-only

(1)   For Mode descriptions, see Count Down (HSCSTS.CountDownFlag) on page 213.

The ML (Low Preset Mask) control bit is used to enable (allow) or disable (not allow) a low preset interrupt from occurring. If this bit is clear (0), and a Low Preset Reached condition is detected by the HSC, the HSC user interrupt is not executed.

This bit is controlled by the user program and retains its value through a power cycle. It is up to the user program to set and clear this bit.

HSC Interrupt Status Information

User Interrupt Enable (HSC0.Enabled)

Description	Data Format	HSC Modes[1]	User Program Access
HSC0.Enabled	bit	0...9	read-only

(1)   For Mode descriptions, see Count Down (HSCSTS.CountDownFlag) on page 213.

The Enabled bit is used to indicate HSC interrupt enable or disable status.

User Interrupt Executing (HSC0.EX)

Description	Data Format	HSC Modes[1]	User Program Access
HSC0.EX	bit	0...9	read-only

(1)   For Mode descriptions, see Count Down (HSCSTS.CountDownFlag) on page 213.

The EX (User Interrupt Executing) bit is set (1) whenever the HSC subsystem begins processing the HSC subroutine due to any of the following conditions:

- Low preset reached
- High preset reached
- Overflow condition – Count up through the overflow value
- Underflow condition – Count down through the underflow value

The HSC EX bit can be used in the control program as conditional logic to detect if an HSC interrupt is executing.

The HSC subsystem clears (0) the EX bit when the controller completes its processing of the HSC subroutine.

User Interrupt Pending (HSC0.PE)

Description	Data Format	HSC Modes[1]	User Program Access
HSC0.PE	bit	0...9	read-only

(1)   For Mode descriptions, see Count Down (HSCSTS.CountDownFlag) on page 213.

The PE (User Interrupt Pending) is a status flag that represents an interrupt is pending. This status bit can be monitored or used for logic purposes in the control program if you must determine when a subroutine cannot be executed immediately. The controller maintains this bit, and sets and clears the bit automatically.

User Interrupt Lost (HSC0.LS)

Description	Data Format	HSC Modes[1]	User Program Access
HSC0.LS	bit	0...9	read/write

(1)   For Mode descriptions, see Count Down (HSCSTS.CountDownFlag) on page 213.

The LS (User Interrupt Lost) is a status flag that represents an interrupt has been lost. The controller can process 1 active and maintain up to 1 pending user interrupt conditions before it sets the lost bit.

The controller sets this bit. It is up to the control program to use, track the lost condition if necessary.

Controller Security

Micro800 controller security generally has three components:

- **Exclusive Access that** prevents simultaneous configuration of the controller by two users
- **Controller Password Protection that secures the Intellectual Property** that is contained within the controller and prevents unauthorized access
- **Disable communication ports** that prevents unauthorized users from accessing the ports through external devices.

Operation Mode

To maintain the secure operation of your Micro800 controllers, operations that can disrupt controller operation are restricted based on the controller operating mode.

Table 98 - Activities Allowed in Different Controller Operating Modes while Online

Current Controller Operation	Activity						
	Firmware Update Request	**Ethernet Port Configuration Setting[1] (through Connected Components Workbench or RSLinx software)**	**Serial and USB Port Configuration Changes**	**Lost Password Recovery**	**Password Change**	**Controller Mode Change**	**I/O Configuration Change**
Controller in Program Mode	Accepted	Accepted	Not Allowed	Accepted	Accepted	Accepted	Not Allowed
Controller without Password Protection in Remote Run Mode	Rejected	Not Allowed	Not Allowed	Not Applicable	Not Applicable	Accepted	Not Allowed
Controller with Password Protection in Remote Run Mode	Rejected	Not Allowed[2]	Not Allowed	Rejected	Rejected	Rejected	Not Allowed
Controller in Hard Run Mode[3]	Rejected	Not Allowed	Not Allowed	Rejected	Rejected	Rejected	Not Allowed

(1)   Ethernet configuration includes IP address, subnet mask, gateway, port speed/duplex and so on.
(2)   Difference between Not Allowed and Rejected is that Not Allowed activities can only be done during offline while Rejected activities can be performed but do not take effect.
(3)   Hard Run Mode can only be in Micro830, Micro850, and Micro870 controllers with the Mode switch to RUN.

Exclusive Access

Exclusive access is enforced on the Micro800 controller regardless of whether the controller is password-protected or not. This means that only one Connected Components Workbench software session is authorized at one time and only an authorized client has exclusive access to the controller application. This ensures that only one software session has exclusive access to the Micro800 application-specific configuration.

Exclusive access is enforced on Micro800 firmware revision 1 and 2. When a Connected Components Workbench software user connects to a Micro800 controller, the controller is given exclusive access to that controller.

Password Protection

By setting a password on the controller, you effectively restrict access to the programming software connections to the controller to software sessions that can supply the correct password. Essentially, Connected Components Workbench software operation such as upload and download are prevented if the controller is secured with a password and the correct password is not provided.

Micro800 controllers with firmware revision 2 and later are shipped with no password but a password can be set through the Connected Components Workbench software (version 2 or later).

In Connected Components Workbench software version 10 or later, a stronger password algorithm is introduced to provide better security. To take full advantage of this enhancement, the Micro800 controller must have firmware revision 10 or later, and the project must also be software version 10 or later.

In Connected Components Workbench software version 20.01.00 or later, the password algorithm is further enhanced to increase the password encryption on the new Micro850 (2080-L50E) and Micro870 (2080-L70E) controllers. When the password is changed, or removed and restored back in the controller, the encrypted code in the controller is different. Therefore the backup copy of the program in the 2080-MEMBAK-RTC2 module must be updated before it can be used for restoring the program, otherwise the restore fails.

The controller password is also backed up to the memory backup module (2080-MEMBAK-RTC and 2080-MEMBAK-RTC2).

IMPORTANT	2080-MEMBAK-RTC is not supported on Micro850 (2080-L50E) and Micro870 controllers.

For instructions on how to set, change, and clear controller passwords, see .

Disable Communication Ports

You can disable the communication ports in the Micro850 and Micro870 controllers to increase the security level of the controller. Both the USB and Ethernet ports can be set to disabled in the Connected Components Workbench software.

IMPORTANT	You must verify that you have a method to connect to the controller after you have disabled the communication ports, so that you can re-enable the ports if needed. You must connect to the controller online to make the change.

Disable USB port   Disable Ethernet port
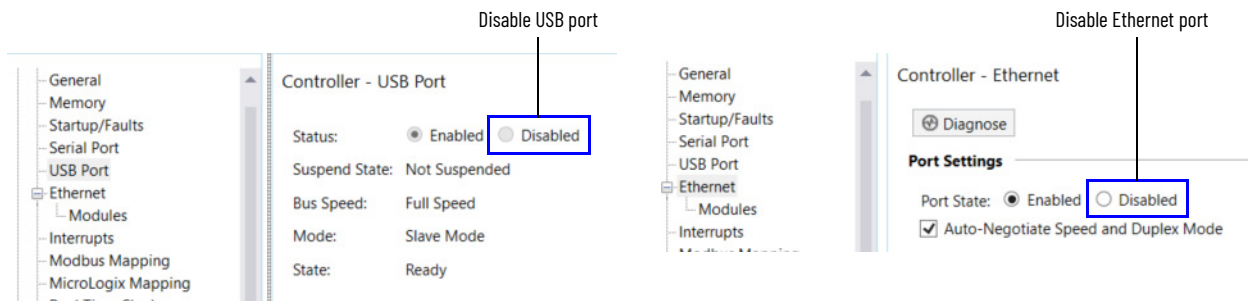
Compatibility

The Controller Password feature is supported on:

- Connected Components Workbench software **version 2** and later
- Micro800 controllers with firmware **revision 2**

For users with earlier versions of the software and/or revisions of the hardware, see the following compatibility scenarios.

Connected Components Workbench Software Version 1 with Micro800 Controller Firmware Revision 2

Connection to a Micro800 controller with firmware revision 2 using an earlier version of the Connected Components Workbench software (version 1) is possible and connections are successful. However, the software is not able to determine whether the controller is locked or not.

If the controller is not locked, access to the user application is allowed, provided that the controller is not busy with another session. If the controller is locked, access to the user application fails. You must upgrade to version 2 of the Connected Components Workbench software.

Connected Components Workbench Software Version 2 with Micro800 Controller Firmware Revision 1

Connected Components Workbench software version 2 can "discover" and connect to Micro800 controllers with firmware revision earlier than revision 2 (that is, not supporting the Controller Password feature). However, the Controller Password feature is not available to these controllers. You cannot see the interfaces that are associated with the Controller Password feature in the Connected Components Workbench software session.

Users are advised to upgrade the firmware. See for instructions.

> ⚠️ **ATTENTION:** Connected Components Workbench software version 9 or earlier with Micro800 controller firmware revision 10 or later.
> If a Micro800 controller with firmware revision 10 or later is locked using the new password algorithm that is introduced in Connected Components Workbench software version 10 or later, it cannot be accessed using Connected Components Workbench software version 9 or earlier. Users are advised to upgrade to the latest version of Connected Components Workbench software.

Work with a Locked Controller

The following workflows are supported on compatible Micro800 controllers (firmware revision 2) and Connected Components Workbench software version 2.

Upload from a Password-Protected Controller

1. Launch the Connected Components Workbench software.
2. In the Project Organizer, expand Catalog by clicking the + sign.
3. Select the target controller.
4. Select Upload.
5. When requested, provide the controller password.

> **IMPORTANT**   When using Connected Components Workbench software version 9 or earlier:
> - You cannot upload a version 10 or later project from the controller.
> - You can upload a version 9 or earlier project from the controller if it was downloaded to the controller using Connected Components Workbench software version 10 or later, but you cannot go online.

Debug a Password-Protected Controller

To debug a locked controller, you have to connect to the controller through the Connected Components Workbench software and provide the password before you can proceed to debug.

1. Launch the Connected Components Workbench software.
2. In the Project Organizer, expand Catalog by clicking the + sign.
3. Select the catalog number of your controller.
4. When requested, provide the controller password.
5. Build and save your project.
6. Debug.

Download to a Password-Protected Controller

1. Launch the Connected Components Workbench software.
2. Click Connect.
3. Select the target controller.
4. When requested, provide the controller password.
5. Build and save the project, if needed.
6. Click Download.
7. Click Disconnect.

IMPORTANT	If the controller has a password locked version 10 or later project, you cannot access the controller using Connected Components Workbench software version 9 or earlier. If you use Connected Components Workbench software version 10 or later to download a version 9 or earlier project, the password in the controller is converted to the old algorithm automatically.

IMPORTANT	If the controller has a password locked version 9 or earlier project and you use Connected Components Workbench software version 10 or later, to download a version 10 or later project, the password in the controller is converted to the new algorithm automatically.

IMPORTANT	If communication is lost during the download, repeat the download and verify that the controller is password protected.

Transfer Controller Program and Password-Protect Receiving Controller

In this scenario, you must transfer the user application from controller1 (locked) to another Micro800 controller with the same catalog number. The transfer of the user application is done through the Connected Components Workbench software by uploading from controller1, then changing the target controller in the Micro800 project, and then downloading to controller2. Finally, controller2 is locked.

1. In the Project Organizer, click the Discover icon.
   The Browse Connections dialog appears.
2. Select target controller1.
3. When requested, enter the controller password for controller1.
4. Build and save the project.
5. Click Disconnect.
6. Power down controller1.
7. Swap controller1 hardware with controller2 hardware.

8. Power up controller2.

9. Click Connect.

10. Select target controller2.

11. Click Download.

12. Lock controller2. See .

Back Up and Restore a Password-Protected Controller

In this workflow, the user application is backed up from a Micro800 controller that is locked to a memory plug-in device.

1. In the Project Organizer, click the Discover icon.
    The Browse Connections dialog appears.

2. Select the target controller.

3. When requested, enter the controller password.

4. Back up controller contents to the memory module.
    The project in the memory module is now password locked.

5. Remove the memory module from controller1 and insert into controller2.

6. Restore contents from the memory module to controller2.
    This operation succeeds only if:

    - the controller has no password – the project can be restored to the controller by setting the "Load on power up" option for the memory module to Load Always.

    - the controller's password matches the project's password.

IMPORTANT	Even though the password matches, the restore operation fails if either one of the controller or project in the memory module is protected using the old password algorithm, and the other is protected using the new password algorithm. You can update the controller using the Reset option to clear the password before restoring the project to the controller.

IMPORTANT	In Connected Components Workbench software version 20.01.00 or later, if the same password is removed and restored back on the new Micro850 (2080-L50E) and Micro870 (2080-L70E) controllers, the backup copy of the program in the 2080-MEMBAK-RTC2 plug-in module must be updated before it can be used to restore the program. Otherwise the restore fails as the internal encryption code changes once the password is removed.

Configure Controller Password

To set, change, and clear the controller password, see the quick start instructions .

IMPORTANT	After creating or changing the controller password, you must power down the controller in order for the password to be saved.

Recover from a Lost Password

If the controller is secured with a password and the password has been lost, then it becomes impossible to access the controller using the Connected Components Workbench software.

To recover, the controller must be set to Program Mode using the keyswitch for Micro830, Micro850, and Micro870 controllers, the 2080-LCD for Micro810 controllers, or the 2080-REMLCD for Micro820 controllers. Then, ControlFLASH™ can be used to update the controller firmware, which also clears the controller memory. In Connected Components Workbench software version 10 or later, the Reset option must be selected for the controller memory to be cleared during the firmware update. If the Upgrade or Downgrade option is selected, the password is retained.

> ⚠️ **ATTENTION:** The project in the controller is lost but a new project can be downloaded.

Using the Memory Module Plug-in

You can use the memory module (2080-MEMBAK-RTC and 2080-MEMBAK-RTC2) to download a program into different controllers.

> **IMPORTANT**   2080-MEMBAK-RTC is not supported on Micro850 (2080-L50E) and Micro870 controllers.

Back up the project

To perform a project backup from a controller to the memory module, follow these steps:

1. Create a program and choose the desired controller.

   The Micro850 controller is used for this example.

2. Double-click the controller icon under Project Organizer to bring up the controller properties window.
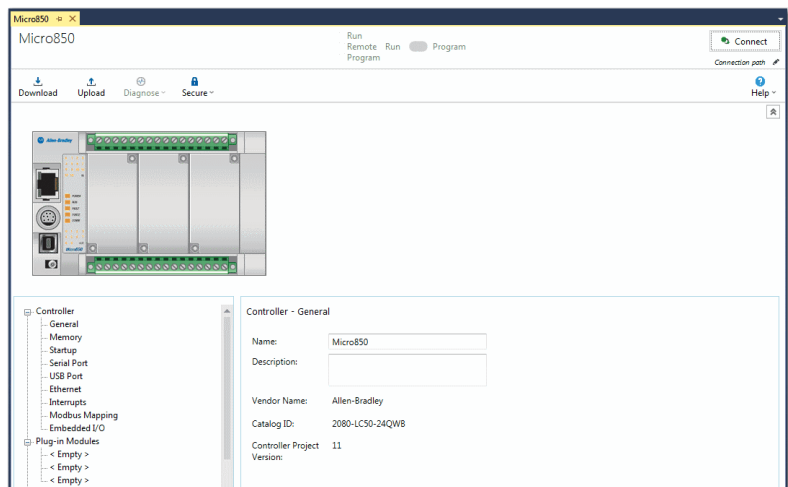
3. Add the memory module to the first slot in the controller.
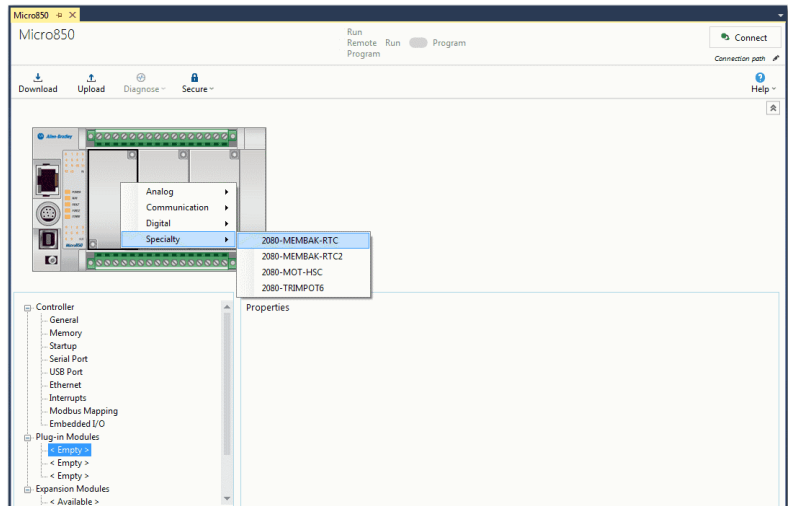
4. Select Configuration under the MEMBAK-RTC properties and select "Load Always" or "Load on Memory Error" for the Load on power up option.
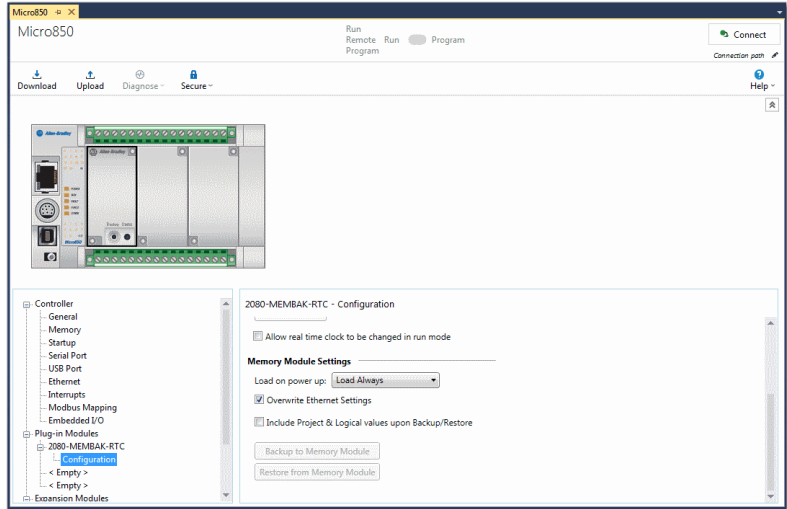
5. Build and Download the project to the controller.

6. While connected to the controller and being in the MEMBAK-RTC properties, make sure that the controller is changed to Program Mode and click "Backup to Memory Module" under Memory Module Settings. Select Yes to download the program into the Memory Module. A window should pop up stating that the operation has been completed successfully.

IMPORTANT	The Password Mismatch status must be at "False", this means that the Controller and Backup project have the same security condition.
	If the status is "True" then the Controller does not load from the Memory module as the security condition is mismatched.

Restore the Project

To restore the project from the memory module to the controller, follow these steps:

1. While connected to the controller and being in the MEMBAK-RTC properties, make sure that the controller is changed to Program Mode and click "Restore from Memory Module" under Memory Module Settings. Select Yes to download the program into the controller. A window should pop up stating that the operation has been completed successfully.

Using the Memory Module to Copy a Project to Multiple Controllers

You can use the memory module to download a project to multiple controllers without connecting them to a PC with Connected Components Workbench software installed. To do this:

1. Back up a project with "Load Always" option enabled.

2. Remove the module and plug it into another controller.

3. Cycle the power. Observe the Status LED on the module lights up for a few seconds while the project is being downloaded from the module to the controller.

4. When the operation is finished you can unplug the module and leave the slot empty, or plug in another MEMBAK-RTC module if you want to use the RTC functionality.

Notes:

Using microSD Cards

This chapter provides a description of microSD card support on Micro830, Micro850, and Micro870 controllers. The last section provides quick start projects for the data log and recipe functions.

Overview

With firmware revision 12.011 or later, Micro830, Micro850, and Micro870 controllers support microSD cards by using the microSD card plug-in (a PartnerNetwork™ Technology partner product) for Micro800 controllers for the following purposes:

- Project backup and restore
- Data log and Recipe

We recommend using the Allen-Bradley 2080-SD-2GB microSD card.

IMPORTANT	For Micro850 (2080-L50E) and Micro870 (2080-L70E) controllers, do not use any microSD card with less than 2 GB of memory.

IMPORTANT	For optimum performance, the microSD card should not be more than 90% full. Regularly check available space on your microSD card and verify that the card is exclusively used for the Micro800 controller and no unnecessary files are present. Regularly delete old data log files and directories.

IMPORTANT	Do not remove the microSD card or power down while operations such as upload, download, delete, search, backup, and restore are ongoing to prevent data loss. A blinking SD status LED indicates that these operations are ongoing. Note the following: • The SD status LED does not blink when updating the firmware from the microSD card. • The SD status LED does not blink continuously for the entire duration of the restore operation.

IMPORTANT	To prevent data loss, the recipe and data log function blocks must indicate the Idle status before the microSD card is removed.

Project Backup and Restore

Project backup and restore on Micro830, Micro850, and Micro870 controllers are supported through the microSD card. Both backup and restore can be initiated or manually triggered and configured through the Connected Components Workbench software, and the ConfigMeFirst.txt file in the microSD card. These backup files are not the same as the Connected Components Workbench project files.

Backup and restore can only occur when the controller is in PROGRAM mode. On controller power-up, restore automatically occurs if the Load Always or Load on Memory Error option has been configured in the Connected Components Workbench software.
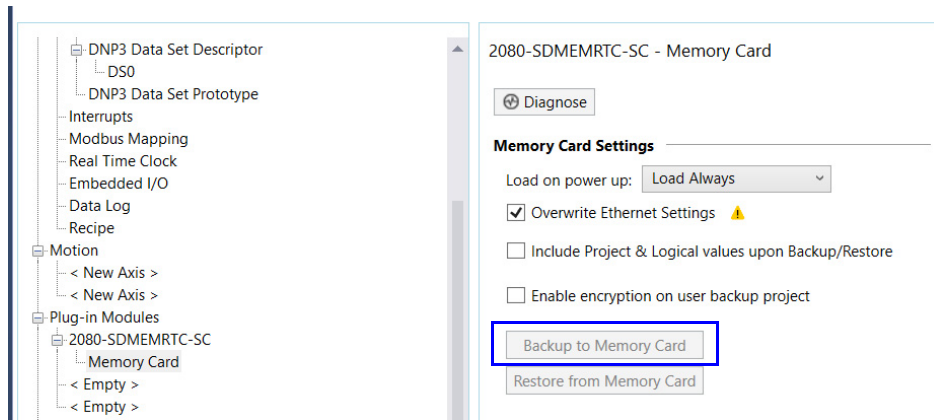
In Connected Components Workbench software version 20.01.00 or later, when you use the Micro850 (2080-L50E) and Micro870 (2080-L70E) controllers with the microSD card plug-in module, the encryption functionality option is added to help enhance the encryption capability on your programs for better security protection.

If you select the option "Enable Encryption on user backup project", the time it takes to restore the program can increase up to 10 times, depending on the size and content of the program.

IMPORTANT	To learn about restore and backup using the Connected Components Workbench software, see the software Online Help.

IMPORTANT	For Micro800 controllers that support microSD cards, IP protection of your project can only be achieved through the POU password protection mechanism in the Connected Components Workbench Developer Edition software and NOT via the Controller Lock feature.

IMPORTANT	If the Load Always setting is enabled and power is lost when restoring a project from the microSD card, the controller will attempt to load the project using the default project name and directory after power is restored. If your project is not using the default name and directory, the operation fails and a fault occurs, or the wrong project is loaded.
	The default project name is the name of the controller, for example "Micro850", and the default directory is "Micro850\USERPRJ".
	If you change the name of the controller from the default, you must configure the UPD setting in the ConfigMeFirst.txt file.

The microSD card stores the controller password in encrypted format. When the password is mismatched, the contents of the microSD card is not restored on the controller.

Table 99 describes the methods that you can use to trigger project backup and restore.

Table 99 - Methods for Backup and Restore

Method	Backup	Restore
Online with Connected Components Workbench software	Yes	Yes
Project configuration on memory card at power-up	No	Load Always and/or Load on Memory Error options
ConfigMeFirst.txt at power-up	Yes (Through the [BKD] command)	Yes (Through the [RSD] command)

Backup and Restore Directory Structure
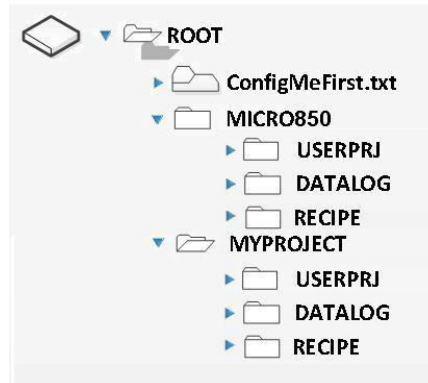
When a user project is backed up, a subdirectory that is named <Micro800>\USERPRJ is created on the microSD card. The folder name takes the name of the project that is specified in the General Page in the Connected Components Workbench software, which is the name of the controller by default. However, if the ConfigMeFirst.txt file specifies another subdirectory (example: MyProject), the project is backed up to that directory. See .
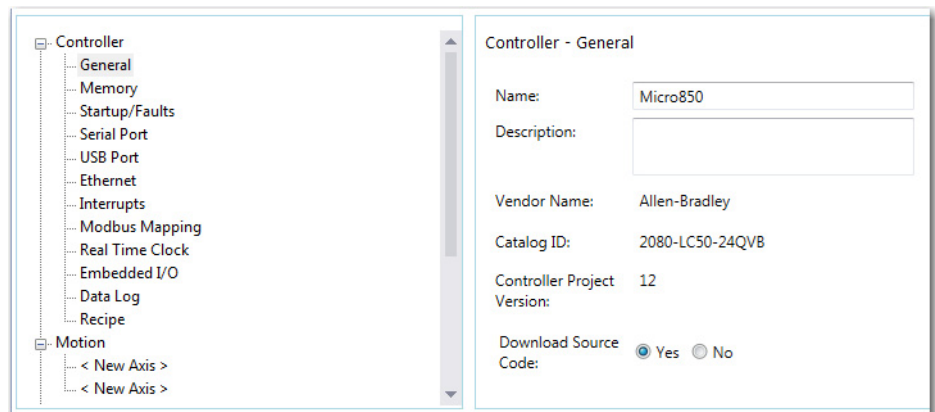
Project restore is done from the subdirectory that is specified in the ConfigMeFirst.txt file or the <Micro800>/USERPRJ default folder, if none is specified in the ConfigMeFirst.txt file. You must verify that the directory is populated with the correct contents before restoring.

The ConfigMeFirst.txt file is a configuration file that is stored on the microSD card that you can optionally create to customize backup, restore, recipe, and data log directories. The following sections include information on how to configure the ConfigMeFirst.txt properly.

IMPORTANT	The Micro800 controller reports a major fault when project backup does not succeed because the memory card size is exceeded.

Power-up Settings in ConfigMeFirst.txt

On power-up, the controller reads and implements the configuration settings that are described in the ConfigMeFirst.txt file. However, the UPD setting also takes effect when the microSD card is inserted. The configuration settings for the ConfigMeFirst.txt file are shown in .

Table 100 - ConfigMeFirst.txt Configuration Settings

Setting	Takes Effect On...	Description
Firmware update settings		
[FWFILE]	Power-up	File path location of the firmware revision on the microSD card. The default location is in the following format: firmware\<catalog number>\<filename of firmware>
[FWDOWN]	Power-up	Sets whether to upgrade or downgrade the controller firmware from the current revision. 0 = Upgrade firmware; 1 = Downgrade firmware **IMPORTANT:** Firmware upgrade happens if the [FWFILE] setting points to a newer revision of firmware file compared to current firmware in the controller, irrespective of the [FWDOWN] setting.
Controller settings		
[PM]	Power-up	Power up and switch to PROGRAM mode.
[CF]	Power-up	Power up and attempt to clear fault.
Project settings		
[BKD = My Proj1]	Power-up	Power up and save the controller project into the backup directory, My Proj1\USERPRJ. Require extra power cycle to clear existing fault first using [CF] setting or other means.
[RSD = MyProj2]	Power-up	Power up and read the project from the restore directory MyProj2\USERPRJ into the controller. Require extra power cycle to clear existing fault first using [CF] setting or other means. This setting overwrites UPD (or its default) load always or load on error restore function.
[UPD = My Proj]	Power-up and Insertion	For normal usage of backup and restore (that is, through Connected Components Workbench software, Load Always, or Load on Memory Error settings), set the user project directory name. For example, My Proj, during power-up or when the microSD card is inserted. This directory is also used by the data logging and recipe function.
Network settings		
[ESFD]	Power-up	Embedded serial factory defaults Power up and revert embedded serial comms to factory defaults.
[IPA = xxx.xxx.xxx.xxx]	Power-up	Power up and set IP address to xxx (must be numbers only).
[SNM = xxx.xxx.xxx.xxx]	Power-up	Power up and set subnet mask to xxx (must be numbers only).
[GWA = xxx.xxx.xxx.xxx]	Power-up	Power up and set gateway address to xxx (must be numbers only).
General settings		
[END]	Power-up	End of setting This setting is **always** required even when the ConfigMeFirst.txt file does not contain any other setting. The SD LED goes off when this setting is not present.

> **IMPORTANT** **Update Settings**
>
> With Connected Components Workbench software version 12 or later, you can update your Micro800 controller from the microSD card in addition to using ControlFLASH. See <u>Firmware Update From MicroSD Card on page 269</u> for instructions.
>
> • [FWFILE] and [FWDOWN] settings must be placed at the beginning of the file.

> **IMPORTANT**   **Directory Settings**
>
> - If no directory has been specified in the ConfigMeFirst.txt file, then backup and restore occurs in the controller name directory (<Micro800>/USERPRJ, by default).
> - If [UPD] is configured in the ConfigMeFirst.txt file, then backup and restore occurs in the [UPD] directory specified.
> - [BKD] setting is implemented even when the controller is locked or password protected.
> - [BKD] directory is automatically created if it does not yet exist.

> **IMPORTANT**   **Powerup Network Parameter Settings**
>
> - [IPA], [SNM] and [GWA] follow the general IP configuration rules.
> - [IPA], when set in ConfigMeFirst.txt, should always be configured with a valid [SNM] and vice versa.
> - When the optional [GWA] setting is used, make sure that [IPA] and [SNM] settings are also present in ConfigMeFirst.txt.
> - The [ESFD], [IPA], [SNM], and [GWA] settings overwrite the respective communication settings from project restore due to [RSD], Load Always or Load on Memory Error.

Sample ConfigMeFirst.txt File

```
ConfigMeFirst.txt - Notepad
File  Edit  Format  View  Help
# This is a comment because there is a "#"
# These settings take effect after power cycle
#
# [PM]                    # Force to Program mode.
# [CF]                    # Clear any faults if possible.
# [ESFD]                  # Set embedded serial port back to factory defaults.
# [IPA=192.168.0.100]     # Ethernet IP address
# [SNM=255.255.255.0]     # Ethernet subnet mask
# [GWA=192.168.0.1]       # Ethernet gateway address
[BKD=MyProj1]             # First, back up current project in machine.
[RSD=MyProj2]             # Then, put new project in machine.
[END]                     # This setting should always exist, even if there are no
                          # other configurations.
```

General Configuration Rules in ConfigMeFirst.txt

- All settings must be in upper case and enclosed in brackets [ ].

- Each line must contain only one setting.

- Settings must always appear first in a line.

- Comments are started with the # symbol.

- No action related to the setting is performed when the setting does not exist, or a # symbol appears before the setting (example, #[PM]).

ConfigMeFirst.txt Errors

The SD status LED goes off when the microSD card is inserted during PROGRAM or RUN mode (or on power-up) and the ConfigMeFirst.txt file is either unreadable or invalid. The ConfigMeFirst.txt file is invalid when it has the following errors:

- Unrecognized setting (that is, the first three configuration rules have not been followed).

- The setting parameters after the = symbol is invalid, does not exist, or out of range.

- The same setting exists twice or more.

- One or more non-setting characters exist within the same bracket.

- Space in between setting characters (example, [P M]).

- Space in between IP address, subnet mask, and gateway address (for example, xxx. x xx.xxx.xxx).
- Only one of the network parameter settings ([IPA], [SNM], or [GWA]) is assigned.
- [END] setting does not exist (even if there are no other settings in the configuration file).

The microSD card becomes unusable until the ConfigMeFirst.txt file becomes readable or the errors are corrected.

Deliver Project Updates to Customers Through Email

A benefit of using the project backup and restore feature is to allow you to deliver project updates to customers through email. You can do so by following the example that is shown below.

Back up project to microSD card

The first step is to back up the project from the controller into the microSD card.

1. In the Connected Components Workbench software, verify that you have downloaded the updated project to your controller.

2. Insert a microSD card into the microSD card slot.

3. Set the controller to program mode.

4. Under the Memory Card option in your controller settings, select Backup to Memory Card.

> **IMPORTANT**   The Backup to Memory Card button is enabled when the controller is in program mode and a microSD card is in the microSD card slot.

5. After the backup is completed, select OK.

The image files are stored in the default location on the microSD card <Micro800>\USERPRJ. This location is where the controller loads from when the Load on power up setting is configured to "Load Always" or "Load on Error".

Alternatively, if you do not want to use Connected Components Workbench software to create the project backup, you can also use the ConfigMeFirst.txt file.

Figure 57 – Example Configuration for Project Backup
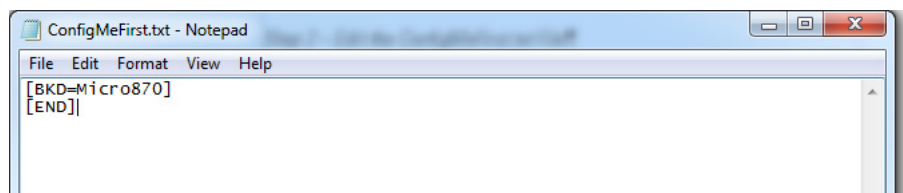
The ConfigMeFirst.txt file also allows you to restore from the backup if you want to configure the Load on power up setting to "Disable".

Send Image Files Through Email

The next step is to retrieve the image files from the microSD card and send them to your customer through email.

1. Remove the microSD card from the controller and read the card using your computer.
2. Navigate to the location where the image files are stored (default is <Micro800>\USERPRJ).
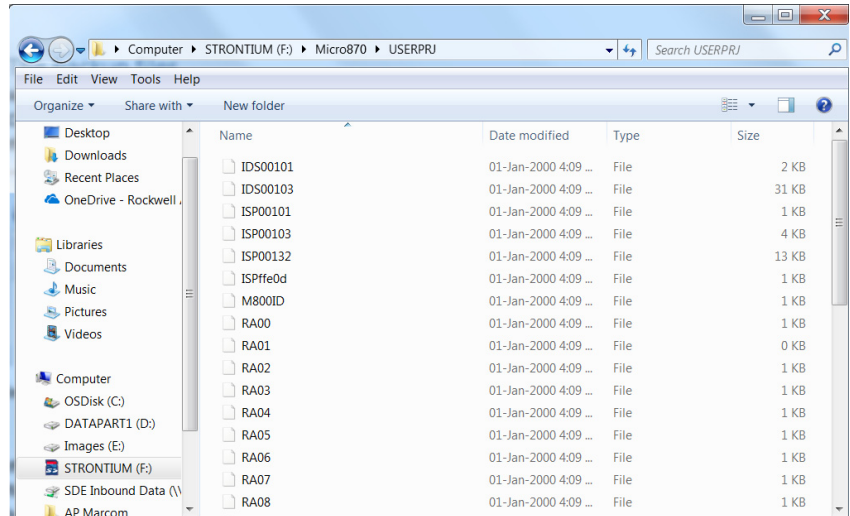
3. Use a compression program to zip these image files and send them to your customer through email.

The customer must extract these image files into the root directory of their microSD card and verify that the location is identical to the original (default is <Micro800>\USERPRJ).

Restore Project from Backup

The last step is to restore the project to your controller from the microSD card. There are two methods to restore the backup, depending on the configuration of the controller.

Existing Controller - Load Always / Load on Error

For this example, the Load on power up setting was configured to "Load Always". This means that the controller loads the project from the memory card whenever it is powered on.

1. Insert the microSD card into the microSD card slot.
2. Cycle power to the controller.
3. When the SD LED displays a steady green light, the project restore is complete.

This method is used for an existing controller that has been configured and you want to update the program.

New Controller

If your controller is new, you can use the ConfigMeFirst.txt file to restore the project backup.

Figure 58 - Example Configuration for Project Restore
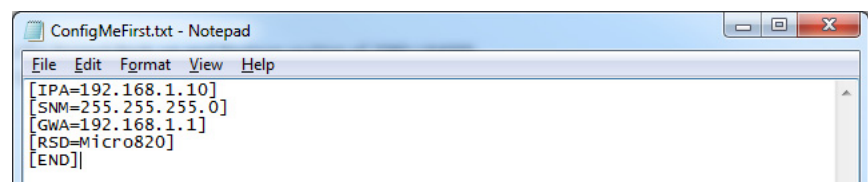
In the example shown above, the ConfigMeFirst.txt file configures the IP address, subnet mask, and gateway of the controller, and restores the project from the location that is specified on the microSD card.

The ConfigMeFirst.txt file must be placed in the same root directory as the backup folder in the microSD card.
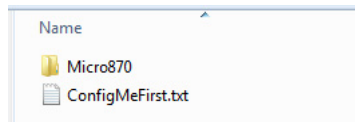
1. Insert the microSD card into the microSD card slot.
2. Cycle power to the controller.
3. When the SD LED displays a steady green light, the project restore is complete.

Data Log

The data logging feature allows you to capture global and local variables with time stamp from the Micro800 controller into the microSD card. You can retrieve the recorded datasets on the microSD card by reading the contents of the microSD card through a card reader or by doing an upload through the Connected Components Workbench software.

A maximum number of 10 datasets are supported for a Micro800 program. Each dataset can contain up to 128 variables, with a maximum of four data string variables per dataset. String variables can have a maximum of 252 characters. All datasets are written to the same file. For more information on how data logs are stored on the microSD card, see the Data Log Directory Structure on page 241.

You can retrieve data log files from the microSD card using a card reader or by uploading the data logs through the Connected Components Workbench software.

IMPORTANT	Uploading data log files in PROGRAM mode is recommended for optimum performance and to prevent file access conflict. For example, if the data log instruction is executing, Connected Components Workbench software does not upload the last data log file.

See the sample quick start project to get you started on the Data Log feature, Use the Data Log Feature on page 248.

IMPORTANT	Data log execution time depends on the user application and its complexity. Users are advised to data log no faster than every 2 seconds for typical applications. Housekeeping takes at least 5 ms per program scan. See Program Execution in Micro800 Controllers on page 125 for more information on program scan and execution rules and sequence. See also Data Log – Data Payload vs. Performance Time on page 244.

IMPORTANT	In cases where there are simultaneous RCP and DLG function block execution or uploads/downloads/searches, the activities are queued up and handled one by one by the program scan. You can observe a slowdown in performance in these cases.

Data Log Directory Structure
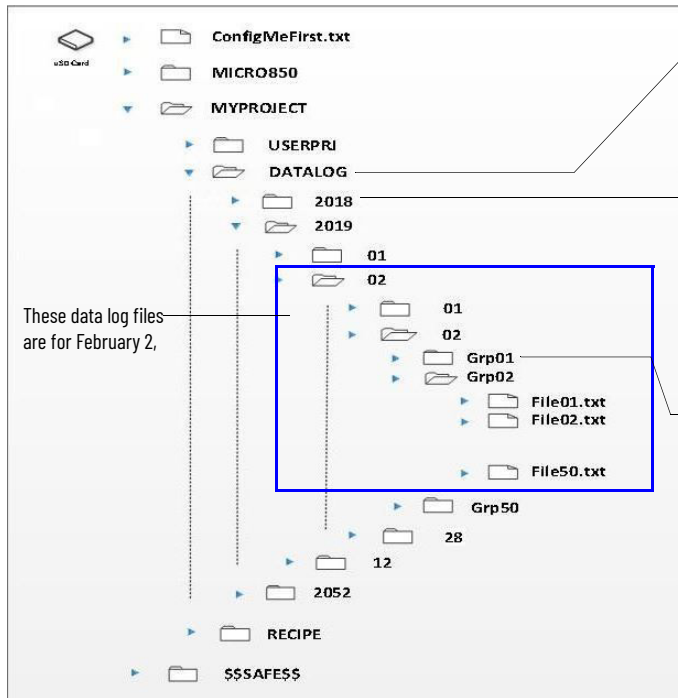
These data log files are for February 2,

The DATALOG folder is created under the current project directory in the microSD card. In this example, the current project directory is MYPROJECT. By default, the current project directory name is taken from the downloaded project's controller name or from the ConfigMeFirst.txt. See ConfigMeFirst.txt Configuration Settings on page 236.

Subdirectories are also created following the controller RTC time stamp. This means that if the RTC date at the time of function block execution is February 02, 2019, the subfolder 2019 is created under DATALOG. Under the 2019 folder, the subfolder 02 (which represents the month of February) is created. Under 02, another subfolder 02 is created, corresponding to the current date.

Under the current working folder, the subfolder Grp01 is created. A maximum of 50 Grpxxx folders can be generated on the microSD card per day.

Under the current Grpxxx working folder, the data log file File01.txt is created. Once this file reaches more than 4 KB, another file, File02.txt, is automatically created to store data. The file size is kept small to minimize data loss in case the card is removed or when there is unexpected power off.

Each Grpxx folder can accommodate up to 50 files. This means that, for example, when the Grp01 folder already stores 50 files, a new folder Grp02 is automatically created to store the next data log files for that day. This automatic folder and file generation goes on until the Grpxx folder reaches 50 for that day.

When a microSD card is inserted, the DLG function block looks for the last Grpxx folder and filexx.txt file, and proceeds to do the data logging based on that information.

Table 101 summarizes data logging performance on Micro800 controllers.

Table 101 - Data Log Specifications

Attribute	Value	
Maximum datasets	10	All datasets are stored in the same file.
Maximum variables per dataset	128	Configured in Connected Components Workbench software.
Minimum size per file	4 KB	—
Maximum files per Grpxx folder[1]	50	When the directory is full, a new directory is automatically created in RUN mode.
Maximum files (Filexx.txt) per day	50	When the file reaches maximum size, a new file is automatically created in RUN mode.
Typical data per day	10 MB	—

(1)   Once the data log limits are reached (that is, 50 Grpxx folders per day, then an error (ErrorID 3: DLG_ERR_DATAFILE_ACCESS) is returned.

Data Log Function (DLG) Block

The data logging function block lets a user program to write run-time global values into the data logging file in microSD card.
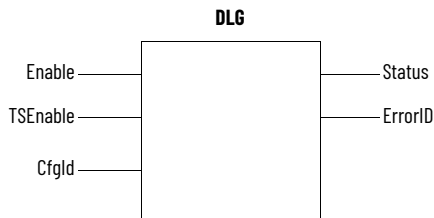
DLG

Enable ———	——— Status
TSEnable ———	——— ErrorID
CfgId ———	

DLG Input and Output Parameters

Parameter	Parameter Type	Data Type	Description
Enable	INPUT	BOOL	Data logging write function enable On the rising edge (that is, the Enable value is triggered from low to high), the function block executes. The precondition for execution is that the last operation has completed.
TSEnable	INPUT	BOOL	Date and time stamp logging enable flag
CfgId	INPUT	USINT	Configured dataset (DSET) number (1...10)
Status	OUTPUT	USINT	Data logging function block status
ErrorID	OUTPUT	UDINT	ErrorID if DLG Write fails.

DLG Function Block Status

Status Code	Description
0	Data logging IDLE status
1	Data logging BUSY status
2	Data logging COMPLETE SUCCEED status
3	Data logging COMPLETE ERROR status

DLG Function Block Errors

Status Code	Name	Description
0	DLG_ERR_NONE	No error.
1	DLG_ERR_NO_SDCARD	microSD card is missing.
2	DLG_ERR_RESERVED	Reserved.
3	DLG_ERR_DATAFILE_ACCESS	Error accessing data log file in microSD card.
4	DLG_ERR_CFG_ABSENT	Data log configuration file is absent.
5	DLG_ERR_CFG_ID	Configuration ID is missing in data log configuration file.
6	DLG_ERR_RESOURCE_BUSY	Same Configuration ID is used with other data log function block call simultaneously
7	DLG_ERR_CFG_FORMAT	Data log configuration file format is wrong.
8	DLG_ERR_RTC	Real-time clock is invalid.
9	DLG_ERR_UNKNOWN	Unspecified error has occurred.

> **IMPORTANT** File access error is returned during DLG function block execution when the card is full.

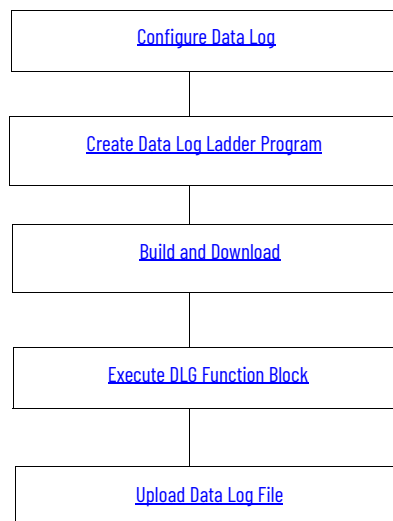**Figure 59 – Data Log Function Block Timing Diagram**
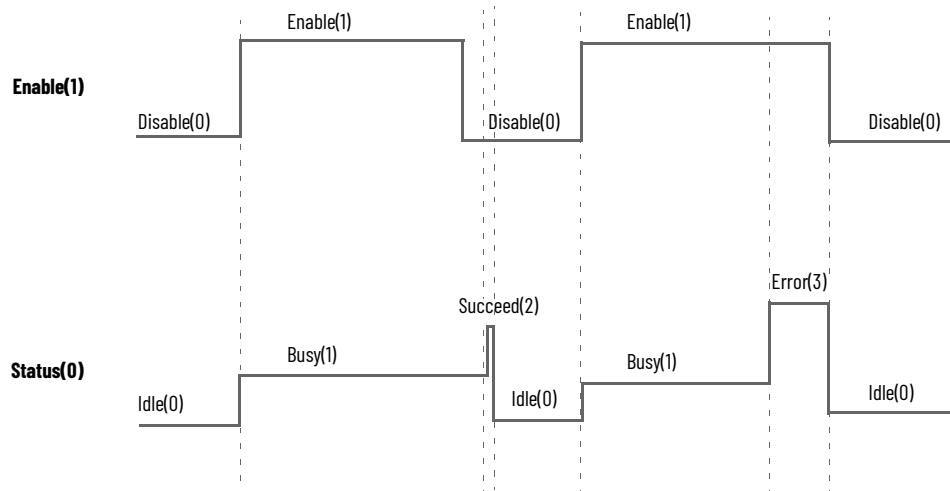
IMPORTANT	**Data Log Function Block Execution**
	• There are three possible states for the Data Log function block: Idle, Busy, and Complete (which includes Complete with Succeed and Complete with Error).
	• For one Data Log function block execution, the typical status starts from Idle, then Busy and finishes with Complete. To trigger another function block execution, the status must return to Idle first.
	• Idle status changes to Busy status only when Enable input signal is in the rising edge. Complete status enters Idle status when Enable input signal is Disable status only.
	• TSEnable and CfgId input parameters are only sampled at the Enable input parameter's rising edge when a new function block execution starts. During function block execution, the input parameters of TSEnable and CfgId are locked and any changes are ignored.
	• When execution completes, the status changes from Busy to Complete. At this stage, if the input Enable is False, status changes to Idle after indicating Complete for exactly one scan time. Otherwise function block status is kept as Complete until input Enable changes to False.
	• The data log file can only be created by the DLG instruction block. The Connected Components Workbench software can only upload and delete the data log file.
	• There are separators in between every data variable in the data file that is defined during configuration in the Connected Components Workbench software. See <u>Supported Data Types for Data Log and Recipe Function Blocks on page 243</u>.
	• Data variable values are sampled when the data logging function block is in Busy state. However, the data logging file is only created when the data logging function block is in the Complete state.

Table 102 – Supported Data Types for Data Log and Recipe Function Blocks

Data Type	Description	Example Format in Output Data Log File
BOOL[1]	Logical Boolean with values TRUE and FALSE	0: FALSE 1: TRUE)
SINT	Signed 8-bit integer value	–128, 127
INT	Signed 16-bit integer value	–32768, 32767
DINT	Signed 32-bit integer value	–2147483648, 2147483647
LINT	Signed 64-bit integer value	–9223372036854775808, 9223372036854775807
USINT(BYTE)	Unsigned 8-bit integer value	0, 255
UINT(WORD)	Unsigned 16-bit integer value	0, 65535

Table 102 - Supported Data Types for Data Log and Recipe Function Blocks (Continued)

Data Type	Description	Example Format in Output Data Log File
UDINT(DWORD)	Unsigned 32-bit integer value	0, 4294967295
ULINT(LWORD)	Unsigned 64-bit integer value	0, 18446744073709551615
REAL	32-bit floating point value	–3.40282347E+38, +3.40282347E+38
LREAL	64-bit floating point value	–1.7976931348623157E+308, +1.7976931348623157E+308
STRING[2]	character string (1 byte per character)	"Rotation Speed"
DATE[1]	Unsigned 32-bit integer value	1234567 (Date variables are stored as 32-bit words, a positive number of seconds beginning at 1970-01-01 at midnight GMT.)
TIME[1]	Unsigned 32-bit integer value	1234567 (Time variables are stored as 32-bit words, positive number of milliseconds).

(1)   BOOL, DATE, TIME data variables are presented in decimal digital format in the microSD Card. You can convert this format to a more friendly format. For example, use the ANY_TO_STRING function block to convert BOOL data type (0, 1) to FALSE or TRUE. You can similarly do the same for DATE and TIME data types.
DATE data type is presented in differential decimal digital value between system baseline time (1970/01/01,00:00:00) and current date value. Unit is millisecond.
Time should be absolute time value. Unit is second.
(2)   String data variables are enclosed in double quotation marks in the data log file.
The example below shows DSET1 using string variables and DSET2 using integers.

```
DSET1,"Temperature", "Humidity", "Pressure"
DSET2, 30, 50, 125
```

Data Log Performance

Data Log – Data Payload vs. Performance Time

Parameter	Number of Characters					
	28	502	518	1028	1493	3676
Average write time per data log file including all overheads	541.75 ms	1043.75 ms	1086.67 ms	1632.36 ms	1972.9 ms	2696.22 ms
Average write time excluding first sample	500.40 ms	963.86 ms	999.14 ms	1472.37 ms	1818.33 ms	2545.92 ms
Average write time excluding all overheads	479.10 ms	493.03 ms	502.78 ms	505.54 ms	519.91 ms	715.68 ms

Time (msec)

Time (msec)
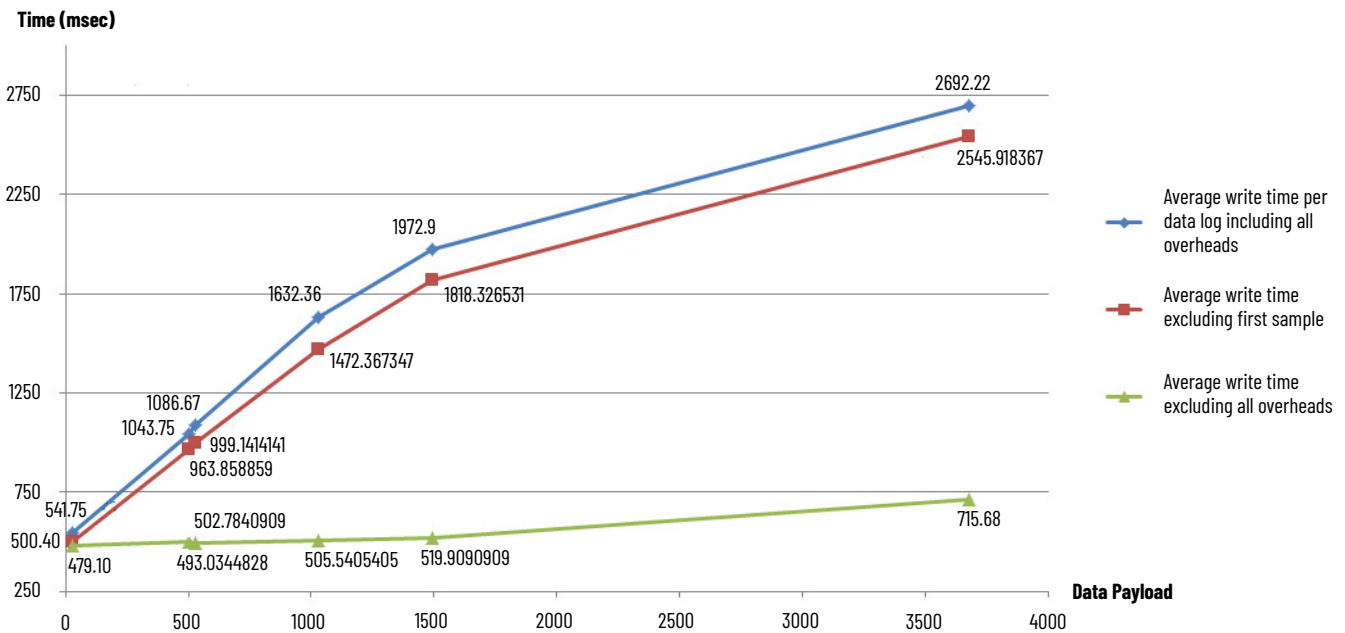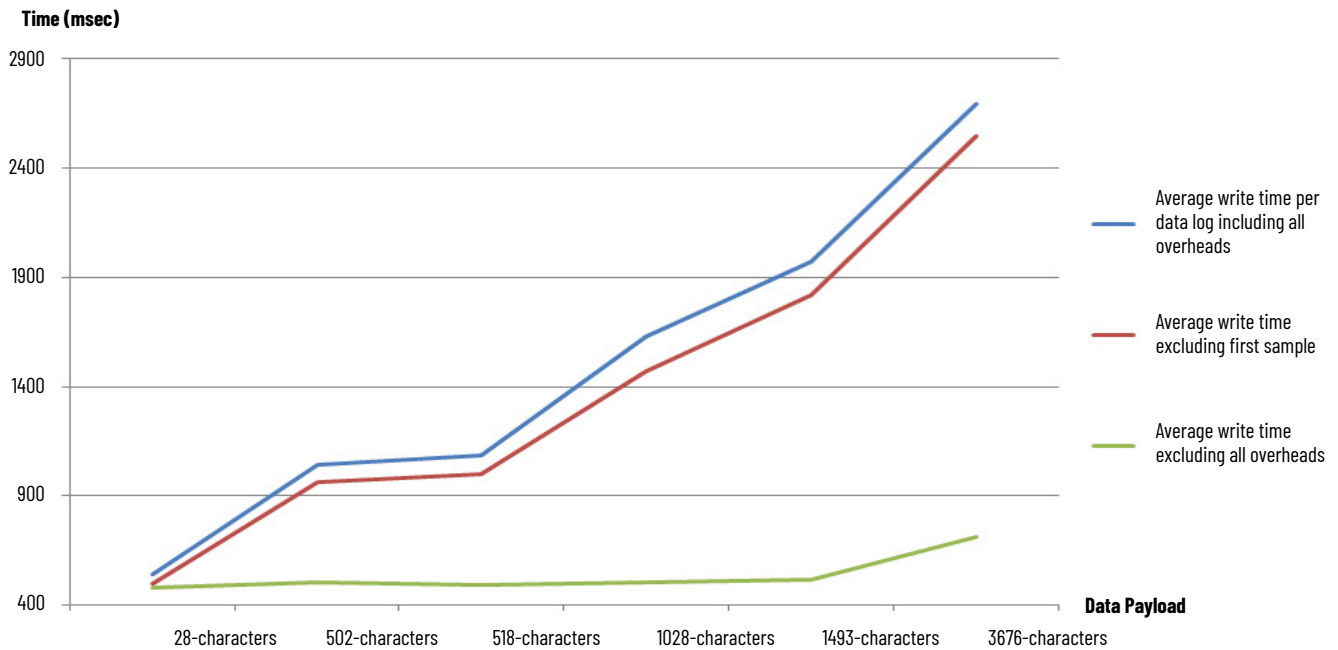
Recipe

Micro800 controllers support the Recipe feature and allows you to store and load a list of data to and/or from recipe data files using the RCP instruction. It also allows you to download, upload, and delete Recipe data on the microSD card through the Connected Components Workbench software.

A maximum number of 10 recipe sets are supported for a Micro800 program. Each recipe can contain up to 128 variables, with a maximum of four data string variables per recipe. String variables can have a maximum of 252 characters. Variations of the recipe are stored in separate files with unique file names. For more information on how recipes are stored on the microSD card, see the .

Table 103 - Recipe Specifications

Attribute	Value	
Maximum number of recipe sets	10	Recipe sets are stored in 10 directories (Rcp_Id01...Rcp_Id10) with a maximum number of 50 recipe files in each directory.
Maximum number of recipes in each set	50	
Maximum number of variables per recipe	128	Configured in Connected Components Workbench software.
Maximum bytes per recipe file	4 KB	

Recipe Directory Structure
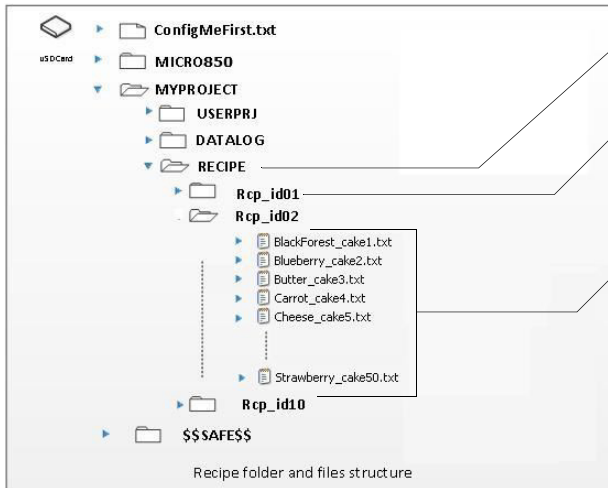
On first execution of RCP, it creates the RECIPE folder under the current project directory on the microSD card.

It also creates 10 subdirectories for each recipe set with a name following the CfgID input value (1...10). If the CfgID value is 1, then the subfolder Rcp_Id01 is created.

Recipe files are then created/written into the folder, with file names that correspond to the input value of the RcpName parameter for the RCP function block, as configured in the Connected Components Workbench software. Each Recipe set can contain up to 50 recipe files or variations. Filenames for recipe files should not exceed 30 characters.

Recipe folder and files structure

Recipe Configuration and Retrieval

You can retrieve recipe files from the microSD card using a card reader or by uploading and downloading the recipe sets through the Connected Components Workbench software.

Recipe Function (RCP) Block

The RCP function block allows a user program to read variable values from an existing recipe data file that is in the recipe folder of the microSD card and update run-time global or local variable values in the controller. The RCP function block also allows the user program to write run-time global or local variable values from the smaller controller into the recipe data file in the microSD card.
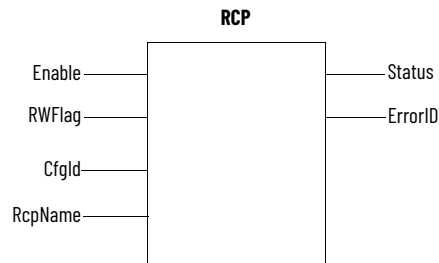
Table 104 - RCP Input and Output Parameters

Parameter	Parameter Type	Data Type	Description
Enable	INPUT	BOOL	Recipe read/write function enable. If Rising Edge (Enable is triggered from "low" to "high"), starts the recipe function block and the precondition is that the last operation is completed.
RWFlag	INPUT	BOOL	TRUE: Recipe writes data variables to the recipe files into the microSD card. FALSE: Recipe reads saved data variables from the microSD card and update these variables accordingly.
CfgId	INPUT	USINT	Recipe set number (1...10)
RcpName	INPUT	STRING	Recipe data filename (maximum 30 characters)
Status	OUTPUT	USINT	Current state of Recipe function block
ErrorID	OUTPUT	UDINT	Detailed ErrorID information if RCP read/write fails.

Table 105 - RCP Function Block Status

Status Code	Description
0	Recipe Idle status
1	Recipe Busy status
2	Recipe Complete Succeed status
3	Recipe Complete Error status

Table 106 - RCP Function Block Errors

ErrorID	Error Name	Description
0	RCP_ERR_NONE	No error
1	RCP_ERR_NO_SDCARD	microSD card is absent.
2	RCP_ERR_DATAFILE_FULL	Recipe files exceed the maximum number of files per recipe set folder.
3	RCP_ERR_DATAFILE_ACCESS	Error to access recipe data file in microSD card
4	RCP_ERR_CFG_ABSENT	Recipe configuration file is absent.
5	RCP_ERR_CFG_ID	Configure ID is absent in the recipe configuration file.
6	RCP_ERR_RESOURCE_BUSY	The Recipe operation resource linked to this Recipe ID is used by another function block operation.
7	RCP_ERR_CFG_FORMAT	Recipe configuration file format is invalid.
8	RCP_ERR_RESERVED	Reserved
9	RCP_ERR_UNKNOWN	Unspecified error has occurred.
10	RCP_ERR_DATAFILE_NAME	Recipe data file name is invalid.
11	RCP_ERR_DATAFOLDER_INVALID	Recipe dataset folder is invalid.
12	RCP_ERR_DATAFILE_ABSENT	Recipe data file is absent.
13	RCP_ERR_DATAFILE_FORMAT	Recipe data file contents are wrong.
14	RCP_ERR_DATAFILE_SIZE	Recipe data file size is too large (>4K).

> **IMPORTANT**   File access error is returned during RCP function block execution when the card is full.

Figure 60 - Recipe Function Block Timing Diagram
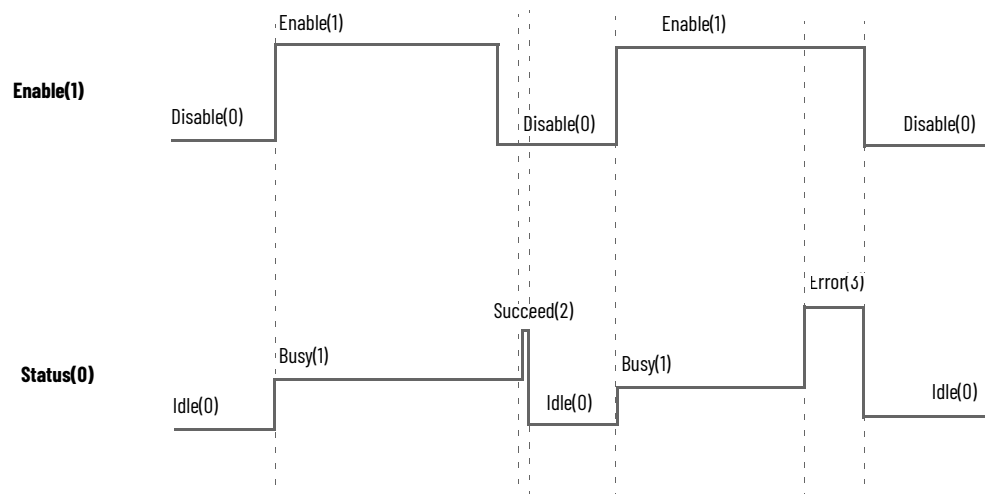
IMPORTANT	RCP Function Block Execution
	• There are three possible states for the Recipe function block: Idle, Busy, Complete (Complete with Succeed and Complete with Error).
	• For one Recipe function block execution, the typical status starts from Idle then Busy and finishes with Complete. To trigger another function block execution, the status must return to Idle first.
	• Idle status changes to Busy status only when Enable input signal is in the rising edge. Complete status enters Idle status when Enable input signal is on Disable status.
	• RWFlag, CfgId, and RcpName input parameters are only sampled at the Enable input parameter's rising edge when a new function block execution starts. During function block execution, the input parameters of RWFlag, CfgId, and RcpName are locked and any changes are ignored.
	• When the function block execution finishes, the function block status changes from Busy to Complete. At this stage, if input Enable is False, the function block status changes to Idle after staying as Complete for exactly one scan time. Otherwise, function block status remains Complete until input Enable changes to False.
	• Recipe function block file name supports a maximum of 30 bytes in length, and only supports upper and lower case letters Aa...Zz, numbers 0...9 and underscore (_).
	• The RcpName input parameter does not allow file extensions (for example, .txt) to be added to its value. The recipe data file is written to the microSD card with the .txt extension.
	• There are separators in between every data variable in the recipe data file that is defined during configuration in the Connected Components Workbench software. Redundant tab, space, carriage return, and line feed characters are strictly not allowed. See Supported Data Types for Data Log and Recipe Function Blocks on page 243.
	• Double quotes are not allowed within a string in a recipe file.

Quick Start Projects for Data Log and Recipe Function Blocks

The following sample quick start projects provide step-by-step instructions on how to use the Data Log and Recipe function blocks in the Connected Components Workbench software to generate and manage your recipe files and data logs.

Use the Data Log Feature

Configure Data Log

Create Data Log Ladder Program

Build and Download

Execute DLG Function Block

Upload Data Log File

Configure Data Log

1. In the Connected Components Workbench software, go to the Properties pane to configure your data log.

2. Select Data Log. Click Add Data Set to add a dataset. Each dataset will be stored in the same file. You can add up to 10 datasets per configuration.

3. Click Add Variable to add variables to the dataset. You can add up to 128 variables to each dataset.
   For this quick start sample project, add the following variables that you have previously created to Dataset 1.

Local Variables

Variable Name	Data Type
data_bool	BOOL
data_int8	INT
data_string	STRING
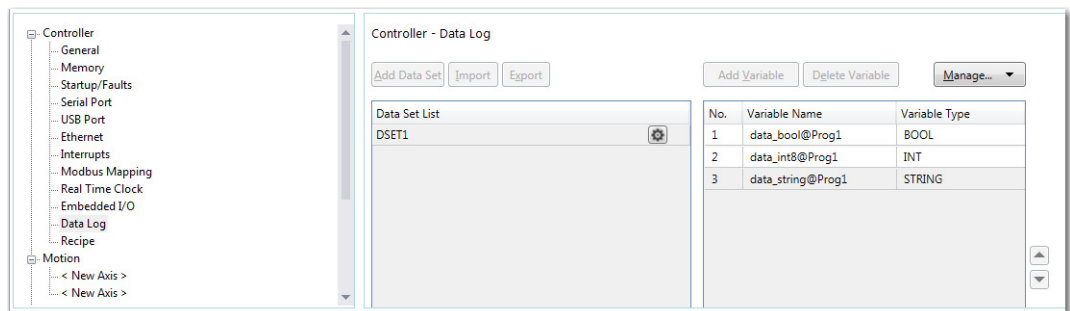
Create Data Log Ladder Program
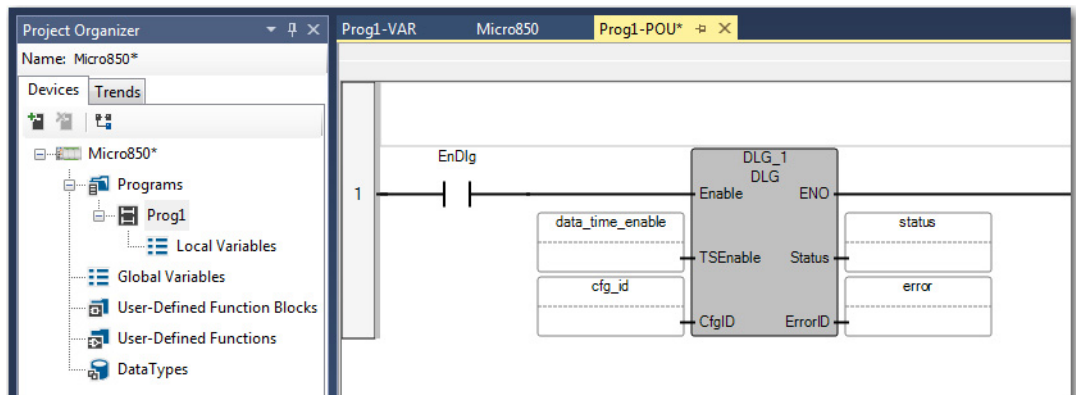
1. Launch the Connected Components Workbench software. Create a user program for your Micro800 controller.

2. Right-click Programs. Select Add New LD: Ladder Diagram. Name the Program (for example, Prog1).
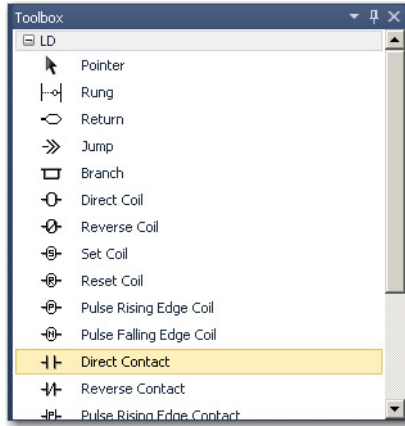
3.   From the Toolbox, double-click Direct Contact to add it to the rung.

4.   From the Toolbox, double-click Block to add it to the rung.

5.   On the Block Selector window that appears, type DLG to filter the DLG function block from the list of available function blocks. Select OK.
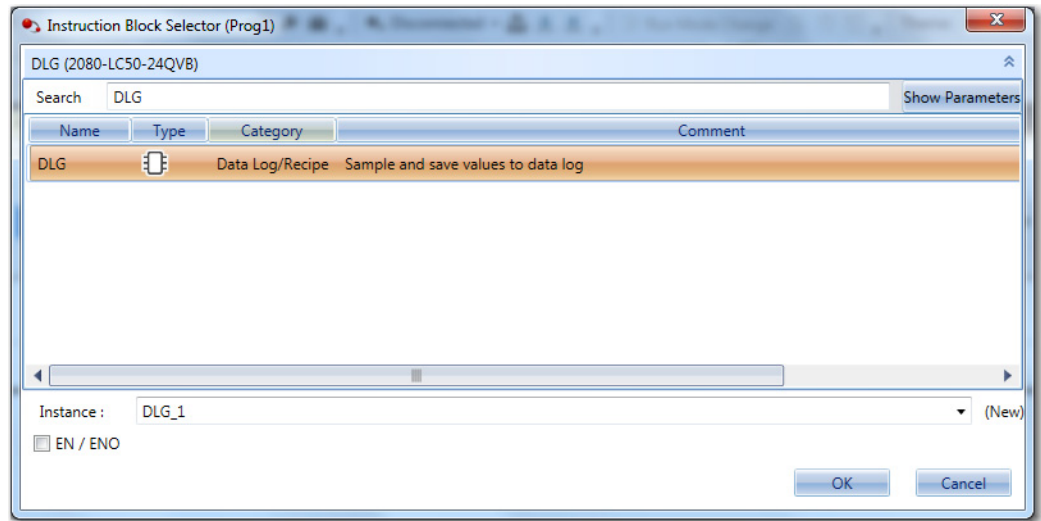
6.   Create the following local variables for your project.
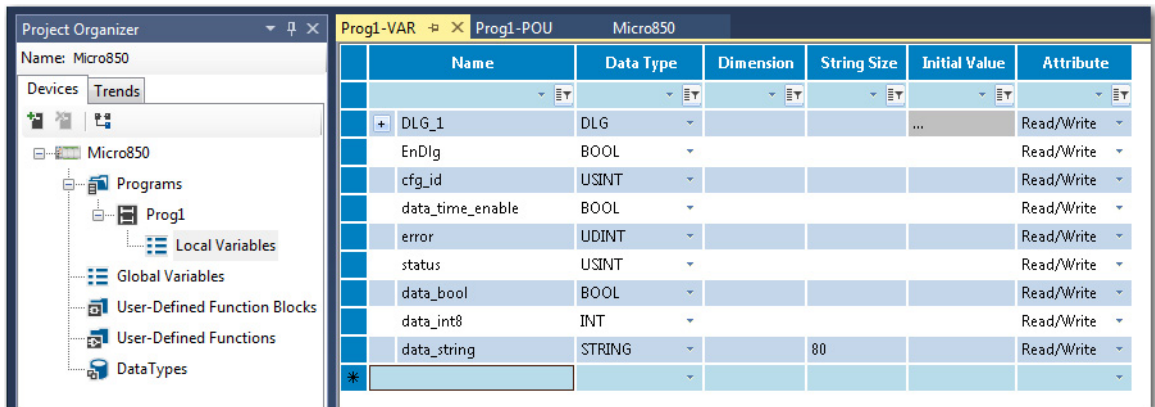
Local Variables

Variable Name	Data Type
EnDlg	BOOL
cfg_id	USINT
data_time_enable	BOOL
error	UDINT
status	USINT
data_bool	BOOL
data_int8	INT
data_string	STRING

7. Assign the variables to the DLG input and output parameters as follows:
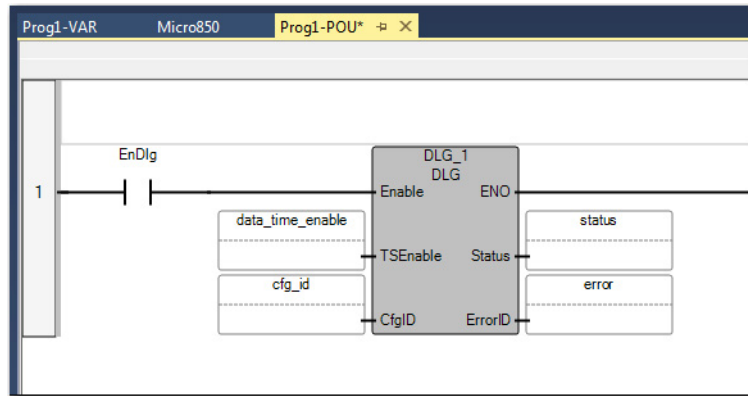
For the CfgID input parameter, you can choose a predefined variable by choosing from the Defined Words in the Connected Components Workbench software. To do so, click the CfgID input box. From the Variable Selector window that appears, click the Defined Words tab and choose from the list of defined words. For example, DSET1 that corresponds to DSET1 in your recipe configuration. See Figure 61.

Figure 61 - Choose a Predefined Variable
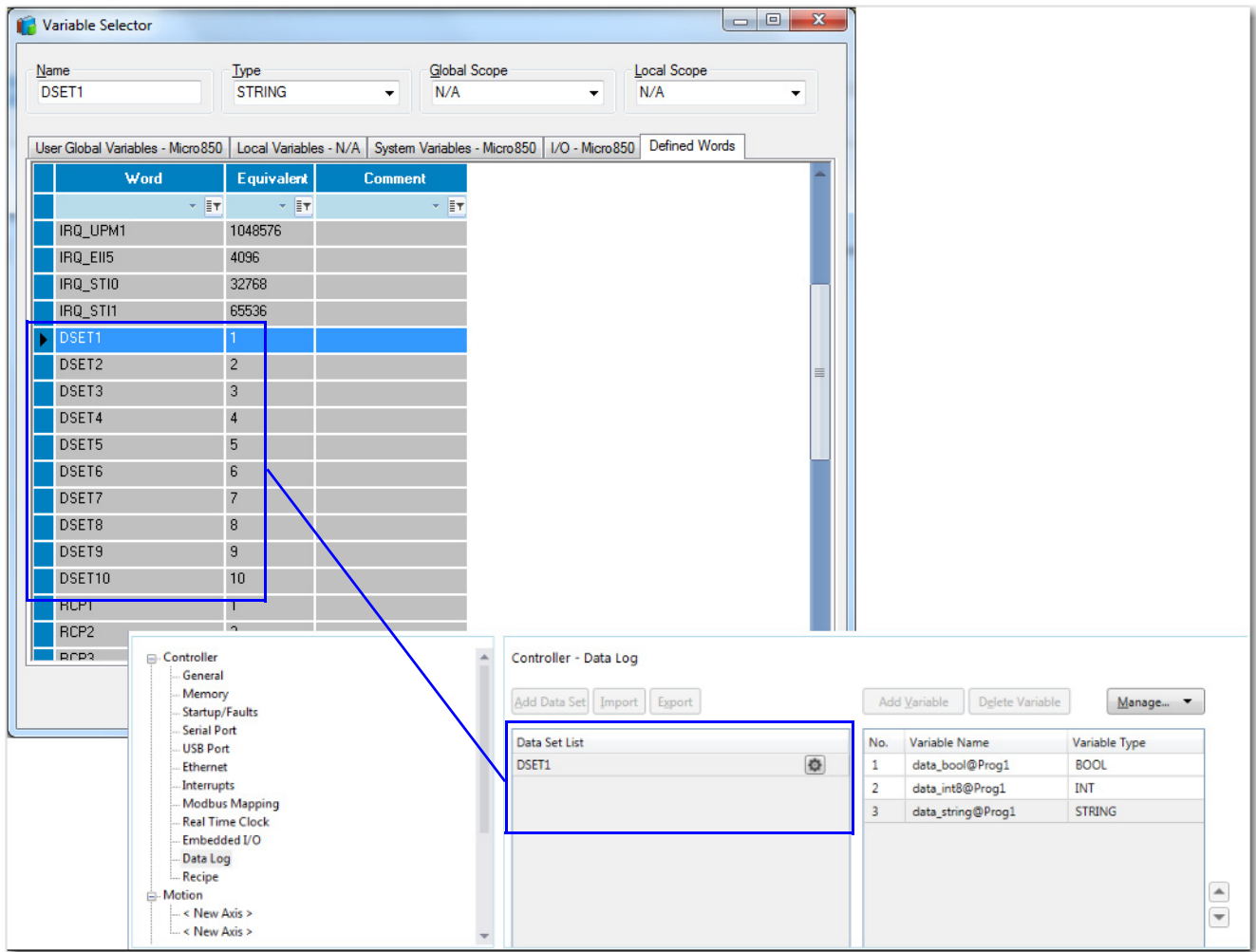
Build and Download

After configuring data log properties, build the program and download to the controller.

Execute DLG Function Block

Execute the DLG function block. Notice the Status output go from 0 (Idle) to 1(Enable), and 2 (Succeed).
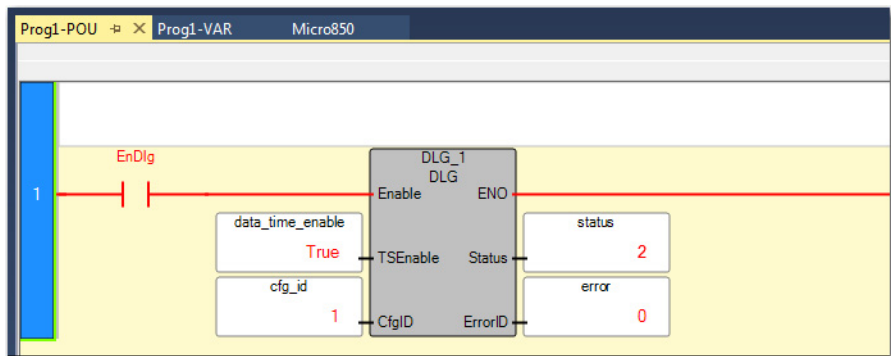
Upload Data Log File

You can retrieve data log files from the microSD card using a card reader or by uploading the data logs through the Connected Components Workbench software.

1. To use the Upload feature, go to the Properties section of your project in the Connected Components Workbench software.

2. Select Data Log. Click Manage and then choose Upload.
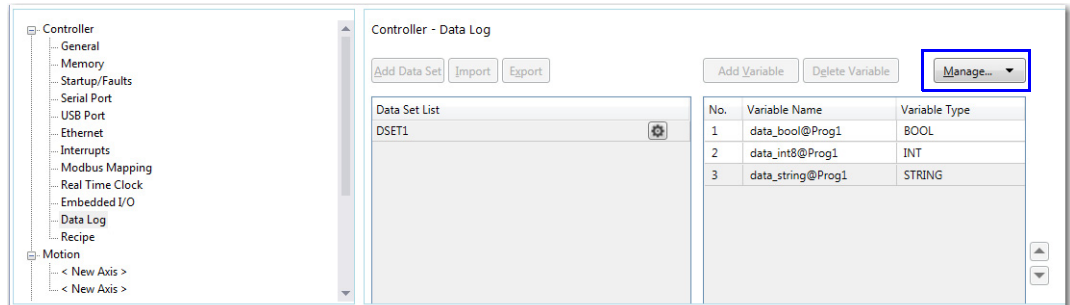
IMPORTANT	The Manage button is not available in DEBUG mode. You must stop DEBUG mode to use the Manage button to upload data log files. Uploading data log files in PROGRAM mode is recommended for performance and file locking reasons.

3. From the Upload window that appears, select the date of the data log files that you would like to upload. You can upload data logs for the entire month by selecting the Whole Month checkbox
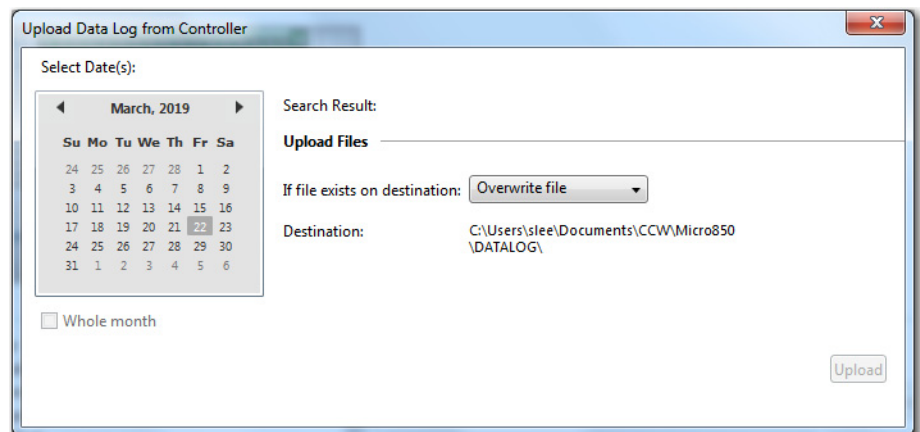
4. If the file exists in your destination folder, select whether you would like to Overwrite file, Skip file, or Preserve both files.

5. Select Upload. The progress bar tells you whether the upload is successful or not.

IMPORTANT	Do not take out the microSD card from the slot while data is being written or retrieved from the card. Ongoing write and retrieval operations are indicated by a flashing SD status LED.

IMPORTANT	For better data log file management, you can use a third-party tool or DOS CMD to merge all your data log files into one file and import as a CSV file in Excel®.

Use the Recipe Feature


```
┌─────────────────────────────┐
│     Configure Recipe         │
└─────────────────────────────┘
              │
┌─────────────────────────────┐
│  Create Recipe Ladder Program│
└─────────────────────────────┘
              │
┌─────────────────────────────┐
│     Build and Download       │
└─────────────────────────────┘
              │
┌─────────────────────────────┐
│   Execute RCP Function Block │
└─────────────────────────────┘
              │
┌─────────────────────────────┐
│     Upload Recipe Files      │
└─────────────────────────────┘
```
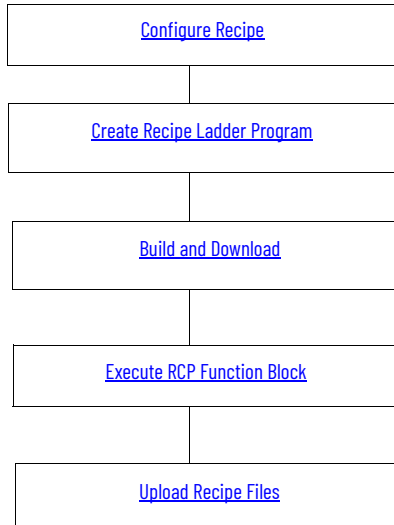
Configure Recipe

1. In the Connected Components Workbench software, go to the Properties pane to configure Recipe.

2. Select Recipe. Select Add Recipe to add a recipe. Each recipe is stored in separate files. You can add up to 10 recipes per configuration.

3. Select Add Variable to add variables to the recipe. You can add up to 128 variables to each recipe.
For this quick start sample project, add the following variables that you have previously created to RCP 1:

Local Variables

Variable Name	Data Type
data_bool	BOOL
data_int8	INT
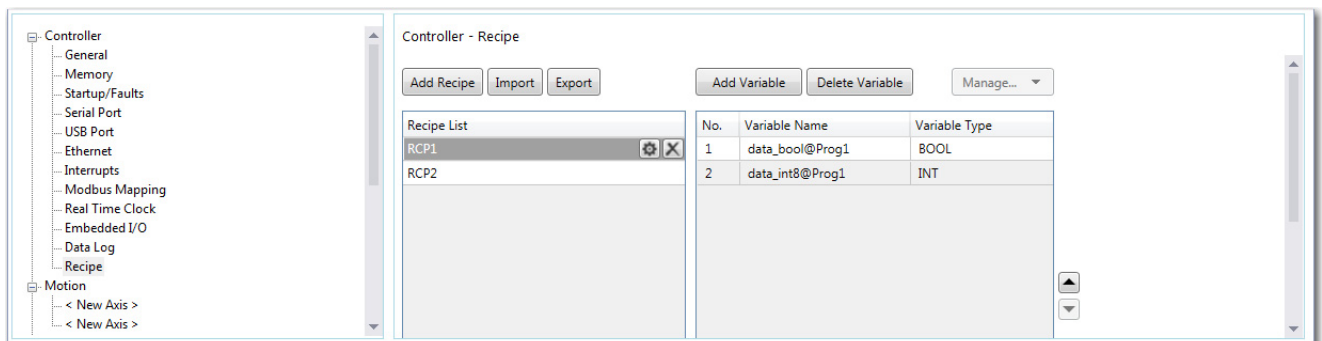
Create Recipe Ladder Program
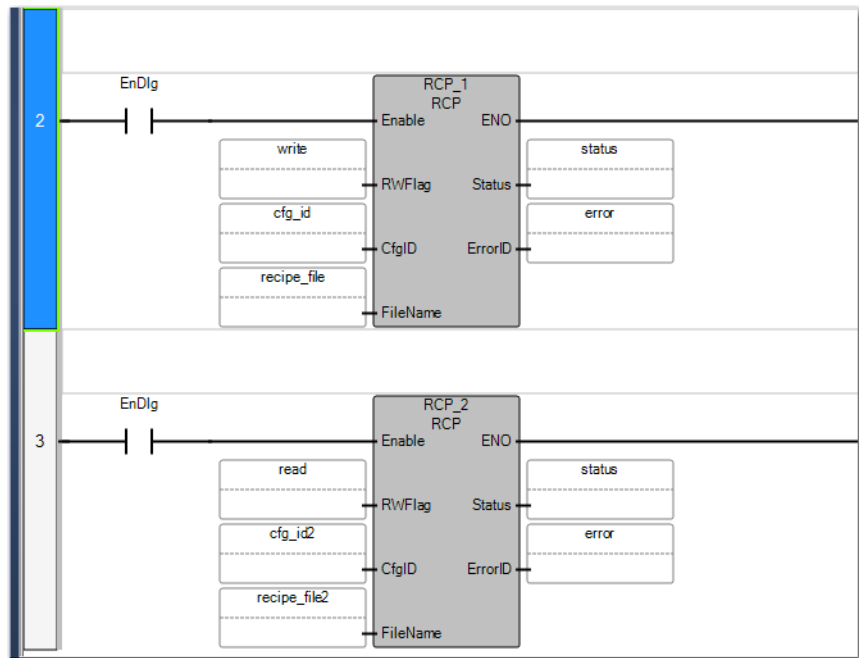
1. Launch the Connected Components Workbench software. Create a user program for your Micro800 controller.

2. Right-click Programs. Select Add New LD: Ladder Diagram. Name the Program (for example, Prog2).

3. From the Toolbox, double-click Direct Contact to add it to the first rung.
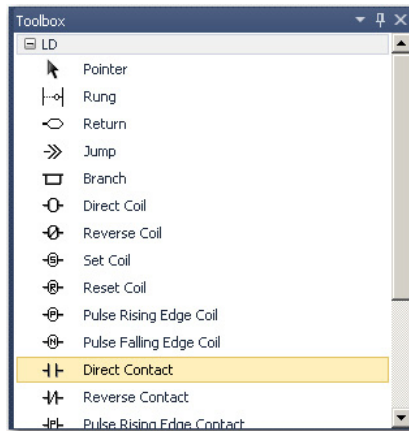
4. From the Toolbox, double-click Block to add it to the rung.

5. On the Block Selector window that appears, type RCP to filter the Recipe function block from the list of available function blocks. Select OK.
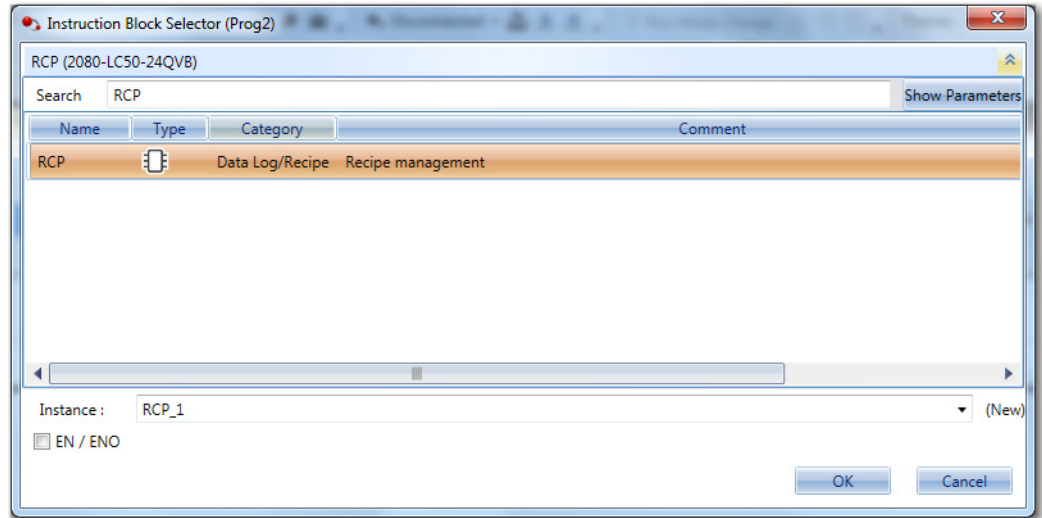
6. From the Toolbox, double-click rung to add another rung.
7. Add a Direct Contact and RCP function block to this second rung by following steps 3...5.
8. Create the following local variables for your program, in addition to the ones that you have already created for data log.

Local Variables

Variable Name	Data Type
recipe_file	STRING
recipe_file2	STRING
cfg_id2	USINT
read	BOOL
write	BOOL

9. Assign the variables to the RCP input and output parameters as follows:
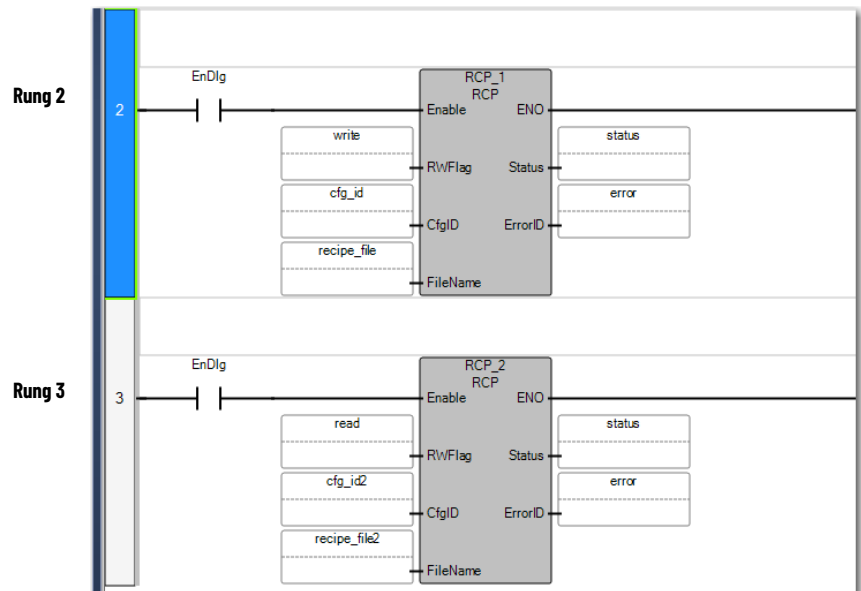
For the CfgID input parameter, you can choose a predefined variable by choosing from the Defined Words in the Connected Components Workbench software. To do so, select the CfgID input box. From the Variable Selector window that appears, select the Defined Words tab and choose from the list of defined words. For example, RCP1 that corresponds to RCP1 in your recipe configuration. See Figure 62.

Figure 62 - Choose a Predefined Variable
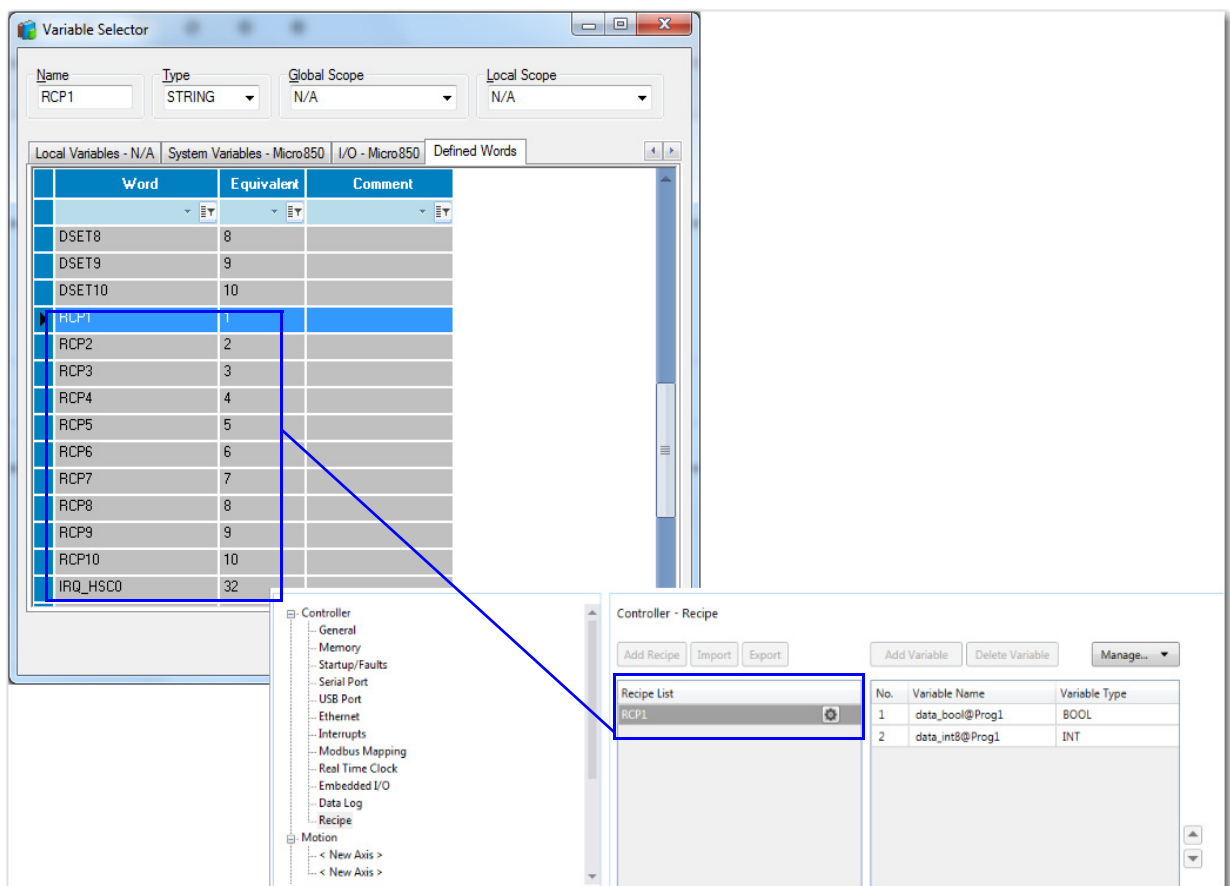
Build and Download

After configuring Recipe, build the program and download to the controller.

Execute RCP Function Block

Execute the RCP function block. Notice the Status output go from 0 (Idle) to
1 (Enable), and 2 (Succeed).
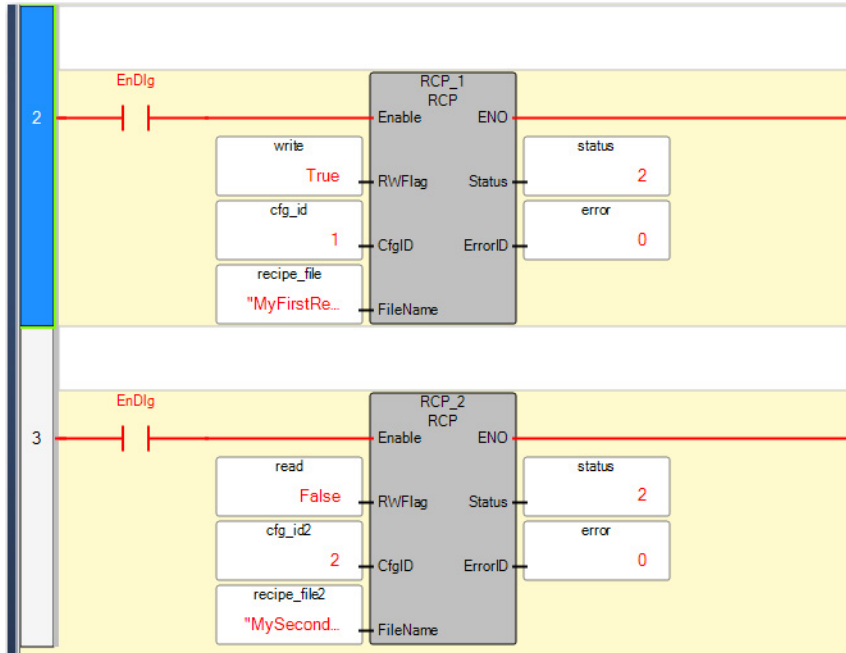
Upload Recipe Files

You can retrieve recipe files from the microSD card using a card reader or by uploading the
recipe files through the Connected Components Workbench software.

1. To use the Upload feature, go to the Properties section of your project in the Connected
   Components Workbench software.
2. Select Recipe. Select Manage and then select Upload.
   Through Manage, you can also choose to Download and Delete recipe files.
3. From the Upload window that appears, select the batch of recipe files that you would
   like to upload.
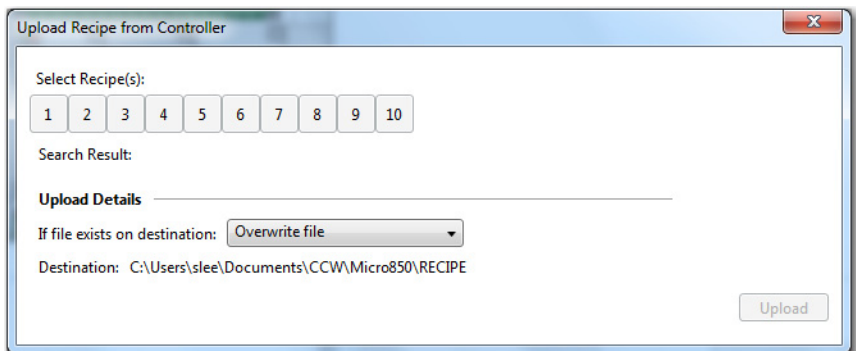
4. If the file exists in your destination folder, select whether you would like to Overwrite
   file, Skip file, or Preserve both Files.

5. Select Upload. The progress bar tell you whether the upload is successful or not.

IMPORTANT	Do not remove the microSD card from the slot while data is being written or retrieved from the card. Ongoing write and retrieval operations are indicated by a flashing SD status LED.

A recipe header file is saved with the uploaded recipes.

Documents library
Recipe

Name ▲	Date modified	Type	Size
📁 Rcp_id01	06/11/2013 2:40 PM	File folder	
📄 recipe_header.txt	06/11/2013 2:42 PM	Text Document	1 KB

recipe_header.txt - Notepad
File Edit Format View Help
RCP1,data_string@Prog1,data_bool@Prog1
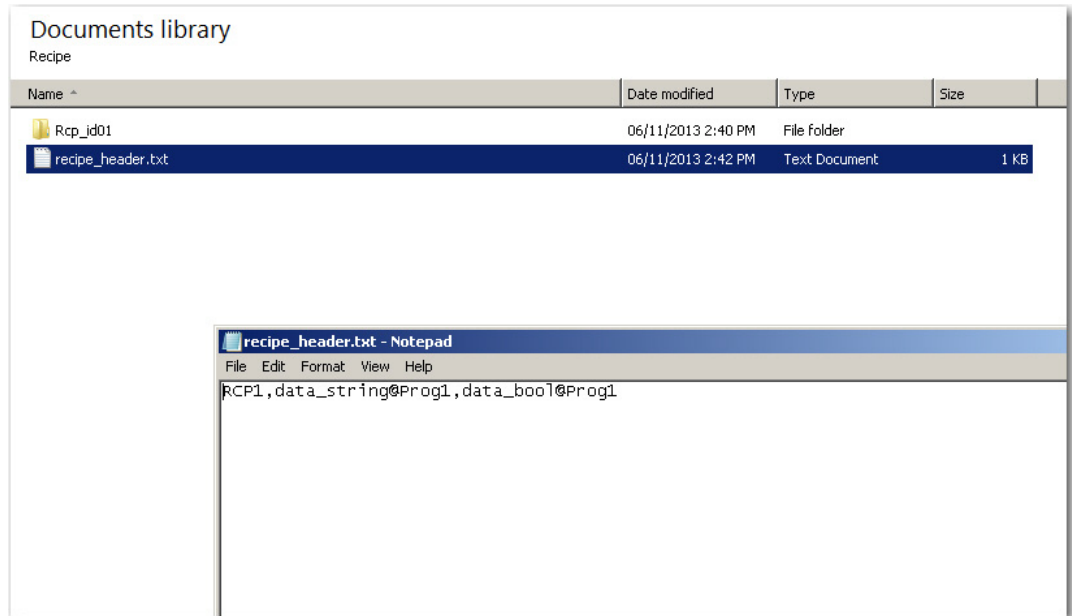
Notes:

Modbus Mapping for Micro800 Controllers

Modbus Mapping

Micro830, Micro850, and Micro870 controllers support Modbus RTU over a serial port through the embedded, non-isolated serial port. The 2080-SERIALISOL isolated serial port plug-in module also supports Modbus RTU. Both Modbus RTU master and slave are supported. Although performance may be affected by the program scan time, the 48-point controllers can support up to six serial ports (one embedded and five plug-ins), and so consequently, six separate Modbus networks.

In addition, the Micro850 and Micro870 controller support Modbus TCP Client/Server through the Ethernet port.

Endian Configuration

Modbus protocol is big-endian in that the most significant byte of a 16-bit word is transmitted first. Micro800 controllers are also big-endian, so byte ordering does not have to be reversed. For Micro800 data types larger than 16 bits (for example, DINT, LINT, REAL, LREAL), multiple Modbus addresses may be required but the most significant byte is always the first.

Mapping Address Space and Supported Data Types

Since Micro800 controllers use symbolic variable names instead of physical memory addresses, a mapping from symbolic variable name to physical Modbus addressing is supported in Connected Components Workbench software. For example, InputSensorA is mapped to Modbus address 100001.

By default Micro800 controllers follow the six-digit addressing specified in the latest Modbus specification. For convenience, conceptually the Modbus address is mapped with the following address ranges. The Connected Components Workbench mapping screen follows this convention.

Table 107 - Mapping Table

Variable Data Type	0 - Coils 000001...065536		1 - Discrete Inputs 100001...165536		3 - Input Registers 300001...365536		4 - Holding Registers 400001...465536	
	Supported	Modbus Address Used	Supported	Modbus Address Used	Supported	Modbus Address Used	Supported	Modbus Address Used
BOOL	Y	1	Y	1				
SINT	Y	8	Y	8				
BYTE	Y	8	Y	8				
USINT	Y	8	Y	8				
INT	Y	16	Y	16	Y	1	Y	1
UINT	Y	16	Y	16	Y	1	Y	1
WORD	Y	16	Y	16	Y	1	Y	1
REAL	Y	32	Y	32	Y	2	Y	2
DINT	Y	32	Y	32	Y	2	Y	2
UDINT	Y	32	Y	32	Y	2	Y	2
DWORD	Y	32	Y	32	Y	2	Y	2
LWORD	Y	64	Y	64	Y	4	Y	4

Table 107 - Mapping Table (Continued)

Variable Data Type	0 - Coils 000001...065536		1 - Discrete Inputs 100001...165536		3 - Input Registers 300001...365536		4 - Holding Registers 400001...465536	
	Supported	Modbus Address Used	Supported	Modbus Address Used	Supported	Modbus Address Used	Supported	Modbus Address Used
ULINT	Y	64	Y	64	Y	4	Y	4
LINT	Y	64	Y	64	Y	4	Y	4
LREAL	Y	64	Y	64	Y	4	Y	4

NOTE: Strings are not supported.

To make it easier to map variables to five-digit Modbus addresses, the Connected Components Workbench mapping tool checks the number of characters that are entered for the Modbus Address. If only five-digits are entered, the address is treated as a five-digit Modbus address. This means that the Coils are mapped from 00001...09999, Discrete Inputs are mapped from 10001...19999, Input Registers are mapped from 30001...39999, and Holding Registers are mapping from 40001...49999.

Example 1, PanelView 800 HMI (Master) to Micro800 (Slave)

The embedded serial port is targeted for use with HMIs using Modbus RTU. The maximum recommended cable distance is 3 meters (10 feet). Use the 2080-SERIALISOL serial port plug-in module if longer distances or more noise immunity is needed.

The HMI is typically configured for Master and the Micro800 embedded serial port is configured for Slave.

From the default Communications Settings for a PanelView 800 HMI (PV800), there are three items that must be checked or modified to configure communications from PV800 to Micro800.

1. Change the protocol from DF1 to Modbus.
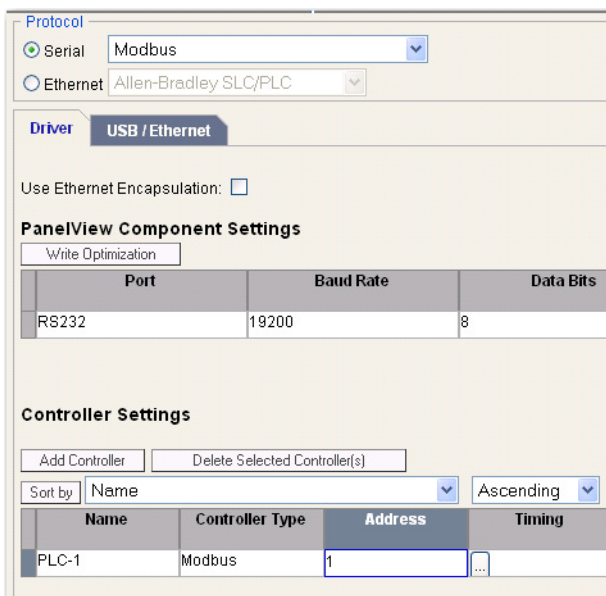
2. Set the Address of Micro800 slave to match the serial port configuration for the controller.
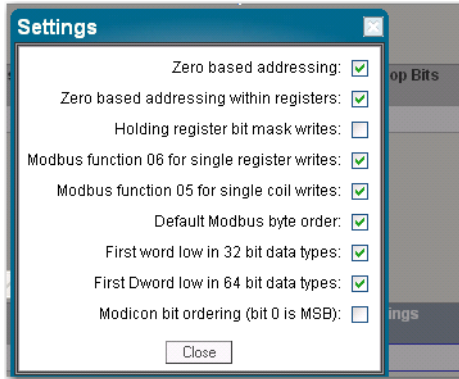
3. Deactivate Tags on Error. This is to prevent the requirement of power cycling PV800 when new Modbus Mappings are downloaded from the Connected Components Workbench software to the Micro800 controller.
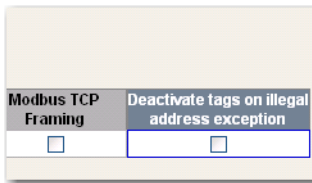
Example 2, Micro800 (Master) to PowerFlex 4M Drive (Slave)

The following is the overview of the steps to be taken for configuring a PowerFlex 4M drive. Parameter numbers that are listed in this section are for a PowerFlex 4M and are different if you are using another PowerFlex 4-class drive.

Table 108 - Parameters in PowerFlex 4-class Drives

Parameter Name	Parameter Number						
	4M	4	40	40P	400	400N	400P
Start Source	P106	P36					
Speed Reference	P108	P38					
Comm Data Rate	C302	A103			C103		
Comm Node Addr	C303	A104			C104		
Comm Loss Action	C304	A105			C105		
Comm Loss Time	C305	A106			C106		
Comm Format	C306	A107			C102		

- Connect the 1203-USB to the PowerFlex drive and to the computer.
- Launch the Connected Components Workbench software, then connect to the drive and set the parameters.

To configure the PowerFlex 4M drive, perform the following steps:

1. Double-click the PowerFlex 4M drive if it is not already open in the Connected Components Workbench software.
2. Select Connect.
3. In the Connection Browser, expand the AB_DF1 DH+™ Driver.
   Select the AB DSI (PF4 Port) and select OK.

4. Once the Drive has connected and been read in, select the Start up wizard and change the following items. Select Finish to save the changes to the drive.

    -   Select the Comm Port as the Speed Reference. Set P108 [Speed Reference] to 5 (Comm Port).
    -   Set Start Source to Comm Port. Set P106 [Start Source] to 5 (Comm Port).
    -   Defaults for the remaining Inputs
    -   Accept Defaults for the remainder and select Finish.

5. Select Parameters from the Connected Components Workbench window.
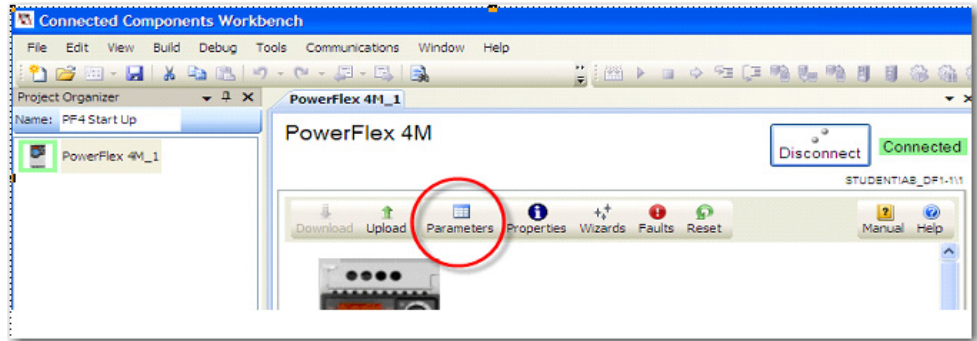
6. The Parameter window opens. Resize it to view the parameters. From this window, you can view and set the data values of Parameters.

7. From the Parameter window, change the Parameters in Table 109 to set the communications for Modbus RTU so that the PowerFlex 4M Drive communicates with the Micro830/850/870 controller via Modbus RTU communication.

Table 109 - Modbus RTU Parameters

Parameter	Description	Setting
C302	Comm. Data Rate (Baud Rate) 4 = 19200 bps	4
C303	Communication Node Address (address range is 1...127)	2
C304	Comm. Loss Action (Action that is taken when loss communication) 0 = Fault with coast stop	0
C305	Comm. Loss Time (Time remain in communication before taking action set in C304) 5 sec (max 60)	5
C306	Comm. Format (Data/Parity/Stop) RTU:8 Data Bit, Parity None, 1 Stop bit	0

8. Disconnect the Communications and save your project.
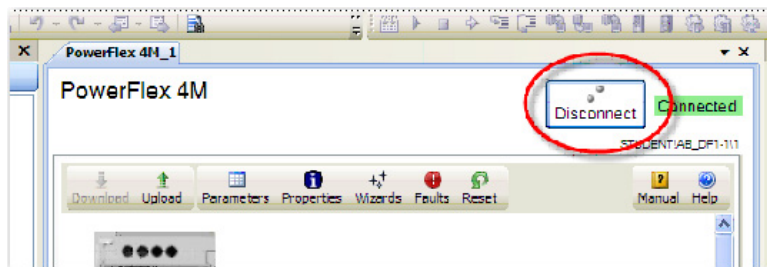
9. Turn off the power to the drive until the PowerFlex 4M display blanks out completely, then restore power to the PowerFlex 4M.

The drive is now ready to be controlled by Modbus RTU communication commands initiated from the Micro830/850/870 controller.

Modbus devices can be 0-based (registers are numbered starting at 0) or 1-based (registers are numbered starting at 1). When PowerFlex 4-class drives are used with Micro800 controllers, the register addresses that are listed in the PowerFlex drive user manuals must be offset by n+1.

For example, the Logic Command word is at address 8192, but your Micro800 program must use 8193 (8192+1) to access it.

EXAMPLE: Modbus Address (n+1 value shown)

8193	Logic Command word (Stop, Start, Jog, and so on)
8194	Speed Reference word

xxx.x format for PowerFlex 4/4M/40 drives, where "123" = 12.3 Hz
xxx.xx format for PowerFlex 40P/400/400N/400P drives, where "123" = 1.23 Hz

8449	Logic Status word (Read, Active, Fault, and so on.)
8452	Speed Feedback word (uses same format as Speed Reference)
8450	Error Code word
(n+1)	To access Parameter

If the respective PowerFlex drive supports Modbus Function Code 16 Preset (Write) Multiple Registers, use one write message with a length of "2" to write the Logic Command (8193) and Speed reference (8194) simultaneously.

Use one Function Code 03 Read Holding Registers with a length of "4" to read the Logic status (8449), Error Code (8450), and Speed Feedback (8452) simultaneously.

See the respective PowerFlex 4-class drive user manual for additional information about Modbus addressing.

Performance

The performance of MSG_MODBUS (Micro800 controller is configured as the master) is affected by the Program Scan because messages are serviced when the message instruction is executed in a program. For example, if the program scan is 100 ms and six serial ports are used, then the theoretical maximum for serial ports is 60 messages/second total. This theoretical maximum may not be possible since MSG_MODBUS is a master/slave request/response protocol, so performance is affected by several variables such as message size, communication rate, and slave response time.

The performance of Micro800 controllers when receiving Modbus request messages (Micro800 controller is configured as the slave) is also affected by the Program Scan. Each serial port is serviced only once per program scan.

Notes:

Quick Starts

This chapter covers some common tasks and quick start instructions that are aimed to make you familiar with the Connected Components Workbench software.

Update Your Micro800 Controller Firmware

The quick start shows you how to update the firmware for a Micro800 controller using Connected Components Workbench software version 10 or later.

From Connected Components Workbench software release 10 onwards, there are two options you can select when updating the firmware:

- Upgrade or Downgrade – This option retains the controller's existing configuration, Ethernet settings, and password.
- Reset – This option clears the controller's existing configuration, Ethernet settings, and password.

The procedure to update the controller is similar for both options.

⚠	**ATTENTION:** Retention of the controller's existing configuration, Ethernet settings, and password is only available when updating from firmware revision 10 to the same or later revision. If updating from firmware revision 10 to 9 or earlier, or updating to firmware revision 10 from an earlier revision, the controller's existing configuration, Ethernet settings, and password are cleared.

IMPORTANT	If you have forgotten the password for the controller, use the Reset option to clear the password.

On the Micro850 and Micro870 controllers, you can update your controllers through the Ethernet port and USB.

IMPORTANT	To update your controller over USB successfully, connect only one controller to your computer, and do not perform the update in a virtual machine such as VMware.

IMPORTANT	Update over USB using FactoryTalk Linx software with a 32-bit operating system is not supported. Use either a 64-bit operating system or RSLinx Classic software.

To begin, launch the Connected Components Workbench software:

1. In the menu, select Device > Update Firmware > Upgrade or Downgrade...
   Alternatively, in the Project Organizer, right-click the controller and select Update Firmware > Upgrade or Downgrade...
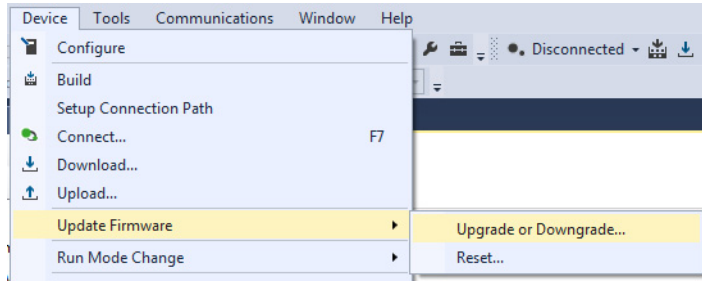
2. If your project does not have a connection path to the controller, the Connection Browser dialog appears. Select your controller, then select OK.
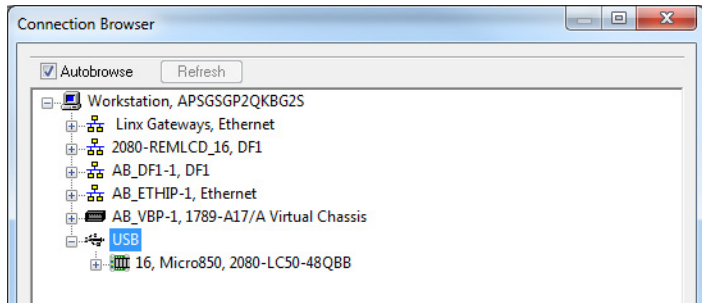
3. In the Upgrade or Downgrade Firmware dialog box, select the desired Target Revision to update the controller.
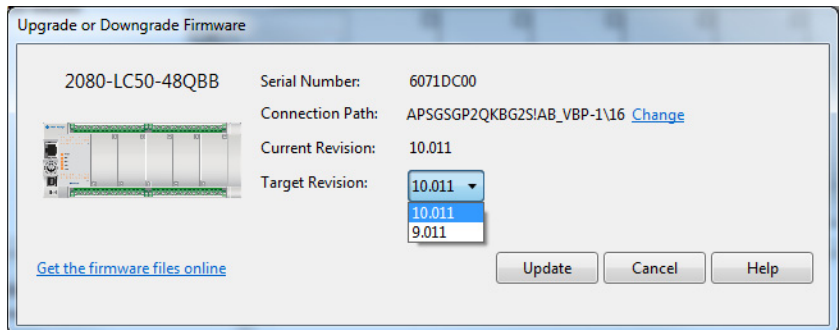
If the desired firmware revision is not shown in the drop-down list, you can download that firmware revision by clicking the "Get the firmware files online" link.

You can also change the Connection Path by selecting the "Change" link.

4. When you have confirmed the settings, select Update to begin updating the controller. The update progress is shown in the dialog box.
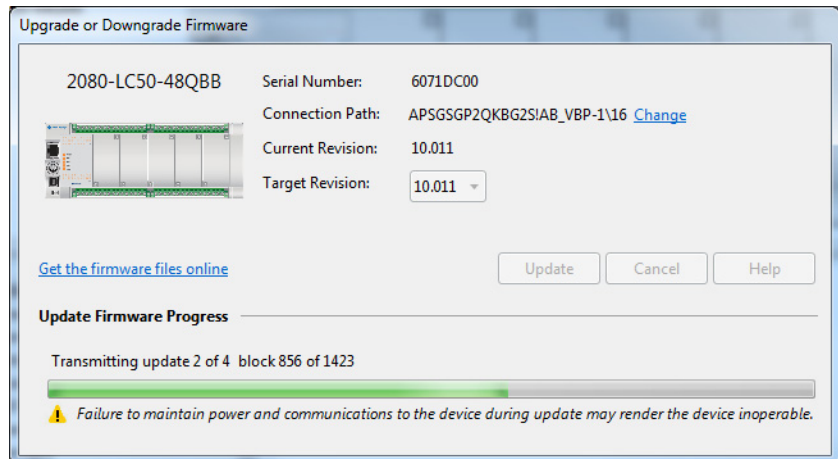
5. After the update is completed, the status is shown in the dialog box.
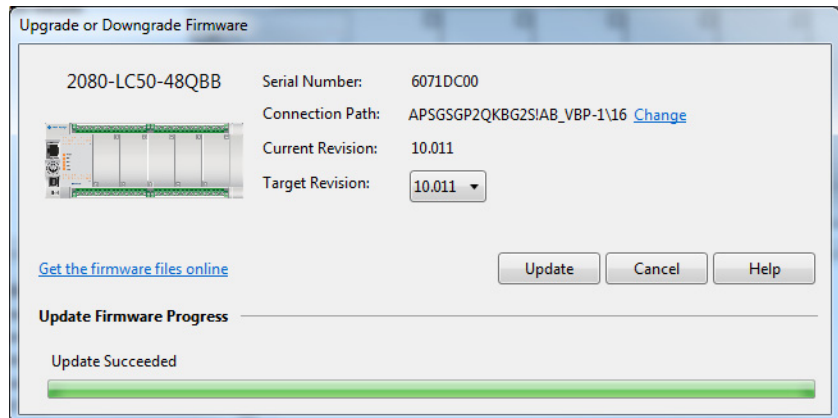
IMPORTANT	After control flashing the controller, some microSD cards may not be detected. Remove and insert the microSD card, or power cycle the controller if this issue is encountered.

Firmware Update From MicroSD Card

With Connected Components Workbench software version 12 or later, and the microSD card plug-in for Micro800 controllers, you can update your Micro830, Micro850, and Micro870 controller from the microSD card and with ControlFLASH. This is two-step process – first you transfer the firmware to the microSD card using the SD Card Utility, then you edit the ConfigMeFirst.txt file to initiate the update process. See the following instructions for performing the update from the microSD card.

Step 1 – Transfer the Firmware to the microSD Card

1. Launch the Connected Components Workbench software.
2. Click Tools > SD Card Utility.
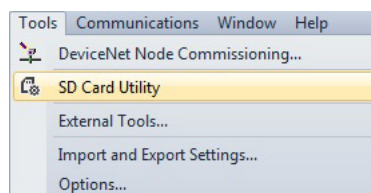
The SD Card Utility window appears.
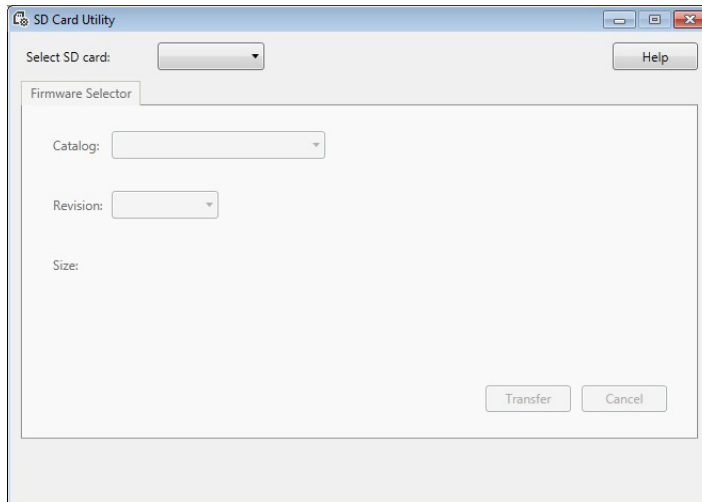
3. Select the drive letter that points to the microSD card on your computer from the dropdown list.
You can check the drive letter by looking in Windows® Explorer. For this example, the microSD card uses the drive letter "G".
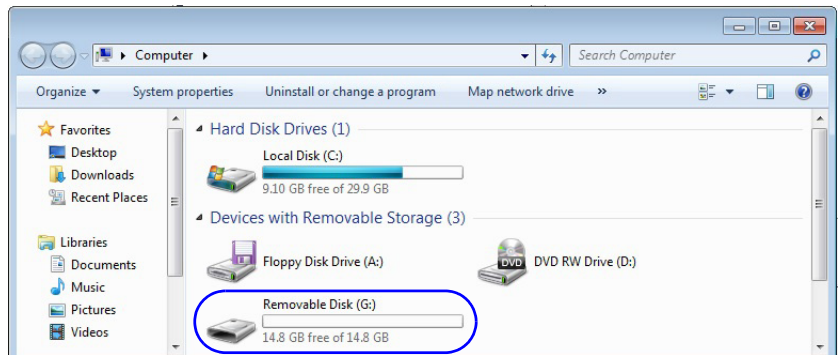
4. Select the catalog number of your Micro800 controller.
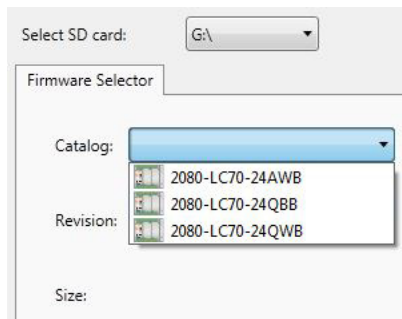
5. Select the firmware revision that you want to update your Micro800 controller with.
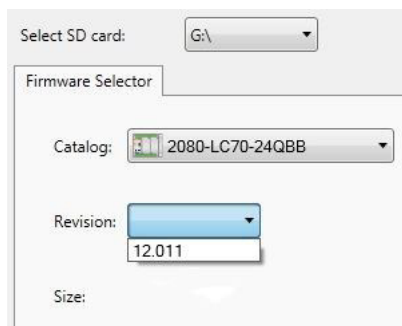
The list of firmware revisions is installed together with the Connected Components Workbench software. If you require a revision that is not listed, download the firmware from the Product Compatibility and Download Center (PCDC) at rok.auto/pcdc and install the included ControlFLASH kit.

IMPORTANT	You must sign in to the Rockwell Automation website before downloading a firmware revision.

Close and relaunch the Connected Components Workbench software, then open the SD Card Utility again. The revision should now appear in the list.

6. Click Transfer.
   The file is copied to the microSD card.

Copying items 4 (19.4 KB) to G:\

⚠ Firmware transfer completes successfully.
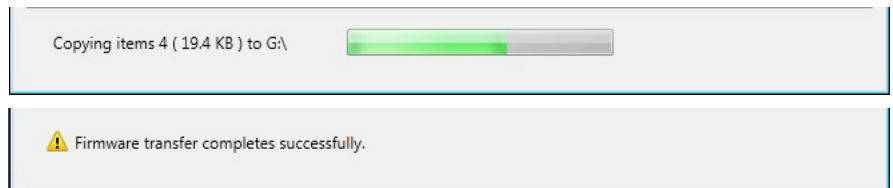
7. Close the SD Card Utility and proceed to the next step to edit the ConfigMeFirst.txt file.

Step 2 – Edit the ConfigMeFirst.txt File

To update the controller with the firmware that you have transferred to the microSD card, you must edit the ConfigMeFirst.txt file with the settings listed below. These settings must be added at the beginning of the file.

Table 110 - New ConfigMeFirst.txt Configuration Settings for Firmware Update

Setting	Takes Effect On...	Description
Firmware update settings		
[FWFILE]	Powerup	File path location of the firmware revision on the microSD card. The default location is in the following format: firmware\<catalog number>\<filename of firmware>
[FWDOWN]	Powerup	Sets whether to upgrade or downgrade the controller firmware from the current revision. 0 = Upgrade firmware; 1 = Downgrade firmware **IMPORTANT:** Firmware Upgrade occurs if the [FWFILE] setting points to a newer revision of the firmware file, compared to the firmware in the controller, irrespective of the [FWDOWN] setting.

Example of ConfigMeFirst.txt File for Firmware Update

```
ConfigMeFirst.txt - Notepad
File   Edit   Format   View   Help
[FWFILE = firmware\2080-LC70-24QBBA\2080-LC70-24QBBA_12_011.nvs]
[FWDOWN = 0]
[END]
```
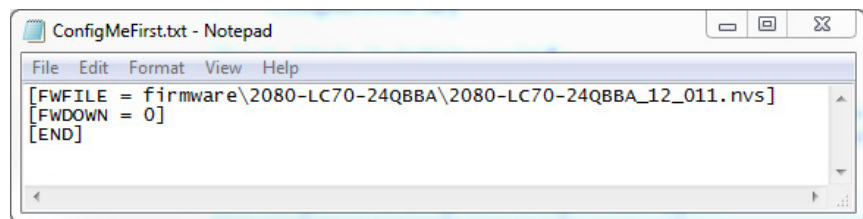
After you have edited the file, insert the microSD card into the controller. Cycle power to the controller and the update process begins. The SD status LED does not blink when updating the firmware from the microSD card is in progress.

When using the ControlFLASH software to downgrading the firmware of a Micro830 or Micro850 series B controller to firmware revision 10.011, the program reports an error and fails at the initial stage. However when upgrading a Micro800 controller using the microSD card with a firmware revision that is not compatible with the series, the controller hard faults. There is no error code reported after you have cycled power to the controller. The controller retains the old firmware.

Table 111 - Fault Status Indicator Description

State	Indicates
Steady Red	Fault
Flashing Green	Run

For a list of firmware and series compatibility, see the release notes for firmware revision 11.011 or later, on the Product Compatibility and Download Center (PCDC) at rok.auto/pcdc.

Establish Communications Between RSLinx and a Micro830/Micro850/ Micro870 Controller Through USB

This quick start shows you how to get RSLinx® RSWho to communicate with a Micro830, Micro850, or Micro870 controller through USB. Micro830, Micro850, and Micro870 controllers use the AB_VBP-x driver.

RSLinx Classic software is installed as part of the Connected Components Workbench software installation process. The minimum version of RSLinx Classic with full Micro800 controller support is 2.57, build 15 (released March 2011).

1. Power up the Micro830/Micro850/Micro870 controller.
2. Plug the USB A/B cable directly between your PC and the Micro830/Micro850/Micro870 controller.
3. Windows should discover the new hardware. Select No, not this time and then select Next.

4. Select Install the software automatically (Recommended), and then select Next.

5.   The Wizard searches for new hardware.
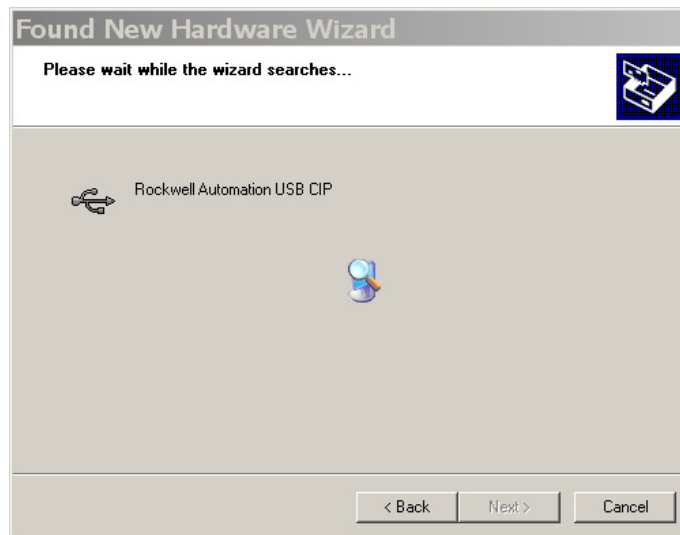
6.   Select Finish when the wizard completes the installation.

7.   Open RSLinx Classic and run RSWho by selecting the 🔲 icon.

If the proper EDS file is installed, the Micro830/Micro850/Micro870 controller should be properly identified and show up under both the Virtual Backplane (VBP) driver and the USB driver, which was automatically created.
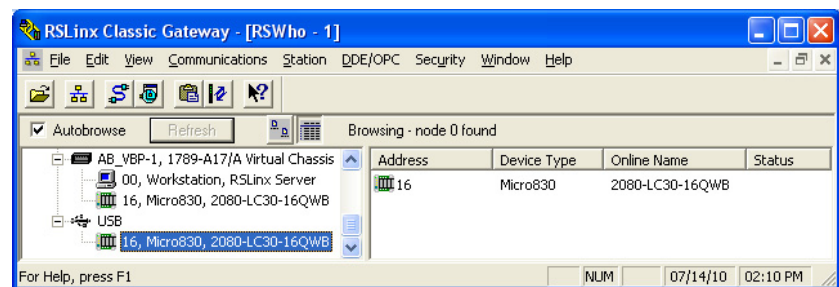
If instead the Micro830/Micro850/Micro870 shows up as a "1756 Module" under the AB_VBP-1 Virtual Chassis driver, then the proper EDS file for this major revision of firmware has not yet been installed or the controller is running pre-release firmware (Major Revision=0).
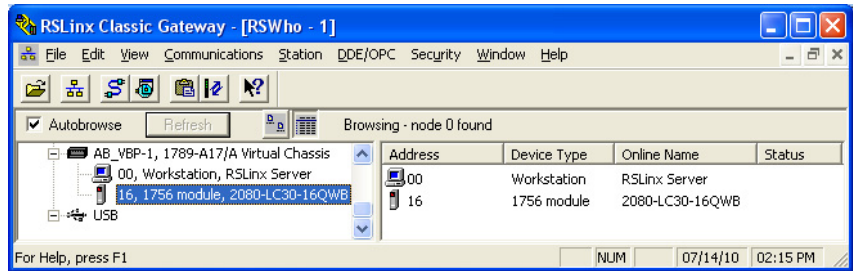
Since the Micro830/Micro850/Micro870 controllers support embedded EDS files, right-click this device, and select Upload EDS file from device.
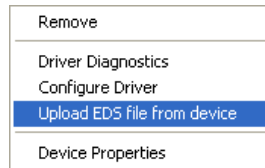
8. On the EDS wizard that appears, select Next to continue.

9. Follow the prompts to upload and install the EDS file.
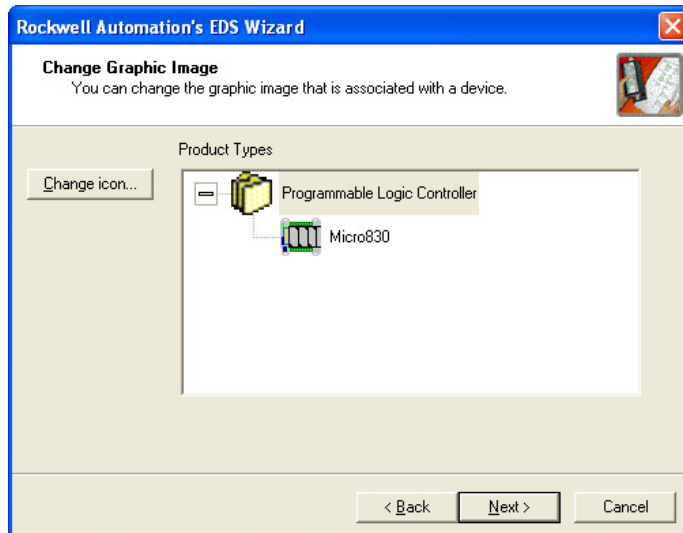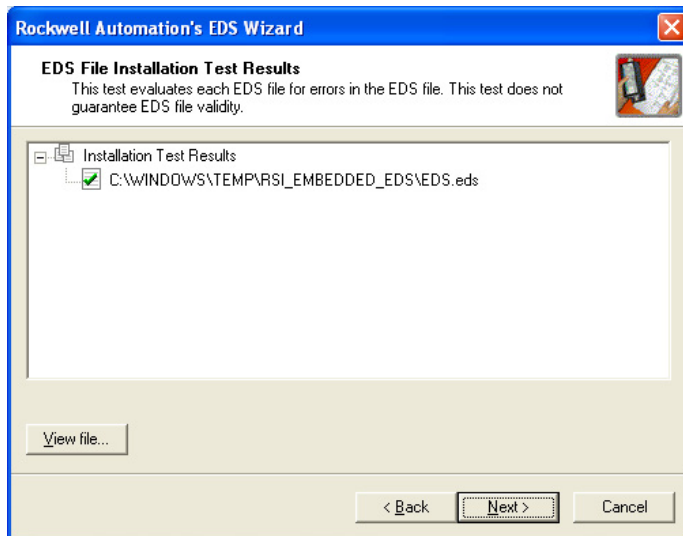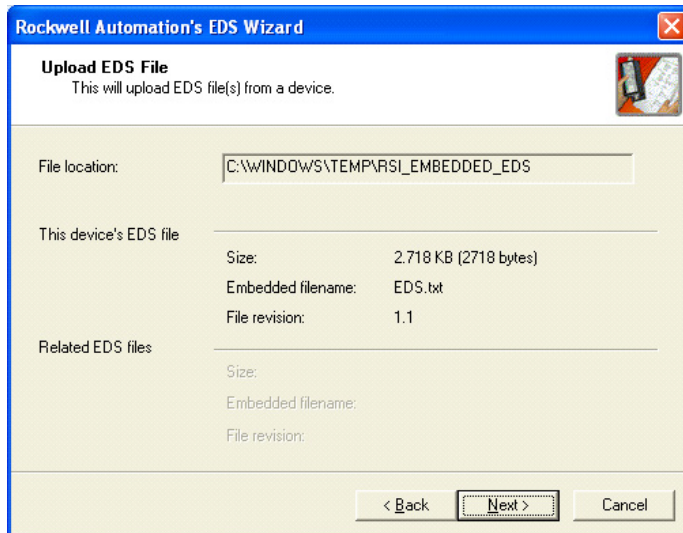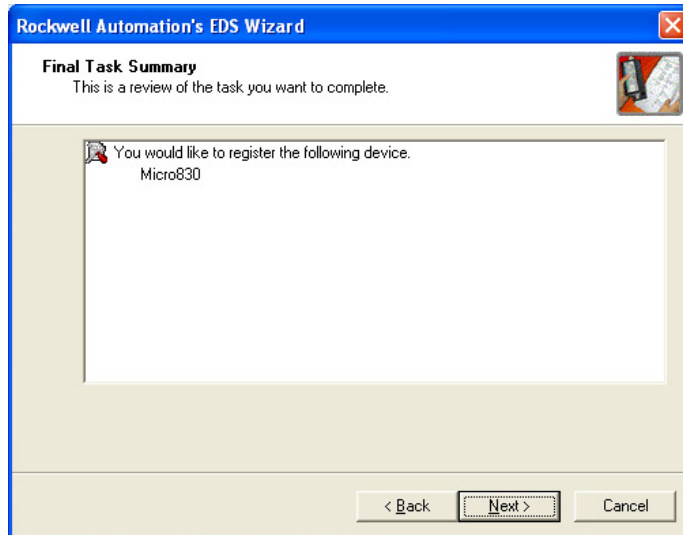
10. Select Finish to complete.
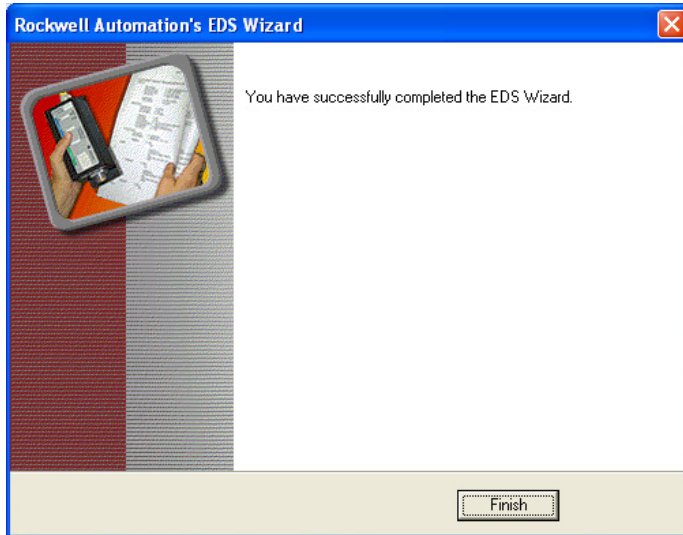
If the Micro830/Micro850/Micro870 still shows up as a 1756 Module, then you are probably running pre-release firmware that is reporting itself as Major Revision 0, which does not match the embedded EDS file. To confirm, right-click the device and select Device Properties (firmware revision is Major.Minor).
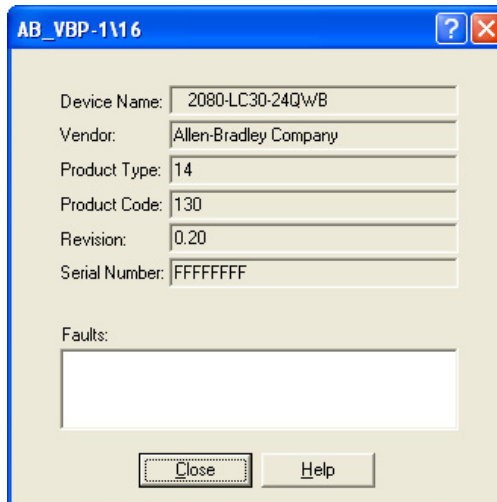
Configure Controller Password

Set, change, and clear the password on a target controller through the Connected Components Workbench software.

IMPORTANT	The following instructions are supported on Connected Components Workbench software version 2 and Micro800 controllers with firmware revision 2.
	For more information about the controller password feature on Micro800 controllers, see <u>Controller Security on page 225</u>.

Set Controller Password

IMPORTANT	After creating or changing the controller password, you must power down the controller in order for the password to be saved.

In the following instructions, the Connected Components Workbench software is connected to the Micro800 controller.

1. On the Connected Components Workbench software, open the project for the target controller.

2. Select Connect to connect to the target controller.
   On the Device Details toolbar, the Secure tooltip message "Set, Change, or Clear Micro800 Controller Password Protection" is displayed.
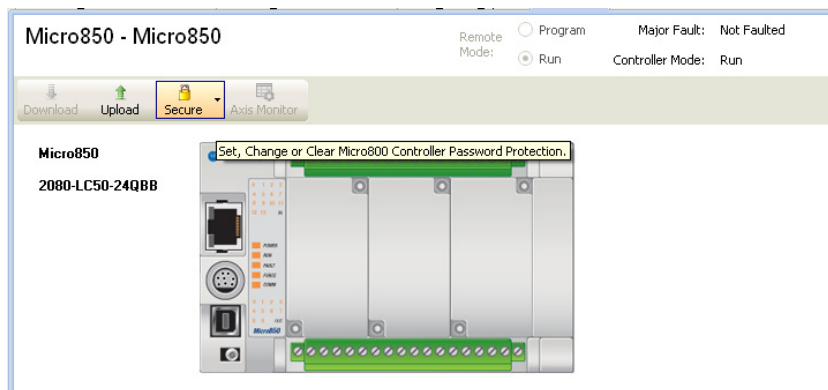
3. Click Secure. Select Set Password.
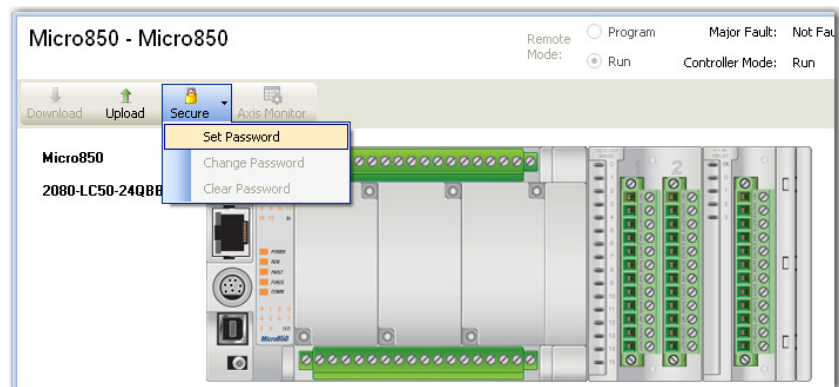
4. The Set Controller Password dialog appears. Provide password. Confirm the password by providing it again in the Confirm field.

> Passwords must have at least eight characters to be valid.

5. Select OK.
    Once a password is created, any new sessions that try to connect to the controller have to supply the password to gain exclusive access to the target controller.

Change Password

With an authorized session, you can change the password on a target controller through the Connected Components Workbench software. The target controller must be in Connected status.

1. On the Device Details toolbar, select Secure. Select Change Password.
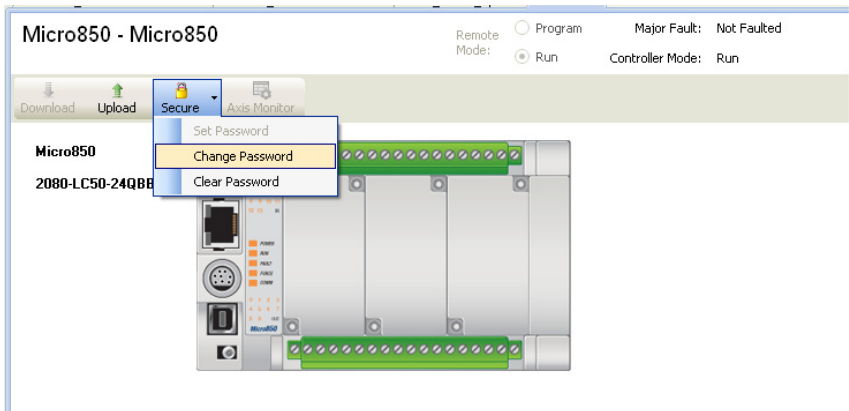
2. The Change Controller Password dialog appears. Enter Old Password, New Password and confirm the new password.

3. Select OK.

The controller requires the new password to grant access to any new session.

Clear Password

With an authorized session, you can clear the password on a target controller through the Connected Components Workbench software.

1. On the Device Details toolbar, select Secure. Select Clear Password.
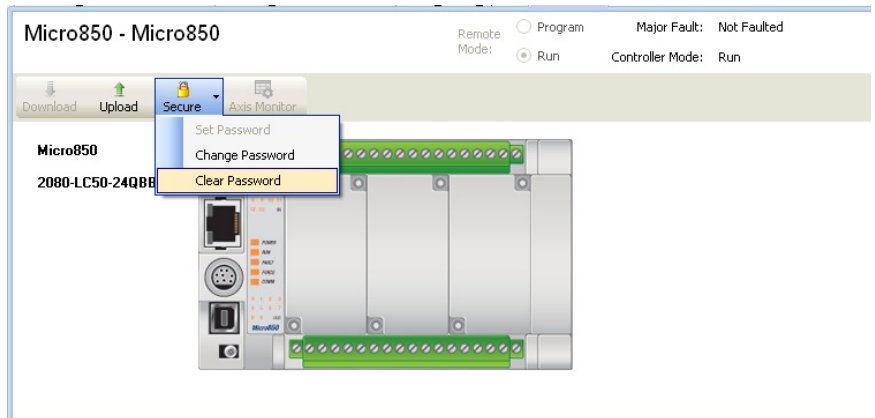
2. The Clear Password dialog appears. Enter your password.
3. Select OK to clear the password.

The controller requires no password on any new session.

Use the High-Speed Counter

To use a HSC device, you first need to establish the HSC counting mode required by your application. See for available modes on Micro800 controllers.

The following sample project guides you through the creation of a project that uses HSC mode 6, a quadrature counter with phased inputs A and B. It shows you how to write a simple ladder program with the HSC function block, create variables, and assign variables and values to your function block. It also guides you through a step-by-step process on how test your program, and enable a Programmable Light Switch (PLS).

This sample project uses a quadrature encoder. The quadrature encoder is used for determining direction of rotation and position for rotating, such as a lathe. The Bidirectional Counter counts the rotation of the Quadrature Encoder.

shows a quadrature encoder that is connected to inputs 0 and 1. The count direction is determined by the phase angle between A and B. If A leads B, the counter increments. If B leads A, the counter decrements.

Figure 63 - Quadrature Encoder on Inputs 0 and 1
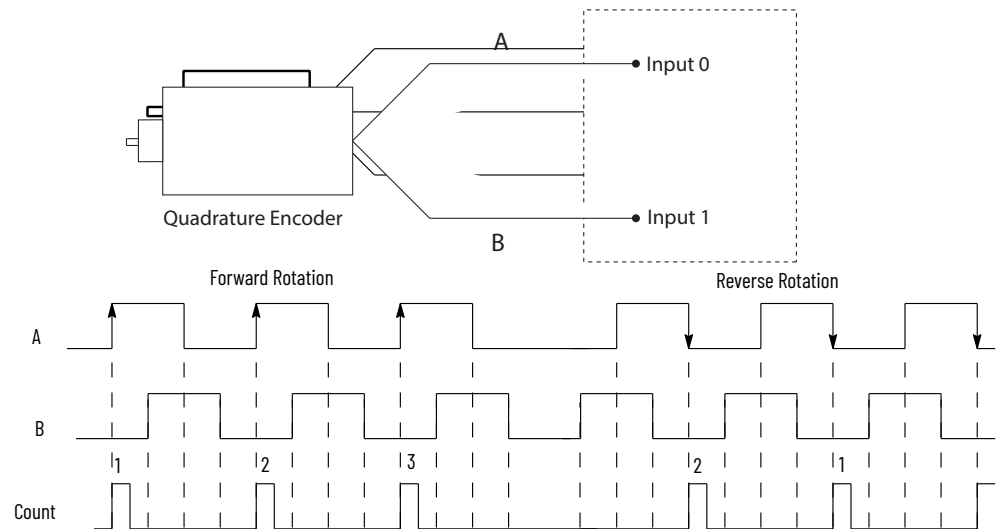
This quick start includes the following sections:

Create the HSC Project and Variables

1. Start the Connected Components Workbench software and open a new project. From the Device Toolbox, go to Catalog > Controllers. Double-click your controller[1] or drag-and-drop it onto the Project Organizer window.
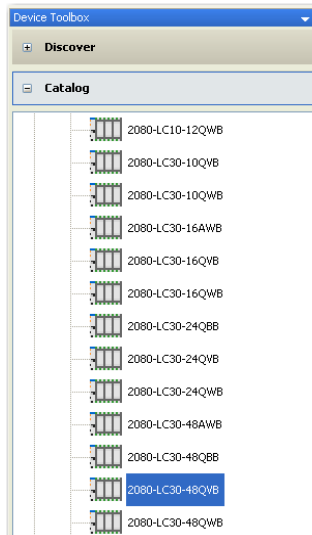
2. Under Project Organizer, right-click Programs. Select Add New LD: Ladder Diagram to add a new ladder logic program.

---

(1)   The HSC module is supported on all Micro830, Micro850, and Micro870 controllers, except on 2080-LCxx-xxAWB types.
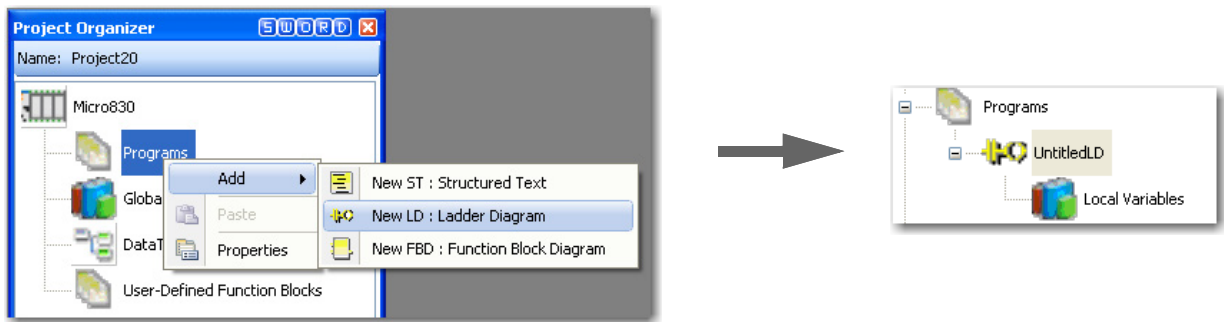
3. Right-click UntitledLD and select Open.
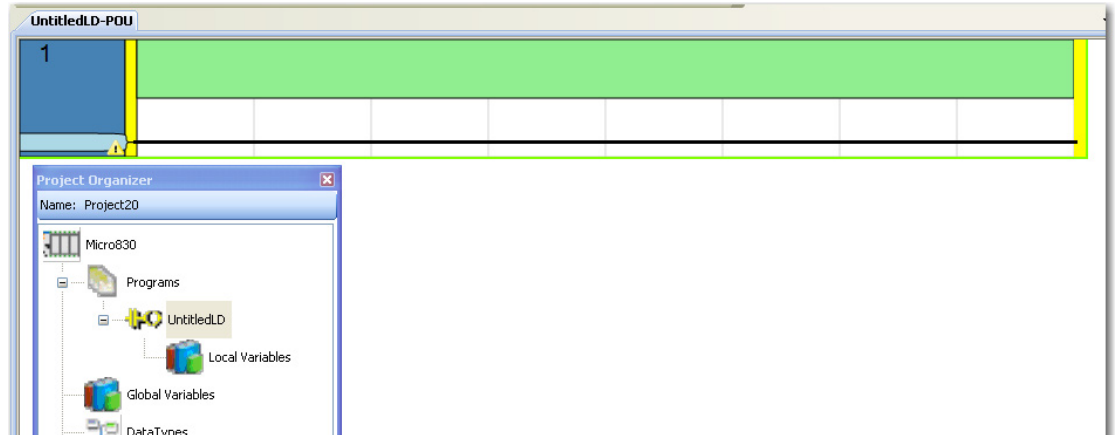
4. From the Toolbox, double-click Direct Contact to add it to the rung or drag-and-drop Direct Contact onto the Rung.
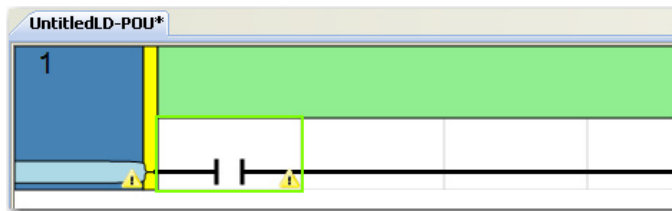
5. Double-click the Direct Contact that you have added to bring up the Variable Selector dialog. Select the I/O - Micro830 tab. Assign the Direct Contact to input 5 by selecting _IO_EM_DI_05. Select OK.
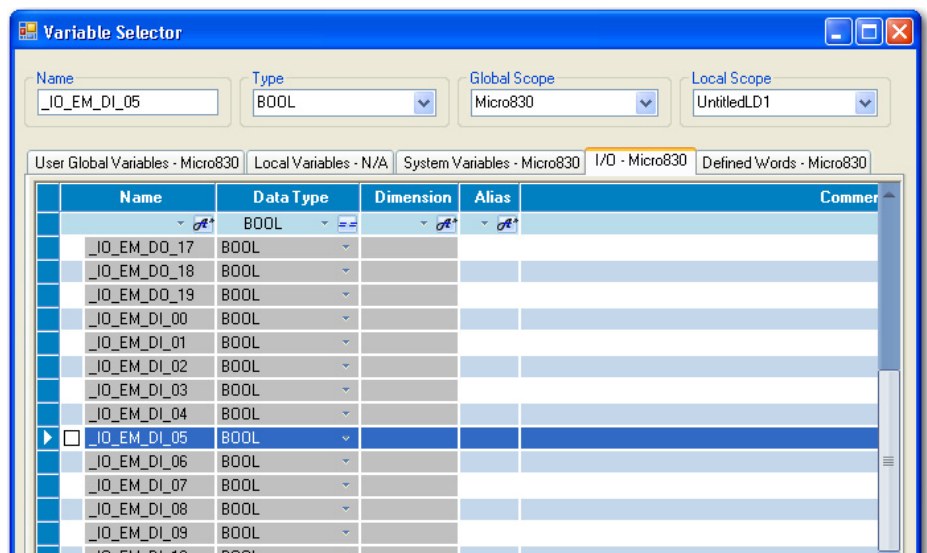
6. To the right of the Direct Contact, add a function block by double-clicking the function block from the Toolbox or dragging and dropping the function block onto the rung.
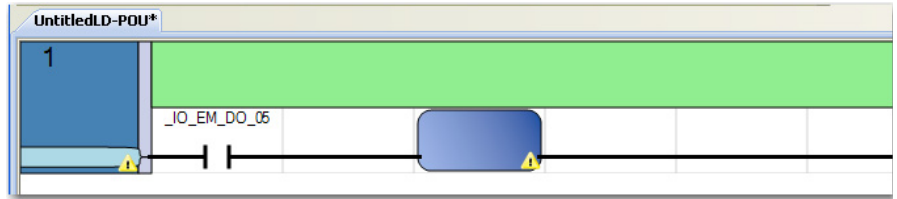
7. Double-click the function block to open up the Instruction Selector dialog. Choose HSC. You can do a quick search for HSC function block by typing "HSC" on the name field. Select OK.
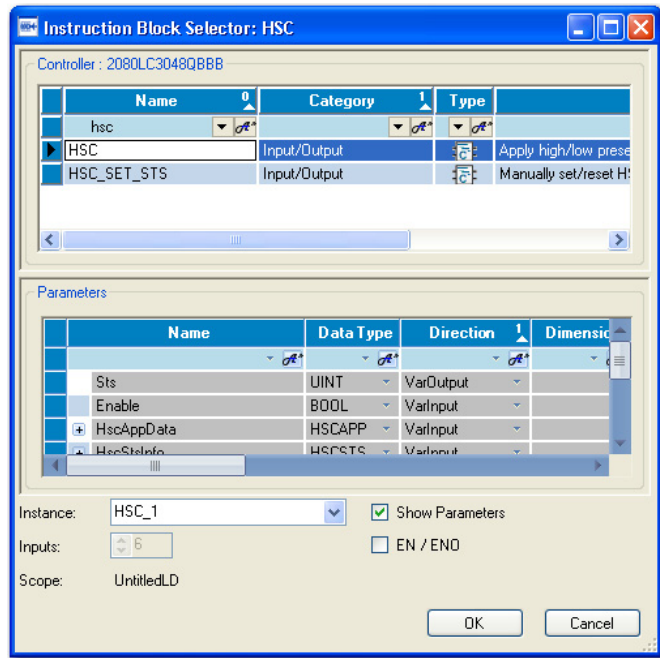
Your ladder rung appears as shown:
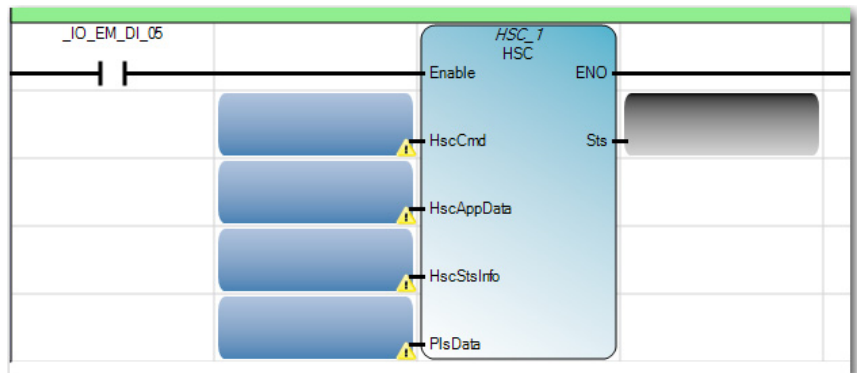
8. On the Project Organizer pane, double-click Local Variables to bring up the Variables window. Add the following variables with the corresponding data types, as specified in Table 112 on page 282.

Table 112 - Variable Data Types

Variable Name	Data Type
MyCommand	USINT
MyAppData	HSCAPP

Table 112 - Variable Data Types

Variable Name	Data Type
MyInfo	HSCSTS
MyPLS	PLS
MyStatus	UINT

After adding the variables, your Local Variables table should look like this:

Assign Values to the HSC Variables

Next, you must assign values to the variables you have created. Typically, a routine is used to assign values to your variables. For illustration purposes, this quick start assigns values through the Initial Value column of the Local Variables table.

> In a real program, you should write a routine to assign values to your variable according to your application.

1. On the Initial Value field for the MyCommand variable, type 1.
   See HSC Commands (HScCmd) on page 218 for more information on the description for each value.

2. Assign values to the MyAppData variables. Expand the list of MyAppData subvariables selecting the + sign. Set the values of the different subvariables as shown in the following screenshot.

IMPORTANT	MyAppData variable has subvariables that determine the settings of the counter. It is **crucial** to know each one to determine how the counter will performs. A quick summary is provided below but you can also see HSC APP Data Structure on page 204 for detailed information.

MyAppData.PlsEnable allows you to either enable or disable the PLS settings. It should be set to FALSE (disabled) if the MyAppData variable is to be used.

MyAppData.HscID allows you to specify which embedded inputs is used depending on the mode and the type of application. See the table [HSC Inputs and Wiring Mapping on page 203](#) to know the different IDs that can be used, and the embedded inputs and its characteristics.

If ID 0 is used, ID 1 cannot be used on the same controller since the inputs are being used by the Reset and Hold.

MyAppData.HscMode allows you to specify the type of operation in which the HSC uses to count. See [HSC Mode (HSCAPP.HSCMode) on page 206](#) for more information about HSC modes. See [Table 113](#) for the list available modes.

Table 113 – HSC Operating Modes

Mode Number	Type
0	Up Counter – The accumulator is immediately cleared (0) when it reaches the high preset. A low preset cannot be defined in this mode.
1	Up Counter with external reset and hold – The accumulator is immediately cleared (0) when it reaches the high preset. A low preset cannot be defined in this mode.
2	Counter with external direction
3	Counter with external direction, reset, and hold
4	Two input counter (up and down)
5	Two input counter (up and down) with external reset and hold
6	Quadrature counter (phased inputs A and B)
7	Quadrature counter (phased inputs A and B) with external reset and hold
8	Quadrature X4 counter (phased inputs A and B)
9	Quadrature X4 counter (phased inputs A and B) with external reset and hold

Modes 1, 3, 5, 7, and 9 only work when an ID of 0, 2, or 4 is set because these modes use reset and hold. Modes 0, 2, 4, 6, and 8 work on any ID. Modes 6…9 only work when an encoder is connected to the controller. Use the HSC ID chart as a reference to wire the encoder to the controller.

MyAppData.HPSetting, **MyAppData.LPSetting**, **MyAppData.OFSetting**, and **MyAppData.UFSetting** are all user-defined variables that represent the counting range of the HSC. [Figure 64](#) gives an example of a range of values that can be set for these variables.

Figure 64 – Range of Values
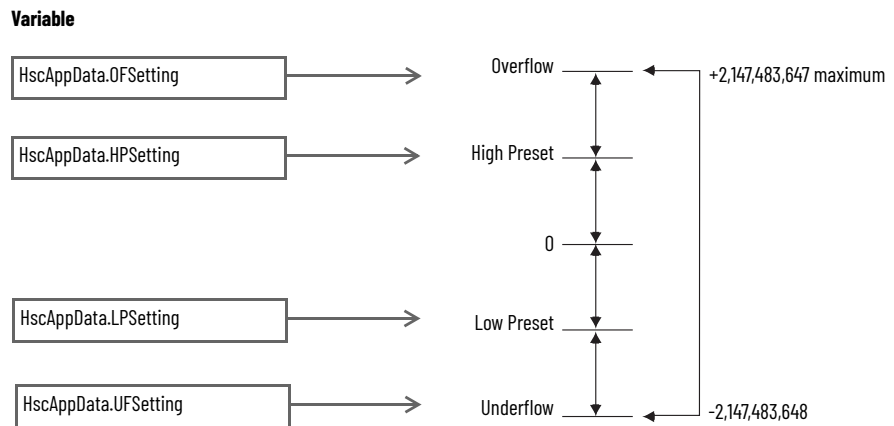
MyAppData.OutputMask along with **MyAppData.HPOutput** and **MyAppData.LPOutput** allows you to specify which embedded outputs can be turned on when a High Preset or Low Preset is reached. These variables use a combination of decimals and binary numbers to specify the embedded outputs that are able to turn on/off.

In our example, we first set the Output Mask to a decimal value of 3 which, when converted to binary, is equal to 0011. This means that now outputs 00 and 01 can be turned On/Off.

We have set the HPOutput to a decimal value of 1, which, when converted to binary, is equal to 0001. This means that when a High Preset is reached, output 00 turns on and stays on until the HSC is reset or the counter counts back down to a Low Preset. The LPOutput works the same way as the HPOutput except an output turns on when a Low Preset is reached.

Assign Variables to the Function Block

1. Return to the ladder diagram and assign the variables you have configured to the corresponding elements of the HSC function block. The HSC function block should appear as shown:
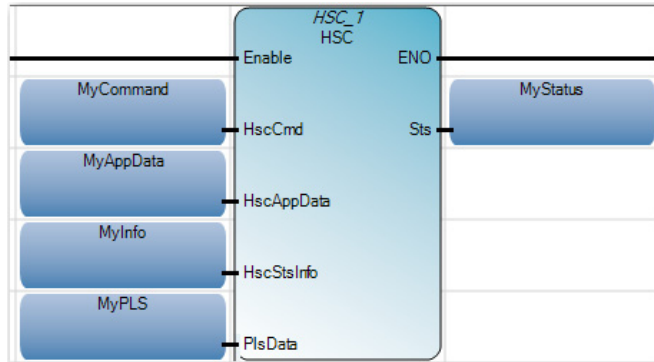
To assign a variable to a particular element in your function block, double-click the empty variable block. On the Variable selector that appears, choose the variable that you have created. For example, for the input element HSCAppData, select the variable MyAppData.

2. Next, select the Micro830 controller under the Project Organizer pane to bring up the Micro830 Controller Properties pane. Under Controller Properties, select Embedded I/O. Set the input filters to a correct value depending on the characteristics of your encoder.
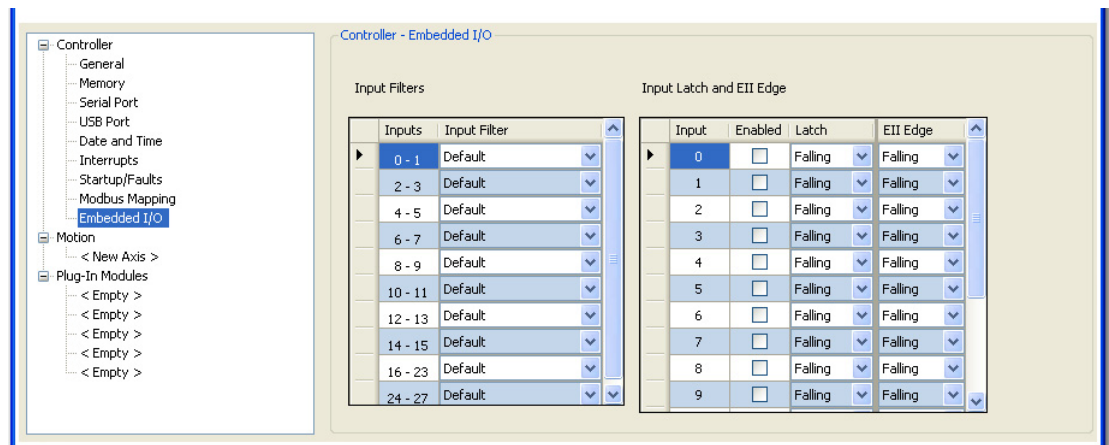
3. Make sure that your encoder is connected to the Micro830 controller.

4. Power up the Micro830 controller and connect it to your PC. Build the program in the Connected Components Workbench software and download it to the controller.

Run the High-Speed Counter

1. To test the program, go into debug mode by doing any of the following:
   - Select Debug menu, then choose Start Debugging,
   - Select the green play button below the menu bar, or
   - Press the F5 Windows key.
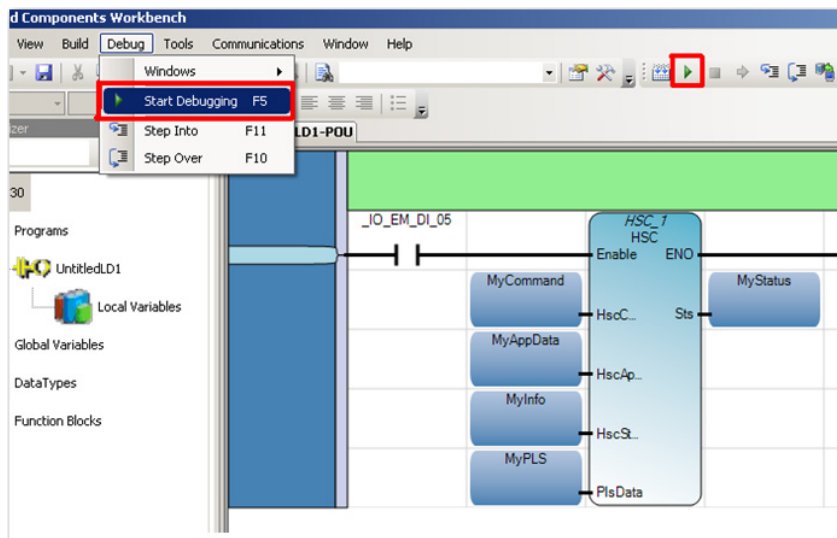
Now that we are on debug mode we can see the values of the HSC output. The HSC function block has two outputs, one is the STS (MyStatus) and the other is the HSCSTS (MyInfo).

2. Double-click the Direct Contact labeled _IO_EM_DI_05 to bring up the Variable Monitoring window.

3. Select the I/O - Micro830 tab. Select the _IO_EM_DI_05 row. Check the boxes Lock and Logical Value so that this input is forced in the ON position.

4. Click the Local Variables tab to see any real time changes being made to the variables. Expand the MyAppData and MyInfo variable list by selecting the + sign.

5. Turn On the encoder to see the counter count up/down. For example, if the encoder is attached to a motor shaft then turn on the motor to trigger the HSC count. The counter value is displayed on MyInfo.Accumulator. MyStatus variable should display a Logical Value of 1, which means that the HSC is running.

See HSC Function Block Status Codes on page 218 for the complete list of status codes. For example, if the MyStatus value is 04, a configuration error exists and the controller will fault. You must check your parameters in this case.

Variable Monitoring

	Name	Logical Value	Physical Value	Initial Value
		...	...	...
+	HSC_1	...	...	...
	MyCommand	1	N/A	1
-	MyAppData	...	...	...
	MyAppData.PlsEnable	☐	N/A	FALSE
	MyAppData.HscID	0	N/A	0
	MyAppData.HscMode	7	N/A	5
	MyAppData.Accumulator	40	N/A	
	MyAppData.HPSetting	40	N/A	40
	MyAppData.LPSetting	-40	N/A	-40
	MyAppData.OFSetting	50	N/A	50
	MyAppData.UFSetting	-50	N/A	-50
	MyAppData.OutputMask	3	N/A	3
	MyAppData.HPOutput	1	N/A	1
	MyAppData.LPOutput	2	N/A	2
-	MyInfo	...	...	...
	MyInfo.CountEnable	☑	N/A	
	MyInfo.ErrorDetected	☐	N/A	
	MyInfo.CountUpFlag	☑	N/A	
	MyInfo.CountDwnFlag	☑	N/A	
	MyInfo.Mode1Done	☐	N/A	
	MyInfo.OVF	☐	N/A	
	MyInfo.UNF	☐	N/A	
	MyInfo.CountDir	☑	N/A	
	MyInfo.HPReached	☑	N/A	
	MyInfo.LPReached	☐	N/A	
	MyInfo.OFCauseInter	☐	N/A	
	MyInfo.UFCauseInter	☐	N/A	
	MyInfo.HPCauseInter	☐	N/A	
	MyInfo.LPCauseInter	☐	N/A	
	MyInfo.PlsPosition	0	N/A	
	MyInfo.ErrorCode	0	N/A	
	MyInfo.Accumulator	40	N/A	
	MyInfo.HP	40	N/A	
	MyInfo.LP	-40	N/A	
	MyInfo.HPOutput	1	N/A	
	MyInfo.LPOutput	2	N/A	
+	MyPLS	...	...	...
▶	MyStatus	1	N/A	

Tabs: Global Variables - Micro830 | Local Variables - UntitledLD1 | System Variables - Micro830 | I/O - Micro830

For this example, once the Accumulator reaches a High Preset value of 40, output 0 turns on, and the HPReached flag turns on. Once the Accumulator reaches a Low Preset value of -40, output 1 turns on and the LPReached flag turns on as well.

Use the Programmable Limit Switch (PLS) Function

The Programmable Limit Switch function allows you to configure the High-Speed Counter to operate as a PLS (programmable limit switch) or rotary cam switch. The PLS is used when you need multiple pairs of high and low presets (up to 255 pairs of high and low presets are supported by the PLS).

1. Start a new project following the same steps and values as the previous project. Set the values for the following variables as follows:

    -   HSCAPP.PlsEnable variable should be set to TRUE.
    -   Set a value only for UFSetting and OFSetting (OutputMask is optional depending if an output is to be set or not). Your new values should follow the example in :

Figure 65 - PLS Values

Name	Data Type	Dimension	Alias	Initial Value	Attribute
⊞ HSC_1	HSC			...	ReadWrite
MyCommand	USINT			1	ReadWrite
⊟ MyAppData	HSCAPP			...	ReadWrite
MyAppData.PlsEnable	BOOL			TRUE	ReadWrite
MyAppData.HscID	UINT			0	ReadWrite
MyAppData.HscMode	UINT			7	ReadWrite
MyAppData.Accumulator	DINT				ReadWrite
MyAppData.HPSetting	DINT				ReadWrite
MyAppData.LPSetting	DINT				ReadWrite
MyAppData.OFSetting	DINT			50	ReadWrite
MyAppData.UFSetting	DINT			-50	ReadWrite
MyAppData.OutputMask	UDINT			255	ReadWrite
MyAppData.HPOutput	UDINT				ReadWrite
MyAppData.LPOutput	UDINT				ReadWrite
⊞ MyInfo	HSCSTS			...	ReadWrite
⊟ MyPLS	PLS	[1..4]		...	ReadWrite
⊟ MyPLS[1]	PLS			...	ReadWrite
MyPLS[1].HscHP	DINT			10	ReadWrite
MyPLS[1].HscLP	DINT			-10	ReadWrite
MyPLS[1].HscHPOutPut	UDINT			1	ReadWrite
MyPLS[1].HscLPOutPut	UDINT			16	ReadWrite
⊟ MyPLS[2]	PLS			...	ReadWrite
MyPLS[2].HscHP	DINT			20	ReadWrite
MyPLS[2].HscLP	DINT			-20	ReadWrite
MyPLS[2].HscHPOutPut	UDINT			2	ReadWrite
MyPLS[2].HscLPOutPut	UDINT			32	ReadWrite
⊟ MyPLS[3]	PLS			...	ReadWrite
MyPLS[3].HscHP	DINT			30	ReadWrite
MyPLS[3].HscLP	DINT			-30	ReadWrite
MyPLS[3].HscHPOutPut	UDINT			4	ReadWrite
MyPLS[3].HscLPOutPut	UDINT			64	ReadWrite
⊟ MyPLS[4]	PLS			...	ReadWrite
MyPLS[4].HscHP	DINT			40	ReadWrite
MyPLS[4].HscLP	DINT			-40	ReadWrite
MyPLS[4].HscHPOutPut	UDINT			8	ReadWrite
MyPLS[4].HscLPOutPut	UDINT			128	ReadWrite
MyStatus	UINT				ReadWrite

In this example, the PLS variable is given a dimension of [1..4]. This means that the HSC can have four pairs of High and Low Presets.

Once again, your High Presets should be set lower than the OFSetting and the Low Preset should be greater than the UFSetting. The HscHPOutPut and HscLPOutPut values determine which outputs turn on when a High Preset or Low Preset is reached.

2. You can now build and download the program into the controller then debug and test it following the instructions for the last project.

Forcing I/Os

Inputs are logically forced. Status indicators do not show forced values, but the inputs in the user program are forced.

Forcing is only possible with I/O and does not apply to user-defined variables and non-I/O variables, and special functions such as HSC and Motion that execute independently from the User Program scan. For example, for motion, Drive Ready input cannot be forced.

Unlike inputs, outputs are physically forced. Status indicators do show forced values and the user program does not use forced values.

Figure 66 illustrates forcing behavior.

Figure 66 - Forcing I/O Behavior
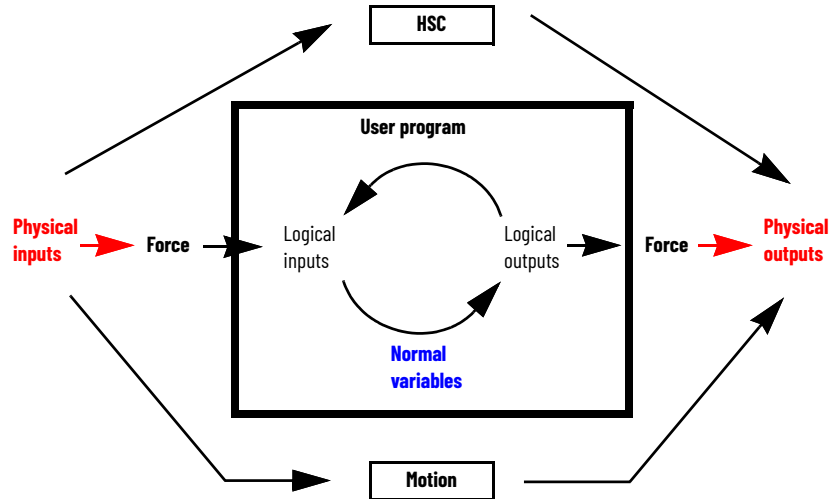
- LED status indicators always match the physical value of I/O
- Normal, non-physical internal variables cannot be forced
- Special functions such as HSC and Motion cannot be forced

> ⚠️ **ATTENTION:** Forcing variables can result in sudden machine movement, possibly injuring personnel or equipment. Use extreme caution when forcing variables.

Checking if Forces (locks) are Enabled

If the Connected Components Workbench software is available, check the Variable Monitor while debugging online. Forcing is performed by first Locking an I/O variable and then setting the Logical Value for Inputs and Physical Value for Outputs. Remember that you cannot force a Physical Input and cannot force a Logical Output.
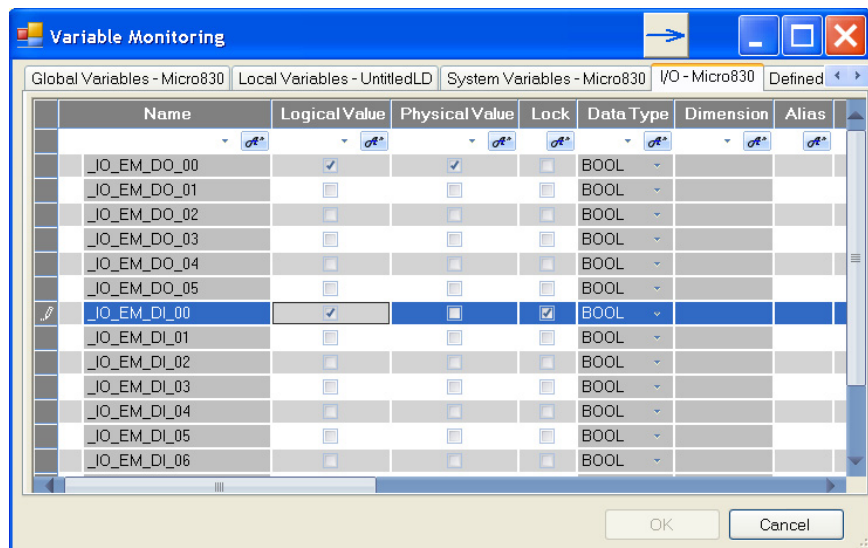
In many cases, the front of the controller is not visible to the operator and the Connected Components Workbench software is not online with the controller. If you want the force status to be visible to the operator, then the User Program must read the force status using the SYS_INFO function block and then display the force status on something that the operator can see, such as the human machine interface (HMI), or stack light. Figure 67 is an example program in Structured Text.

Figure 67 - Structured Text Example Program

```
1   (* Read System Information including Force Enable bit *)
2   SYS_INFO_1(TRUE);
3
4   (* Turn on Warning Light if Forces are Enabled *)
5   If SYS_INFO_1.Sts.ForcesInstall = TRUE THEN
6       _IO_EM_DO_05 := TRUE;
7   ELSE
8       _IO_EM_DO_05 := FALSE;
9   END_IF;
```

If the front of the controller is visible, and not blocked by the cabinet enclosure, Micro830, Micro850, and Micro870 controllers have a Force LED indicator.

I/O Forces After a Power Cycle

After you cycle power to a controller, all I/O forces are cleared from memory.

Use Run Mode Change

Run Mode Change allows you to make small changes to the logic of a running project and immediately testing it out on the controller, without having to go into Program mode or disconnecting from the controller.

IMPORTANT	The following requirements must be met to use Run Mode Change:
	• Micro820/Micro830/Micro850 controller firmware revision 8 or higher, and
	• Connected Components Workbench Developer Edition software, version 8 or higher.

The following sample project guides you through the creation of a simple application for a Micro850 controller without any plug-in modules, and how to use the Run Mode Change feature.

Create the Project

1. Create a project for a Micro830/Micro85/Micro8700 controller without any plug-ins. Observe that the controller is disconnected.
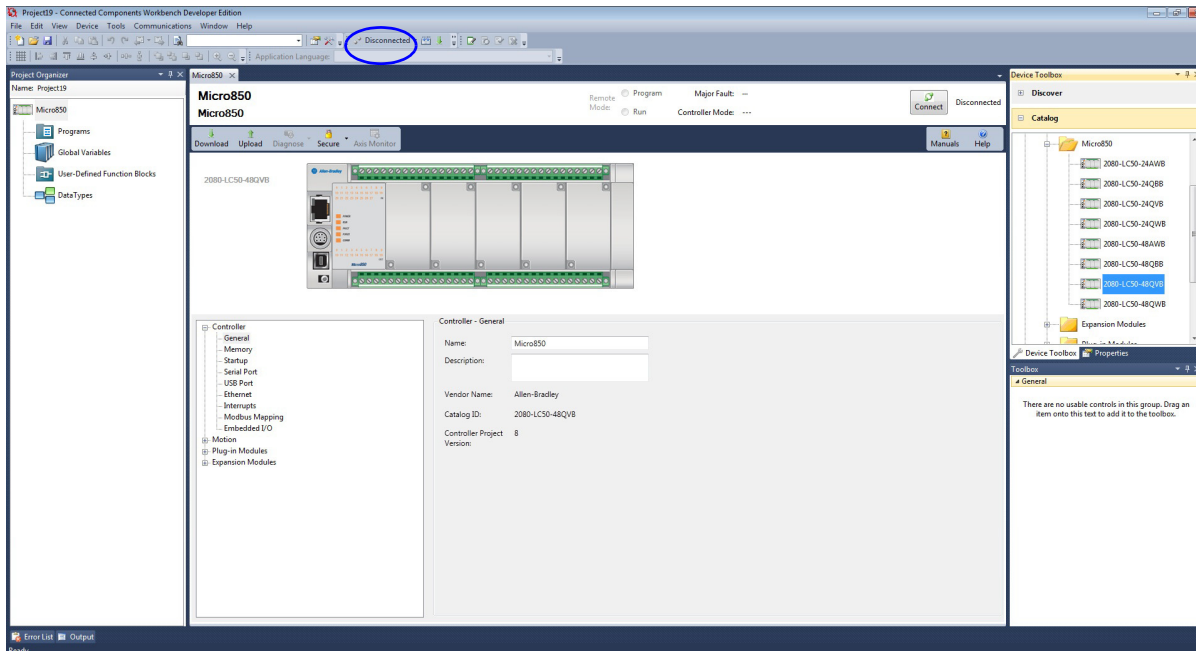
2. Right-click Programs and select Add > New LD: Ladder Diagram.

3. From the Toolbox, double-click Direct Coil to add it to the rung, or drag-and-drop Direct Coil onto the rung.

4. Double-click the newly added Direct Coil to bring up the Variable Selector dialog and select "_IO_EM_DO_00".
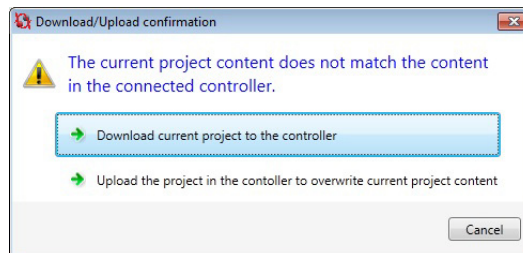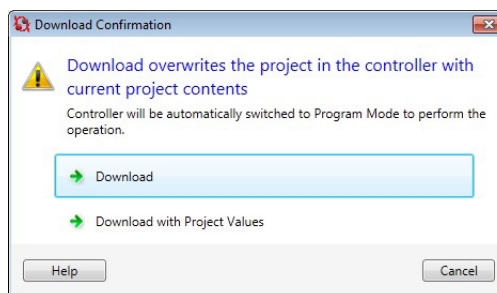

5. Build the project.
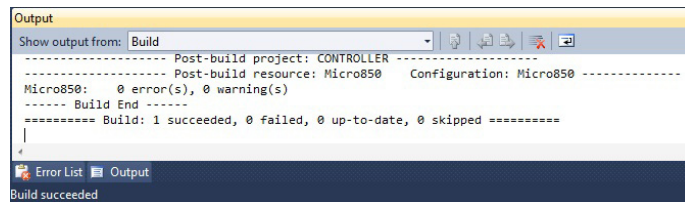
6. Download the project to the controller.
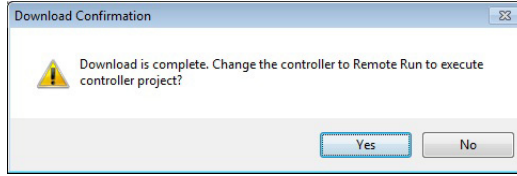In the Connection Browser dialog, select the Micro850 controller.
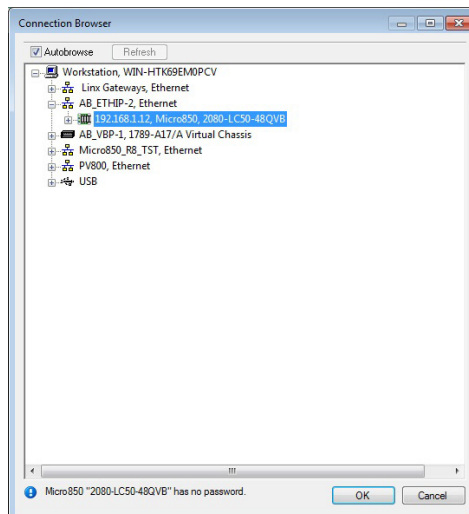
7. Select Download current project to the controller.

8. Select Download to confirm.

9. When the project has been downloaded to the controller, a prompt asking to change the controller to Remote Run mode appears. Select Yes.

10. Observe that the controller is now in Debug mode.

> **IMPORTANT** From Connected Components Workbench software version 8 onwards, selecting "Yes" to change the controller to Remote Run mode after a downloading a project automatically switches it to Debug mode.

Edit the Project Using Run Mode Change

Run Mode Change Toolbar

Run Mode Change   Test Logic Changes   Accept Changes   Undo Changes
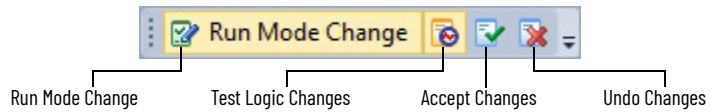
1. Click the Run Mode Change icon.
   Observe that the controller goes into Edit mode and is still connected.
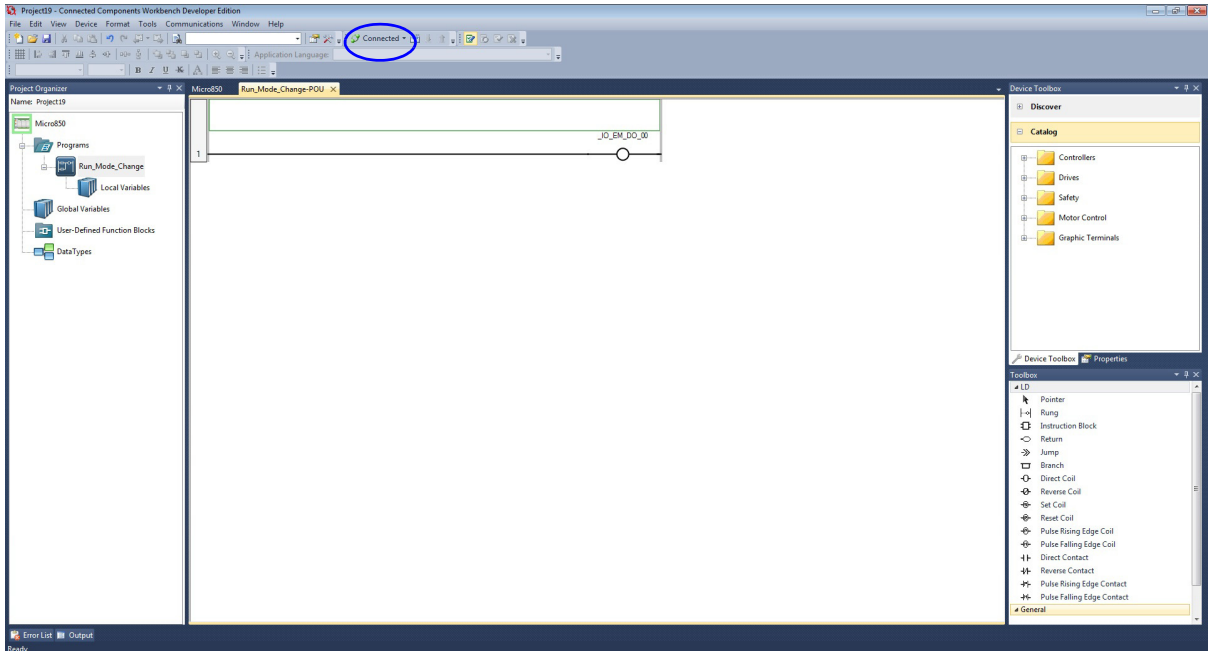
If you add a new variable during RMC, external data access and changing the access type (default is Read/Write) of this new variable is not available until you have chosen to Accept or Undo the Test Logic changes.

2. From the Toolbox, double-click Instruction Block to add it to the rung, or drag and drop Instruction Block onto the rung.

3. Double-click the newly added Instruction Block and select "Timer On/Off "(TONOFF).
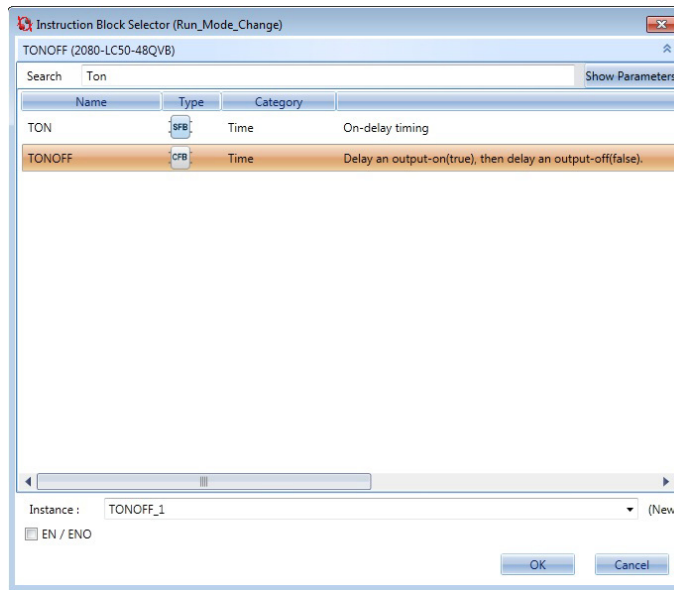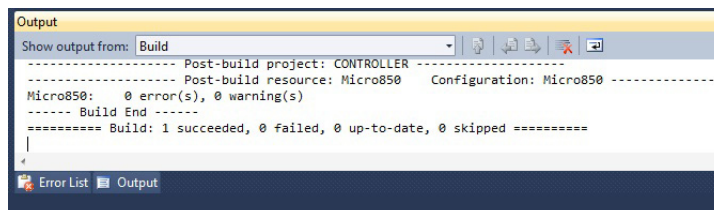

Configure the Instruction Block to trigger every one second.

4. From the Toolbox, double-click Reverse Contact to add it to the rung, or drag-and-drop Reverse Contact onto the run. Place it to the left of the recently added Instruction Block.
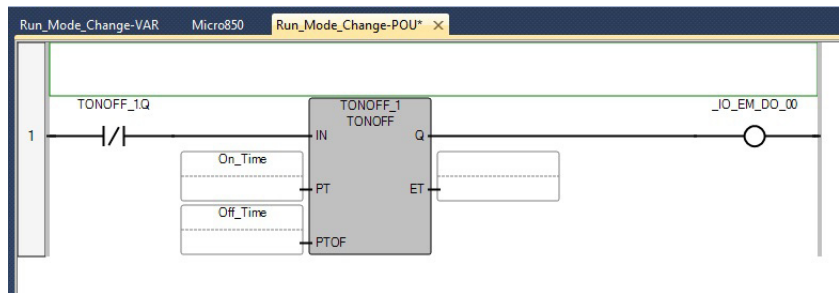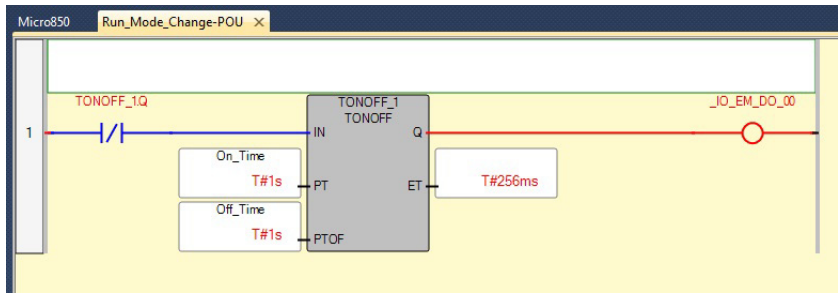

5. Select the Test Logic Changes [icon] icon to build the project and download it to the controller.

IMPORTANT	When a Test Logic is performed, or undoing changes after the Test Logic is completed, active communication instructions are aborted while the changes are downloaded to the controller.

6. The controller automatically goes into Debug mode and displays the updated project.

7. You can now choose to either Undo or Accept the changes to the project.

To Undo the Changes

1. Select the Undo Changes [icon] icon.
2. The changes are discarded and the original project is restored to the controller.

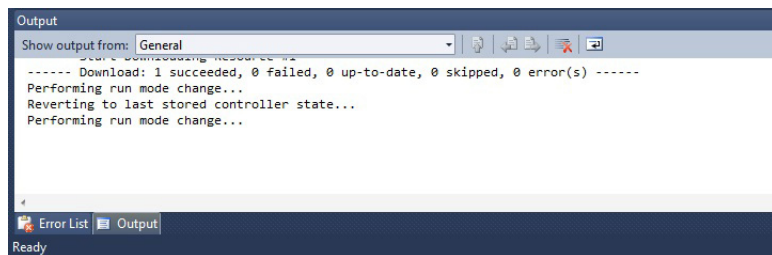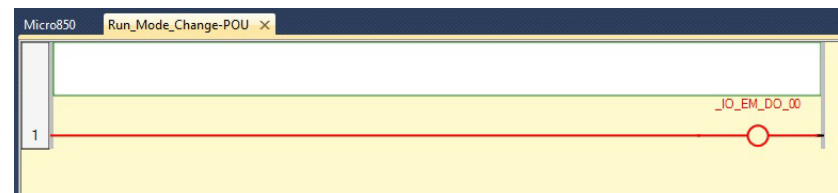| **IMPORTANT**	When a Test Logic is performed, or undoing changes after the Test Logic is completed, active communication instructions are aborted while the changes are downloaded to the controller.

Observe that the original project is shown and the controller is in Debug mode.

To Accept the Changes

1. Select the Accept Changes [icon] icon.
2. Observe that only the Run Mode Change icon is now enabled and the controller remains in Debug mode.

User Interrupts

Interrupts allow you to interrupt your program based on defined events. This chapter contains information about using interrupts, the interrupt instructions, and interrupt configuration.

For more information on HSC Interrupt, see .

Information About Using Interrupts

The purpose of this section is to explain some fundamental properties of the User Interrupts, including:

- What is an interrupt?
- When can the controller operation be interrupted?
- Priority of User Interrupts
- Interrupt Configuration
- User Fault Routine

What is an Interrupt?

An interrupt is an event that causes the controller to suspend the Program Organization Unit (POU) it is performing, perform another POU, and then return to the suspended POU at the point where it suspended. The Micro830, Micro850, and Micro870 controllers support the following User Interrupts:

- User Fault Routine
- Event Interrupts (8)
- High-Speed Counter Interrupts (6)
- Selectable Timed Interrupts (4)
- Plug-in Module Interrupts (5)

An interrupt must be configured and enabled to execute. When any one of the interrupts is configured (and enabled) and later occurs, the user program:
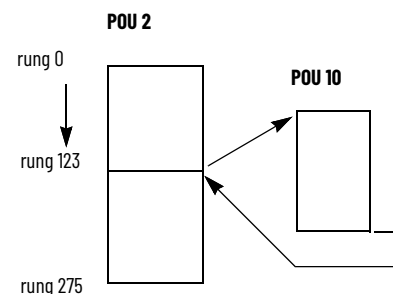
1. Suspends its execution of the current POU,
2. Performs a predefined POU based on which interrupt occurred, and
3. Returns to the suspended operation.

Interrupt Operation Example
POU 2 is the main control program.
POU 10 is the interrupt routine.
- An Interrupt Event occurs at rung 123.
- POU 10 is executed.
- POU 2 execution resumes immediately after POU 10 is scanned.

Specifically, if the controller program is executing normally and an interrupt event occurs:

1. The controller stops its normal execution.
2. Determines which interrupt occurred.
3. Goes immediately to the beginning of the POU specified for that User Interrupt.
4. Begins executing the User Interrupt POU (or set of POU/function blocks if the specified POU calls a subsequent function block).
5. Completes the POU.
6. Resumes normal execution from the point where the controller program was interrupted.

When Can the Controller Operation be Interrupted?

The Micro830 controllers allow interrupts to be serviced at any point of a program scan. Use UID/ UIE instructions to protect the program block that should not be interrupted.

Priority of User Interrupts

When multiple interrupts occur, the interrupts are serviced based on their individual priority.

When an interrupt occurs and another interrupt has already occurred but has not been serviced, the new interrupt is scheduled for execution based on its priority relative to the other pending interrupts. At the next point in time when an interrupt can be serviced, all interrupts are executed in the sequence of highest priority to lowest priority.

If an interrupt occurs while a lower priority interrupt is being serviced (executed), the currently executing interrupt routine is suspended, and the higher priority interrupt is serviced. Then the lower priority interrupt is allowed to complete before returning to normal processing.

If an interrupt occurs while a higher priority interrupt is being serviced (executed), and the pending bit has been set for the lower priority interrupt, the currently executing interrupt routine continues to completion. Then the lower priority interrupt runs before returning to normal processing.

Table 114 - Priorities From Highest to Lowest

User Fault Routine	Highest Priority
Event Interrupt0	
Event Interrupt1	
Event Interrupt2	
Event Interrupt3	
High-Speed Counter Interrupt0	
High-Speed Counter Interrupt1	
High-Speed Counter Interrupt2	
High-Speed Counter Interrupt3	
High-Speed Counter Interrupt4	
High-Speed Counter Interrupt5	
Event Interrupt4	
Event Interrupt5	
Event Interrupt6	
Event Interrupt7	
Selectable Timed Interrupt0	
Selectable Timed Interrupt1	
Selectable Timed Interrupt2	
Selectable Timed Interrupt3	
Plug-In Module Interrupt0, 1, 2, 3, 4	**Lowest Priority**

User Interrupt Configuration

User interrupts can be configured and set as AutoStart from the Interrupts window.
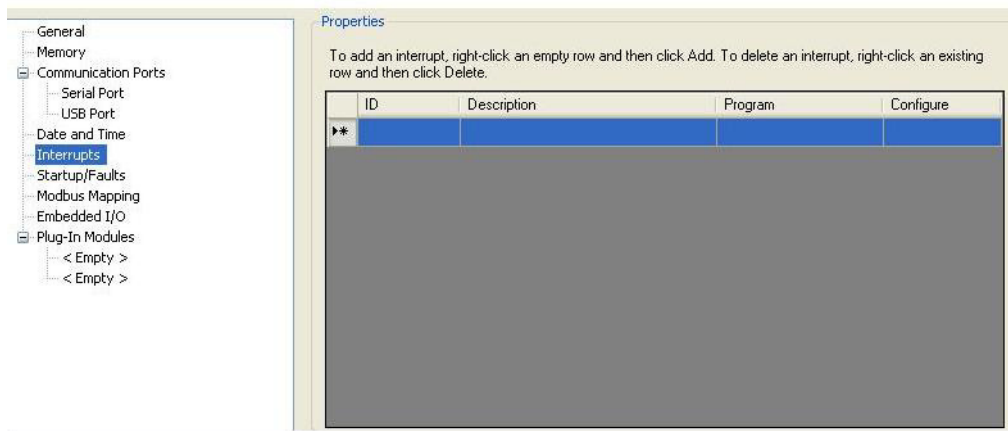
User Fault Routine

The user fault routine gives you the option of doing the cleanup before a controller shutdown, when a specific user fault occurs. The fault routine is executed when any user fault occurs. The fault routine is not executed for non-user faults.

The controller goes to Fault mode after a User Fault Routine is executed, and the User Program execution stops.

Creating a User Fault Subroutine

To use the user fault subroutine:

1. Create a POU.
2. In the User Interrupt Configuration window, configure this POU as a User Fault routine.
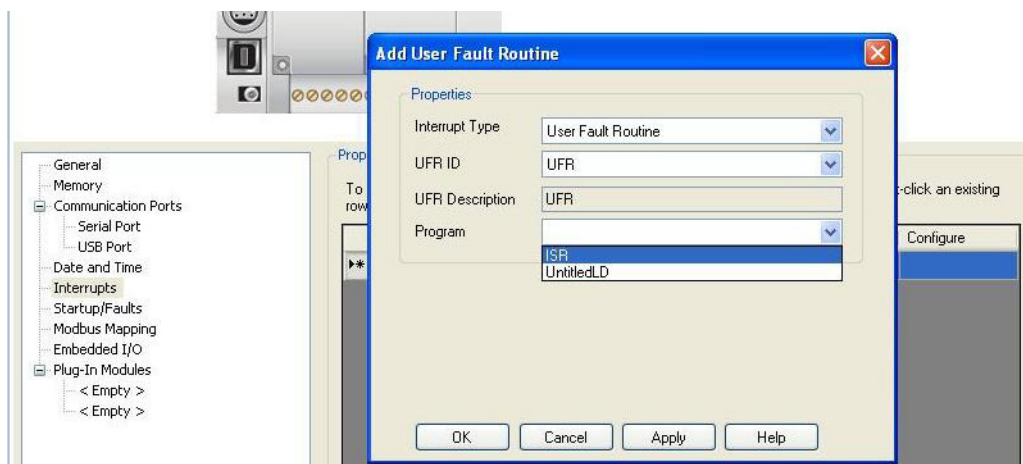
User Interrupt Instructions

Table 115 - List of User Interrupt Instructions

Instruction	Used To:	Page
STIS – Selectable Timed Start	Use the Selectable Timed Interrupt Start (STIS) instruction to the start the STI timer from the control program, rather than starting automatically.	298
UID – User Interrupt Disable	Use the User Interrupt Disable (UID) and the User Interrupt Enable (UIE) instructions to create zones in which user interrupts cannot occur.	299
UIE – User Interrupt Enable		300
UIF – User Interrupt Flush	Use the UIF instruction to remove the selected pending interrupts from the system.	301
UIC – User Interrupt Clear	Use this function to clear the Interrupt Lost bit for the selected User Interrupt.	302

STIS - Selectable Timed Start
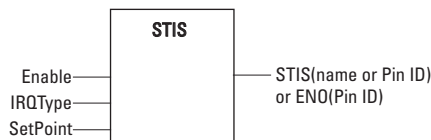
STIO is used in this document to define how STIS works.

Table 116 - STIS Parameters

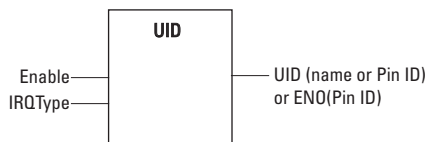| Parameter	Parameter Type	Data Type	Parameter Description
Enable	Input	BOOL	Enable Function When Enable = TRUE, the function is performed. When Enable = FALSE, the function is not performed.
IRQType	Input	UDINT	Use the STI defined DWORD IRQ_STIO, IRQ_STI1, IRQ_STI2, IRQ_STI3
SetPoint	Input	UINT	The user timer interrupt interval time value in milliseconds. When SetPoint = 0, STI is disabled. When SetPoint = 1...65535, STI is enabled.
STIS or ENO	Output	BOOL	Rung Status (same as Enable)

The STIS instruction can be used to start and stop the STI function or to change the time interval between STI user interrupts. The STI instruction has two operands:

- **IRQType** — This is the STI ID that you want to drive.
- **SetPoint** — This is the amount of time (in milliseconds) which must expire before executing the selectable timed user interrupt. A value of zero disables the STI function. The time range is from 0...65,535 milliseconds.

The STIS instruction applies the specified setpoint to the STI function as follows (STIO is used here as an example):

- If a zero setpoint is specified, the STI is disabled and STIO.Enable is cleared (0).
- If the STI is disabled (not timing) and a value greater than 0 is entered into the setpoint, the STI starts timing to the new setpoint and STIO.Enable is set (1).
- If the STI is timing and the setpoint is changed, the new setting takes effect immediately, restarting from zero. The STI continues to time until it reaches the new setpoint.

UID - User Interrupt Disable

```
              UID
           ┌──────────┐
 Enable────┤          ├──── UID (name or Pin ID)
IRQType────┤          │       or ENO(Pin ID)
           └──────────┘
```

The UID instruction is used to disable selected user interrupts. Table 117 shows the types of interrupts with their corresponding disable bits:

Table 117 - Types of Interrupts Disabled by the UID Instruction

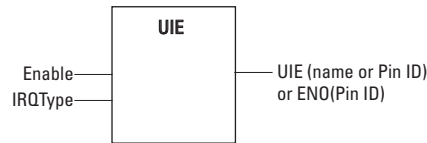| Interrupt Type	Element	Decimal Value	Corresponding Bit
Plug-In Module	UPM4	8388608	bit 23
Plug-In Module	UPM3	4194304	bit 22
Plug-In Module	UPM2	2097152	bit 21
Plug-In Module	UPM1	1048576	bit 20
Plug-In Module	UPM0	524288	bit 19
STI - Selectable Timed Interrupt	STI3	262144	bit 18
STI - Selectable Timed Interrupt	STI2	131072	bit 17
STI - Selectable Timed Interrupt	STI1	65536	bit 16
STI - Selectable Timed Interrupt	STI0	32768	bit 15
EII - Event Input Interrupt	Event 7	16384	bit 14
EII - Event Input Interrupt	Event 6	8192	bit 13
EII - Event Input Interrupt	Event 5	4096	bit 12
EII - Event Input Interrupt	Event 4	2048	bit 11
HSC - High-Speed Counter	HSC5	1024	bit 10
HSC - High-Speed Counter	HSC4	512	bit 9
HSC - High-Speed Counter	HSC3	256	bit 8
HSC - High-Speed Counter	HSC2	128	bit 7
HSC - High-Speed Counter	HSC1	64	bit 6
HSC - High-Speed Counter	HSC0	32	bit 5
EII - Event Input Interrupt	Event 3	16	bit 4
EII - Event Input Interrupt	Event 2	8	bit 3
EII - Event Input Interrupt	Event 1	4	bit 2
EII - Event Input Interrupt	Event 0	2	bit 1
UFR - User Fault Routine Interrupt	UFR	1	bit 0 (reserved)

To disable interrupts:

1. Select the interrupts that you want to disable.
2. Find the Decimal Value for the interrupts that you selected.
3. Add the Decimal Values if you selected multiple types of interrupt.
4. Enter the sum into the UID instruction.

For example, to disable EII Event 1 and EII Event 3:
EII Event 1 = 4, EII Event 3 = 16
4 + 16 = 20 (enter this value)

UIE - User Interrupt Enable

```
              ┌──────────┐
              │   UIE    │
  Enable──────┤          ├────── UIE (name or Pin ID)
  IRQType─────┤          │       or ENO(Pin ID)
              └──────────┘
```

Use the UIE instruction to enable selected user interrupts. Table 118 shows the types of interrupts with their corresponding enable bits:

Table 118 - Types of Interrupts Enabled by the UIE Instruction

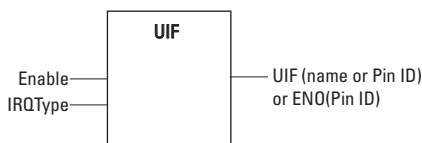| Interrupt Type	Element	Decimal Value	Corresponding Bit
Plug-In Module	UPM4	8388608	bit 23
Plug-In Module	UPM3	4194304	bit 22
Plug-In Module	UPM2	2097152	bit 21
Plug-In Module	UPM1	1048576	bit 20
Plug-In Module	UPM0	524288	bit 19
STI - Selectable Timed Interrupt	STI3	262144	bit 18
STI - Selectable Timed Interrupt	STI2	131072	bit 17
STI - Selectable Timed Interrupt	STI1	65536	bit 16
STI - Selectable Timed Interrupt	STI0	32768	bit 15
EII - Event Input Interrupt	Event 7	16384	bit 14
EII - Event Input Interrupt	Event 6	8192	bit 13
EII - Event Input Interrupt	Event 5	4096	bit 12
EII - Event Input Interrupt	Event 4	2048	bit 11
HSC - High-Speed Counter	HSC5	1024	bit 10
HSC - High-Speed Counter	HSC4	512	bit 9
HSC - High-Speed Counter	HSC3	256	bit 8
HSC - High-Speed Counter	HSC2	128	bit 7
HSC - High-Speed Counter	HSC1	64	bit 6
HSC - High-Speed Counter	HSC0	32	bit 5
EII - Event Input Interrupt	Event 3	16	bit 4
EII - Event Input Interrupt	Event 2	8	bit 3
EII - Event Input Interrupt	Event 1	4	bit 2
EII - Event Input Interrupt	Event 0	2	bit 1
		1	bit 0 (reserved)

To enable interrupts:

1. Select the interrupts that you want to enable.
2. Find the Decimal Value for the interrupt that you selected.
3. Add the Decimal Values if you selected multiple types of interrupt.
4. Enter the sum into the UIE instruction.

For example, to enable EII Event 1 and EII Event 3:
EII Event 1 = 4, EII Event 3 = 16
4 + 16 = 20 (enter this value)

UIF - User Interrupt Flush

```
                    ┌──────────┐
                    │   UIF    │
                    │          │──── UIF (name or Pin ID)
   Enable ──────────┤          │     or ENO(Pin ID)
   IRQType──────────┤          │
                    │          │
                    └──────────┘
```

Use the UIF instruction to flush (remove pending interrupts from the system) selected user interrupts. Table 119 shows the types of interrupts with their corresponding flush bits:

Table 119 - Types of Interrupts Disabled by the UIF Instruction

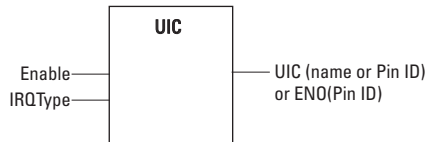| Interrupt Type	Element	Decimal Value	Corresponding Bit
Plug-In Module	UPM4	8388608	bit 23
Plug-In Module	UPM3	4194304	bit 22
Plug-In Module	UPM2	2097152	bit 21
Plug-In Module	UPM1	1048576	bit 20
Plug-In Module	UPM0	524288	bit 19
STI - Selectable Timed Interrupt	STI3	262144	bit 18
STI - Selectable Timed Interrupt	STI2	131072	bit 17
STI - Selectable Timed Interrupt	STI1	65536	bit 16
STI - Selectable Timed Interrupt	STI0	32768	bit 15
EII - Event Input Interrupt	Event 7	16384	bit 14
EII - Event Input Interrupt	Event 6	8192	bit 13
EII - Event Input Interrupt	Event 5	4096	bit 12
EII - Event Input Interrupt	Event 4	2048	bit 11
HSC - High-Speed Counter	HSC5	1024	bit 10
HSC - High-Speed Counter	HSC4	512	bit 9
HSC - High-Speed Counter	HSC3	256	bit 8
HSC - High-Speed Counter	HSC2	128	bit 7
HSC - High-Speed Counter	HSC1	64	bit 6
HSC - High-Speed Counter	HSC0	32	bit 5
EII - Event Input Interrupt	Event 3	16	bit 4
EII - Event Input Interrupt	Event 2	8	bit 3
EII - Event Input Interrupt	Event 1	4	bit 2
EII - Event Input Interrupt	Event 0	2	bit 1
UFR - User Fault Routine Interrupt	UFR	1	bit 0 (reserved)

To flush interrupts:

1. Select the interrupts that you want to flush.
2. Find the Decimal Value for the interrupt that you selected.
3. Add the Decimal Values if you selected multiple types of interrupt.
4. Enter the sum into the UIF instruction.

For example, to disable EII Event 1 and EII Event 3:
EII Event 1 = 4, EII Event 3 = 16
4 + 16 = 20 (enter this value)

UIC – User Interrupt Clear

```
                    ┌─────────┐
                    │   UIC   │
          Enable────┤         ├──── UIC (name or Pin ID)
         IRQType────┤         │     or ENO(Pin ID)
                    └─────────┘
```

This C function clears the Interrupt Lost bit for the selected User Interrupts.

Table 120 – Types of Interrupts Disabled by the UIC Instruction

Interrupt Type	Element	Decimal Value	Corresponding Bit
Plug-In Module	UPM4	8388608	bit 23
Plug-In Module	UPM3	4194304	bit 22
Plug-In Module	UPM2	2097152	bit 21
Plug-In Module	UPM1	1048576	bit 20
Plug-In Module	UPM0	524288	bit 19
STI - Selectable Timed Interrupt	STI3	262144	bit 18
STI - Selectable Timed Interrupt	STI2	131072	bit 17
STI - Selectable Timed Interrupt	STI1	65536	bit 16
STI - Selectable Timed Interrupt	STI0	32768	bit 15
EII - Event Input Interrupt	Event 7	16384	bit 14
EII - Event Input Interrupt	Event 6	8192	bit 13
EII - Event Input Interrupt	Event 5	4096	bit 12
EII - Event Input Interrupt	Event 4	2048	bit 11
HSC - High-Speed Counter	HSC5	1024	bit 10
HSC - High-Speed Counter	HSC4	512	bit 9
HSC - High-Speed Counter	HSC3	256	bit 8
HSC - High-Speed Counter	HSC2	128	bit 7
HSC - High-Speed Counter	HSC1	64	bit 6
HSC - High-Speed Counter	HSC0	32	bit 5
EII - Event Input Interrupt	Event 3	16	bit 4
EII - Event Input Interrupt	Event 2	8	bit 3
EII - Event Input Interrupt	Event 1	4	bit 2
EII - Event Input Interrupt	Event 0	2	bit 1
UFR - User Fault Routine Interrupt	UFR	1	bit 0 (reserved)

Using the Selectable Timed Interrupt (STI) Function

Configure the STI function from the Interrupt Configuration window.
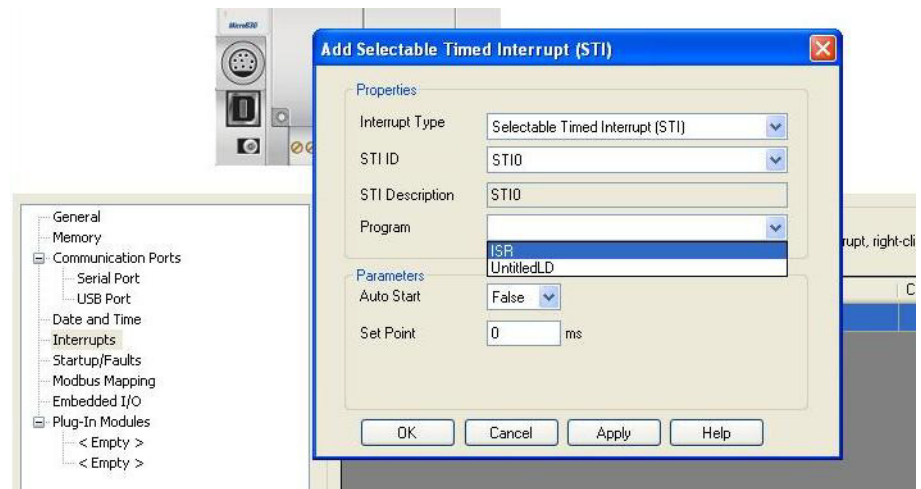
The Selectable Timed Interrupt (STI) provides a mechanism to solve time critical control requirements. The STI is a trigger mechanism that allows you to scan or solve control program logic that is time sensitive.

Example of where you would use the STI are:

- PID type applications, where a calculation must be performed at a specific time interval.
- A block of logic that must be scanned more often.

How an STI is used is typically driven by the demands/requirements of the application. It operates using the following sequence:

1. Select a time interval.
2. When a valid interval is set and the STI is properly configured, the controller monitors the STI value.
3. When the time period has elapsed, the controller's normal operation is interrupted.
4. The controller then scans the logic in the STI POU.
5. When the STI POU is completed, the controller returns to where it was before the interrupt and continues normal operation.

Selectable Time Interrupt (STI) Function Configuration and Status

This section covers the configuration and status management of the STI function.

STI Function Configuration

STI Program POU

This is the name of the Program Organizational Unit (POU) which is executed immediately when this STI Interrupt occurs. You can choose any pre-programmed POU from the dropdown list.

STI Auto Start (STI0.AS)

Sub-element Description	Data Format	User Program Access
AS - Auto Start	Binary (bit)	Read-only

The AS (Auto Start) is a control bit that can be used in the control program. The auto start bit is configured with the programming device and stored as part of the user program. The auto start bit automatically sets the STI Timed Interrupt Enable (STI0.Enabled) bit when the controller enters any executing mode.

STI Set Point Milliseconds Between Interrupts (STI0.SP)

Sub-element Description	Data Format	Range	User Program Access
SP - Set Point Msec	Word (INT)	0...65,535	Read/write

When the controller transitions to an executing mode, the set point in milliseconds (SP) value is loaded into the STI. If the STI is configured correctly, and enabled, the POU in the STI configuration is executed at this interval. You can change the value from the control program by using the STIS instruction.

> The minimum value cannot be less than the time required to scan the STI POUplus the Interrupt Latency.

STI Function Status Information

You can monitor the STI Function status bits either in the User Program, or in the Connected Components Workbench software, in Debug mode.

STI User Interrupt Executing (STI0.EX)

Sub-element Description	Data Format	User Program Access
EX - User Interrupt Executing	Binary (bit)	Read-only

The User Interrupt Executing (EX) bit is set whenever the STI mechanism completes timing and the controller is scanning the STI POU. The EX bit is cleared when the controller completes processing the STI subroutine.

The STI EX bit can be used in the control program as conditional logic to detect if an STI interrupt is executing.

STI User Interrupt Enable (STI0.Enabled)

Sub-element Description	Data Format	User Program Access
Enabled - User Interrupt Enable	Binary (bit)	Read-only

The User Interrupt Enable bit is used to indicate STI enable or disable status.

STI User Interrupt Lost (STI0.LS)

Sub-element Description	Data Format	User Program Access
LS - User Interrupt Lost	Binary (bit)	Read/write

The LS is a status flag that indicates an interrupt was lost. The controller can process 1 active and maintain up to 1 pending user interrupt conditions before it sets the lost bit.

The controller sets this bit. It is up to the control program to use, track, the lost condition if necessary.

STI User Interrupt Pending (STI0.PE)

Sub-element Description	Data Format	User Program Access
PE - User Interrupt Pending	Binary (bit)	Read-only

The PE is a status flag that represents an interrupt is pending. This status bit can be monitored or used for logic purposes in the control program if you must determine when a subroutine cannot execute immediately.

The controller set this bit and clears this bit automatically. The controller can process 1 active and maintain up to 1 pending user interrupt conditions before it sets the lost bit.
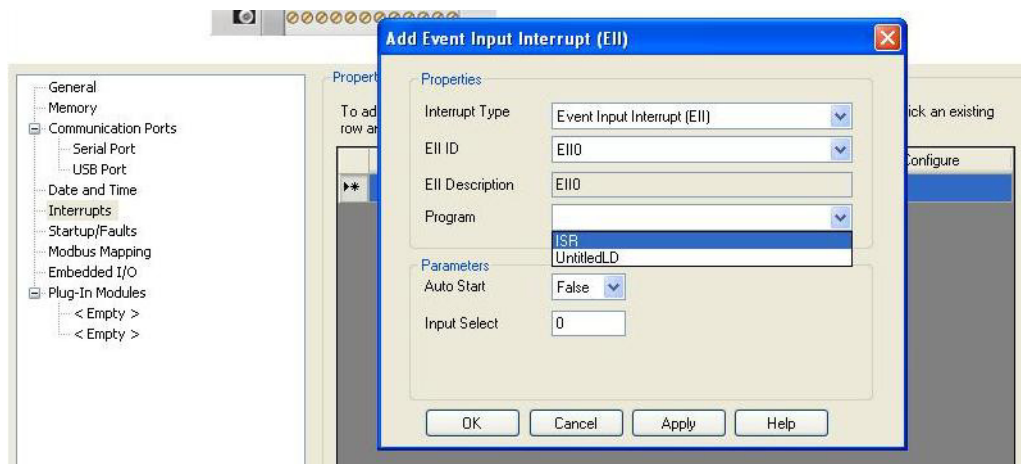
Using the Event Input Interrupt (EII) Function

The Event Input Interrupt (EII) is a feature that allows you to scan a specific POU when an input condition is detected from a field device.

EII0 is used in this document to define how EII works.

Configure EII Input Edge from the Embedded I/O configuration window.

Configure the EII from the Interrupt Configuration window.

Event Input Interrupt (EII) Function Configuration and Status

EII Function Configuration

The Event Input Interrupt Function has the following related configuration parameters.

EII Program POU

This is the name of the Program Organizational Unit (POU) which is executed immediately when this EII Interrupt occurs. You can choose any pre-programmed POU from the dropdown list.

EII Auto Start (EII0.AS)

Sub-element Description	Data Format	User Program Access
AS - Auto Start	Binary (bit)	Read-only

Auto Start (AS) is a control bit that can be used in the control program. The auto start bit is configured with the programming device and stored as part of the user program. The auto start bit automatically sets the Event User Interrupt Enable bit when the controller enters any executing mode.

EII Input Select (EII0.IS)

Sub-element Description	Data Format	User Program Access
IS - Input Select	Word (INT)	Read-only

The Input Select (IS) parameter is used to configure each EII to a specific input on the controller. Valid inputs are 0...N, where N is either 15, or the maximum input ID, whichever is smaller.

You can only configure this parameter with the programming device. The parameter cannot be changed from the control program.

EII Function Status Information

You can monitor the EII Function status bits either in the User Program, or in the Connected Components Workbench software, in Debug mode.

EII User Interrupt Executing (EII0.EX)

Sub-element Description	Data Format	User Program Access
EX - User Interrupt Executing	Binary (bit)	Read-only

The User Interrupt Executing (EX) bit is set whenever the EII mechanism detects a valid input and the controller is scanning the EII POU. The EII mechanism clears the EX bit when the controller completes its processing of the EII subroutine.

You can use the EII EX bit in the control program as conditional logic to detect if an EII interrupt is executing.

EII User Interrupt Enable (EII0.Enabled)

Sub-element Description	Data Format	User Program Access
Enabled - User Interrupt Enable	Binary (bit)	Read-only

The Enabled (User Interrupt Enable) bit indicates the EII enable or disable status.

EII User Interrupt Lost (EII0.LS)

Sub-element Description	Data Format	User Program Access
LS - User Interrupt Lost	Binary (bit)	Read/write

User Interrupt Lost (LS) is a status flag that represents an interrupt has been lost. The controller can process 1 active and maintain up to 1 pending user interrupt conditions before it sets the lost bit.

The controller sets the bit. It is up to the control program to use or track, the lost condition if necessary.

EII User Interrupt Pending (EII0.PE)

Sub-element Description	Data Format	User Program Access
PE - User Interrupt Pending	Binary (bit)	Read-only

User Interrupt Pending (PE) is a status flag that represents an interrupt is pending. You can monitor the status bit or use it for logic purposes in the control program if you must determine when a subroutine cannot execute immediately.
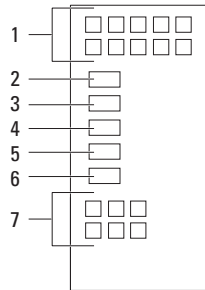
The controller automatically sets and clears the bit. The controller can process 1 active and maintain up to 1 pending user interrupt conditions before it sets the lost bit.
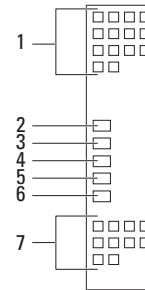
Troubleshooting

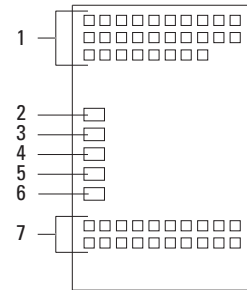## Status Indicators on the Controller

Micro830 Controllers
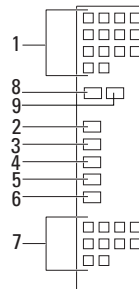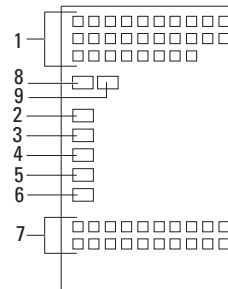

Micro850 Controllers

Micro870 Controllers
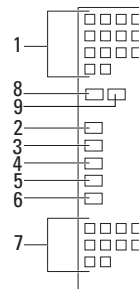
Status Indicator Description

	Description	State	Indicates
1	Input status	Off	Input is not energized
		On	Input is energized (terminal status)
2	Power status	Off	No input power, or power error condition
		Green	Power on
3	Run status	Off	Not executing the user program
		Green	Executing the user program in run mode
		Flashing green	Memory backup/restore in progress
4	Fault status	Off	No fault detected
		Red	Nonrecoverable fault that requires a power cycle
		Flashing Red	Recoverable fault
5	Force status	Off	No force conditions are active.
		Amber	Force conditions are active.
6	Serial communications status	Off	No traffic for RS-232/RS-485
		Green	Traffic through RS-232/RS-485 The indicator only blinks when transmitting data. It does not blink when receiving data.
7	Output status	Off	Output is not energized
		On	Output is energized (logic status)
8	Module status	Steady Off	No power
		Flashing Green	Standby
		Steady Green	Device operational
		Flashing Red	Minor fault (minor and major recoverable faults)
		Steady Red	Major fault (nonrecoverable fault)
		Flashing Green and Red	Self-test The device is performing power on self test (POST). During POST, the network status indicator alternates flashing green and red. The duration of the self-test depends on the size of the project in the controller.
9	Network status	Steady Off	Not powered, no IP address. The device is powered off, or is powered on but with no IP address.
		Flashing Green	No connections An IP address is configured, but no Ethernet application is connected.
		Steady Green	Connected At least one EtherNet/IP session is established.
		Flashing Red	Connection timeout (not implemented).
		Steady Red	Duplicate IP The device has detected that its IP address is being used by another device in the network. This status is applicable only if the device's duplicate IP address detection (ACD) feature is enabled.
		Flashing Green and Red	Self-test The device is performing power on self test (POST). During POST, the network status indicator alternates flashing green and red. The duration of the self-test depends on the size of the project in the controller.

Normal Operation

The POWER and RUN indicators are on. If a force condition is active, the FORCE indicator turns on and remains on until all forces are removed.

Error Codes

This section lists possible error codes for your controller and recommended actions for recovery. Information about the fault is stored in a fault log, which can be accessed from the Diagnostics page in the Connected Components Workbench software. The fault log contains brief information about the last fault, and detailed information about the last 10 nonrecoverable faults that occurred.

If an error persists after performing the recommended action, contact your local Rockwell Automation technical support representative. For contact information, go to rok.auto/support.

Fault Types

There are two basic types of faults that can occur:

- Recoverable – A recoverable fault can be cleared without having to power cycle the controller. The fault LED flashes red when a recoverable fault occurs.
- Nonrecoverable – A nonrecoverable fault requires the controller to be power cycled before clearing the fault. After the controller has been power cycled or reset, check the fault log in the Diagnostic page of the Connected Components Workbench software, then clear the fault. The fault LED is steady red when a nonrecoverable fault occurs.

Table 121 - List of Error Codes for Micro800 Controllers

Error Code	Fault Type	Description	Recommended Action
0xF000	Recoverable	The controller was unexpectedly reset due to a noisy environment or an internal hardware failure. If the system variable _SYSVA_USER_DATA_LOST is set, the controller is able to recover the user program but the user data is cleared. If not, the Micro800 controller program is cleared.	Perform one of the following: • See Corrective Actions for Recoverable Faults on page 314. • Check wiring to eliminate any noise.
0xF001	Recoverable	The controller program has been cleared. This happened because: • A power-down occurred during program download or data transfer from the memory module. • The cable was removed from the controller during program download. • The RAM integrity test failed.	Perform one of the following: • See Corrective Actions for Recoverable Faults on page 314. • Transfer the program using the memory module restore utility.
0xF002	Nonrecoverable	The controller hardware watchdog was activated. The controller hardware watchdog timeout happens if the program scan is more than 3 seconds. If the system variable _SYSVA_USER_DATA_LOST is set, the controller is able to recover the user program but the user data is cleared. If not, the Micro800 controller program is cleared.	See Corrective Actions for Nonrecoverable Faults on page 314.
0xF003	Recoverable	One of the following occurred: • The memory module hardware faulted. • The memory module connection faulted. • The memory module was incompatible with the Micro800 controller's firmware revision.	Perform one of the following: • Remove the memory module and plug it in again. • Obtain a new memory module. • See Corrective Actions for Recoverable Faults on page 314. • Upgrade the Micro800 controller's firmware revision to be compatible with the memory module. For information on firmware revision compatibility, go to rok.auto/pcdc.
0xF004	Recoverable	A failure occurred during the memory module data transfer.	Perform one of the following: • See Corrective Actions for Recoverable Faults on page 314. • Attempt the data transfer again. If the error persists, replace the memory module. • For Embedded RTC failure, restart the controller.
0xF005	Recoverable	The user program failed an integrity check while the Micro800 controller was in Run mode.	Perform one of the following: • See Corrective Actions for Recoverable Faults on page 314. • Check wiring.
0xF006	Recoverable	The user program is incompatible with the Micro800 controller's firmware revision.	Perform one of the following: • See Corrective Actions for Recoverable Faults on page 314. • Contact your local Rockwell Automation technical support representative.
0xF010	Recoverable	The user program contains a function/function block that is not supported by the Micro800 controller.	Perform one of the following: • See Corrective Actions for Recoverable Faults on page 314. • Contact your local Rockwell Automation technical support representative.

Table 121 – List of Error Codes for Micro800 Controllers (Continued)

Error Code	Fault Type	Description	Recommended Action
0xF014	Recoverable	A memory module memory error occurred.	• See Corrective Actions for Recoverable Faults on page 314. • Reprogram the memory module. If the error persists, replace the memory module.
0xF015	Nonrecoverable	An unexpected software error occurred.	Perform one of the following: • See Corrective Actions for Nonrecoverable Faults on page 314. • Check wiring.
0xF016	Nonrecoverable	An unexpected hardware error occurred.	Perform one of the following: • See Corrective Actions for Nonrecoverable Faults on page 314. • Check wiring.
0xF017	Nonrecoverable	An unexpected software error occurred due to an unexpected hardware interrupt. If the system variable _SYSVA_USER_DATA_LOST is set, the controller is able to recover the user program but the user data is cleared. If not, the Micro800 controller program is cleared.	Perform one of the following: • See Corrective Actions for Nonrecoverable Faults on page 314. • Check wiring.
0xF018	Nonrecoverable	An unexpected software error occurred due to SPI communication failure. If the system variable _SYSVA_USER_DATA_LOST is set, the controller is able to recover the user program but the user data is cleared. If not, the Micro800 controller program is cleared.	Perform one of the following: • See Corrective Actions for Nonrecoverable Faults on page 314. • Check wiring.
0xF019	Nonrecoverable	An unexpected software error occurred due to memory or other controller resource issue.	See Corrective Actions for Nonrecoverable Faults on page 314.
0xF01A	Recoverable	The controller was unexpectedly reset during Run Mode Change (RMC) due to a noisy environment or an internal hardware failure. If the system variable _SYSVA_USER_DATA_LOST is set, the controller is able to recover the user program but the user data is cleared. If not, the Micro800 controller program is cleared.	See Corrective Actions for Recoverable Faults on page 314.
0xF020	Recoverable	The base hardware is faulted or is incompatible with the Micro800 controller's firmware revision.	See Corrective Actions for Recoverable Faults on page 314.
0xF021	Recoverable	The I/O configuration in the user program is invalid or does not exist in the Micro800 controller.	See Corrective Actions for Recoverable Faults on page 314.
0xF022	Recoverable	The user program in the memory module is incompatible with the Micro800 controller's firmware revision.	Perform one of the following: • See Corrective Actions for Recoverable Faults on page 314. • Replace the memory module.
0xF023	Nonrecoverable	The controller program has been cleared. This happened because: • A power down occurred during program download or transfer from the memory module. • The Flash Integrity Test failed (Micro810 only).	Perform one of the following: • See Corrective Actions for Nonrecoverable Faults on page 314. • Download or transfer the program.
0xF030 0xF031 0xF032 0xF033	Recoverable	Power down information in persistent memory may not be written properly due to a noisy environment or an internal hardware failure. If the system variable _SYSVA_USER_DATA_LOST is set, the controller is able to recover the user program but the user data is cleared. If not, the Micro800 controller program is cleared.	See Corrective Actions for Recoverable Faults on page 314.
0xF050	Recoverable	The embedded I/O configuration in the user program is invalid.	See Corrective Actions for Recoverable Faults on page 314.
0xF100	Recoverable	There is a general configuration error, which is detected in the motion configuration that is downloaded from the Connected Components Workbench software, such as number of axes, or motion execution interval being configured out of range.	Perform the following: • See Corrective Actions for Recoverable Faults on page 314. • Correct the axes configuration in the user program.
0xF110	Recoverable	There is motion resource missing, such as Motion_DIAG variable not defined.	Perform the following: • See Corrective Actions for Recoverable Faults on page 314. • Correct the axes configuration in the user program.
0xF12z[1]	Recoverable	Motion configuration for axis z cannot be supported by this controller model, or the axis configuration has some resource conflict with some other motion axis, which has been configured earlier.	Perform the following: • See Corrective Actions for Recoverable Faults on page 314. • Remove all axes and reconfigure motion with the guidance from the User Manual.

Table 121 – List of Error Codes for Micro800 Controllers (Continued)

Error Code	Fault Type	Description	Recommended Action
0xF15z[1]	Recoverable	There is a motion engine logic error (firmware logic issue or memory crash) for one axis that is detected during motion engine cyclic operation. One possible reason can be motion engine data/memory crash.	See <u>Corrective Actions for Recoverable Faults on page 314</u>.
0xF210	Recoverable	The expansion I/O terminator is missing.	Perform the following: 1. Power off the controller. 2. Attach the expansion I/O terminator on the last expansion I/O module on the system. 3. Power on the controller. 4. See <u>Corrective Actions for Recoverable Faults on page 314</u>.
0xF230	Recoverable	The maximum number of expansion I/O modules has been exceeded.	Perform the following: 1. Power off the controller. 2. Check that the number of expansion I/O modules is not more than four. 3. Power on the controller. 4. See <u>Corrective Actions for Recoverable Faults on page 314</u>.
0xF250	Recoverable	There is a nonrecoverable error and the expansion I/O modules could not be detected.	See <u>Corrective Actions for Recoverable Faults on page 314</u>.
0xF26z[2]	Recoverable	An expansion I/O master fault is detected on the system.	See <u>Corrective Actions for Recoverable Faults on page 314</u>.
0xF27z[2]	Recoverable	A nonrecoverable communication fault has occurred on the expansion I/O module.	Perform one of the following: • See <u>Corrective Actions for Recoverable Faults on page 314</u>. • Replace the slot number z module.
0xF28z[2]	Recoverable	Expansion I/O communication rate error.	Perform one of the following: • See <u>Corrective Actions for Recoverable Faults on page 314</u>. • Replace the slot number z module.
0xF29z[2]	Recoverable	A module fault is detected on your expansion I/O module.	Perform one of the following: • See <u>Corrective Actions for Recoverable Faults on page 314</u>. • Replace the slot number z module.
0xF2Az[2]	Recoverable	Expansion I/O power failure	Perform one of the following: • See <u>Corrective Actions for Recoverable Faults on page 314</u>. • Replace the slot number z module.
0xF2Bz[2]	Recoverable	Expansion I/O configuration fault.	Perform one of the following: • See <u>Corrective Actions for Recoverable Faults on page 314</u>. • Correct the expansion IO module configuration in the user program to match that of the actual hardware configuration. • Check the expansion I/O module operation and condition. • Replace the expansion I/O module.
0xF300	Recoverable	The memory module is present but memory module is empty and restore operation is requested.	Perform the following: • See <u>Corrective Actions for Recoverable Faults on page 314</u>. • Check that there is a valid project in the memory module. • Download a user program and use the backup function to the memory module.
0xF301	Recoverable	The memory module's project is not compatible with the controller.	Perform one of the following: • See <u>Corrective Actions for Recoverable Faults on page 314</u>. • Check that there is a user program with a controller that has the correct controller catalog configured. • Download a user program and use the backup function to the memory module.
0xF302	Recoverable	The password is mismatched between memory module and controller. Only applies to the Micro820 controller when Remote LCD performs the restore operation. This fault does not apply to Micro800 controller firmware revision 10 and later.	Perform one of the following: • See <u>Corrective Actions for Recoverable Faults on page 314</u>. • Check that the user program in the memory module has the correct password. • Download a user program with a password and use the backup function to the memory module. • Use the Connected Components Workbench software to enter the correct password into the controller and perform the restore operation again.
0xF303	Recoverable	The memory module is not present and restore operation is requested.	Check that the memory module is present.
0xF0Az[3]	Recoverable	The plug-in I/O module experienced an error during operation.	Perform the following: • Check the condition and operation of the plug-in I/O module. • See <u>Corrective Actions for Recoverable Faults on page 314</u>.

Table 121 – List of Error Codes for Micro800 Controllers (Continued)

Error Code	Fault Type	Description	Recommended Action
0xF0Bz[3]	Recoverable	The plug-in I/O module configuration does not match the actual I/O configuration detected.	Perform one of the following: • See <u>Corrective Actions for Recoverable Faults on page 314</u>. • Correct the plug-in I/O module configuration in the user program to match that of the actual hardware configuration. • Check the condition and operation of the plug-in I/O module. • Replace the plug-in I/O module.
0xF0Dz[3]	Recoverable	When power was applied to the plug-in I/O module or the plug-in I/O module was removed, a hardware error occurred.	Perform the following: 1. See <u>Corrective Actions for Recoverable Faults on page 314</u>. 2. Correct the plug-in I/O module configuration in the user program. 3. Build and download the program using Connected Components Workbench software. 4. Put the Micro800 controller into Run mode.
0xF0Ez[3]	Recoverable	The plug-in I/O module configuration does not match the actual I/O configuration detected.	Perform the following: 1. See <u>Corrective Actions for Recoverable Faults on page 314</u>. 2. Correct the plug-in I/O module configuration in the user program. 3. Build and download the program using Connected Components Workbench software. 4. Put the Micro800 controller into Run mode.
0xF830	Recoverable	An error occurred in the EII configuration.	Perform the following: • See <u>Corrective Actions for Recoverable Faults on page 314</u>. • Review and change the EII configuration in the Micro800 controller properties.
0xF840	Recoverable	An error occurred in the HSC configuration.	Perform the following: • See <u>Corrective Actions for Recoverable Faults on page 314</u>. • Review and change the EII configuration in the Micro800 controller properties.
0xF850	Recoverable	An error occurred in the STI configuration.	Perform the following: • See <u>Corrective Actions for Recoverable Faults on page 314</u>. • Review and change the EII configuration in the Micro800 controller properties.
0xF860	Recoverable	A data overflow occurred. A data overflow error is generated when the ladder, structured text, or function block diagram execution encounters a divide-by-zero.	Perform the following: 1. See <u>Corrective Actions for Recoverable Faults on page 314</u>. 2. Correct the program to verify that there is no data overflow. 3. Build and download the program using Connected Components Workbench software. 4. Put the Micro800 controller into Run mode.
0xF870	Recoverable	An index address was out of data space.	Perform the following: 1. See <u>Corrective Actions for Recoverable Faults on page 314</u>. 2. Correct the program to verify that there is no index used to access an array element beyond the array boundaries. 3. Build and download the program using Connected Components Workbench software. 4. Put the Micro800 controller into Run mode.
0xF0878	Recoverable	An index used to access a bit is beyond the boundaries of the data type it is used on.	Perform the following: 1. See <u>Corrective Actions for Recoverable Faults on page 314</u>. 2. Correct the program to verify that there is no index used to access a bit beyond the boundaries of the data type. 3. Build and download the program using Connected Components Workbench software. 4. Put the Micro800 controller into Run mode.
0xF880	Recoverable	A data conversion error occurred.	Perform the following: 1. See <u>Corrective Actions for Recoverable Faults on page 314</u>. 2. Correct the program to verify that there is no data conversion error. 3. Build and download the program using Connected Components Workbench software. 4. Put the Micro800 controller into Run mode.
0xF888	Recoverable	The call stack of the controller cannot support the sequence of calls to function blocks in the current project. Too many blocks are within another block.	Perform the following: • See <u>Corrective Actions for Recoverable Faults on page 314</u>. • Change the project to reduce the quantity of blocks being called within a block.
0xF898	Recoverable	An error occurred in the user interrupt configuration for the plug-in I/O module.	Perform the following: • See <u>Corrective Actions for Recoverable Faults on page 314</u>. • Correct the user interrupt configuration for the plug-in I/O module in the user program to match that of the actual hardware configuration.

Table 121 - List of Error Codes for Micro800 Controllers (Continued)

Error Code	Fault Type	Description	Recommended Action
0xF8A0	Recoverable	The TOW parameters are invalid.	Perform the following: 1. See Corrective Actions for Recoverable Faults on page 314. 2. Correct the program to verify that there are no invalid parameters. 3. Build and download the program using Connected Components Workbench software. 4. Put the Micro800 controller into Run mode.
0xF8A1	Recoverable	The DOY parameters are invalid.	Perform the following: 1. See Corrective Actions for Recoverable Faults on page 314. 2. Correct the program to verify that there are no invalid parameters. 3. Build and download the program using Connected Components Workbench software. 4. Put the Micro800 controller into Run mode.
0xF8A3	Recoverable	Major fault on the controller when Class 1 connection to the controller fails while in Run mode.	Determine which Ethernet module has a faulted connection and correct the fault condition.
0xFFzz[4]	Recoverable	A user-created fault from the Connected Components Workbench software has occurred.	See Corrective Actions for Recoverable Faults on page 314.
0xD00F	Recoverable	A particular hardware type (for example, embedded I/O) was selected in the user program configuration, but did not match the actual hardware base.	See Corrective Actions for Recoverable Faults on page 314.
0xD011	Recoverable	The program scan time exceeded the watchdog timeout value.	Perform the following: • See Corrective Actions for Recoverable Faults on page 314. • Determine if the program is caught in a loop and correct the problem. Fault may occur if your Structured Text program contains a For loop with the upper limit set to the maximum value of the variable. For example, the variable is a USINT and the limit is set to 255, or the variable is a UINT and the limit is set to 65535. To correct the fault, perform the following: 1. Correct the program to verify that the upper limit is not reached. One method is to use a data type with a larger maximum value. 2. Build and download the program using Connected Components Workbench software. 3. Put the Micro800 controller into Run mode. If your program is designed to have a scan time of longer than 3 seconds, in the user program, increase the watchdog timeout value that is set in the system variable _SYSVA_TCYWDG and then build and download the program using Connected Components Workbench software.

(1)   z indicates the logic axis ID. (0...3)
(2)   z indicates the slot number of the expansion I/O. If z=0, then the slot number cannot be identified.
(3)   z is the slot number of the plug-in module. If z = 0, then the slot number cannot be identified.
(4)   zz indicates the last byte of the program number. Only program numbers up to 0xFF can be displayed. For program numbers 01x00 to 0xFFFF, only the last byte is displayed.)

Corrective Action for Recoverable and Nonrecoverable Faults

Corrective Actions for Recoverable Faults

Perform the following:
1. Optionally save the fault log from Connected Components Workbench software.
2. Clear the recoverable fault using Connected Components Workbench software.
3. If the problem persists, contact technical support with the fault log.

Corrective Actions for Nonrecoverable Faults

Perform the following:
1. Cycle power to your Micro800 controller.
2. The controller goes to recoverable fault. Optionally save the fault log from Connected Components Workbench software.
3. Clear the recoverable fault using Connected Components Workbench software.
4. If the program is lost, build and download your program using Connected Components Workbench software.
5. If the problem persists, contact technical support with the fault log.

Retrieve a Fault Log

You can retrieve a fault log for your controller by using the Connected Components Workbench software, version 9 or later.

Perform the following:
1. Launch the Connected Components Workbench software.
2. Connect to your Micro800 controller.
3. In Project Organizer, right-click the Micro800 controller.
4. Select Diagnose > Fault.
   The Fault Diagnostics tab displays.
5. Select Get Fault Log.
6. Save the fault log (.txt) file.

Retrieve a Core Dump on Major Fault

If a major nonrecoverable fault occurs, you can retrieve a core dump of the controller and submit this file to Rockwell Automation Tech Support for help in troubleshooting and debugging. The size of the core dump is limited to 8 KB.

This feature is available for controllers with firmware revision 22.011 or later. You must also use Connected Components Workbench software version 22.00 or later.

After a major fault occurs and you have cycled power to the controller, perform the following:
1. Launch the Connected Components Workbench software.
2. Connect to your Micro800 controller.
3. In Project Organizer, right-click the Micro800 controller.
4. Select Diagnose > Fault.
   The Fault Diagnostics tab displays.
5. Select Core Dump Upload.
6. Save the core dump (.bin) file.

IMPORTANT	Updating your controller firmware revision clears the core dump.

Controller Error Recovery Model

Use the following error recovery model to help you diagnose software and hardware problems in the micro controller. The model provides common questions that you might ask to help troubleshoot your system. See the recommended pages within the model for further help.

Controller Error Recovery Model

```
                    ┌─────────────┐
                    │    Start    │
                    └──────┬──────┘
                           │
                           ▼
                    ╱────────────╲
                   ╱   Is the      ╲      No      ┌─────────────────────┐
                  ╱   Power LED      ╲───────────►│  Check the wiring.  │
                   ╲    on?          ╱            └─────────────────────┘
                    ╲──────────────╱
                           │ Yes
                           ▼
              ┌────────────────────────────┐
              │ Check the Fault LED.        │
              │ Flashing red = Recoverable  │
              │ Steady red = Nonrecoverable │
              └─────────────┬──────────────┘
                            │
                            ▼
                    ╱────────────────╲
                   ╱                   ╲    No    ┌──────────────────────────┐
                  ╱  Is fault recoverable? ╲─────►│ Power cycle the controller.│
                   ╲                    ╱         └──────────────────────────┘
                    ╲─────────────────╱
                           │ Yes
                           ▼
          ┌────────────────────────────────────┐
          │ Diagnose fault in Connected Components│
          │ Workbench software and see page 309 for│
          │ probable cause and recommended action. │
          └─────────────────┬──────────────────┘
                            │
                            ▼
              ┌────────────────────────────┐
              │      Clear the fault.       │
              └─────────────┬──────────────┘
                            │
                            ▼
              ┌────────────────────────────┐
              │ Correct the condition       │
              │ causing the fault.          │
              └─────────────┬──────────────┘
                            │
                            ▼
              ┌────────────────────────────┐
              │ Test and verify             │
              │ system operation.           │
              └─────────────┬──────────────┘
                            │
                            ▼
                    ┌─────────────┐
                    │     End     │
                    └─────────────┘
```
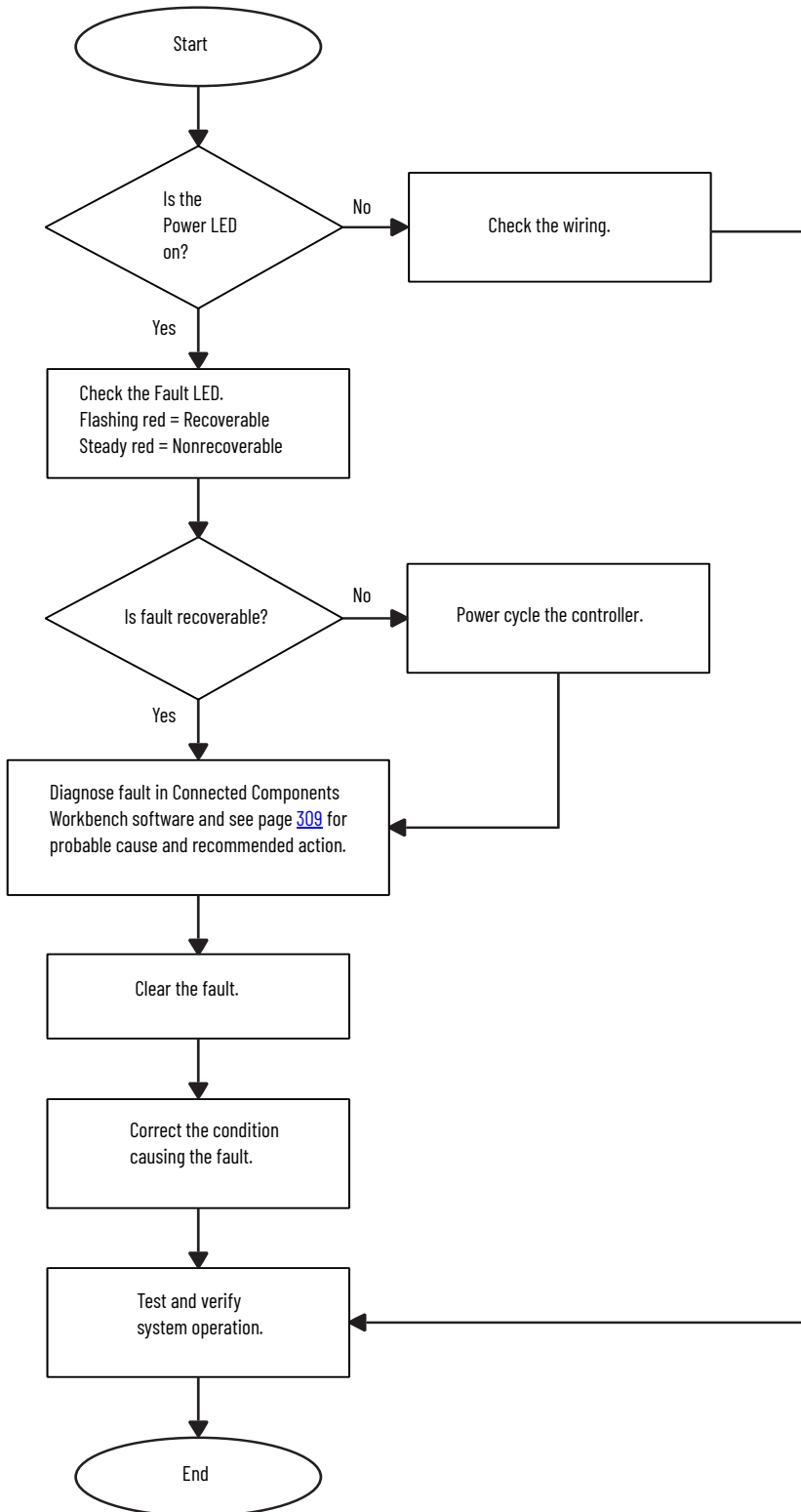
Ethernet Diagnostics

In Connected Components Workbench software, when the project in online, you can view the status of the embedded Ethernet ports on the controller.

To see the Communication Diagnostic window, do the following.

1. Access the controller configuration.
2. Select Diagnose and then select Communication.
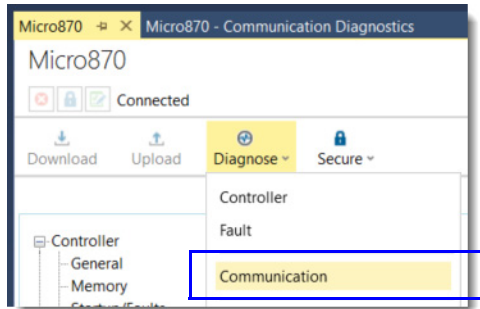
General Diagnostic Information

In the Communication Diagnostic window, change Communications to Ethernet, and change Protocols to General.

Table 122 - General Diagnostic Parameters [1]

Parameters	Description
Interface Counters	
Unicast Packets (Outbound)	The number of unicast packets sent from the interface
Nonunicast Packets (Inbound)	The number of nonunicast packets received on the interface
Nonunicast Packets (Outbound)	The number of nonunicast packets transmitted on the interface
Media Counters	
SQE Test Errors	The number of times an SQE test error message was generated
IP Statistics	
IP Total Packets Sent	Total number of IP packets successfully transmitted
IP Total Packets Received	Total number of IP packets successfully received
IP Total Bytes Sent	Total number of IP bytes successfully transmitted
IP Total Bytes Received	Total number of IP bytes successfully received
IP Sent Packets Dropped	Total number of IP packets dropped at sent
IP Received Packets Dropped	Total number of IP packets dropped at received
IP Total Fragments Sent	Total number of IP fragments sent
IP Total Fragments Received	Total number of IP fragments received
IP Received Checksum Error	Total number of IP packets received with checksum errors
IP Invalid Packets	Total number of invalid IP packets
TCP Statistics	
TCP Packets Sent	Total number of TCP packets sent
TCP Packets Received	Total number of TCP packets received
TCP Bytes Sent	Total number of TCP bytes sent
TCP Bytes Received	Total number of TCP bytes received
TCP Connections Dropped	Total number of TCP connections dropped
TCP Received Packets Dropped	Total number of TCP packets dropped
TCP Connections	Total number of TCP connections
TCP Disconnections	Total number of TCP disconnections
TCP Checksum Error	Total number of TCP packets with checksum errors
TCP Invalid Packets	Total number of invalid TCP packets
TCP Retransmit Packets	Total number of TCP packets retransmitted
UDP Statistics	
UDP Packets Sent	Total number of UDP packets sent
UDP Packets Received	Total number of UDP packets received
UDP Bytes Sent	Total number of UDP bytes sent
UDP Bytes Received	Total number of UDP bytes received
UDP Received Packets Dropped	Total number of UDP received packets dropped
UDP Checksum Error	Total number of UDP packets with checksum errors
UDP Invalid Packets	Total number of invalid UDP packets

(1)   These parameters are only available with firmware revision 21.011 or later.

EtherNet/IP Overview Diagnostic Information

In the Communication Diagnostic window, change Communications to Ethernet, and change Protocols to EtherNet/IP Overview. This feature is only available with firmware revision 21.011 or later.

Table 123 – EtherNet/IP Overview Diagnostic Parameters

Parameters	Description
TCP Connections	
Active TCP Connections	Current number of active TCP connections for EtherNet/IP messaging
Max TCP Connections	Maximum TCP connections that are supported for EtherNet/IP messaging
Unconnected	
Sent Packets Per Second	Number of EtherNet/IP unconnected messages (packets) sent in the last second
Received Packets Per Second	Number of EtherNet/IP unconnected messages (packets) received in the last second
Total Packets Sent	Cumulative number of EtherNet/IP unconnected messages (packets) sent
Total Packets Received	Cumulative number of EtherNet/IP unconnected messages (packets) received
Connected	
Sent Packets Per Second	Number of EtherNet/IP connected messages (packets) sent in the last second
Received Packets Per Second	Number of EtherNet/IP connected messages (packets) received in the last second
Total Packets Sent	Cumulative number of EtherNet/IP connected messages (packets) sent
Total Packets Received	Cumulative number of EtherNet/IP connected messages (packets) received

Table 123 – EtherNet/IP Overview Diagnostic Parameters (Continued)

Parameters	Description
I/O Packets Per Second	
I/O Packets Per Second Send (Actual)	Number of Class1 EtherNet/IP packets the system transmitted in the last second.
I/O Packets Per Second Received (Actual)	Number of Class1 EtherNet/IP packets the system received in the last second.
I/O Packets Per Second Total (Actual)	Total number of Class1 EtherNet/IP packets the system transmitted/received in the last second.
I/O Packets Per Second Rejected (Actual)	Number of Class1 EtherNet/IP packets the system rejected in the last second.
I/O Packets Counts	
I/O Packets Sent	Cumulative number of Class1 EtherNet/IP packets the system transmitted.
I/O Packets Received	Cumulative number of Class1 EtherNet/IP packets the system received.
I/O Packets Rejected	Cumulative number of Class1 EtherNet/IP packets the system rejected. These packets were messages received and then rejected because the connection was closed or there was a duplicate multicast address.
I/O Packets Missed	Cumulative number of Class1 EtherNet/IP packets that were not received in order. Each UDP packet has a sequence number and if a packet is missing (corrupted or dropped), the system recognizes this void upon receipt of the next packet received. Missed counter increments by the number of packets missed.
I/O Packets Total	Cumulative number of Class1 EtherNet/IP packets the system transmitted/received. The total is the sum of all packets that were sent, received, inhibited, and rejected.
Application Connections	
Type	Defines the class of the connection in terms of Class1 or Class3.
Status	Defines the current state of the connection in terms of Active and Inactive.
UpTime	Elapsed time for the connection that has been maintained.
Serial Number	Unique identifier for each connection.
O->T Packet Size	Size of originator to target packet data in bytes.
T->O Packets Size	Size of target to originator packet data in bytes.
Missed Packet Rx	Number of packets that are missed by that particular connection.

System Diagnostic Information

In the Communication Diagnostic window, change Communications to System to view the CIP Connection and Implicit messaging I/O system level diagnostics information. This feature is only available with firmware revision 21.011 or later.

Table 124 – System Diagnostic Parameters

Parameters	Description
CIP connection Statistics	
Active Total	Sum of CIP Explicit messaging and I/O connections in use.
Active Messaging	Total number of CIP Explicit (Class 3 to Message Router) messaging connections in use.
Active IO	Total number of CIP I/O (Implicit - Class1) connections in use.
Maximum CIP Connections	Maximum number of CIP connections that are supported by the system.
Connections Timeout	Number of connections that become nonexistent due to time out.
I/O Resource Utilization	
I/O Comms Utilization (Actual)	Actual communication utilization bandwidth that is currently in used the by I/O resource.
I/O Comms Utilization (Theoretical)	Theoretical communication utilization bandwidth that should be used by the I/O resource.
I/O Rate (Actual)	Actual PPS (Packets Per Second) at which I/O communication is happening.
I/O Rate (Theoretical)	Theoretical PPS (Packets Per Second) at which I/O communication should happen.

The I/O Comms Utilization values help you check the resources that are used by the Ethernet Implicit messaging feature in the Micro800 controller. If the application observes packet drops or missed packets in the Ethernet diagnostics, this value helps to provide an understanding if the utilization is close to the 100%. We recommend keeping the utilization value less than 60% to have enough resources in the controller to perform Class 3 messaging, interrupts, and program execution. The easiest way to reduce the utilization value is to increase the RPI of the devices configured under Ethernet-Modules.

Controller Diagnostic Information

The Controller Diagnostic page helps provide an overview of the information about the controller. This page includes details on the controller identity, project and memory module startup configuration, memory settings, controller uptime, and communication utilization.
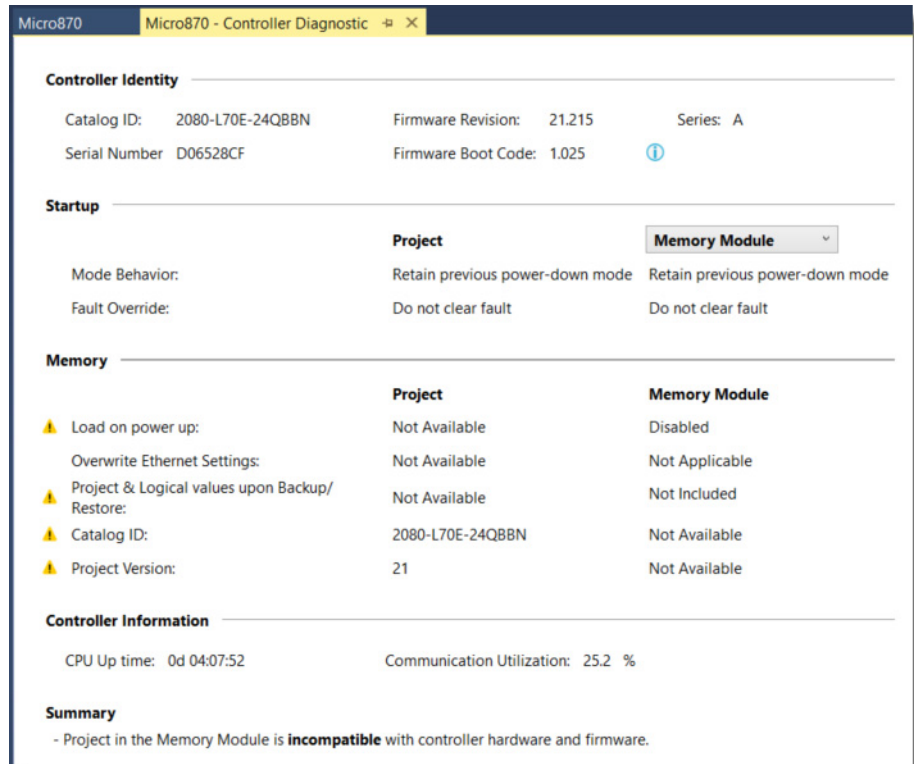
Table 125 - Controller Diagnostic Parameters [1]

Parameters	Description
Controller Information	
Communication Utilization	Communication buffer space utilization that is currently used by the system.
CPU Up Time	Time that the system is up since last power-up or reset.

(1)   These parameters are only available with firmware revision 21.011 or later.

Calling Rockwell Automation for Assistance

If you need to contact Rockwell Automation or local distributor for assistance, it is helpful to obtain the following (before calling):

* Controller type, series letter, revision letter, and firmware (FRN) number of the controller.
* Controller indicator status.

PID Function Blocks

The PID function block has parameter naming similar to RSLogix 500 and is recommended for users who are already familiar with programming in RSLogix 500. The IPIDCONTROLLER function block has the advantage of supporting autotune.

Table 126 - Comparison Between IPIDCONTROLLER and PID

IPIDCONTROLLER	PID	Description
Common parameters		
Process	PV	Process Variable feedback
Setpoint	SP	Setpoint input
Output	CV	CV output
Gains.DirectActing	Control	Control direction of process (cooling versus heating)
Gains.ProportionalGain	Gains.Kc	Controller gain for both P and I
Gains.TimeIntegral	Gains.Ti	Time integral value for I
Gains.TimeDerivative	Gains.Td	Time derivative value for D
Gains.DerivativeGain	Gains.FC	A higher filter constant makes CV output more responsive to error. Acts like a derivative gain.
AbsoluteError	AbsoluteError	Absolute value of error
PID-specific parameters		
	CVMin	For limiting CV
	CVMax	For limiting CV
	AutoManual	TRUE = Normal operation of PID FALSE = Manual operation using CVManual
	CVManual	CV when in manual mode
—	Enable	TRUE = Start execution with current input parameters. FALSE = CV equals zero.
	Active	TRUE = PID state is running. FALSE = PID state is stopped.
	Error	TRUE = PID has an error. FALSE = PID has no errors.
	ErrorID	PID ErrorID
IPIDCONTROLLER-specific parameters		
Auto		TRUE = Normal operation of PID FALSE = Output tracks Feedback
Feedback		Feedback of the control being applied to the process. Usually it's the PID's CV after any limits or manual control has been applied.
AutoTune	—	TRUE = Autotune FALSE = No Autotune
ATParameters		Autotune parameters
ATWarning		Autotune warning
OutGains		Gains from Autotune
Initialize		Used for AutoTune

PID Function Block

This function block diagram shows the arguments in the PID function block.

```
                      PID
    ──  Enable              Active     ──
    ──  PV                  CV         ──
    ──  SP                  AbsoluteError ──
    ──  AutoManual          Error      ──
    ──  CVManual            Error ID   ──
    ──  CVMax
    ──  CVMin
    ──  Gains
    ──  Control
    ──  LInit
```
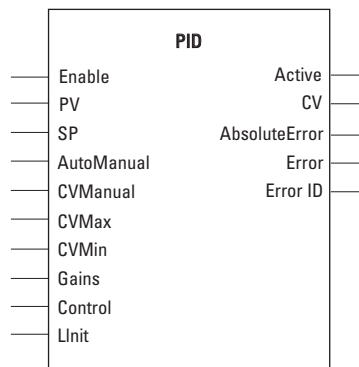
Table 127 explains the arguments that are used in this function block.

Table 127 - PID Arguments

Parameter	Parameter Type	Data Type	Description
Enable	Input	BOOL	Enable instruction TRUE = Start execution with current input parameters. FALSE = CV equals zero.
PV	Input	REAL	Process Value. This value is typically read from an analog input module. The SI unit must be the same as Setpoint.
SP	Input	REAL	The setpoint value for the process
AutoManual	Input	BOOL	Auto or manual mode selection: TRUE = Normal operation of PID FALSE = Manual operation using CVManual
CVManual	Input	REAL	Control value input defined for manual mode operation. The valid range for CVManual is: CVMin < CVManual < CVMax
CVMin	Input	REAL	Control value minimum limit If CV < CVMin, then CV = CVMin. If CVMin > CVMax, an error occurs.
CVMax	Input	REAL	Control value maximum limit If CV > CVMax, then CV = CVMax. If CVMax < CVMin, an error occurs.
Gains	Input	PID_GAINS	Gains of PID for controller Use the PID_GAINS data type to configure the Gains parameter.
Control	Input	BOOL	Control direction of the process: TRUE = Direct acting, such as Cooling FALSE = Reverse acting, such as Heating
LInit	Input	BOOL	Reserved for future use
Active	Output	BOOL	Status of the PID controller: TRUE = PID state is running FALSE = PID state is stopped
CV	Output	REAL	The control value output If any error occurred, CV is 0.
AbsoluteError	Output	REAL	Absolute error is the difference between process value (PV) and setpoint (SV) value.
Error	Output	BOOL	Indicates the existence of an error condition TRUE = PID has an error. FALSE = PID has no errors.
ErrorID	Output	USINT	A unique numeric that identifies the error. The errors are defined in PID error codes.

Table 128 – GAIN_PID Data Type

Parameter	Parameter Type	Data Type	Description
Kc	Input	REAL	Controller gain for PID Proportional and Integral are dependent on this gain (>= 0.0001). Increasing Kc improves response time but also increases overshoot and oscillation of the PID. If Kc is invalid, an error occurs.
Ti	Input	REAL	Time integral constant in seconds (>= 0.0001). Increasing Ti decreases the overshoot and oscillation of the PID. If Ti is invalid, an error occurs.
Td	Input	REAL	Time derivative constant in seconds (>= 0.0). When Td equals 0, then there is no derivative action and PID becomes a PI controller. Increasing Td reduces the overshot and removes the oscillation of the PID controller. If Td is invalid, an error occurs.
FC	Input	REAL	Filter constant (>= 0.0) Recommended range for FC is 0...20. Increasing FC smooths the response of the PID controller. If FC is invalid, an error occurs.

Table 129 – PID Error Codes

Error Code	Description
0	PID is working normally.
1	Kc is invalid.
2	Ti is invalid.
3	Td is invalid.
4	FC is invalid.
5	CVMin > CVMax, or CVMax < CVMin
6	CVManual < CVMin CVManaul is invalid.
7	CVManual > CVMax CVManual is invalid.

IPIDCONTROLLER Function Block

This function block diagram shows the arguments in the IPIDCONTROLLER function block.

```
              IPIDCONTROLLER
  ──── EN                          ENO ────
  ──── Process                  Output ────
  ──── SetPoint          AbsoluteError ────
  ──── FeedBack              ATWarning ────
  ──── Auto                   OutGains ────
  ──── Initialize
  ──── Gains
  ──── AutoTune
  ──── ATParameters
```
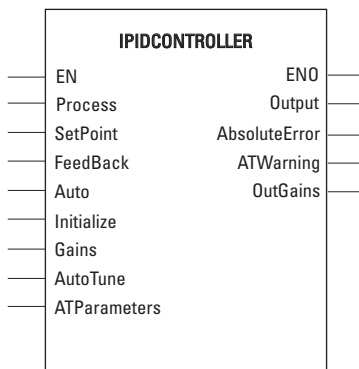
Table 130 explains the arguments that are used in this function block.

Table 130 - IPIDCONTROLLER Arguments

Parameter	Parameter Type	Data Type	Description
EN	Input	BOOL	Function block enable TRUE = Execute function. FALSE = Do not execute function. Applicable to Ladder Diagram programs
Process	Input	REAL	Process value, which is the value that is measured from the process output.
SetPoint	Input	REAL	The setpoint value for the process
Feedback	Input	REAL	Feedback signal, which is the value of the control variable that is applied to the process. For example, the feedback can be IPIDCONTROLLER output.
Auto	Input	BOOL	Operating modes of PID controller: TRUE = Normal operation of PID FALSE = Output tracks Feedback
Initialize	Input	BOOL	A change in value (TRUE to FALSE or FALSE to TRUE) causes the controller to eliminate any proportional gain during that cycle. It also initializes AutoTune sequences.
Gains	Input	GAIN_PID	Gains PID for IPIDCONTROLLER Use the GAIN_PID data type to define the parameters for the Gains input.
AutoTune	Input	BOOL	TRUE = Autotune FALSE = No Autotune
ATParameters	Input	AT_Param	AutoTune parameters Use AT_Param data type to define the parameters for the ATParameters input.
Output	Output	Real	Output value from the controller
AbsoluteError	Output	Real	Absolute error (Process - SetPoint) from the controller
ATWarnings	Output	DINT	Warning for the AutoTune sequence. Possible values are: 0 = No auto tune done 1 = In auto tune mode 2 = Auto tune done –1 = Error 1: Input automatically set to TRUE, no auto tune possible. –2 = Error 2: Auto tune error, the ATDynamSet expired.
OutGains	Output	GAIN_PID	Gains calculated from AutoTune Sequences Use the GAIN_PID data type to define the OutGains output.
ENO	Output	BOOL	Enable output Applicable to Ladder Diagram programs

Table 131 - GAIN_PID Data Type

Parameter	Type	Description
DirectActing	BOOL	Types of acting: TRUE = Direct acting, output moves in the same direction as error. That is, the actual process value is greater than the setpoint and the appropriate controller action is to increase the output. For example, Chilling. FALSE = Reverse acting, output moves opposite direction as error. That is, the actual process value is greater than the Setpoint and the appropriate controller action is to decrease the output. For example, Heating.
ProportionalGain	REAL	Proportional gain for PID (>= 0.0001) **Proportional gain for PID (P_Gain)** A higher proportional gain causes a larger change in the output based on the difference between the PV (measured process value) and SV (setpoint value). The higher the gain, the faster the error is decreased, but this may result in instability such as oscillations. The lower the gain, the slower the error is decreased, but the system is more stable and less sensitive to large errors. The P_Gain usually is the most important gain to adjust and the first gain to adjust while tuning.
TimeIntegral	REAL	Time integral value for PID (>= 0.0001) **Time integral value for PID** A smaller integral time constant causes a faster change in the output based on the difference between the PV (measured process value) and SV (setpoint value) integrated over this time. A smaller integral time constant decreases the steady state error (error when SV is not being changed) but increases the chances of instability such as oscillations. A larger integral time constant slows down the response of the system and makes it more stable, but PV approaches the SV at a slower rate.
TimeDerivative	REAL	Time derivative value for PID (> 0.0) **Time derivative value for PID (Td)** A smaller derivative time constant causes a faster change in the output based on the rate of change of the difference between PV (measured process value) and SV (setpoint value). A smaller derivative time constant makes a system more responsive to sudden changes in error (SV is changed) but increases the chances of instability such as oscillations. A larger time constant makes a system less responsive to sudden changes in error and the system is less susceptible to noise and step changes in PV. TimeDerivative (Td) is related to the derivative gain but allows the derivative contribution to PID to be tuned using time so the sample time must be considered.
DerivativeGain	REAL	Derivative gain for PID (>= 0.0) **Derivative gain for PID (D_Gain)** A higher derivative gain causes a larger change in the output based on the rate of change of the difference between the PV (measured process value) and SV (setpoint value). A higher gain makes a system more responsive to sudden changes in error but increases the chances of instability such as oscillations. A lower gain makes a system less responsive to sudden changes in error and makes the system less susceptible to noise and step changes in the PV. **If the derivative gain is set to zero, it disables the derivative portion of the PID.**

Table 132 - AT_Param Data Type

Parameter	Type	Description
Load	REAL	Load parameter for autotuning. This is the output value when starting AutoTune.
Deviation	REAL	Deviation for autotuning. This is the standard deviation that is used to evaluate the noise band needed for AutoTune (noise band = 3* Deviation)[1]
Step	REAL	Step value for AutoTune. Must be greater than noise band and less than ½ load.
ATDynamSet	REAL	Waiting time in seconds before abandoning auto tune
ATReset	BOOL	Determines whether the output value is reset to zero after an AutoTune sequence: TRUE = Resets output to zero. FALSE = Leaves output at Load value.

(1)   The application engineer can estimate the value of ATParams.Deviation by observing the value of Process input. For example, in a project that involves the control of temperature, if the temperature stabilizes around 22 °C, and a fluctuation of 21.7…22.5 °C is observed, the value of ATParams.Deviation is (22.5…21.7)/2 = 0.4.

How to Autotune

Before you autotune, you must:

- Verify that your system is constant when there is no control. For example, for temperature control, the process value should remain at room temperature when there is no control output.
- Configure the setpoint to 0.
- Set Auto Input to False.
- Set the Gain parameter as follows:

Table 133 - GAIN Parameter Values

GAIN Parameter	Value
DirectActing	According to the operation: TRUE (for example, Cooling), or FALSE (for example, Heating)
DerivativeGain	0.5
ProportionalGain	0.0001
TimeIntegral	0.0001
TimeDerivative	0.0

- Set the AT_Parameter as follows:

Table 134 - AT_Parameter Values

AT Parameter	Recommendation
Load	Every 'Load' provides a saturated process value over a period of time. Adjust the load to the value for the saturated process value that you want. **IMPORTANT**: If a load of 40 gives you a process value of 30 °C over a period of time, and you want to tune your system to 30 °C, you should set the load to 40.
Deviation	This parameter plays a significant role in the autotune process. The method of deriving this value is explained later in this section. It is not necessary to set this parameter before autotuning. However, if you already know the deviation, it is fine to set it first.
Step	Step value should be between 3*Deviation and ½ load. The step provides an offset for the load during autotuning. It should be set to a value high enough to create a significant change in process value.
ATDynamSet	Set this value to a reasonably long time for the autotune process. Every system is different, so allow more time to a system with a process value that takes longer to react to change.
ATReset	Set this parameter to TRUE to reset the output to zero after the autotune process completes. Set this parameter to FALSE to leave the output at load value after the autotune process completes.

During autotune, the controller automatically sets the process value to zero. To autotune, perform the following steps:

1. Set the Initialize input to TRUE.
2. Set the AutoTune input to TRUE.
3. Wait for the Process input to stabilize or reach a steady state.
4. Note the temperature fluctuation of the process value.
5. Calculate the deviation value with reference to the fluctuation. For example, if the temperature stabilizes around 22 °C (72 °F) with a fluctuation of 21.7…22.5 °C (71…72.5 °F), the value of 'ATParams.Deviation' is:

$$\text{For °C: } \frac{22.5…21.7}{2} = 0.4 \qquad \text{For °F: } \frac{72.5…71}{2} = 0.75$$

6. Set the deviation value, if you have not set it yet.
7. Change the initialize input to FALSE.
8. Wait until the 'AT_Warning' shows 2. The autotune process is successful.
9. Get the tuned value from the 'OutGains'.

How Autotune Works

The autotune process begins when 'Initialize' is set to FALSE (Step 7.) At this moment, the control output increases by the amount of 'Step' and the process waits for the process value to reach or exceeds 'first peak'.

First peak is defined as:

For Direct Operation: First peak = PV1 - (12 x Deviation)
For Reverse Operation: First peak = PV1 + (12 x Deviation)
Where PV1 is the process value when Initialize is set to FALSE.

Once the process value reaches the first peak, the control output reduces by the amount of Step and waits for the process value to drop to the second peak.

Second peak is defined as:

For Direct Operation: Second peak = PV1 - (3 x Deviation)
For Reverse Operation: Second peak = PV1 + (3 x Deviation)
Once the process value reaches or falls below the second peak, calculations commence and a set of gain is generated to parameter OutGains.

Troubleshooting an Autotune Process

You can understand the autotune process from the sequences of control output. Here are some known sequences of control output and what it means if the autotune fails. For the ease of illustrating the sequence of control output, we define:

- Load: 50
- Step: 20

Output Sequence 1: 50 ⬆70 ⬇30

Sequence Condition	Autotune Result	Action for Autotune Fail
Process value reached 'first peak' and 'second' peak in time	Likely successful	NA

Output Sequence 2: 50 ⬆70 ⬇50

Sequence Condition	Autotune Result	Action for Autotune Fail
Process value not able to reach 'first peak'	Likely unsuccessful	Reduce Deviation or Increase Step

Output Sequence 3: 50 ⬆70 ⬇30 ⬆50

Sequence Condition	Autotune Result	Action for Autotune Fail
Process value not able to reach second peak	Likely unsuccessful	Increase Deviation or increase Step

Output Sequence 4: 50 ⬆70

Sequence Condition	Autotune Result	Action for Autotune Fail
Process value not able to reach First peak in time	Likely unsuccessful	Increase ATDynamSet

PID Application Example

Figure 68 - Example of a PID Application
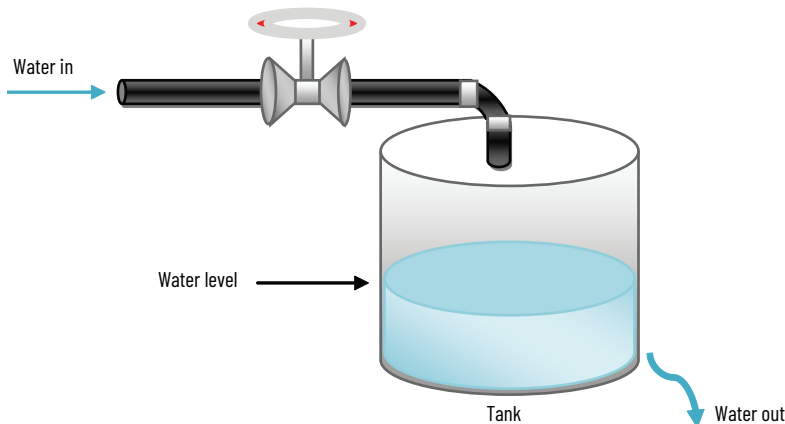
Figure 68 shows a basic water level control system to maintain a preset water level in the tank. A solenoid valve is used to control incoming water, filling the tank at a preset rate. Similarly, outflowing water is controlled at a measurable rate.

IPID Autotuning for First and Second Order Systems

Autotune of IPID can only work on first and second order systems.

A first order system can be described by an independent energy storage element. Examples of first order systems are the cooling of a fluid tank; the flow of fluid from a tank; a motor with constant torque driving a disk flywheel or an electric RC lead network. The energy storage elements for these systems are heat energy, potential energy, rotational kinetic energy, and capacitive storage energy, respectively.

This may be written in a standard form such as $f(t) = \tau dy/dt + y(t)$, where τ is the system time constant, f is the forcing function, and y is the system state variable.

In the cooling of a fluid tank example, it can be modeled by the thermal capacitance C of the fluid and thermal resistance R of the walls of the tank. The system time constant is RC, the forcing function is the ambient temperature, and the system state variable is the fluid temperature.

A second order system can be described by two independent energy storage elements that exchange stored energy. Examples of second order systems are: a motor driving a disk flywheel with the motor that is coupled to the flywheel via a shaft with torsional stiffness, or an electric circuit composed of a current source driving a series LR (inductor and resistor) with a shunt C (capacitor). The energy storage elements for these systems are the rotational kinetic energy and torsion spring energy for the former, and the inductive and capacitive storage energy for the latter. Motor drive systems and heating systems can be typically modeled by the LR and C electric circuit.

PID Code Sample

Figure 69 - Sample of PID Code to Control a PID Application
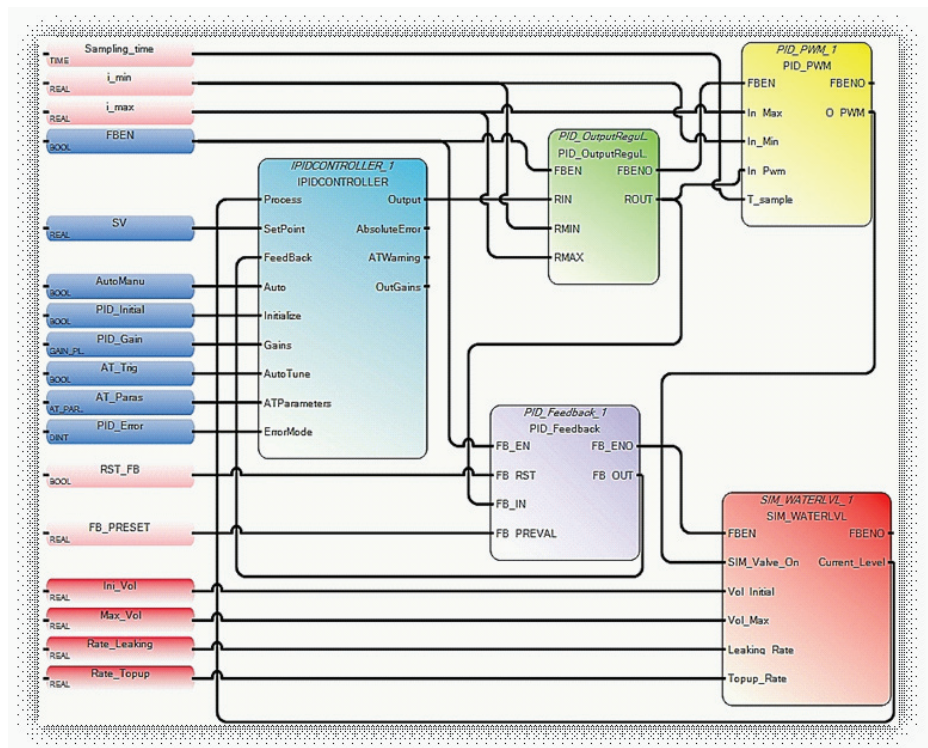
Figure 69 shows the sample code for controlling the PID application example (Figure 68). The sample code, developed using Function Block Diagrams, consists of a pre-defined function block (IPIDCONTROLLER), and four user-defined function blocks. The four user-defined function blocks are:

- PID_OutputRegulator

  Regulates the output of IPIDCONTROLLER within a safe range to verify that there is no damage to the hardware that is used in the process.

  If RMIN ≤ RIN ≤ RMAX, then ROUT = RIN,
  If RIN < RMIN, then ROUT = RMIN,
  If RIN > RMAX, then ROUT = RMAX.

- PID_Feedback

  Acts as a multiplexer.

  If FB_RST is false, then FB_OUT = FB_IN,
  If FB_RST is true, then FB_OUT = FB_PREVAL.

- PID_PWM

  Provides a PWM function, which converts a real value to a time-related ON/OFF output.

- SIM_WATERLVL

  Simulates the process that is shown in the PID application example (Figure 68).

IMPORTANT	User Program Scan Time
	The autotuning method must cause the output of the control loop to oscillate. To identify the oscillation period, the IPID must be called frequently enough to sample the oscillation adequately. The scan time of the user program must be less than half the oscillation period. In essence, you must adhere to the Shannon or Nyquist-Shannon sampling theorem.
	Also, you must trigger the function block at a relatively constant time interval. You can achieve this with an STI interrupt function.

Notes:

System Loading

Table 135 - Micro830, Micro850, and Micro870 Power Requirements

Controller/Module	Power Requirement		
Micro830, Micro850, and Micro870 (without plug-in/expansion I/O) 10/16-point 24-point 48-point	5 W 8 W 11 W		
Plug-in modules, each	1.44 W		
Expansion I/O (system bus power consumption)	2085-IQ16 2085-IQ32T 2085-IA8 2085-IM8 2085-OA8 2085-OB16 2085-OV16 2085-OW8 2085-OW16 2085-IF4 2085-IF8 2085-OF4 2085-IRT4	– – – – – – – – – – – – –	0.85 W 0.95 W 0.75 W 0.75 W 0.90 W 1.00 W 1.00 W 1.80 W 3.20 W 1.70 W 1.75 W 3.70 W 2.00 W

Calculate Total Power for Your Micro830/Micro850/Micro870 Controller

To calculate Total Power for your Micro830, Micro850, and Micro870 controller, use the following formula:

> **Total Power** =Main Unit Power + No. of Plug-ins * Plug-in Power + Sum of Expansion I/O Power

Example 1:
Derive Total Power for a 24-point Micro830 controller with two plug-ins.

> **Total Power** = 8 W+ 1.44 W * 2 + 0 = **10.88 W**

Example 2:
Derive Total Power for a 48-point Micro850 controller, with 3 plug-ins, and 2085-IQ16, and 2085-IF4 expansion I/O modules attached.

> **Total Power** = 11 W + 3 * 1.44 W + 0.85 W + 1.7 W = **17.87 W**

Calculate External AC Power Supply Loading for your Micro830 Controller

To calculate External AC Power Supply Loading:

- Get total sensor current loading. For this example, assume it is 250 mA.
- Calculate Total Power Loading by Sensor using this formula:
  (24V * 250 mA) 6 W.
- Derive External AC Power Supply Loading using this formula:
  AC Power Supply Loading = Total Power that is calculated for a Micro800 system with Plug-in + Total power loading by Sensor.

As an example, a 48-point Micro850 controller with 2 plug-ins, 2085-IQ16 and 2085-IF4 expansion I/O, and 250 mA sensor current (6 W sensor power) have the following Total Loading for AC Power Supply:

Total loading for AC power supply = 17.87 W + 6 W = **23.87 W**

> ⚠ **ATTENTION:** Maximum loading to AC Power Supply is limited to 38.4 W with maximum surrounding ambient temperature limited to 65 °C (149 °F).

Connect to Networks using DF1

IMPORTANT	This appendix only applies to Micro850 (2080-L50E) and Micro870 (2080-L70E) controllers.

The following protocols are supported on the embedded serial port, including any 2080-SERIALISOL plug-in module that is installed, on the newer Micro850 (2080-L50E) and Micro870 (2080-L70E) controllers.

- DF1 Full-duplex
- DF1 Half-duplex Master/Slave
- DF1 Radio Modem

DF1 Full-duplex Protocol

DF1 Full-duplex protocol provides a point-to-point connection between two devices. DF1 Full-duplex protocol combines data transparency (American National Standards Institute ANSI – X3.28-1976 specification subcategory D1) and 2-way simultaneous transmission with embedded responses (subcategory F1).

The controller supports the DF1 Full-duplex protocol via RS-232 connection to external devices, such as computers, or other controllers that support DF1 Full-duplex.

DF1 support is achieved through the CIP serial interface in the Micro800 controllers.

DF1 Full-duplex protocol (also referred to as DF1 point-to-point protocol) is useful where RS-232 point-to-point communication is required. DF1 protocol controls message flow, detects and signals errors, and retries if errors are detected.

Example DF1 Full-duplex Connections
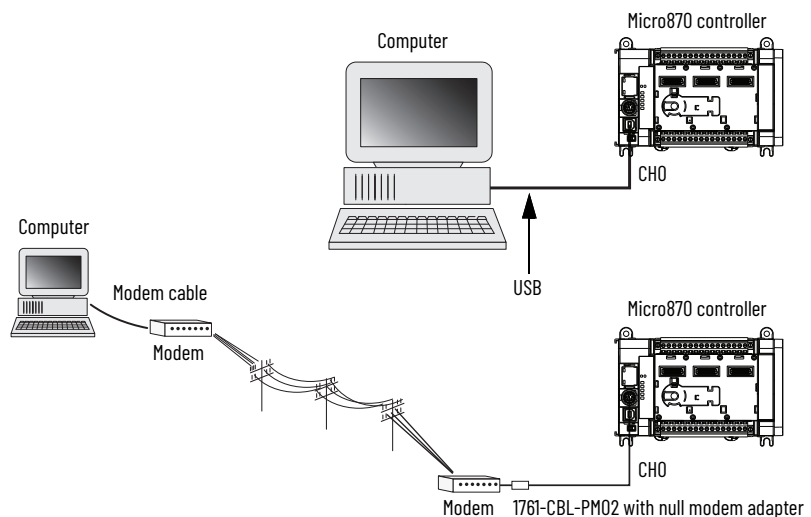
DF1 Half-duplex Protocol

DF1 Half-duplex protocol is a multi-drop single master/multiple slave network. DF1 Half-duplex protocol supports data transparency (American National Standards Institute ANSI – X3.28-1976 specification subcategory D1). In contrast to DF1 Full-duplex, communication takes place in one direction at a time. You can use the RS-232/RS-485 port on the controller as both a Half-duplex programming port and a Half-duplex peer-to-peer messaging port.

DF1 Half-duplex Operation

A DF1 Half-duplex master device initiates all communication by "polling" each slave device. The slave device may only transmit when it is polled by the master. It is the master's responsibility to poll each slave on a regular and sequential basis to allow slave devices an opportunity to communicate.

An additional feature of the DF1 Half-duplex protocol is that it is possible for a slave device to enable a MSG write or read to/from another slave. When the initiating slave is polled, the MSG is sent to the master. The master recognizes that the message is not intended for it, but for another slave, so the master immediately forwards the message to the intended slave. The master does this automatically; you do not need to program the master to move data between slave nodes. This slave-to-slave transfer can also be used by programming software to allow slave-to-slave upload and download of programs to controllers (including the master) on the DF1 Half-duplex link.

The controller can act as the master or as a slave on a Half-duplex network. When the controller is a slave device, a master device is required to run the network. Several other Allen-Bradley products support the DF1 Half-duplex master protocol.

DF1 Half-duplex supports up to 255 devices (address 0...254) with address 255 reserved for master broadcasts. As a DF1 Half-duplex slave device, the controller supports broadcast reception. As a DF1 Half-duplex master, the controller supports both the reception and initiation of broadcast write commands (via the MSG instruction). The controller also supports Half-duplex modems using RTS/CTS hardware handshaking.

IMPORTANT	Firmware revision 20 for Micro850 and Micro870 controllers does not support the broadcast function in Half-duplex master. This feature will be supported in a future firmware revision.

Considerations When Communicating as a DF1 Slave on a Multi-drop Link

When communication is between either your programming software and a Micro800 controller or between two Micro800 controllers via slave-to-slave communication on a larger multi-drop link, the devices depend on a DF1 Half-duplex master to give each of them access in a timely manner. As the number of slave devices increases, the time between when slave devices are polled also increases. This increase in time may also be large if you are using low communication rate. As these time periods grow, you must increase the poll timeout and reply timeout values for slave devices.

IMPORTANT	Program download is not supported in the DF1 Half-duplex and Radio Modem for Micro850 (2080-L50E) and Micro870 (2080-L70E) controllers firmware revision 20 or later.

Using Modems with Micro800 Programmable Controllers

The types of modems you can use with Micro800 controllers include the following:
- Dial-up phone modems
  A Micro800 controller, on the receiving end of the dial-up connection, can be configured for DF1 Full-duplex protocol with or without handshaking. The modem connected to the Micro800 controller should support auto-answer.

- Leased-line modems
Leased-line modems are used with dedicated phone lines that are typically leased from the local phone company. The dedicated lines may be in a point-to-point topology supporting Full-duplex communications between two modems or in a multi-drop topology supporting Half-duplex communications between three or more modems.

- Radio modems
Radio modems may be implemented in a point-to-point topology supporting either Half-duplex or Full-duplex communications, or in a multi-drop topology supporting half-duplex communications between three or more modems. Micro800 controllers also support DF1 Radio Modem protocol.

- Line drivers
Line drivers, also called short-haul modems, do not actually modulate the serial data, but rather condition the electrical signals to operate reliably over long transmission distances (up to several miles). Line drivers are available in Full-duplex and Half-duplex models.

For point-to-point Full-duplex modem connections that do not require any modem handshaking signals to operate, use the DF1 Full-duplex protocol with no handshaking. For point-to-point Full-duplex modem connections that require RTS/CTS handshaking, use the DF1 Full-duplex protocol with handshaking.

For radio modem connections, use the DF1 Radio Modem protocol, especially if store and forward capability is required.

For general multi-drop modem connections, or for point-to-point modem connections that require RTS/CTS handshaking, use the DF1 Half-duplex slave protocol. In this case, one (and only one) of the other devices must be configured for the DF1 Half-duplex master protocol.

> Micro850 (2080-L50E) and Micro870 (2080-L70E) controllers support RTS/CTS modem handshaking when configured for DF1 Full-duplex protocol with the control line parameter set to Full-duplex Modem Handshaking or DF1 Half-duplex slave protocol with the control line parameter set to Half-duplex Modem.
>
> These controllers also support the Data Carrier Detect (DCD) line for the DF1 radio modem protocol. No other modem handshaking lines (such as Data-Set™ Ready and Data Terminal Ready) are supported by Micro800 controllers.

Modem Control Line Operation

The following explains the operation of the Micro800 controllers when you configure the RS-232 serial port for the following applications.

DF1 Full-duplex

When configured for DF1 Full-duplex, the following control line operation takes effect:

No Handshake Selected

RTS is always inactive (low). Receptions and transmissions take place regardless of the state of CTS input. Only make this selection when the Micro800 controllers are directly connected to another device that does not require handshaking signals.

Full-duplex (RTS always ON) Selected

RTS is always active (high).

Transmissions require CTS to be active.

DF1 Half-duplex Slave

When configured for DF1 Half-duplex slave, the following control line operation takes effect:

No Handshake Selected

RTS is always inactive. Receptions and transmissions take place regardless of the state of CTS input. Only make this selection when the controller is directly connected to another device that does not require handshaking signals.

Half-duplex without Continuous Carrier (RTS/CTS) Selected

RTS is only activated during transmissions (and any programmed delays before or after transmissions). Transmissions require CTS to be active.

DF1 Half-duplex Master

When configuring for the DF1 Half-duplex master, the following control line operation takes effect:

No Handshake Selected

RTS is always inactive. Receptions and transmissions take place regardless of the state of CTS input. Only make this selection when the controller is directly connected to another device that does not require handshaking signals.

Full-duplex Modem (RTS always ON) Selected

RTS is always active (high).

Transmissions require CTS to be active.

Half-duplex without Continuous Carrier (RTS/CTS) Selected

RTS is only active during transmissions (and any programmed delays before and after transmissions).

Transmissions require CTS to be active

DF1 Radio Modem

When you configure the Micro800 controllers for DF1 Radio Modem, the following control line operation takes effect:

No Handshake Selected

RTS is always inactive. Receptions and transmissions take place regardless of the state of CTS input. This selection should only be made when the controller is directly connected to another device that does not require handshaking signals.

Half-duplex without Continuous Carrier (RTS/CTS) Selected

RTS is activated during transmission and during any programmed delays before or after transmissions. Programmed delays include RTS Send Delay and RTS Off Delay.

Transmissions require CTS to be active. If CTS is inactive at the onset of transmission, one second will be provided to wait for CTS to become active before the message packet is discarded.

Half-duplex with DCD Handshake Selected

RTS is activated during transmissions and during any programmed delays before and after transmissions. Programmed delays include RTS Send Delay and RTS Off Delay. The DCD input signal is monitored to determine if transmissions are acceptable. If DCD is active, receptions are possible.

Transmissions require CTS to be active and DCD to be inactive. If DCD is active at the onset of transmission, a configured delay (DCD Wait Delay) will wait for DCD to become inactive before discarding the packet. If CTS is inactive at the onset of transmission, one second will be provided to wait for CTS to become active before the message packet is discarded.

Configure DF1 Half-Duplex Parameters

RTS Send Delay and RTS Off Delay

Through your programming software, the parameters RTS Send Delay and RTS Off Delay let you set how long RTS is on before transmission, and how long to keep it on after transmission is complete. These parameters only apply when you select Half-duplex Modem. For maximum communication throughput, leave these parameters at zero.

For use with Half-duplex Modems that require extra time to turnaround or key-up their transmitter even after they have activated CTS, the RTS Send Delay specifies (in 20 millisecond increments) the amount of delay time after activating RTS to wait before checking to see if CTS has been activated by the modem. If CTS is not yet active, RTS remains active, and as long as CTS is activated within one second, the transmission occurs. After one second, if CTS is still not activated, then RTS is set to inactive and the transmission is aborted.

For modems that do not supply a CTS signal but still require RTS to be raised before transmission, jumper RTS to CTS and use the shortest delay possible without losing reliable operation.

IMPORTANT	If an RTS Send Delay of 0 is selected, then transmission starts as soon as CTS is activated. If CTS does not go active within one second after RTS is raised, RTS is set inactive and the transmission is stopped.

Certain modems drop their carrier link when RTS is set inactive even though the transmission has not finished. The RTS Off Delay parameter specifies in 20 millisecond increments the delay between when the last serial character is sent to the modem and when RTS is deactivated. This gives the modem extra time to transmit the last character of a packet.

⚠	**ATTENTION:** For almost all modem applications, the RTS Off Delay should be left at 0. Never select an RTS Off Delay that is greater than the RTS Send Delay in the other devices on the network, or you may incur two devices trying to transmit simultaneously.

Configure a Standard-Mode DF1 Half-duplex Master Station

Choose standard mode if you want to query slave stations for information based on user-configured polling ranges. This mode is often used in general point-to-multipoint configurations.

To configure the controller as a master station using standard-communication mode, place the controller into program mode and follow the steps below using your programming software:

1. To bring up the configuration page, select Serial Port.
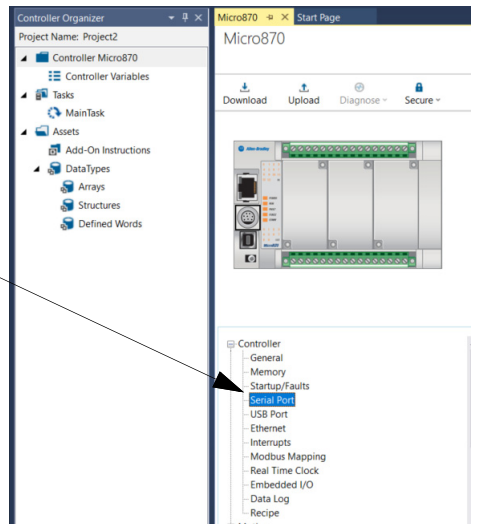
2. On the Serial Port configuration page, select Half-Duplex Master for DF1 Mode.
3. Choose a Standard Polling Mode.
4. Configure the rest of the communication driver according to Table 136.
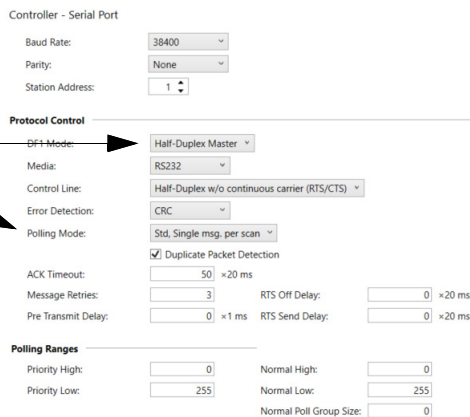
Table 136 shows the parameters for configuring a Micro850 (2080-L50E) or Micro870 (2080-L70E) controller as a master station using standard-communication mode to talk to slave stations.

Table 136 - Configure a Micro800 Controller as a Master Using Standard-communication Mode

Parameter	Selections
Baud Rate	Select a communication rate that all devices in your system support. Configure all devices in the system for the same communication rate.
Parity	Parity provides additional message packet error detection. To implement even parity checking, choose Even. To implement no parity checking, choose None.
Node Address	A node address identifies the controller on the DF1 half-duplex link. Each station on a link must have a unique address. Choose an address between 0_{10} and 254_{10}. Node address 255_{10} is the broadcast address, and cannot be selected as a station's individual address.
Control Line	Defines the mode in which the driver operates. Choose a method appropriate for your system configuration: • If you are not using a modem, choose NO HANDSHAKE. • If the master modem is full duplex, choose FULL-DUPLEX (RTS ALWAYS ON). • If all modems in the system are half-duplex, choose HALF-DUPLEX WITHOUT CONTINUOUS CARRIER (RTS/CTS). See Modem Control Line Operation on page 337 for a description of the control line operation settings.

Error Detection	With this selection, you choose how the controller checks the accuracy of each DF1 packet transmission.
	BCC: This algorithm provides a medium level of data security. It cannot detect:
	   – Transposition of bytes during transmission of a packet
	   – The insertion or deletion of data values of zero within a packet
	CRC: This algorithm provides a higher level of data security.
	Select an error detection method that all devices in your configuration can use. When possible, choose CRC.

Table 136 - Configure a Micro800 Controller as a Master Using Standard-communication Mode (Continued)

Parameter	Selections
Polling Mode	If you want to receive: • Only one message from a slave station per its turn, choose STANDARD (SINGLE MESSAGE TRANSFER PER NODE SCAN). Choose this method only if it is critical to keep the poll list scan time to a minimum. • As many messages from a slave station as it has, choose STANDARD (MULTIPLE MESSAGE TRANSFER PER NODE SCAN).
Duplicate Packet Detect	Duplicate Detect lets the controller detect if it has received a message that is a duplicate of its most recent message from another station. If you choose duplicate detect, the controller acknowledges (ACK) the message but doens not act on it since it has already performed the message's task when it received the command from the first message. If you want to detect duplicate packets and discard them, check this parameter. If you want to accept duplicate packets and execute them, leave this parameter unchecked.
ACK Timeout	The amount of time, in 20 millisecond increments, that you want the controller to wait for an acknowledgment to the message it has sent before the controller retries the message or the message errors out. This timeout value is also used for the poll response timeout. See Minimum DF1 Half-duplex Master ACK Timeout on page 341 for recommendations to minimize this value.
RTS Off Delay	Defines the amount of time, in 20 millisecond increments that elapses between the end of the message transmission and the de-assertion of the RTS signal. This time delay is a buffer to make sure that the modem has transmitted the message but should normally be left at zero. See RTS Send Delay and RTS Off Delay on page 339 for further guidelines for setting this parameter.
RTS Send Delay	Defines the amount of time, in 20 millisecond increments that elapses between the assertion of the RTS signal and the beginning of the message transmission. This time allows the modem to prepare to transmit the message. The Clear to Send (CTS) signal must be high for transmission to occur. See RTS Send Delay and RTS Off Delay on page 339 for further guidelines for setting this parameter.
Pre-Transmit Delay	Defines the amount of time in 1 millisecond increments that elapses between when the controller has a message to send and when it asserts the RTS signal.
Message Retries	Defines the number of times a master station retries either: • A message before it declares the message undeliverable • A poll packet to an active station before the master station declares that station to be inactive.
Priority Polling Range – High	Select the last slave station address to priority poll.
Priority Polling Range – Low	Select the first slave station address to priority poll. Entering 255 disables priority polling.
Normal Polling Range – High	Select the last slave station address to normal poll.
Normal Polling Range – Low	Select the first slave station address to normal poll. Entering 255 disables normal polling.
Normal Poll Group Size	Enter the quantity of active stations in the normal poll range that you want polled during a scan through the normal poll range before returning to the priority poll range. If no stations are configured in the Priority Polling Range, leave this parameter at 0.

> **IMPORTANT**   The unconnected timeout value in the message instruction should always be larger than the pre-transmit delay.

Minimum DF1 Half-duplex Master ACK Timeout

The governing timeout parameter to configure for a DF1 Half-duplex master is the serial port ACK Timeout. The ACK Timeout is the amount of time that you want the controller to wait for an acknowledgment of its message transmissions. Set in 20 millisecond intervals, the value is the amount of time the master waits for:

- An ACK to be returned by a slave when the master has sent it a message, or
- A poll response or message to be returned by a slave when the master has sent it a poll packet.

The timeout must be long enough that after the master has transmitted the last character of the poll packet, there is enough time for a slave to transmit (and the master receive) a maximum-sized packet before the time expires.

To calculate the minimum ACK timeout, you must know:

- The modem communication rate
- Maximum-sized data packet (the maximum number of data words that a slave write command or read reply packet might contain)
- The RTS/CTS or turnaround delay of the slave modem
- The configured RTS Send Delay in the slave
- The program scan time of the slave

Determining Minimum Master ACK Timeout

To determine the minimum ACK Timeout, you must first calculate the transmission time by multiplying the maximum-sized data packet for your controller by the modem rate in ms/byte. For example, we assume a Micro800 controller (103 data words or 224 bytes total packet size including overhead) and a 9600 bps modem, which transmits at approximately 1 ms/byte. Therefore, the message transmission time is 224 ms. For approximate modem transmission rates, see the following table.

Table 137 - Approximate Modem Transmission Rates

Speed (bps)	Rate, approx (ms/byte)
4800	2 ms/byte
9600	1 ms/byte
19200	0.5 ms/byte

Next, you must determine the average slave program scan time. In Connected Components Workbench software, double-click the controller variables in the Controller Organizer and locate the system tag _SYSVA_TCYCYCTIME in the Variable tab. For this example, assume that the program scan time is 20 ms. The program scan time varies by application.

Finally, you must determine the larger of two values, either the configured slave RTS Send Delay or the turnaround time of the slave modem. The RTS Send Delay time can be found by in the Configuration screen of the Micro800 embedded serial port. The RTS Send Delay time is in intervals of 20 ms, so with a value of 3 in the box, the RTS Send Delay time is 20 ms multiplied by 3. Using this value (60 ms) for our example, and assuming that the turnaround time of the modem is 50 ms (which varies by modem), choose to use the RTS Send Delay time of 60 ms for your calculation.

Having determined the maximum message transmission time (224 ms), the average slave program scan time (20 ms) and the largest of either the RTS Send Delay (60 ms) or the modem turnaround time, the minimum ACK timeout is the sum of these values.

Parameter	Example Values (in ms)	
Max Message Transmission Time	224	
Average Program Scan Time	20	
RTS Send Delay	60	
Modem Turnaround Time	50	} Use only the largest of these two values.
Calculated ACK Timeout	304	
Round up to Nearest 20 ms	320	

DF1 Half-Duplex Master Communication Diagnostics

Communication diagnostics is available while connected to the controller by clicking the Diagnose communication status button. Table 138 explains information regarding the diagnostic counter data displayed.

1. Click Diagnose communication status to bring up the DF1 Half-duplex master diagnostics.
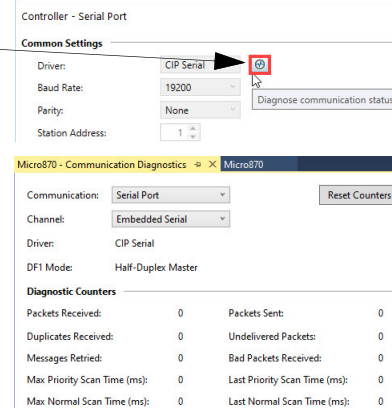
2. See Table 138 for details concerning the DF1 Half-duplex master Communication Diagnostics screen.

Table 138 - DF1 Half-duplex Master Communication Diagnostics Parameters

Status Field	Definition
Packets Sent	The total number of DF1 messages sent by the controller (including message retries)
Packets Received	The number of messages received with no errors
Last Normal Scan Time (ms)	Time in millisecond increments of last scan through Normal Poll List
Last Priority Scan Time (ms)	Time in millisecond increments of last scan through Priority Poll List
Message Retried	The number of message retries sent by the controller
Undelivered Packets	The number of messages that were sent by the controller but not acknowledged by the destination device
Duplicate Received	The number of times the controller received a message packet identical to the previous message packet
Bad Packet Received	The number of incorrect data packets received by the controller for which no ACK was returned
Max Normal Scan Time (ms)	Maximum time in millisecond increments to scan the Normal Poll List
Max Priority Scan Time (ms)	Maximum time in millisecond increments to scan the Priority Poll List

Configure a Message-based Mode DF1 Half-duplex Master Station

Choose message-based communication mode if you want to use MSG instructions in user programming to communicate with one station at a time. If your application uses satellite transmission or cellular transmission, consider choosing message-based. Communication to a slave station can be initiated on an as-needed basis.

With message-based mode, you do not have an active node file that you can use to monitor station status. Also, you cannot implement slave station-to-slave station messaging or slave programming.

To configure the controller for a master station using message-based communication, place the controller in program mode and follow the steps below in the Connected Components Workbench software.

1. To bring up the configuration page, click Serial Port.
2. On the Serial Port configuration page, select Half-Duplex Master for your DF1 Mode.
3. Choose a Message-based Polling Mode.
4. Configure the rest of the communication driver according to Table 139.
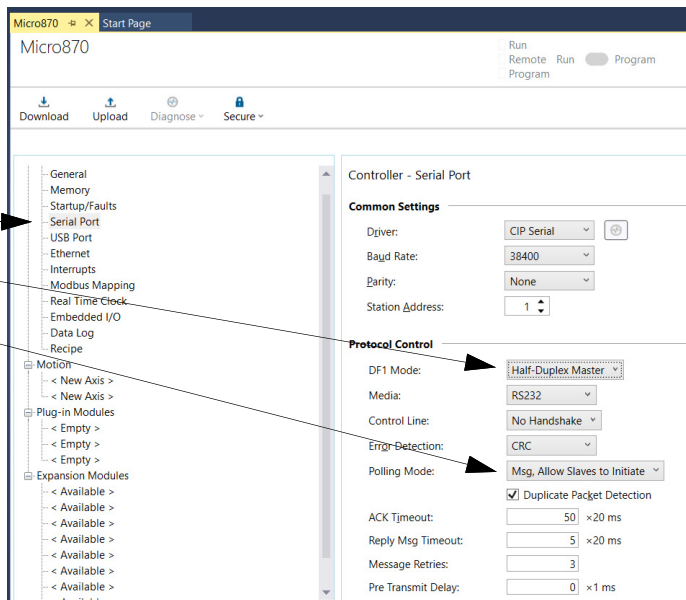
Define the parameters shown in Table 139 when configuring a Micro800 controller as a master station using message-based communication mode to talk to slave stations.

Table 139 - Configure a Micro800 Controller as a Master Using Message-based Communication Mode

Parameter	Selections
Baud Rate	Select a communication rate that all devices in your system support. Configure all devices in the system for the same communication rate.
Parity	Parity provides additional message packet error detection. To implement even parity checking, choose Even. To implement no parity checking, choose None.
Node Address	A node address identifies the controller on the DF1 Half-duplex link. Each station on a link must have a unique address. Choose an address between 0_{10} and 254_{10}. Node address 255_{10} is the broadcast address, and cannot be selected as a station's individual address.
Media	Select the communication media for the DF1 protocol: • RS-232 • RS-485 (only available when DF1 mode is Half-duplex)
Control Line	This parameter defines the mode in which the driver operates. Choose a method appropriate for your system's configuration: • If you are not using a modem, choose NO HANDSHAKE. • If the master modem is full-duplex, choose FULL-DUPLEX (RTS ALWAYS ON). • If all modems in the system are half-duplex, choose HALF-DUPLEX WITHOUT CONTINUOUS CARRIER (RTS/CTS). See Modem Control Line Operation on page 337 for descriptions of control line operation settings.

Error Detection	With this selection, you choose how the controller checks the accuracy of each DF1 packet transmission.
	BCC: This algorithm provides a medium level of data security. It cannot detect:
	  – Transposition of bytes during transmission of a packet
	  – The insertion or deletion of data values of zero within a packet
	CRC: This algorithm provides a higher level of data security.
	Select an error detection method that all devices in your configuration can use.
	When possible, choose CRC.

Table 139 - Configure a Micro800 Controller as a Master Using Message-based Communication Mode (Continued)

Parameter	Selections
Polling Mode	If you want to: • Accept unsolicited messages from slave stations, choose MESSAGE BASED (ALLOW SLAVES TO INITIATE MESSAGES) Slave station-initiated messages are acknowledged and processed after all master station-initiated (solicited) messages. **Note:** Slave stations can only send messages when they are polled. If the message-based master station never sends a slave station a message, the master station never sends the slave station a poll. Therefore, to obtain a slave station-initiated message from a slave station regularly, you should choose to use standard communication mode instead. • Ignore unsolicited messages from slave stations, choose MESSAGE BASED (DO NOT ALLOW SLAVES TO INITIATE MESSAGES) Slave station-initiated messages are acknowledged and discarded. The master station acknowledges the slave station-initiated message so that the slave station removes the message from its transmit queue, which allows the next packet that is slated for transmission into the transmit queue.
Duplicate Packet Detect	Duplicate Detect lets the controller detect if it has received a message that is a duplicate of its most recent message from another station. If you choose duplicate detect, the controller acknowledges (ACK) the message but does not act on it since it has already performed the message's task when it received the command from the first message. If you want to detect duplicate packets and discard them, check this parameter. If you want to accept duplicate packets and execute them, leave this parameter unchecked.
Reply Message Wait Timeout	Defines the amount of time in 20 millisecond increments that the master station will wait after receiving an ACK (to a master-initiated message) before polling the slave station for a reply. Choose a time that is, at minimum, equal to the longest time that a slave station must format a reply packet. This would typically be the maximum scan time of the slave station.
ACK Timeout	Defines the amount of time in 20 millisecond increments that you want the controller to wait for an acknowledgment to the message it has sent before the controller retries the message or the message errors out. This timeout value is also used for the poll response timeout. See See Minimum DF1 Half-duplex Master ACK Timeout on page 341 for recommendations to minimize this value.
RTS Off Delay	Defines the amount of time in 20 millisecond increments that elapses between the end of the message transmission and the de-assertion of the RTS signal. This time delay is a buffer to make sure that the modem has transmitted the message but should normally be left at zero. See RTS Send Delay and RTS Off Delay on page 339 for further guidelines for setting this parameter.
RTS Send Delay	Defines the amount of time in 20 millisecond increments that elapses between the assertion of the RTS signal and the beginning of the message transmission. This time allows the modem to prepare to transmit the message. The Clear to Send (CTS) signal must be high for transmission to occur. See RTS Send Delay and RTS Off Delay on page 339 for further guidelines for setting this parameter.
Pre-Transmit Delay	Defines the amount of time in 1 millisecond increments that elapses between when the controller has a message to send and when it asserts the RTS signal.
Message Retries	Defines the number of times a master station retries a message before it declares the message undeliverable.

> **IMPORTANT**   The unconnected timeout value in the message instruction should always be larger than the pre-transmit delay.

Configure a Slave Station

To choose the controller as a slave station, follow the steps below using your programming software:

1. To bring up the configuration page, select Serial Port.
2. On the Serial Port configuration page, select Half-Duplex Slave for your DF1 Mode.
3. Configure the rest of the communication driver according to Table 140.
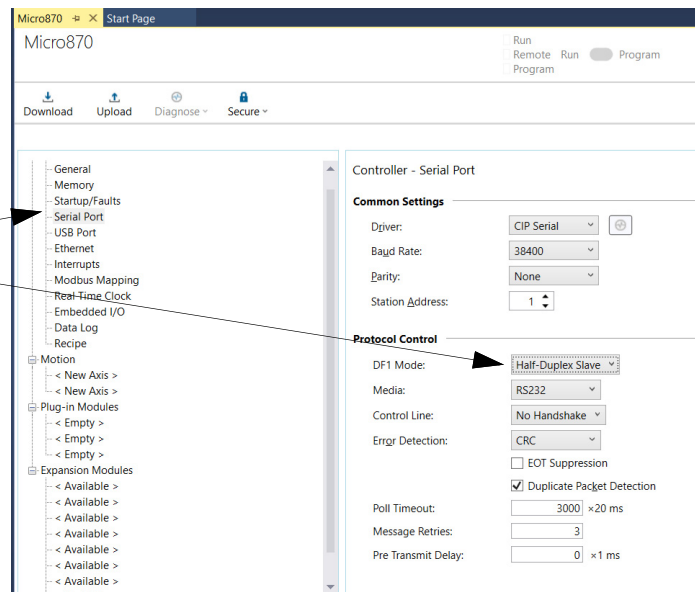
Define these parameters when configuring a Micro800 controller as a slave station.

Table 140 - Configure a Micro800 Controller as a Slave Station

Parameter	Selections
Baud Rate	Select a communication rate that all devices in your system support. Configure all devices in the system for the same communication rate.
Parity	Parity provides additional message packet error detection. To implement even parity checking, choose Even. To implement no parity checking, choose None.
Node Address	A node address identifies the controller on the DF1 half-duplex link. Each station on a link must have a unique node address. Choose an address between 0_{10} and 254_{10}. Node address 255_{10} is the broadcast address, which you cannot select as a station's individual address.
Control Line	This parameter defines the mode in which the driver operates. Choose a method appropriate for your system's configuration: • If you are not using a modem, choose NO HANDSHAKE. • If the master modem is full-duplex and the slave modem is half-duplex, choose HALF-DUPLEX WITHOUT CONTINUOUS CARRIER (RTS/CTS). See page Modem Control Line Operation on page 337 for descriptions of the control line operation settings.
Error Detection	With this selection, you choose how the controller checks the accuracy of each DF1 packet transmission. **BCC:** This algorithm provides a medium level of data security. It cannot detect: – Transposition of bytes during transmission of a packet – The insertion or deletion of data values of zero within a packet **CRC:** This algorithm provides a higher level of data security. Select an error detection method that all devices in your configuration can use. When possible, choose CRC.
Duplicate Packet Detect	Duplicate Detect lets the controller detect if it has received a message that is a duplicate of its most recent message from the master station. If you choose duplicate detect, the controller acknowledges (ACK) the message but does not act on it since it has already performed the message's task when it received the command from the first message. If you want to detect duplicate packets and discard them, check this parameter. If you want to accept duplicate packets and execute them, leave this parameter unchecked.

Table 140 - Configure a Micro800 Controller as a Slave Station (Continued)

Parameter	Selections
Poll Timeout	The timer keeps track of how often the station is polled. If the station has a message to send, it starts a timer. If the poll timeout expires before the message timeout, which you specify in the MSG control block, the MSG error bit is set and the message is removed from the transmit queue. If the message timeout, which you specify in the MSG control block, expires before the poll timeout expires, the MSG error bit and MSG timeout bit are set. The poll timeout can be disabled by entering a zero. See Configure Poll Timeout on page 347 for recommendations to minimize this value.
RTS Off Delay	Defines the amount of time in 20 millisecond increments that elapses between the end of the message transmission and the de-assertion of the RTS signal. This time delay is a buffer to make sure that the modem has transmitted the message, but should normally be left at zero. See RTS Send Delay and RTS Off Delay on page 339 for further guidelines for setting this parameter.
RTS Send Delay	Defines the amount of time in 20 millisecond increments that elapses between the assertion of the RTS signal and the beginning of the message transmission. This time allows the modem to prepare to transmit the message. The Clear to Send (CTS) signal must be high for transmission to occur. See RTS Send Delay and RTS Off Delay on page 339 for further guidelines for setting this parameter.
Message Retries	Defines the number of times a slave station resends its message to the master station before the slave station declares the message undeliverable.
Pre-Transmit Delay	Defines the amount of time in 1 millisecond increments that elapses between when the controller has a message to send and when it asserts the RTS signal.
EOT Suppression	If you want to minimize traffic on the network, you can choose to have the slave station not send EOT packets to the master station. When EOT packets are suppressed, the master station automatically assumes that a slave station has no data to give if the slave station does not send a message packet as a response to a poll. A disadvantage of suppressing EOTs is that the master station cannot distinguish between an active station that has no data to transmit and an inactive station. A possible application for suppressing EOTs is the following: conserving power with a radio modem because the radio transmitter does not have to power up to transmit a DLE EOT packet ("no data to give" packet). To suppress EOTs, check this parameter. To have the controller send EOTs, leave this parameter unchecked.

Configure Poll Timeout

The Poll Timeout is only used when the DF1 Half-duplex slave is initiating MSG instructions in ladder logic. This implies that the master is most likely configured for Standard Polling Mode. The minimum Poll Timeout value is dependent on the maximum master poll scan rate. Since the master's polling and the slave's triggering of a MSG instruction are asynchronous events, it is possible that in the instant just after the slave was polled, the MSG instruction gets triggered. This means the MSG instruction remains queued-up for transmission until the master has polled every other slave first. Therefore, the minimum slave Serial Port Poll Timeout value is equal to the maximum master Scan Poll Rate rounded up to the next 20 ms increment.

Minimum Serial Port Poll Timeout = Maximum Master Scan Poll Rate

DF1 Half-duplex Slave Communication Diagnostics

Communication diagnostics is available while connected to the controller by clicking the Diagnose communication status button. Table 141 explains information regarding the diagnostic counter data displayed.

1. Select Diagnose communication status to bring up the DF1 Half-duplex slave diagnostics.
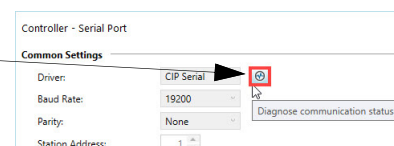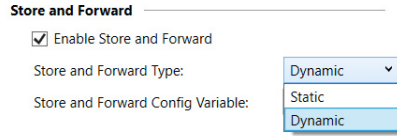
2. See Table 141 for details concerning the DF1
Half-duplex slave Communication
Diagnostics screen.
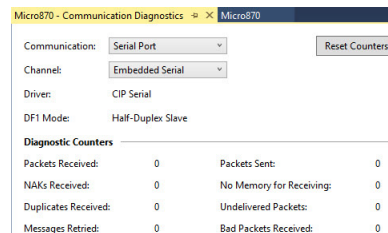
Table 141 - DF1 Half-duplex Slave Communication Diagnostics Parameters

Status Field	Definition
Packets Sent	The total number of DF1 messages sent by the controller (including message retries)
Packets Received	The number of messages received with no errors
NAKs Received	The number of NAKs received by the controller
No Memory for Receiving	The number of times the controller could not receive a message because it did not have available memory
Messages Retried	The number of message retries sent by the controller
Undelivered Packets	The number of messages that were sent by the controller but not acknowledged by the destination device
Duplicate Received	The number of times the controller received a message packet identical to the previous message packet
Bad Packets Received	The number of incorrect data packets received by the controller for which no ACK was returned

Configure a Radio Modem Station

To configure a Micro800 controller channel 1 for DF1 Radio Modem, do the following using your programming software:

1. To bring up the configuration page, select Serial Port.
2. On the Serial Port configuration page, select Radio Modem for your DF1 Mode.
3. Configure the rest of the communication driver according to Table 142.
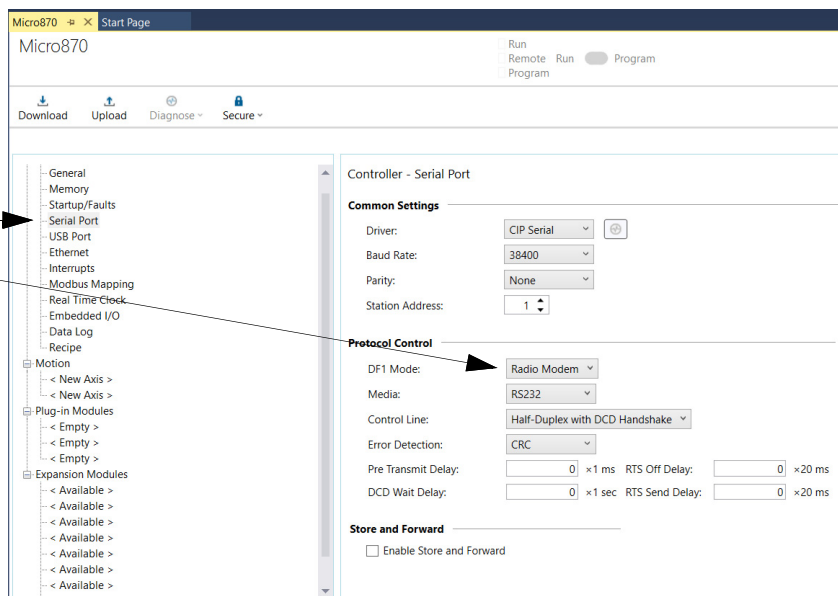
Table 142 - Configure a Micro800 controller for DF1 Radio Modem Communication

Parameter	Default	Selections
Baud Rate	19,200	Select a communication rate that all devices in your system support. Configure all devices in the system for the same communication rate.
Parity	None	Parity provides additional message packet error detection. To implement even parity checking, choose Even. To implement odd parity checking, choose Odd. To implement no parity checking, choose None.
Node Address	1	A node address identifies the controller on the DF1 half-duplex link. Each station on a link must have a unique node address. Choose an address between 0_{10} and 254_{10}. Node address 255_{10} is the broadcast address, which you cannot select as a station's individual address.
Enable Store and Forward	Not selected (Disabled)	See Configure the Store and Forward Table on page 350 for more information.

Table 142 – Configure a Micro800 controller for DF1 Radio Modem Communication (Continued)

Parameter	Default	Selections
Control Line	No Handshake	This parameter defines the mode in which the driver operates. Choose a method appropriate for your system's configuration: • If you are not using a modem, choose NO HANDSHAKE. • Half-duplex without Continuous Carrier (RTS/CTS) • Half-duplex with DCD Handshaking See Modem Control Line Operation on page 337 for descriptions of the control line operation settings.

Error Detection	CRC	With this selection, you choose how the controller checks the accuracy of each DF1 packet transmission.
		BCC: This algorithm provides a medium level of data security. It cannot detect:
		  – Transposition of bytes during transmission of a packet
		  – The insertion or deletion of data values of zero within a packet
		CRC: This algorithm provides a higher level of data security.
		Select an error detection method that all devices in your configuration can use.
		When possible, choose CRC.

RTS Off Delay	0	Defines the amount of time in 20 millisecond increments that elapses between the end of the message transmission and the de-assertion of the RTS signal. This time delay is a buffer to make sure that the modem has transmitted the message, but should normally be left at zero. See RTS Send Delay and RTS Off Delay on page 339 for further guidelines for setting this parameter.
RTS On Delay	0	Defines the amount of time in 20 millisecond increments that elapses between the assertion of the RTS signal and the beginning of the message transmission. This time allows the modem to prepare to transmit the message. The Clear to Send (CTS) signal must be high for transmission to occur. See RTS Send Delay and RTS Off Delay on page 339 for further guidelines for setting this parameter.
Pre-Transmit Delay	0	Defines the amount of time in 1 millisecond increments that elapses between when the controller has a message to send and when it asserts the RTS signal (if handshaking is selected) or begins transmitting (if no handshaking is selected).

DF1 Radio Modem Communication Diagnostics

Communication diagnostics is available while connected to the controller by clicking the Diagnose communication status button. Table 143 explains information regarding the diagnostic counter data displayed.

1. Select Diagnose communication status to bring up the DF1 Radio Modem diagnostics.
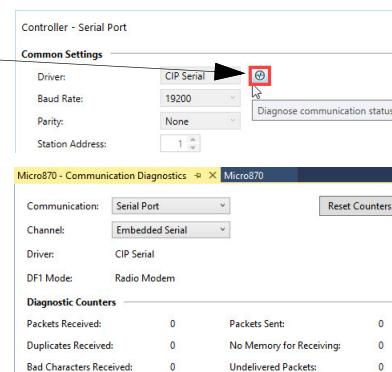
2. See Table 143 for details concerning the DF1 Radio Modem Communication Diagnostics screen.

Table 143 – DF1 Radio Modem Communication Diagnostics Parameters

Status Field	Definition
Packets Sent	The total number of DF1 messages sent by the controller
Packets Received	The number of messages received with no errors
No Memory for Receiving	The number of times the controller could not receive a message because it did not have available memory
Undelivered Packets	The number of messages that were sent by the controller but not acknowledged by the destination device
Duplicate Received	The number of times the controller received a message packet identical to the previous message packet
Bad Characters Received	The number of data characters received with transmission errors by the controller

Configure the Store and Forward Table

The Store and Forward function in Micro800 controllers provide two methods to configure – Dynamic, which requires creating a Store and Forward table, or Static, which you can directly configure in the Connected Components Workbench software.

Store and Forward

☑ Enable Store and Forward

Store and Forward Type: Dynamic
Store and Forward Config Variable: Static
                                      Dynamic

Use the static method when you want to configure the Store and Forward tables while the controller is offline. Use the dynamic method when you want to configure the Store and Forward tables while the controller is in RUN mode. The dynamic Store and Forward table occupies a BOOL array [0...255] or DWORD array [0...7] data type. Each bit in this array corresponds to a DF1 Radio Modem node address.

To configure a Micro800 controller to Store and Forward message packets between two other nodes, the bits corresponding to the addresses of those two other nodes must be set. For instance, if node 2 is used to Store and Forward message packets between nodes 1 and 3, then both Bit 1 and Bit 3 must be set in the Store and Forward table (see Figure 70).

IMPORTANT	Once Store and Forward is enabled, duplicate packet detection is also automatically enabled. Whenever Store and Forward is used within a radio modem network, every node should have Store and Forward enabled, even if all bits in the file are cleared, so that duplicate packets are ignored.

Figure 70 – Applying Store and Forward with DF1 Radio Modem Protocol
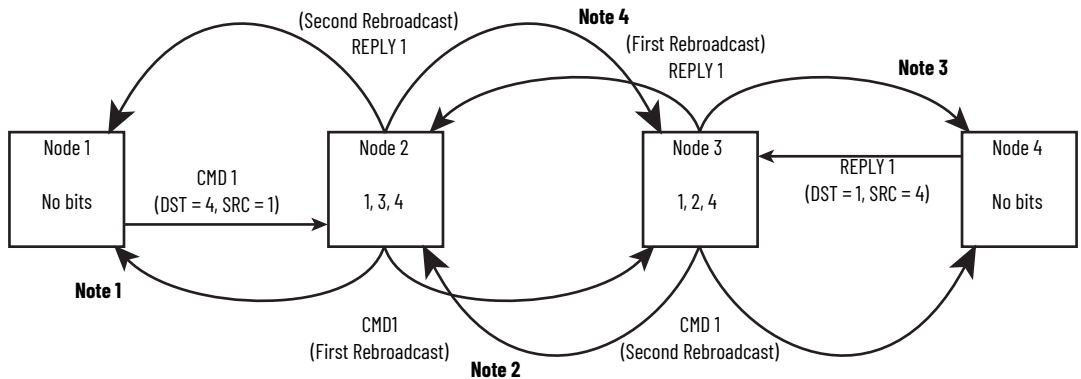
Note 1 – The link layer of Node 1 blocks the re-transmission of a packet that is received with the SRC byte equal to the receiving node's station address. Packets received that originate from the receiving node should never be re-transmitted.
Note 2 – To prevent Node 2 from re-transmitting a duplicate packet, the link layer of Node 2 updates the duplicate packet table with the last 20 packets received.
Note 3 – The link layer of Node 4 blocks the re-transmission of a packet that is received with the SRC byte equal to the receiving node's station address. Packets received that originate from the receiving node should never be re-transmitted.
Note 4 – To prevent Node 3 from re-transmitting a duplicate packet, the link layer of Node 3 updates the duplicate packet table with the last 20 packets received.

Figure 71 - Store and Forward Table for Node 2, Static Method

Store and Forward

☑ Enable Store and Forward

Store and Forward Type: [Static   ▾]

0 - 31	32 - 63	64 - 95	96 - 127	128 - 159	160 - 191	192 - 223	224 - 255

0	No Forward	8	No Forward	16	No Forward	24	No Forward
1	Forward	9	No Forward	17	No Forward	25	No Forward
2	No Forward	10	No Forward	18	No Forward	26	No Forward
3	Forward	11	No Forward	19	No Forward	27	No Forward
4	Forward	12	No Forward	20	No Forward	28	No Forward
5	No Forward	13	No Forward	21	No Forward	29	No Forward
6	No Forward	14	No Forward	22	No Forward	30	No Forward
7	No Forward	15	No Forward	23	No Forward	31	No Forward

Figure 72 - Store and Forward Table for Node 2, Dynamic Method

Dynamic method with BOOL array

Dynamic_BOOL		...	...	...		BOOL	[0..255]
Dynamic_BOOL[0]		☐	N/A		☐	BOOL	
Dynamic_BOOL[1]		☑	N/A		☐	BOOL	
Dynamic_BOOL[2]		☐	N/A		☐	BOOL	
Dynamic_BOOL[3]		☑	N/A		☐	BOOL	
Dynamic_BOOL[4]		☑	N/A		☐	BOOL	
Dynamic_BOOL[5]		☐	N/A		☐	BOOL	
Dynamic_BOOL[6]		☐	N/A		☐	BOOL	
Dynamic_BOOL[7]		☐	N/A		☐	BOOL	
Dynamic_BOOL[8]		☐	N/A		☐	BOOL	
Dynamic_BOOL[9]		☐	N/A		☐	BOOL	
Dynamic_BOOL[10]		☐	N/A		☐	BOOL	
Dynamic_BOOL[11]		☐	N/A		☐	BOOL	
Dynamic_BOOL[12]		☐	N/A			BOOL	

Dynamic method with DWORD array

Dynamic_DWORD		...	...	...		DWORD	[0..7]
Dynamic_DWORD[0]		26	N/A		☐	DWORD	
Dynamic_DWORD[1]		0	N/A		☐	DWORD	
Dynamic_DWORD[2]		0	N/A		☐	DWORD	
Dynamic_DWORD[3]		0	N/A		☐	DWORD	
Dynamic_DWORD[4]		0	N/A		☐	DWORD	
Dynamic_DWORD[5]		0	N/A		☐	DWORD	
Dynamic_DWORD[6]		0	N/A		☐	DWORD	
Dynamic_DWORD[7]		0	N/A		☐	DWORD	

Notes:

User-defined Function Block Motion Instructions

PowerFlex 520-series User-defined Function Block Details

This section provides details for each instruction.

RA_PF523_VEL and RA_PF525_VEL

These instructions allow simple PowerFlex 523 or PowerFlex 525 drive control in Velocity Mode.
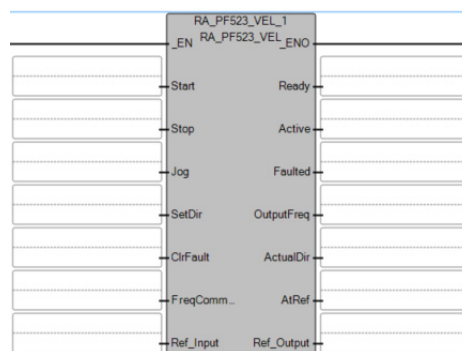
Table 144 - UDFB for RA_PF523_VEL and RA_PF525_VEL

Operand	Type	Description
Instance	RA_PF523_VEL_Y or RA_PF525_VEL_Y	Unique instance of the RA_PF523_VEL UDFB or RS_PF525_VEL UDFB
_EN	BOOL	Trigger of UDFB to start executing the instruction
Start	BOOL	Commands the drive to start.
Stop	BOOL	Performs a normal stop.
Jog	BOOL	Commands the drive to jog.
SetDir	BOOL	Commands the direction of drive. 0 = Reverse Command 1 = Forward Command
ClrFault	BOOL	Clears the drive fault.
FreqCommand	REAL	Controls the reference speed of the drive. In units of 0.01 Hz.
Ref_Input	PowerFlex523_I or PowerFlex525V_I	References the input data structure of the drive to control.
_ENO	BOOL	Indicates that the UDFB is executing.
Ready	BOOL	Indicates that the drive is ready for operation.
Active	BOOL	Indicates that the drive is operating.
Faulted	BOOL	Indicates fault state
OutputFreq	REAL	Displays the reference speed of the drive.
ActualDir	STRING	Indicates the rotating direction.
AtRef	BOOL	Indicates that the drive is at reference speed.
Ref_Output	PowerFlex523_O or PowerFlex525V_O	References the output data structure of the drive to control.

RA_PF525_POS

This instruction allows simple PowerFlex 525 drive control in Position Mode.
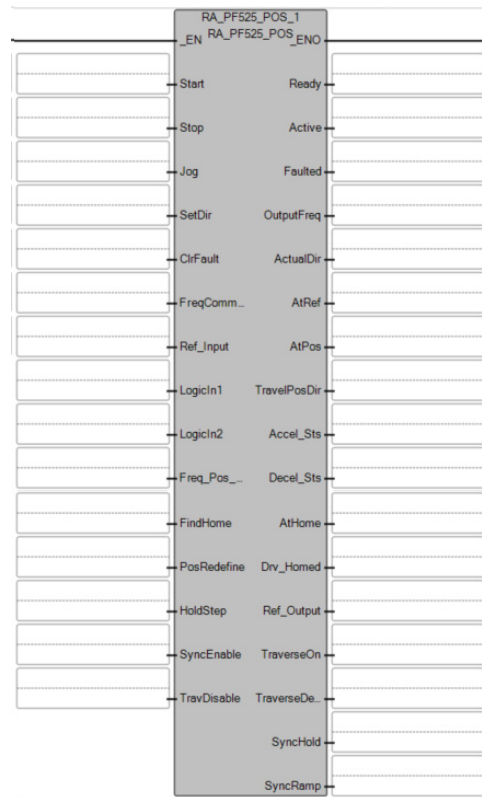
Table 145 - UDFB for RA_PF525_POS

Operand	Type	Description
Instance	RA_PF525_POS_X	Unique instance of the RA_PF525_POS UDFB
_EN	BOOL	Trigger of UDFB to start executing the instruction
Start	BOOL	Commands the drive to start.
Stop	BOOL	Performs a normal stop.
Jog	BOOL	Commands the drive to jog.
SetDir	BOOL	Commands the direction of drive. 0 = Reverse Command 1 = Forward Command
ClrFault	BOOL	Clears the drive fault.
FreqCommand	REAL	Controls the reference speed of the drive. In units of 0.01 Hz.
Ref_Input	PowerFlex525P_I	References the input data structure of the drive to control.
LogicIn1	BOOL	Provides an identical function as the "Logic In1" Digital Input option.
LogicIn2	BOOL	Provides an identical function as the "Logic In2" Digital Input option.
Freq_PosSel1	BOOL	Frequency and Position selection
Freq_PosSel2	BOOL	000 = Frequency and Position Step 0 001 = Frequency and Position Step 1
Freq_PosSel3	BOOL	010 = Frequency and Position Step 2 011 = Frequency and Position Step 3 100 = Frequency and Position Step 4 101 = Frequency and Position Step 5 110 = Frequency and Position Step 6 111 = Frequency and Position Step 7
FindHome	BOOL	Next start command causes the drive to find home. 1 = Find Home
HoldStep	BOOL	Overrides other inputs and causes the drive to remain at its current step (running at zero speed once it reaches its position) until released. 1 = Hold Step

Table 145 - UDFB for RA_PF525_POS (Continued)

Operand	Type	Description
PosRedefine	BOOL	Resets the home position to the current position of the machine. Set this bit to 0 after the homing routine is completed. 1 = Pos Redefine
SyncEnable	BOOL	Holds the existing frequency when Sync Time is set to enable Speed Synchronization. 1 = Sync Enable
TravDisable	BOOL	Disable Traverse function 1= Traverse Disable
_ENO	BOOL	Indicates that the UDFB is executing.
Ready	BOOL	Indicates that the drive is ready for operation.
Active	BOOL	Indicates that the drive is operating.
Faulted	BOOL	Indicates fault state.
OutputFreq	REAL	Displays the reference speed of the drive.
ActualDir	STRING	Indicates the rotating direction.
AtRef	BOOL	Indicates that the drive is at the reference speed.
AtPos	BOOL	Indicates that the drive is at the commanded position.
TravelPosDir	BOOL	Indicates the Travel Position direction.
Accel_Sts	BOOL	Indicates acceleration state.
Decel_Sts	BOOL	Indicates deceleration state.
AtHome	BOOL	Indicates that drive is at the reference home.
DriveHomed	BOOL	Indicates whether the drive has been homed since power-up.
Ref_Output	PowerFlex525P_O	References the output data structure of the drive to control.
SyncHold	BOOL	Indicates if the frequency is holding.
SyncRamp	BOOL	Indicates if the frequency is accelerating to the new commanded frequency in drive parameter A571 [Sync time].
TraverseOn	BOOL	Indicates if Traverse is enabled.
TraverseDecel	BOOL	Indicates if the drive is decelerating in Traverse Mode.

Kinetix 5100 Drive Device Object UDFB

The Device Object UDFB comes with the UDFB motion instructions that is downloaded from the PCDC. Each Kinetix 5100 drive requires a unique instance of the Device Object UDFB. This UDFB provides one software interface that is used by each drive that commands the drive and provides the drive status that you can use with your application logic.

Figure 73 - Device Object UDFB
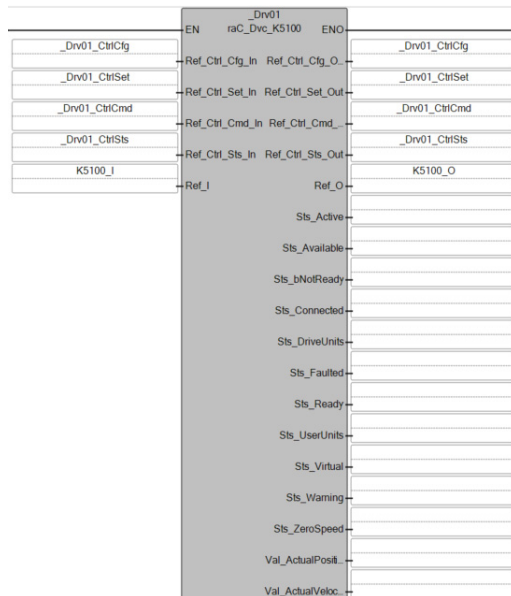
The Kinetix 5100 drive UDFB motion instructions aim to provide necessary simple motion functions. Table 146 lists the Kinetix 5100 drive UDFBs.

Table 146 - UDFB Motion Instruction List for Kinetix 5100 Drives

Name	Description
raC_Opr_K5100_MSO	Motion Servo On Use the Motion Servo On instruction to activate the drive output and to activate the drive servo loops.
raC_Opr_K5100_MSF	Motion Servo Off Use the Motion Servo Off instruction to deactivate the drive output and to deactivate the drive servo loops.
raC_Opr_K5100_MAJ	Motion Axis Jog Use the Motion Axis Jog instruction to accelerate or decelerate the motor at a constant speed without termination.
raC_Opr_K5100_MAT	Motion Axis Torque Use the Motion Axis Torque instruction to use torque limiting while a predefined speed is used to move the motor.
raC_Opr_K5100_MAM	Motion Axis Move Use the Motion Axis Move instruction to move the motor to a specified position.
raC_Opr_K5100_MAH	Motion Axis Home Use the Motion Axis Home instruction to home the motor.
raC_Opr_K5100_MAG	Motion Axis Gear Use the Motion Axis Gear instruction to set the gear ratio between a pulse-source and follower drive. **IMPORTANT:** This UDFB changes the drive E-gear ratio; Slave/Follower ID151 (P1.044) and Master ID152 (P1.045) Counts. If your drive is positioning, be aware that the units are impacted because the E-gear ratio controls the counts/motor rotation value.
raC_Opr_K5100_MAS	Motion Axis Stop Use the Motion Axis Stop instruction to stop a specific motion process on the motor or to stop the motor completely.
raC_Opr_K5100_MAFR	Motion Axis Fault Reset Use the Motion Axis Fault Reset instruction to clear many motion faults for the drive. Some faults cannot be cleared until you cycle power to the drive. The faults, which you can be clear with raC_*xxx*_K5100_MAFR, are listed in the fault list section.
raC_Opr_K5100_MAI	Motion Axis Index Use the Motion Axis Index instruction to execute the specified PR (index) function of the drive. Use K5100C configuration software or explicit messaging to set the PR (index) parameters. The raC_Opr_K5100_MAI instruction specifies the PR (index) number to be executed.

Configure UDFBs for Kinetix 5100 Drives

Follow the steps described in Import the User-defined Function Block Instruction Files on page 146 to install the Device Object and Motion Instruction UDFBs. When the Operation UDFBs are created, use the sample logic that is created as guidance for creating your application logic.

raC_Dvc_K5100

When the Device Object UDFB is created, it references Data Types that interface with the drive to exchange command and status information. You must create each Device Object as a unique instance and create the interface tags in your program Local Variables / Global Variables. You must also map the module defined Global Variables (K5100_I and K5100_O) to Ref_I and Ref_O.

Figure 74 - Device Object Interface Tags
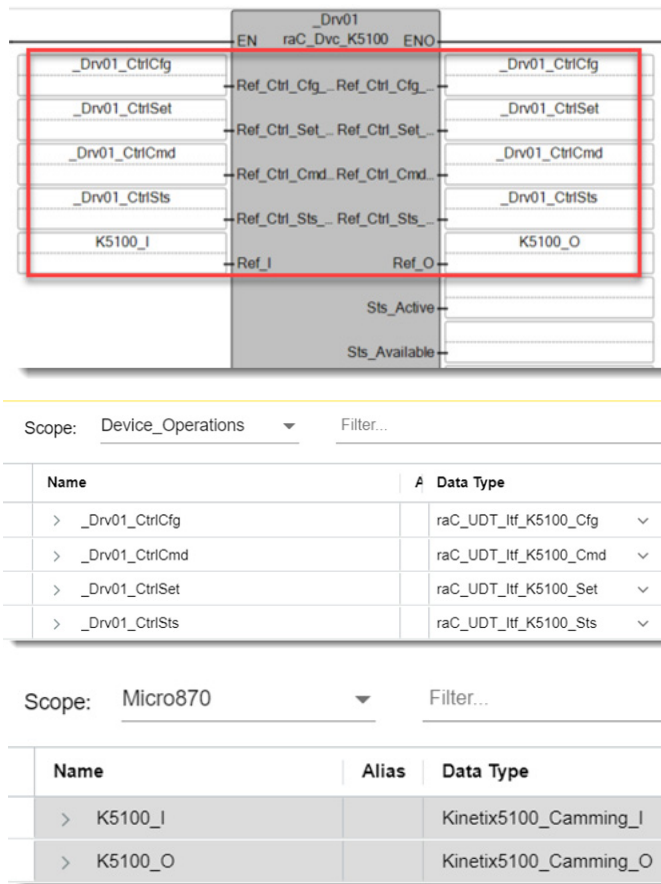
raC_UDT_Itf_K5100_Cfg

raC_UDT_Itf_K5100_Cfg is the user-defined data type for device configuration. Its members provide selection between drive units (counts) or user units (PU).

This selection is useful because the Kinetix 5100 drive natively supports only drive units. When the Operating Units = 1, the Motion Resolution and ConversionConstant values are used. Position Scaling originates from the KNX5100C software and is used together with the Cfg tags to derive user scaling units.

Table 147 - raC_UDT_Itf_K5100_Cfg Data Types

Member	Description	Data Type
OperatingUnits	0 = Drive Units; 1 = UserUnits	DINT
MotionResolution	Motion Counts per Motor Revolution	DINT
ConversionConstant	Motion Counts per Position Unit	LREAL

Example Configuration with Position Units

The E-gear ratio (KNX5100C -> Function List -> E-gear Ratio) is always used to provide a representation of positioning (units or counts) or to define a Pulse-Pulse Following relationship (MAG/PT). When the E-gear ratio is changed, the positioning of the drive is changed. When not using the MAG or PT operation mode, the E-gear ratio is used to define position scaling.

When Operating Units = 1, Position Units are used, and we can define application units instead of using drive counts. In KNX5100C software, the E-gear ratio is defined to provide Position scaling. This is encoder counts (or pulses)/motor rotation.

By using the KNX5100C software, navigate to Settings > E-gear Ratio.

Figure 75 - Position Units Configuration
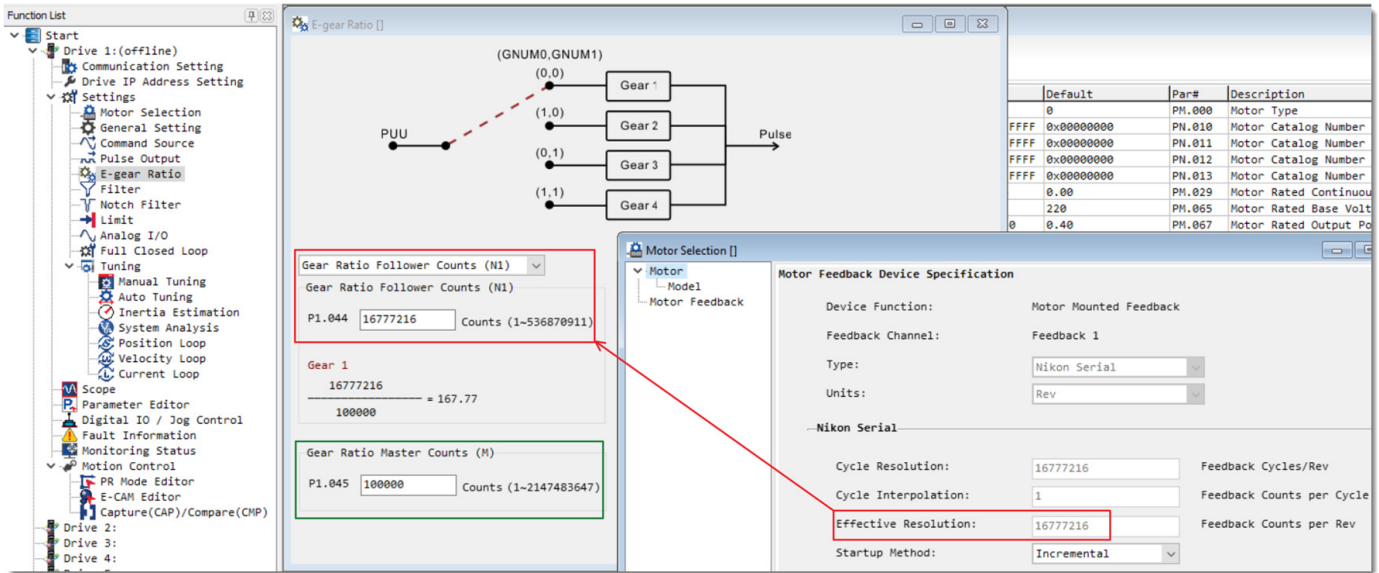
All Position Unit configurations must:

- Configure GearRatioFollowerCounts ID151 (P1.044) to be the same as the motor feedback effective resolution.
- Configure GearRatioMasterCounts ID152 (P1.045) to provide motor feedback counts/ motor rotation.
- You define this value and can be any count value, default values with high-resolution encoders are 100,000 counts/motor rotation. The E-gear configuration is used with the Device Object Cfg tags.

Figure 76 - Position Units Configuration Tag

The Device Object Cfg values must:

- Set Cfg.MotionResolution = GearRatioMasterCounts ID152 (P1.045) -> Motion Counts/ Motor Revolution.
- Set Cfg.ConversionConstant based on the Counts/Position Unit -> Motion Counts/ Position Unit that is required for your application.

The example in Figure 76 results in Position Units = motor rotations. Now, entry values that originally used drive counts can be entered as motor rotations.

raC_UDT_Itf_K5100_Set

raC_UDT_Itf_K5100_Set is the user-defined data type for device settings. Its members provide application program access to allow or inhibit commands and settings from other external sources. The table below shows member names, descriptions, and tag data types.

For example, to inhibit write commands from the other external sources write a 1 to the ModuleName_CtrlSet.InhibitCmd program tag from your application program. This write prevents a jog command from other external sources.

Table 148 - raC_UDT_Itf_K5100_Set Data Types

Member	Description	DataType
bInhibit	Bit overlay for external access restriction	DINT
InhibitCmd	1 = Inhibit user Commands from external sources; 0 = Allow Control. This member is only used with the optional device faceplate.	BOOL
InhibitSet	1 = Inhibit user Settings from external sources; 0 = Allow This member is only used with the optional device faceplate.	BOOL
OperatingMode	Determines the drive operating mode when 'Start Motion' has a zero-to-one transition. 1 - Position mode 2 - Speed mode 3 - Home mode 4 - Torque mode 5 - Gear mode (Fixed Ratio, based on present E-gear ratio) 6 - Index mode 7 - Reserved 8 - Gear Mode (Variable Ratio, based on Master/Slave tag values) 9 - Enhanced MAT mode	DINT
MoveType	Specifies the type of move. 0 = Absolute 1 = Incremental 2 = Rotary Shortest Path 3 = Rotary Positive 4 = Rotary Negative 7 = Relative 8 = Capture	DINT
PositionCommandOverlap	Allows overlapping of successive movements.	BOOL
PositionCommandOverride	Allows interruption of current movement, replacing it with a new movement.	BOOL
CapturedPositionSelect	Captures position selection (First capture or second capture).	BOOL
Position	Determines the command position.	REAL
Velocity	Determines the command speed.	REAL
Accel	Determines the command acceleration.	REAL
Decel	Determines the command deceleration.	REAL
Torque	Determines the command torque.	DINT
TorqueRampTime	Determines the command torque ramp time.	DINT
StartingIndex	This entry is the PR (Position Register) that the drive should execute.	DINT
HomingMethod	Homing Method	DINT
HomeReturnSpeed	Determines the command home return speed.	REAL
CamMasterReference	Future: Determines the master position reference of CAM.	DINT
CamExecutionSchedule	Future: Determines the method that is used to execute the CAM profile.	DINT
CamExecutionMode	Future: Determines if the cam profile is executed only one time or repeatedly.	DINT
CamStopMode	Future: Determines the stop mode of CAM.	BOOL
CamSlaveScaling	Future: Scales the total distance that is covered by the slave axis through the cam profile.	DINT
CamLockPosition	Future: Determines the starting location in the cam profile	DINT
CamMasterLockPosition	Future: Determines the master location where the slave axis locks to the mater axis.	DINT
CamMasterLeadingCounts	Future: Determines the leading counts (master axis) before the cam profile is executed.	DINT
CamMasterUnlockCounts	Future: Determines the unlock counts (master axis) when the cam profile is executed.	DINT

Table 148 - raC_UDT_Itf_K5100_Set Data Types (Continued)

Member	Description	DataType
CamMasterCyclicLeadingCounts	Future: Determines the cyclic leading counts (master axis) during the cam profile is executed.	DINT
GearRatioSlaveCounts	Integer value representing slave counts. This value is P1.044 Gear Ratio Follower Counts from the E-gear ratio in KNX5100C software.	DINT
GearRatioMasterCounts	Integer value representing master counts. This value is P1.045 Gear Ratio Master Counts from the E-gear ratio in KNX5100C software.	DINT

raC_UDT_Itf_K5100_Cmd

raC_UDT_Itf_K5100_Cmd is the user-defined data type for device commands. Its members provide application program access to common basic device commands.

Table 149 shows member names, descriptions, and tag data types.

All commands are available whether operating the device physically or virtually.

While it is possible, it is not typical to modify any of these UDT values directly. The Motion Operation UDFBs manipulate these values as a result of their operation.

Table 149 - raC_UDT_Itf_K5100_Cmd Data Types

Member	Description	Data Type
bCmd	Commands (Bit Overlay)	DINT
Physical	1 = Operate as a physical device	BOOL
Virtual	1 = Operate as a virtual device	BOOL
ResetWarn	1 = Reset device warning	BOOL
ResetFault	1 = Reset device trip or fault	BOOL
Activate	1 = Activate Output Power Structure	BOOL
Deactivate	1 = DeActivate Output Power Structure	BOOL
StartMotion	A zero-to-one transition means that the motion command is issued from the external controller.	BOOL
StopMotion	A zero-to-one transition stops any active motion command in the drive.	BOOL

raC_UDT_Itf_K5100_Sts

raC_UDT_Itf_K5100_Sts is the user-defined data type for device status. Its members provide application program access to device states, status, and diagnostic data. Table 150 shows member names, descriptions, and tag data types.

Table 150 - raC_UDT_Itf_K5100_Sts Data Types

Input	Description	Data Type
eState	Enumerated state value: 0 = Unused 1 = Initializing 2 = Disconnected 3 = Disconnecting 4 = Connecting 5 = Idle 6 = Configuring 7 = Available	DINT
FirstWarning	Capture the First Alarm Bit to trigger. Display the respective description and time stamp on the faceplate. Log the same in the Event Queue.	raC_UDT_Event
FirstFault	Capture the Fault Code of the device. Display the respective code, description, and time stamp on faceplate. Log the same in the Event Queue.	raC_UDT_Event

Table 150 - raC_UDT_Itf_K5100_Sts Data Types (Continued)

Input	Description	Data Type
eCmdFail	Enumerated command failure code	DINT
bSts	Status (Bit Overlays)	DINT
Physical	1 = Operating as a physical device	BOOL
Virtual	1 = Operating as a virtual device	BOOL
Connected	1 = PAC to device connection has been established.	BOOL
Available	1 = The automation device is available for interaction with the user program	BOOL
Warning	1 = A warning is active on the automation device	BOOL
Faulted	1 = A fault is active on the automation device	BOOL
Ready	1 = Device is ready to be Activated	BOOL
Active	1 = Device power structure is active	BOOL
ZeroSpeed	1 = Motor is within zero speed tolerance (this tolerance is defined in KNX5100C software)	BOOL
Homed	Indicates whether the drive completed the home operation.	BOOL
AtReference	Depending on the motion command (position, speed, torque), AtReference is 1 when the actual reference = command reference.	BOOL
CommandInProgress	Toggles state when a motion command is active in the drive. This bit changes state (toggles between 0 and 1) when a new command is executed from the drive. **IMPORTANT**: Once this bit changes state, it remains in that state during the command; it toggles to the opposite state (and remains in that state) once a new command is received.	BOOL
FaultCode	Active Fault Code in the drive	DINT
WarningCode	Active Warning Code in the drive	DINT
OperatingMode	Indicate which operating mode is used.	DINT
MotorType	Indicate which type of motor is connected to the drive. Rotary Motor = 1 Linear Motor =2 (Future)	DINT
ActualPosition	Actual position of the motor. Units depend on the Cfg settings. These can be drive counts or Position Units.	REAL
ActualVelocity	Actual speed of the motor. Units depend on the Cfg settings. These can be 0.1 RPM/sec or Position Units.	REAL
ActualTorque	When the operating mode is 4, Torque Mode, this represents the % motor torque.	REAL
ActiveIndex	Indicates the currently executing Position Register PR (index).	DINT
ParameterMonitor1Value	Parameter Monitor 1 Value You can use ID60 (P0.035) to specify the mapping parameter instance ID number. The content of the parameter that is specified by ID60 (P0.035) is shown in ID55 (P0.025).	DINT
ParameterMonitor2Value	Parameter Monitor 2 Value You can use ID61 (P0.036) to specify the mapping parameter instance ID number. The content of the parameter that is specified by ID61 (P0.036) is shown in ID56 (P0.026).	DINT
ParameterMonitor3Value	Parameter Monitor 3 Value You can use ID62 (P0.037) to specify the mapping parameter instance ID number. The content of the parameter that is specified by ID62 (P0.037) is shown in ID56 (P0.027).	DINT
ParameterMonitor4Value	Parameter Monitor 4 Value You can use ID63 (P0.038) to specify the mapping parameter instance ID number. The content of the parameter that is specified by ID63 (P0.038) is shown in ID57 (P0.028).	DINT
ParameterMonitor5Value	Parameter Monitor 5 Value You can use ID64 (P0.039) to specify the mapping parameter instance ID number. The content of the parameter that is specified by ID64 (P0.039) is shown in ID57 (P0.028).	DINT

UDFB Motion Instruction Details

Use the Device Object states to represent the axis, as these states incorporate the instruction and drive information in one location, which results in an accurate drive state representation.

This section provides details for each instruction.

raC_Opr_K5100_MSO

Use the Motion Servo On (raC_Opr_K5100_MSO) instruction to activate the drive amplifier for the specified axis and to activate the servo control loops.

Figure 77 - MSO Diagram
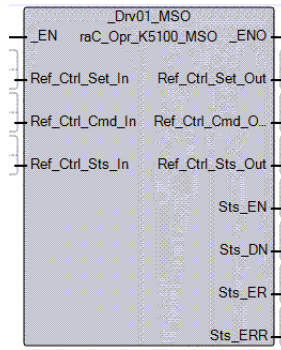
Table 151 - MSO Operands

Operand	Type	Format	Description
_EN	BOOL	Tag	True when the rung is enabled.
Ref_Ctrl_Set_In	raC_UDT_Itf_K5100_Set	Tag	Interface from device object
Ref_Ctrl_Cmd_In	raC_UDT_Itf_K5100_Cmd	Tag	Interface from device object
Ref_Ctrl_Sts_In	raC_UDT_Itf_K5100_Sts	Tag	Interface from device object
_ENO	BOOL	Tag	True when this UDFB output is enabled.
Ref_Ctrl_Set_Out	raC_UDT_Itf_K5100_Set	Tag	Interface to device object
Ref_Ctrl_Cmd_Out	raC_UDT_Itf_K5100_Cmd	Tag	Interface to device object
Ref_Ctrl_Sts_Out	raC_UDT_Itf_K5100_Sts	Tag	Interface to device object
Sts_EN (Enable)	BOOL	Tag	This bit is set when the rung makes a false-to-true transition. The bit remains set as the message transaction to activate the drive is initiated and in process. It remains set while the rung-in condition is true and no faults are active.
Sts_DN (Done)	BOOL	Tag	This bit is set when the rung makes a false-to-true transition and the cmd to activate the drive has been acknowledged.
Sts_ER (Error)	BOOL	Tag	This bit is set when the rung makes a false-to-true transition and there is an error that has occurred with the instruction. (This instruction error can be a result of a fault on the drive itself). See Sts_ERR for details on the cause of the error.
Sts_ERR	DINT	Tag	Instruction error codes. See Kinetix 5100 Drive UDFB Error Codes (Table 163) for details.

Use the raC_Opr_K5100_MSO instruction to activate the motor. This instruction must be used while there are no active faults on the drive and the drive is in a Ready State. The resulting state of the drive is reflected when Ref_Ctrl_Sts_In.Active is one.

IMPORTANT	The instruction execution can take multiple scans to execute because it requires multiple RPI updates to complete the request. The Done (Sts_DN) bit is not set immediately, but only after the request is completed.

Error Codes:
- 100 - Kinetix 5100 drive is not ready

- 101 - Kinetix 5100 drive is faulted
- 102 - Another raC_Opr_K5100_MSO message is executing
- 103 - raC_Opr_K5100_MSF is executing
- 129 - Motor not connected

raC_Opr_K5100_MSF

Use the Motion Servo Off (raC_Opr_K5100_MSF) instruction to deactivate the drive output for the specified axis and to deactivate the motor.

Figure 78 - MSF Diagram
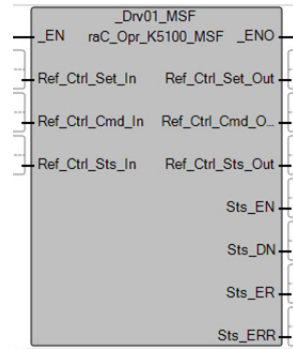
Table 152 - MSF Operands

Operand	Type	Format	Description
_EN	BOOL	Tag	True when the rung is enabled.
Ref_Ctrl_Set_In	raC_UDT_Itf_K5100_Set	Tag	Interface from device object
Ref_Ctrl_Cmd_In	raC_UDT_Itf_K5100_Cmd	Tag	Interface from device object
Ref_Ctrl_Sts_In	raC_UDT_Itf_K5100_Sts	Tag	Interface from device object
_ENO	BOOL	Tag	True when this UDFB output is enabled.
Ref_Ctrl_Set_Out	raC_UDT_Itf_K5100_Set	Tag	Interface to device object
Ref_Ctrl_Cmd_Out	raC_UDT_Itf_K5100_Cmd	Tag	Interface to device object
Ref_Ctrl_Sts_Out	raC_UDT_Itf_K5100_Sts	Tag	Interface to device object
Sts_EN (Enable)	BOOL	Tag	This bit is set when the rung makes a false-to-true transition and remains set as the message transaction to activate the drive is initiated and in process. It remains set while the rung-in condition is true and no faults are active.
Sts_DN (Done)	BOOL	Tag	This bit is set when the rung makes a false-to-true transition and the cmd to activate the drive has been acknowledged.
Sts_ER (Error)	BOOL	Tag	This bit is set when the rung makes a false-to-true transition and there is an error that has occurred with the instruction. (This instruction error can be a result of a fault on the drive itself). See Sts_ERR for details on the cause of the error.
Sts_ERR	DINT	Tag	Instruction error codes. See Kinetix 5100 Drive UDFB Error Codes (Table 163) for details.

The raC_Opr_K5100_MSF instruction deactivates the motor. This instruction must be used when there are no active faults on the drive and the drive is in the Ready state. The resulting state of the drive is reflected when Ref_Ctrl_Sts_In.Active is zero.

> **IMPORTANT**   The instruction execution can take multiple scans to execute because it requires multiple RPI updates to complete the request. The Done (Sts_DN) bit is not set immediately, but only after the request is completed.

Error Codes:

- 100 - Kinetix 5100 drive is not ready
- 101 - Kinetix 5100 drive is faulted
- 104 - Another raC_Opr_K5100_MSF message is executing
- 129 - Motor not connected

raC_Opr_K5100_MAFR

Use the Motion Axis Fault Reset (raC_Opr_K5100_MAFR) instruction to clear some drive faults. When the fault is no longer active in the drive, this instruction clears the fault. This instruction does not clear any faults that are still active in the drive.

Figure 79 - MAFR Diagram
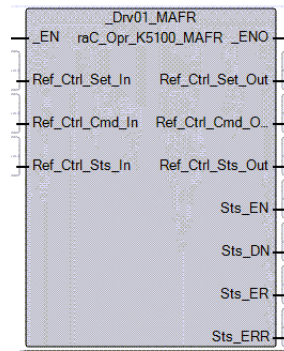
Table 153 - MAFR Operands

Operand	Type	Format	Description
_EN	BOOL	Tag	True when the rung is enabled.
Ref_Ctrl_Set_In	raC_UDT_Itf_K5100_Set	Tag	Interface from device object
Ref_Ctrl_Cmd_In	raC_UDT_Itf_K5100_Cmd	Tag	Interface from device object
Ref_Ctrl_Sts_In	raC_UDT_Itf_K5100_Sts	Tag	Interface from device object
_ENO	BOOL	Tag	True when this UDFB output is enabled.
Ref_Ctrl_Set_Out	raC_UDT_Itf_K5100_Set	Tag	Interface to device object
Ref_Ctrl_Cmd_Out	raC_UDT_Itf_K5100_Cmd	Tag	Interface to device object
Ref_Ctrl_Sts_Out	raC_UDT_Itf_K5100_Sts	Tag	Interface to device object
Sts_EN (Enable)	BOOL	Tag	This bit is set when the rung makes a false-to-true transition and remains set as the message transaction to activate the drive is initiated and in process. It remains set while the rung-in condition is true and no faults are active.
Sts_DN (Done)	BOOL	Tag	This bit is set when the rung makes a false-to-true transition and the cmd to activate the drive has been acknowledged.
Sts_ER (Error)	BOOL	Tag	This bit is set when the rung makes a false-to-true transition and there is an error that has occurred with the instruction. (This instruction error can be a result of a fault on the drive itself). See Sts_ERR for details on the cause of the error.
Sts_ERR	DINT	Tag	Instruction error codes. See Kinetix 5100 Drive UDFB Error Codes (Table 163) for details.

The raC_Opr_K5100_MAFR instruction attempts to clear any active fault on the specified axis. If the active fault condition is still present, the drive remains faulted.

Error Codes:

- 100 - Kinetix 5100 drive is not ready
- 106 - Another raC_Opr_K5100_MAFR message is executing

raC_Opr_K5100_MAS

Use the Motion Axis Stop (raC_Opr_K5100_MAS) instruction to stop motion on an axis. The drive remains active when the stop instruction is complete.

Figure 80 - MAS Diagram
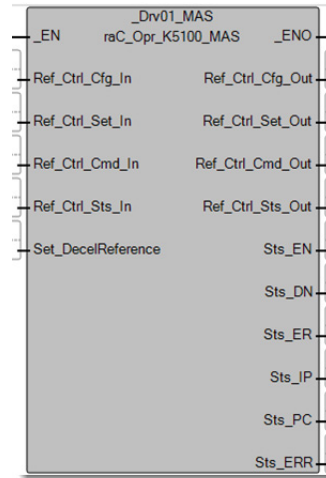
Table 154 - MAS Operands

Operand	Type	Format	Description
_EN	BOOL	Tag	True when the rung is enabled.
Ref_Ctrl_Cfg_In	raC_UDT_Itf_K5100_Cfg	Tag	Interface from device object
Ref_Ctrl_Set_In	raC_UDT_Itf_K5100_Set	Tag	Interface from device object
Ref_Ctrl_Cmd_In	raC_UDT_Itf_K5100_Cmd	Tag	Interface from device object
Ref_Ctrl_Sts_In	raC_UDT_Itf_K5100_Sts	Tag	Interface from device object
Set_DecelReference	LREAL	Immediate value or Tag	Deceleration reference for stopping the Axis.
_ENO	BOOL	Tag	True when this UDFB output is enabled.
Ref_Ctrl_Cfg_Out	raC_UDT_Itf_K5100_Cfg	Tag	Interface to device object
Ref_Ctrl_Set_Out	raC_UDT_Itf_K5100_Set	Tag	Interface to device object
Ref_Ctrl_Cmd_Out	raC_UDT_Itf_K5100_Cmd	Tag	Interface to device object
Ref_Ctrl_Sts_Out	raC_UDT_Itf_K5100_Sts	Tag	Interface to device object
Sts_EN (Enable)	BOOL	Tag	This bit is set when the rung makes a false-to-true transition and remains set as the message transaction to activate the drive is initiated and in process. This bit remains set while the rung-in condition is true and no faults are active.
Sts_DN (Done)	BOOL	Tag	This bit is set when the rung makes a false-to-true transition and the cmd to activate the drive has been acknowledged.
Sts_ER (Error)	BOOL	Tag	This bit is set when the rung makes a false-to-true transition and there is an error that has occurred with the instruction. (This instruction error can be a result of a fault on the drive itself). See Sts_ERR for details on the cause of the error.

Table 154 - MAS Operands (Continued)

Operand	Type	Format	Description
Sts_IP (In Progress)	BOOL	Tag	This bit is set when the rung makes a false-to-true transition, the Stop message transaction is successful, and the motor begins to decelerate. This bit remains set as the motor is executing the stop.
Sts_PC (Process Completed)	BOOL	Tag	This bit is set when the rung makes a false-to-true transition, the Sts_IP is set, and Zero Speed is reached. Zero Speed is defined using KNX5100C software > General Setting.
Sts_ERR	DINT	Tag	Instruction error codes. See Kinetix 5100 Drive UDFB Error Codes (Table 163) for details.

Use the raC_Opr_K5100_MAS instruction when you want a controlled stop for any controlled motion. The instruction stops the motion without disabling the motor. This UDFB stops any motion that is generated by motion UDFBs including the MAJ, MAM, or MAG instructions.

Error Codes:
- 100 - Kinetix 5100 drive is not ready
- 101 - Kinetix 5100 drive is faulted
- 103 - MSF is executing
- 105 - Drive is disabled
- 113 - DecelReference is out of range
- 129 - Motor is not connected
- 140 - Operation is not supported when the device is virtual
- 141 - Motor type not supported (Linear)

raC_Opr_K5100_MAJ

Use the Motion Axis Jog (raC_Opr_K5100_MAJ) instruction to move an axis at a constant speed until the command is ended.

Figure 81 - MAJ Diagram
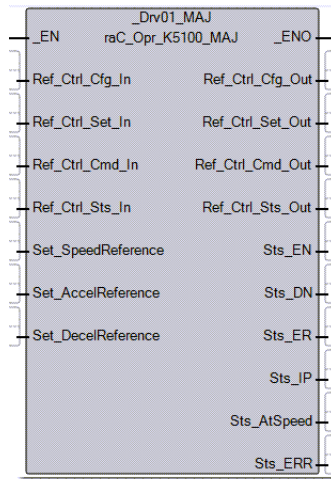
Table 155 - MAJ Operands

Operand	Type	Format	Description
_EN	BOOL	Tag	True when the rung is enabled.
Ref_Ctrl_Cfg_In	raC_UDT_Itf_K5100_Cfg	Tag	Interface from device object
Ref_Ctrl_Set_In	raC_UDT_Itf_K5100_Set	Tag	Interface from device object
Ref_Ctrl_Cmd_In	raC_UDT_Itf_K5100_Cmd	Tag	Interface from device object
Ref_Ctrl_Sts_In	raC_UDT_Itf_K5100_Sts	Tag	Interface from device object
Set_SpeedReference	LREAL	Immediate value or Tag	Units are 0.1 rpm for rotary motors. Range: -80,000...+80,000
Set_AccelReference	LREAL	Immediate value or Tag	Units are 0.1 rpm/s for rotary motors. Range: 458...30,000,000
Set_DecelReference	LREAL	Immediate value or Tag	Units are 0.1 rpm/s for rotary motors. Range: 458...30,000,000
_ENO	BOOL	Tag	True when this UDFB output is enabled.
Ref_Ctrl_Cfg_Out	raC_UDT_Itf_K5100_Cfg	Tag	Interface to device object
Ref_Ctrl_Set_Out	raC_UDT_Itf_K5100_Set	Tag	Interface to device object
Ref_Ctrl_Cmd_Out	raC_UDT_Itf_K5100_Cmd	Tag	Interface to device object
Ref_Ctrl_Sts_Out	raC_UDT_Itf_K5100_Sts	Tag	Interface to device object
Sts_EN (Enable)	BOOL	Tag	This bit is set when the rung makes a false-to-true transition and remains set as the message transaction to activate the drive is initiated and in process. It remains set while the rung-in condition is true and no faults are active.
Sts_DN (Done)	BOOL	Tag	This bit is set when the rung makes a false-to-true transition and the cmd to activate the drive has been acknowledged.
Sts_ER (Error)	BOOL	Tag	This bit is set when the rung makes a false-to-true transition and there is an error that has occurred with the instruction. (This instruction error can be a result of a fault on the drive itself). See Sts_ERR for details on the cause of the error.

Table 155 - MAJ Operands (Continued)

Operand	Type	Format	Description
Sts_IP (In Progress)	BOOL	Tag	This bit is set when the rung makes a false-to-true transition, the Stop message transaction is successful, and the motor begins to decelerate. This bit remains set as the motor is executing the stop.
Sts_AtSpeed	BOOL	Tag	This bit is set when the rung makes a false-to-true transition, the Sts_IP is set, and the Target Speed is reached. This bit remains set while the Jog is active and the AtSpeed condition is true.
Sts_ERR	DINT	Tag	Instruction error codes. See Kinetix 5100 Drive UDFB Error Codes (Table 163) for details.

Use the MAJ instruction to move an axis at a constant speed until the command is ended.

Error Codes:

- 100 - Kinetix 5100 drive is not ready
- 101 - Kinetix 5100 drive is faulted
- 103 - raC_Opr_K5100_MSF is running
- 105 - Drive is disabled
- 107 - raC_Opr_K5100_MAS is executing
- 108 - Other motion UDFB is sending the command
- 111 - SpeedReference is out of range
- 112 - AccelReference is out of range
- 113 - DecelReference is out of range
- 129 - Motor is not connected
- 140 - Operation is not supported when the device is virtual
- 141 - Motor type not supported (Linear)

raC_Opr_K5100_MAM

Use the Motion Axis Move (raC_Opr_K5100_MAM) instruction to move (index) an axis to a specified position.

Figure 82 - MAM Diagram
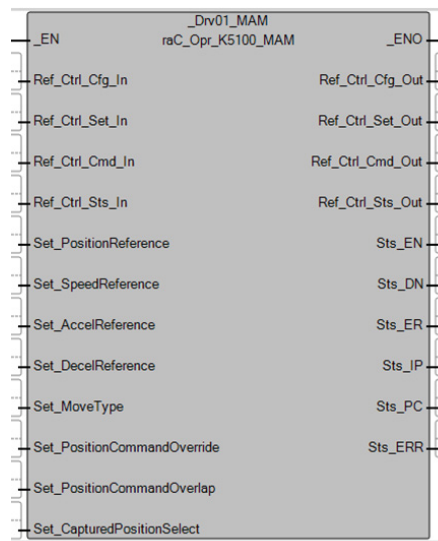
Table 156 - MAM Operands

Operand	Type	Format	Description
_EN	BOOL	Tag	True when the rung is enabled.
Ref_Ctrl_Cfg_In	raC_UDT_Itf_K5100_Cfg	Tag	Interface from device object
Ref_Ctrl_Set_In	raC_UDT_Itf_K5100_Set	Tag	Interface from device object
Ref_Ctrl_Cmd_In	raC_UDT_Itf_K5100_Cmd	Tag	Interface from device object
Ref_Ctrl_Sts_In	raC_UDT_Itf_K5100_Sts	Tag	Interface from device object
Set_PositionReference	LREAL	Immediate value or Tag	Set the Target Distance/Position Reference (PUU) Range: −2,147,483,648...+2,147,483,647
Set_SpeedReference	LREAL	Immediate value or Tag	Units are 0.1 rpm for rotary motors. Range: −80,000...+80,000
Set_AccelReference	LREAL	Immediate value or Tag	Units are 0.1 rpm/s for rotary motors. Range: 458...30,000,000
Set_DecelReference	LREAL	Immediate value or Tag	Units are 0.1 rpm/s for rotary motors. Range: 458...30,000,000
Set_MoveType	INT	Immediate value or Tag	Specifies the type of move: 0 = Absolute 1 = Incremental 2 = Rotary Shortest Path 3 = Rotary Positive 4 = Rotary Negative 7 = Relative 8 = Capture
Set_PositionCommandOverride	BOOL	Immediate value or Tag	0 = Do not interrupt previous movement 1 = Interrupt previous movement
Set_PositionCommandOverlap	BOOL	Immediate value or Tag	0 = Current movement is not overlapped with next 1 = Current movement is overlapped with next movement
Set_CapturedPositionSelect	BOOL	Immediate value or Tag	0 = First High-speed capture (triggered by DI9) 1 = Second High-speed capture (triggered by DI10)
_ENO	BOOL	Tag	True when this UDFB output is enabled.
Ref_Ctrl_Cfg_Out	raC_UDT_Itf_K5100_Cfg	Tag	Interface to device object
Ref_Ctrl_Set_Out	raC_UDT_Itf_K5100_Set	Tag	Interface to device object
Ref_Ctrl_Cmd_Out	raC_UDT_Itf_K5100_Cmd	Tag	Interface to device object
Ref_Ctrl_Sts_Out	raC_UDT_Itf_K5100_Sts	Tag	Interface to device object
Sts_EN (Enable)	BOOL	Tag	This bit is set when the rung makes a false-to-true transition and remains set as the message transaction to activate the drive is initiated and in process. It remains set while the rung-in condition is true and no faults are active.
Sts_DN (Done)	BOOL	Tag	This bit is set when the rung makes a false-to-true transition and the cmd to activate the drive has been acknowledged.
Sts_ER (Error)	BOOL	Tag	This bit is set when the rung makes a false-to-true transition and there is an error that has occurred with the instruction. This instruction error can be a result of a fault on the drive itself. See Sts_ERR for details on the cause of the error.

Table 156 - MAM Operands (Continued)

Operand	Type	Format	Description
Sts_IP (In Progress)	BOOL	Tag	This bit is set when the rung makes a false-to-true transition, the Stop message transaction is successful, and the motor begins to decelerate. This bit remains set as the motor is executing the stop.
Sts_PC (Process Completed)	BOOL	Tag	This bit is set when the rung makes a false-to-true transition, the Sts_IP is set, and the motor reaches the Target Position.
Sts_ERR	DINT	Tag	Instruction error codes. See Kinetix 5100 Drive UDFB Error Codes (Table 163) for details.

The raC_Opr_K5100_MAM instruction moves an axis to using a target position specified and uses the Move Type to perform the move (index).

Error Codes:

- 100 - Kinetix 5100 drive is not ready
- 101 - Kinetix 5100 drive is faulted
- 103 - MSF is executing
- 105 - Drive is disabled
- 107 - raC_Opr_K5100_MAS is executing
- 108 - Other motion UDFB is sending the command
- 111 - SpeedReference is out of range
- 112 - AccelReference is out of range
- 113 - DecelReference is out of range
- 119 - Move Type is out of range
- 126 - Homing not completed
- 129 - Motor is not connected to drive
- 140 - Operation is not supported when the device is virtual
- 141 - Motor type not supported (Linear)
- 150 - PositionReference in counts exceeds maximum allowed value (2147481984)

Set_MoveType:

- Four types of move operations (Set_MoveType = 0,1,7,8) are performed as shown.
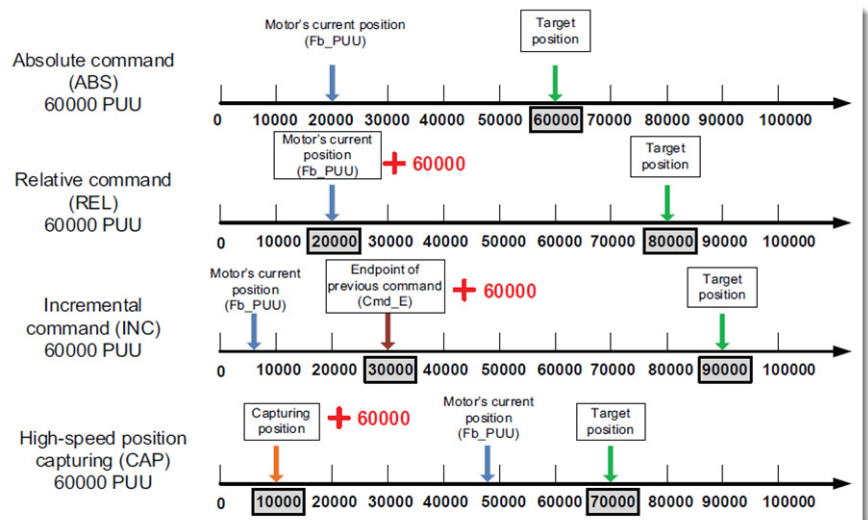
- Three types of move operations (Set_MoveMethod = 2,3,4) are performed as shown. Define 'Indexing Coordinate':
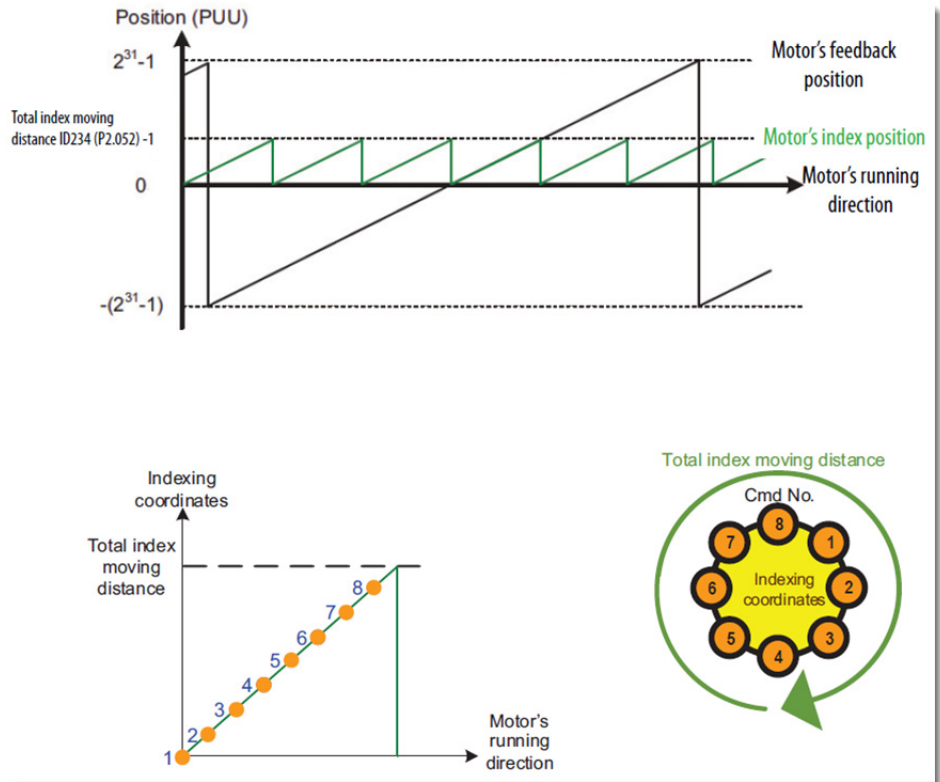
- Rotate Positive or Rotate Negative or Rotate Shortest Path.

  The Rotary move types are used to provide a way to index while observing the natural rollover of the feedback device. For example, if the motor could only index positive, the Rotary Positive is used. When the feedback device transitions through its natural unwind (typically 2.1 billion counts), the movements always indexes positive.

IMPORTANT	Currently, the Kinetix 5100 drive does not have a user defined Unwind function. The rotary selections in this UDFB do not refer to user-defined rotary axis types.
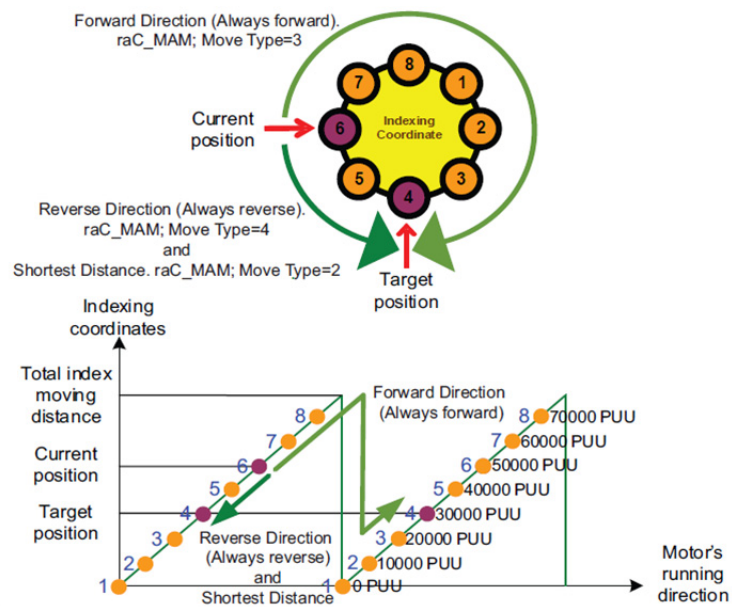
- Position Command with Overlap option.

  The executing index is interrupted during its deceleration. The new index is started before the deceleration is complete
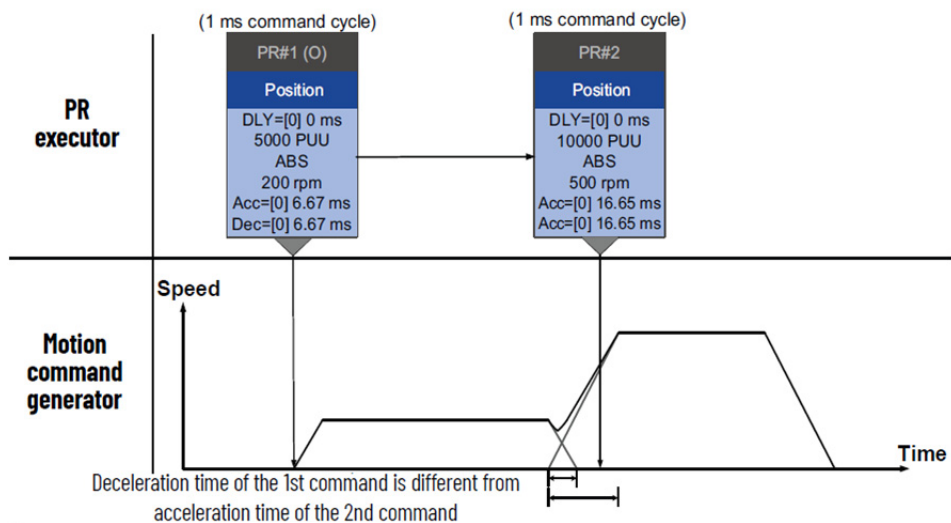
- Position Command with Interrupt option.

  The executing index (Index 1) is stopped. The new index (Index 2) is executed using its dynamics. This is shown in the graphic below. The red arrow is the point where the drive receives the command for Index 2.
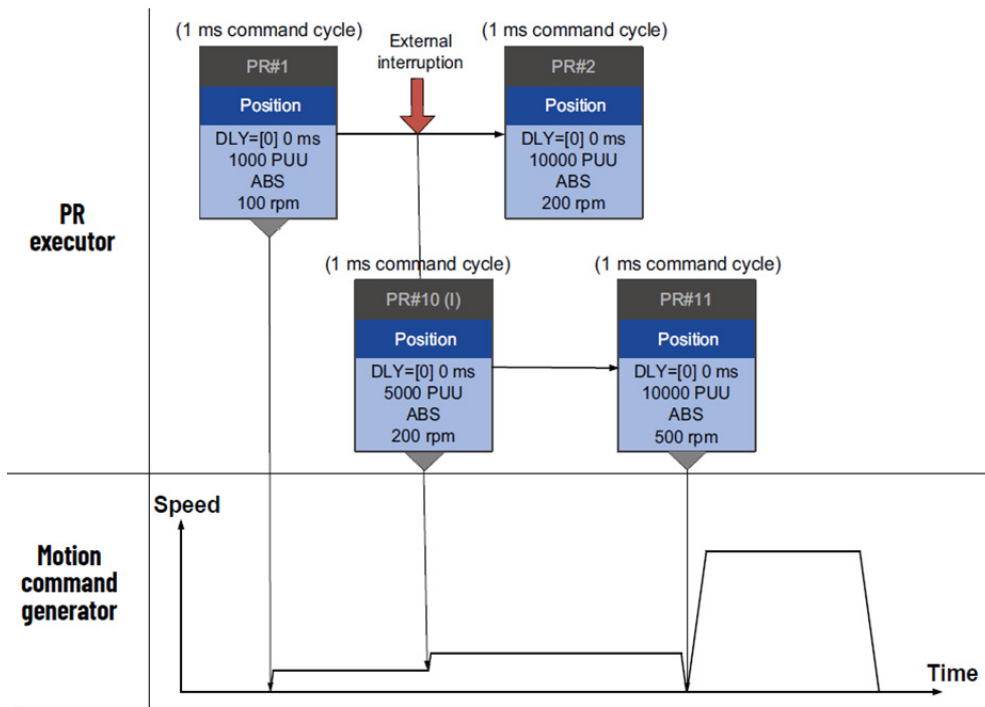
raC_Opr_K5100_MAI

It may be useful to execute a motion control internal register (PR) while in the IO operation mode. This can be for performing an action that is not able to be performed by one of the UDFBs. To use this, you must pre-configure the PR in the drive using KNX5100C software, before you establish the IO Mode connection.

Figure 83 - MAI Diagram
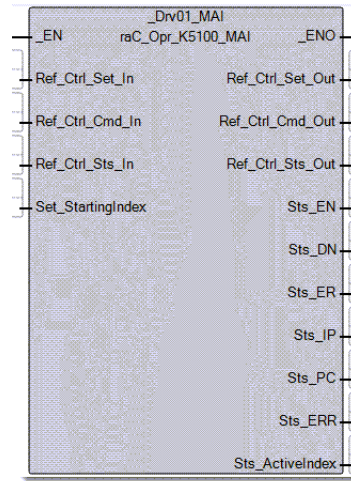
Table 157 - MAI Operands

Operand	Type	Format	Description
_EN	BOOL	Tag	True when the rung is enabled.
Ref_Ctrl_Set_In	raC_UDT_Itf_K5100_Set	Tag	Interface from device object
Ref_Ctrl_Cmd_In	raC_UDT_Itf_K5100_Cmd	Tag	Interface from device object
Ref_Ctrl_Sts_In	raC_UDT_Itf_K5100_Sts	Tag	Interface from device object
Set_StartingIndex	INT	Immediate value or Tag	Enter the pre-configured PR# to execute.
_ENO	BOOL	Tag	True when this UDFB output is enabled.
Ref_Ctrl_Set_Out	raC_UDT_Itf_K5100_Set	Tag	Interface to device object
Ref_Ctrl_Cmd_Out	raC_UDT_Itf_K5100_Cmd	Tag	Interface to device object
Ref_Ctrl_Sts_Out	raC_UDT_Itf_K5100_Sts	Tag	Interface to device object
Sts_ActiveIndex	INT	Tag	Reads the current PR# that is executing in the drive.
Sts_EN (Enable)	BOOL	Tag	This bit is set when the rung makes a false-to-true transition and the message transaction to MAI is initiated and in process. It remains high until the rung-in condition is false and no faults are active.
Sts_DN (Done)	BOOL	Tag	This bit is set when the rung makes a false-to-true transition and the message transaction to MAI the drive (Sts_EN) is complete.
Sts_ER (Error)	BOOL	Tag	This bit is set when the rung makes a false-to-true transition and there is an error that has occurred with the instruction. This instruction error can be a result of a fault on the drive itself. See Sts_ERR for details on the cause of the error.
Sts_IP (In Progress)	BOOL	Tag	This bit is set when the rung makes a false-to-true transition, the MAI message transaction is successful, and the PR command has been sent to the drive. This bit remains set until the AtReference bit is set.
Sts_PC (Process Completed)	BOOL	Tag	This bit is set when the rung makes a false-to-true transition, the Sts_IP is set, and the MAI has sent the PR execution and the AtReference bit is set.
Sts_ERR	DINT	Tag	Instruction error codes. See Kinetix 5100 Drive UDFB Error Codes (Table 163) for details.

Use the Motion Axis Index (raC_Opr_K5100_MAI) instruction to execute motion control by internal register (PR) in Kinetix 5100 drives. The 99 built-in command registers select the PR command source.

Error Codes:
- 100 - Kinetix 5100 drive is not ready
- 101 - Kinetix 5100 drive is faulted
- 103 - MSF is executing
- 105 - Drive is disabled
- 107 - raC_Opr_K5100_MAS is executing
- 108 - Other motion UDFB is sending the command
- 115 - StartingIndex is out of range
- 129 - Motor is not connected
- 140 - Operation is not supported when the device is virtual

raC_Opr_K5100_MAG

Use the Motion Axis Gear (MAG) to execute a pulse-pulse relationship with the drive. The MAG uses the E-gear ratio that is configured in the KNX5100C software. The E-gear Ratio dialog box is shown in Figure 75. When you use the MAG instruction, the drive behaves like it is in PT (Position Terminal - or Pulse Train) mode and the drive uses the E-gear ratio to respond to master pulses.

Figure 84 describes the values in the E-gear Ratio dialog box. Not all values that are shown here are used with the Motion Operation UDFB.

Figure 84 - E-gear Ratio Dialog Box
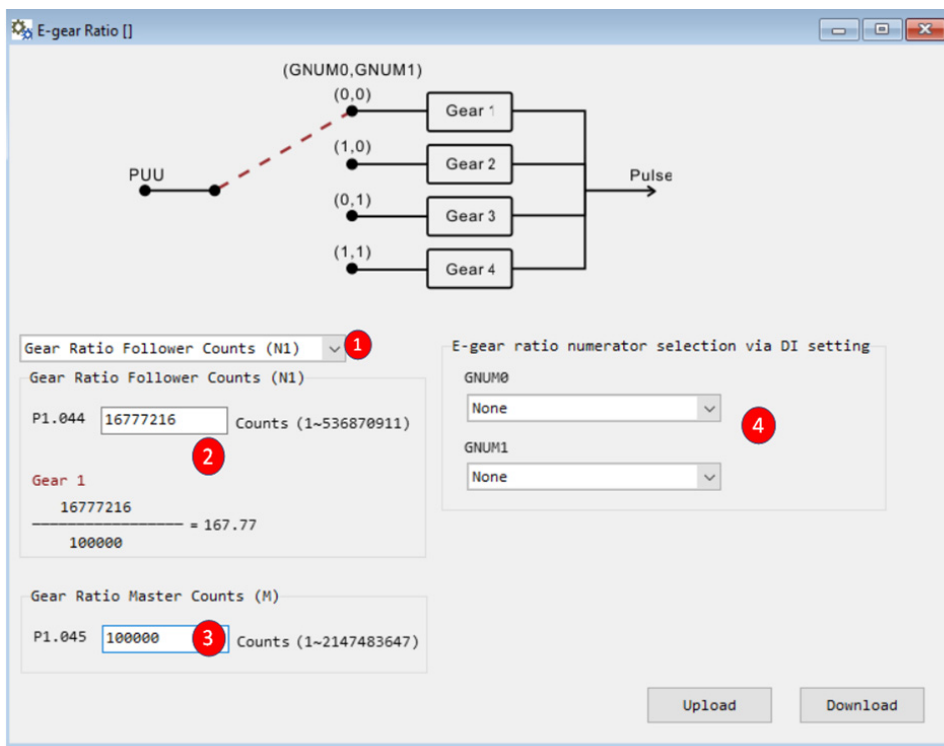
Table 158 - E-gear Ratio Dialog Box Settings

Item	Description
1	Gear Ratio Selection dropdown menu - You can choose from four different ratios (N1...N4). Not used with UDFB.

Table 158 - E-gear Ratio Dialog Box Settings (Continued)

Item	Description
2	Gear Ratio Follower Counts (N1) - Set this value as the motor feedback resolution.
3	Gear Ratio Master Counts (M) - Set this value as the counts/motor resolution. This value is set for whatever your application requires. Typical values are 100,000 counts for a high-resolution encoder.
4	GNUM0/1 - These values are mapped to the Digital Inputs that represent binary weighted values to select the Gear Ratio value. Not used with UDFB.

The PT Mode is a pulse-pulse follower relationship. When you use the variable ratio and change the ratio, there is no positioning ability. Therefore, when you are finished using the MAG UDFB, your position scaling (which also uses the E-gear ratio) might change if you use the variable Cfg_GearingMode and change the Master ratio because of your application requirements.

The MAG Set_SlaveCounts is sometimes called the Gear Ratio Follower Counts or numerator (shown as item 2 in Figure 84). Use it to determine the internal 'ratio' of the drive (shown as 167.77 in Figure 84).

For our purposes, the E-gear ratio Follower = MAG Set_SlaveCounts = motor feedback resolution (from the KNX5100C > Function List > Motor Selection > Feedback dialog box).

Figure 85 - Set the Follower Counts
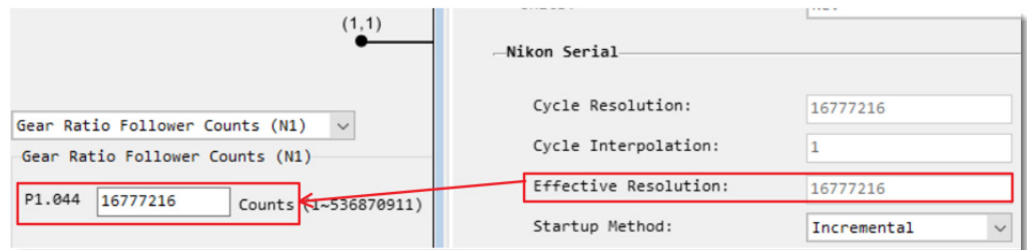
The MAG Set_MasterCounts is sometimes called the Gear ratio Master counts or denominator (shown as item 3 in Figure 84). Any gearing relationship must consider the actual motor mechanics, like a gearbox, or actuator pitch, and use those mechanics to relate back to a motor rotation. Gear Ratio Master counts are desired counts/motor rotation. Desired counts are not used for positioning; but define how many counts your motor moves in one rotation based on the number of feedback pulses you expect to receive from the source input, which is used to determine your gearing relationship. So, this Master counts value is used to define the pulse-pulse relationship.

IMPORTANT	The MAG can affect your positioning. The issuing Kinetix 5100 drive (slave) uses the E-gear ratio to define how it follows pulses from a source (a master). While the result is that the issuing Kinetix 5100 drive (slave) follows pulses from another source master, the way the function operates can affect positioning of the drive. Regardless of Operation Mode, the E-gear ratio is always used to provide a representation of positioning (units or counts) or to define a Pulse-Pulse Following relationship (MAG/PT). When the E-gear ratio is changed, the positioning of the axis is changed.

Gearing example

The master in the example system is a 4000 ppr encoder. When the encoder makes one revolution, we expect the Slave1 drive to see: 4000 pulses.

Our application requirement is that we want to follow this encoder at a 1:2 relationship. So, when the master encoder moves one encoder revolution, the motor rotates two times.

The Master PPR is not entered anywhere, but is required that we know this value. We calculate the MAG Set_MasterCounts value knowing the Master PPR counts and the relationship we want in the Slave1 motor.

We set the MAG Set_SlaveCounts = Motor Feedback Resolution = 16,777,216.

We set MAG Set_MasterCounts = 2000, so when the Slave1 drive sees 2000 master pulses, the Slave1 motor moves one rotation, and thus, as the Master encoder moves 4000 pulses, Slave1 would move two rotations.

You can use two modes of the MAG function. These modes are defined by the Cfg_GearingMode entry. This entry is set for Fixed initially. You must intentionally change this setting. Fixed mode does not impact positioning because it uses the existing E-gear ratio in the Kinetix 5100 drive. Therefore, we can follow a master source at this fixed ratio and when gearing is disabled, we can continue positioning without losing the position scaling for the drive.

Variable mode lets you change the E-gear ratio by manipulating the master/slave counts values. The variable mode lets you change the ratio. When you change the ratio, you affect the motor positioning because the E-gear ratio is also used to define Position Scaling. If you require positioning after using the variable gearing, issue a Homing Sequence to re-establish an origin.

Figure 86 - MAG Diagram
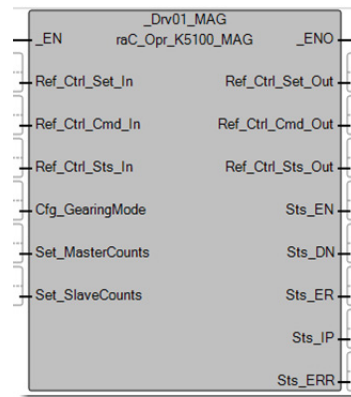
Table 159 - MAG Operands

Operand	Type	Format	Description
_EN	BOOL	Tag	True when the rung is enabled.
Ref_Ctrl_Set_In	raC_UDT_Itf_K5100_Set	Tag	Interface from device object
Ref_Ctrl_Cmd_In	raC_UDT_Itf_K5100_Cmd	Tag	Interface from device object
Ref_Ctrl_Sts_In	raC_UDT_Itf_K5100_Sts	Tag	Interface from device object
Cfg_GearingMode	BOOL	Immediate value or Tag	0 = Fixed 1 = Variable
Set_MasterCounts	DINT	Immediate value or Tag	Sets the value of the E-gear ratio: Denominator ID152 (P1.045). Set this value to represent the desired counts/motor rotation. This value defines the number of pulses/motor rotation and when used with the feedback pulses that you expect to see from the source input (also pulses/ revolution) provides a gearing relationship.
Set_SlaveCounts	DINT	Immediate value or Tag	Sets the value of the E-gear ratio: Numerator ID151 (P1.044). Set this value the same as the Motor Feedback Resolution.
_ENO	BOOL	Tag	True when this UDFB output is enabled.
Ref_Ctrl_Set_Out	raC_UDT_Itf_K5100_Set	Tag	Interface to device object
Ref_Ctrl_Cmd_Out	raC_UDT_Itf_K5100_Cmd	Tag	Interface to device object
Ref_Ctrl_Sts_Out	raC_UDT_Itf_K5100_Sts	Tag	Interface to device object
Sts_EN (Enable)	BOOL	Tag	This bit is set when the rung makes a false-to-true transition and the message transaction to MAG is initiated and in process. It remains high until the rung-in condition is false and no faults are active.

Table 159 - MAG Operands (Continued)

Operand	Type	Format	Description
Sts_DN (Done)	BOOL	Tag	This bit is set when the rung makes a false-to-true transition and the message transaction to MAG (Sts_EN) is complete.
Sts_ER (Error)	BOOL	Tag	This bit is set when the rung makes a false-to-true transition and there is an error that has occurred with the instruction. This instruction error can be a result of a fault on the drive itself. See Sts_ERR for details on the cause of the error.
Sts_IP (In Progress)	BOOL	Tag	This bit is set when the rung makes a false-to-true transition, the MAG message transaction is successful, and the drive begins following. This bit remains set as the motor is executing the gearing. It remains set while the MAG is active, regardless of the rung-in condition.
Sts_ERR	DINT	Tag	Instruction error codes. See Kinetix 5100 Drive UDFB Error Codes (Table 163) for details.

Error Codes:

- 100 - Kinetix 5100 drive is not ready
- 101 - Kinetix 5100 drive is faulted
- 103 - MSF is executing
- 105 - Drive is disabled
- 107 - raC_Opr_K5100_MAS is executing
- 108 - Other motion UDFB is sending the command
- 129 - Motor is not connected
- 131 - Slave count is out of range 1 to (2^{29}...1)
- 132 - Master count is out of range 1 to (2^{31}...1)
- 140 - Operation is not supported when the device is virtual
- 141 - Motor type not supported (Linear)

raC_Opr_K5100_MAH

Use the Motion Axis Home (raC_Opr_K5100_MAH) instruction to command a homing procedure in the drive. Homing is used to define an origin for your motor and to establish an absolute positioning reference for the motor.

Figure 87 - MAH Diagram
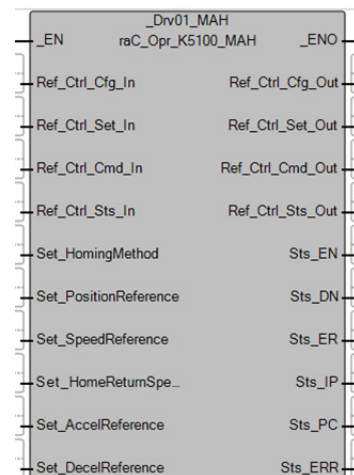
Table 160 – MAH Operands

Operand	Type	Format	Description
_EN	BOOL	Tag	True when the rung is enabled.
Ref_Ctrl_Cfg_In	raC_UDT_Itf_K5100_Cfg	Tag	Interface from device object
Ref_Ctrl_Set_In	raC_UDT_Itf_K5100_Set	Tag	Interface from device object
Ref_Ctrl_Cmd_In	raC_UDT_Itf_K5100_Cmd	Tag	Interface from device object
Ref_Ctrl_Sts_In	raC_UDT_Itf_K5100_Sts	Tag	Interface from device object
Set_HomingMethod	SINT	Immediate value or Tag	0…38. Sets this value for the Homing Method that is used with K5100 drive. See Table 161.
Set_PositionReference	LREAL	Immediate value or Tag	The feedback position when a homing procedure is completed. Range: -2,147,483,648…+2,147,483,647
Set_SpeedReference	LREAL	Immediate value or Tag	The first (high) speed reference. Units are 0.1 rpm for rotary motors. Range: 1…20,000
Set_HomeReturnSpeed	LREAL	Immediate value or Tag	The second (low) speed reference. Units are 0.1 rpm for rotary motors. Range: 1…5000
Set_AccelReference	LREAL	Immediate value or Tag	Units are 0.1 rpm/s for rotary motors. Range: 458…30,000,000
Set_DecelReference	LREAL	Immediate value or Tag	Units are 0.1 rpm/s for rotary motors. Range: 458…30,000,000
_ENO	BOOL	Tag	True when this UDFB output is enabled.
Ref_Ctrl_Cfg_Out	raC_UDT_Itf_K5100_Cfg	Tag	Interface to device object
Ref_Ctrl_Set_Out	raC_UDT_Itf_K5100_Set	Tag	Interface to device object
Ref_Ctrl_Cmd_Out	raC_UDT_Itf_K5100_Cmd	Tag	Interface to device object
Ref_Ctrl_Sts_Out	raC_UDT_Itf_K5100_Sts	Tag	Interface to device object
Sts_EN (Enable)	BOOL	Tag	This bit is set when the rung makes a false-to-true transition and the message transaction to Home is initiated and in process. It remains high until the rung-in condition is false and no faults are active.
Sts_DN (Done)	BOOL	Tag	This bit is set when the rung makes a false-to-true transition and the message transaction to Home the drive (Sts_EN) is complete.
Sts_ER (Error)	BOOL	Tag	This bit is set when the rung makes a false-to-true transition and there is an error that has occurred with the instruction. This instruction error can be as a result of a fault on the drive itself. See Sts_ERR for details on the cause of the error.
Sts_IP (In Progress)	BOOL	Tag	This bit is set when the rung makes a false-to-true transition, the Home message transaction is successful, and the homing begins. This bit remains set if the homing is executing.
Sts_PC (Process Complete)	BOOL	Tag	This bit is set when the rung makes a false-to-true transition, and the Homing Sequence is completed.
Sts_ERR	DINT	Tag	Instruction error codes. See Kinetix 5100 Drive UDFB Error Codes (Table 163) for details.

Table 161 – Homing Methods

Value	Description
0	Homing in forward direction and regard PL as homing origin. Return to Z pulse.
1	Homing in forward direction and regard PL as homing origin. Go forward to Z pulse.
2	Homing in forward direction and regard PL as homing origin. Do not look for Z pulse.
3	Homing in reverse direction and regard NL as homing origin. Return to Z pulse.

Table 161 - Homing Methods (Continued)

Value	Description
4	Homing in reverse direction and regard NL as homing origin. Go forward to Z pulse.
5	Homing in reverse direction and regard NL as homing origin. Do not look for Z pulse.
6	Homing in forward direction, ORG: OFF->ON as homing origin. Return to Z pulse. Shows error when encounter limit.
7	Homing in forward direction, ORG: OFF->ON as homing origin. Return to Z pulse. Reverse direction when encounter limit.
8	Homing in forward direction, ORG: OFF->ON as homing origin. Go forward to Z pulse. Shows error when encounter limit.
9	Homing in forward direction, ORG: OFF->ON as homing origin. Go forward to Z pulse. Reverse direction when encounter limit.
10	Homing in forward direction, ORG: OFF->ON as homing origin. Do not look for Z pulse. Shows error when encounter limit.
11	Homing in forward direction, ORG: OFF->ON as homing origin. Do not look for Z pulse. Reverse direction when encounter limit.
12	Homing in reverse direction, ORG: OFF->ON as homing origin. Return to Z pulse. Shows error when encounter limit.
13	Homing in reverse direction, ORG: OFF->ON as homing origin. Return to Z pulse. Reverse direction when encounter limit.
14	Homing in reverse direction, ORG: OFF->ON as homing origin. Go forward to Z pulse. Shows error when encounter limit.
15	Homing in reverse direction, ORG: OFF->ON as homing origin. Go forward to Z pulse. Reverse direction when encounter limit.
16	Homing in reverse direction, ORG: OFF->ON as homing origin. Do not look for Z pulse. Shows error when encounter limit.
17	Homing in reverse direction, ORG: OFF->ON as homing origin. Do not look for Z pulse. Reverse direction when encounter limit.
18	Look for Z pulse in forward direction and regard it as homing origin. Shows error when encounter limit.
19	Look for Z pulse in forward direction and regard it as homing origin. Reverse direction when encounter limit.
20	Look for Z pulse in reverse direction and regard it as homing origin. Shows error when encounter limit.
21	Look for Z pulse in reverse direction and regard it as homing origin. Reverse direction when encounter limit.
22	Homing in forward direction, ORG: ON->OFF as homing origin. Return to Z pulse. Shows error when encounter limit.
23	Homing in forward direction, ORG: ON->OFF as homing origin. Return to Z pulse. Reverse direction when encounter limit.
24	Homing in forward direction, ORG: ON->OFF as homing origin. Go forward to Z pulse. Shows error when encounter limit.
25	Homing in forward direction, ORG: ON->OFF as homing origin. Go forward to Z pulse. Reverse direction when encounter limit.
26	Homing in forward direction, ORG: ON->OFF as homing origin. Do not look for Z pulse. Shows error when encounter limit.
27	Homing in forward direction, ORG: ON->OFF as homing origin. Do not look for Z pulse. Reverse direction when encounter limit.
28	Homing in reverse direction, ORG: ON->OFF as homing origin. Return to Z pulse. Shows error when encounter limit.
29	Homing in reverse direction, ORG: ON->OFF as homing origin. Return to Z pulse. Reverse direction when encounter limit.
30	Homing in reverse direction, ORG: ON->OFF as homing origin. Go forward to Z pulse. Shows error when encounter limit.
31	Homing in reverse direction, ORG: ON->OFF as homing origin. Go forward to Z pulse. Reverse direction when encounter limit.
32	Homing in reverse direction, ORG: ON->OFF as homing origin. Do not look for Z pulse. Shows error when encounter limit.
33	Homing in reverse direction, ORG: ON->OFF as homing origin. Do not look for Z pulse. Reverse direction when encounter limit.
34	Define current position as the origin.
35	Look for the collision point in forward direction and regard it as the origin. Return to Z pulse. Shows error when encounter negative limit.

Table 161 - Homing Methods (Continued)

Value	Description
36	Look for the collision point in forward direction and regard it as the origin. Do not look for Z pulse.
37	Look for the collision point in reverse direction and regard it as the origin. Return to Z pulse. Shows error when encounter positive limit.
38	Look for the collision point in reverse direction and regard it as the origin. Do not look for Z pulse.

Error Codes:

- 100 - Kinetix 5100 drive is not ready
- 101 - Kinetix 5100 drive is faulted
- 103 - raC_Opr_K5100_MSF is running
- 105 - Drive is disabled
- 107 - raC_Opr_K5100_MAS is executing
- 108 - Other motion UDFB is sending the command
- 111 - SpeedReference is out of range
- 112 - AccelReference is out of range
- 113 - DecelReference is out of range
- 122 - HomingMethod is out of range
- 129 - Motor is not connected
- 140 - Operation is not supported when the device is virtual
- 141 - Motor type not supported (Linear)
- 150 - PositionReference in counts exceeds maximum allowed value (2147481984)

raC_Opr_K5100_MAT

The Motion Axis Torque (raC_Opr_K5100_MAT) instruction lets you use torque limiting while a pre-defined speed is used to move the motor. The first time the pre-defined torque limit is reached, the Sts_TorqueReached bit is set. While the Sts_TorqueReached bit is set, the MAT operation remains active until it is ended by an raC_Opr_K5100_MAS (Motion Axis Stop)/MSF (Motion Servo Off), or a drive fault. The torque and speed entries are bidirectional.

Figure 88 - MAT Diagram
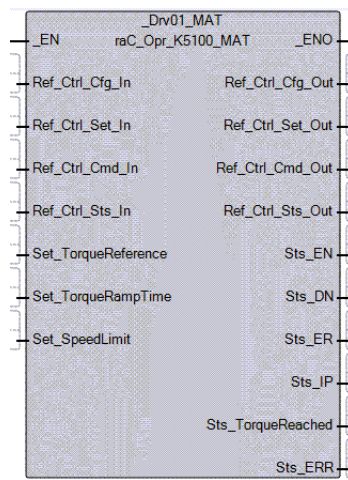
Table 162 - MAT Operands

Operand	Type	Format	Description
_EN	BOOL	Tag	True when the rung is enabled.
Ref_Ctrl_Cfg_In	raC_UDT_Itf_K5100_Cfg	Tag	Interface from device object
Ref_Ctrl_Set_In	raC_UDT_Itf_K5100_Set	Tag	Interface from device object

Table 162 - MAT Operands (Continued)

Operand	Type	Format	Description
Ref_Ctrl_Cmd_In	raC_UDT_Itf_K5100_Cmd	Tag	Interface from device object
Ref_Ctrl_Sts_In	raC_UDT_Itf_K5100_Sts	Tag	Interface from device object
Set_TorqueReference	DINT	Immediate value or Tag	The limited value of motor torque, in the unit of 0.1% of the motor rated torque. Range: -4000...+4000
Set_TorqueRampTime	DINT	Immediate value or Tag	Torque Ramp Time, the time (ms) it takes to ramp up from 0 to the TorqueReference. Range: 1...65500
Set_SpeedLimit	DINT	Immediate value or Tag	Speed limit that is used during the constant torque operation: unit is 0.1 rpm for the rotary motor. Range: -80,000...+80,000
_ENO	BOOL	Tag	True when this UDFB output is enabled.
Ref_Ctrl_Cfg_Out	raC_UDT_Itf_K5100_Cfg	Tag	Interface to device object
Ref_Ctrl_Set_Out	raC_UDT_Itf_K5100_Set	Tag	Interface to device object
Ref_Ctrl_Cmd_Out	raC_UDT_Itf_K5100_Cmd	Tag	Interface to device object
Ref_Ctrl_Sts_Out	raC_UDT_Itf_K5100_Sts	Tag	Interface to device object
Sts_EN (Enable)	BOOL	Tag	This bit is set when the rung makes a false-to-true transition and remains set as the message transaction to execute the MAT is initiated and in process. It remains high until the rung-in condition is false and no faults are active.
Sts_DN (Done)	BOOL	Tag	This bit is set when the rung makes a false-to-true transition and the message transaction to the drive (Sts_EN) is complete.
Sts_ER (Error)	BOOL	Tag	This bit is set when the rung makes a false-to-true transition and there is an error that has occurred with the instruction. (This instruction error can be as a result of a fault on the drive itself). See Sts_ERR for details on the cause of the error.
Sts_IP (In Progress)	BOOL	Tag	This bit is set when the rung makes a false-to-true transition, the MAT message transaction is successful, and the motor begins to move. This bit remains set while the MAT operation is active.
Sts_TorqueReached	BOOL	Tag	This bit is set when the rung makes a false-to-true transition, the Sts_IP is set, and the Set_TorqueReference value is reached. This bit is set (and remains set) on the first occurrence of this condition.
Sts_ERR	DINT	Tag	Instruction error codes. See Kinetix 5100 Drive UDFB Error Codes (Table 163) for details.

Error Codes:

- 100 - Kinetix 5100 drive is not ready
- 101 - Kinetix 5100 drive is faulted
- 103 - raC_Opr_K5100_MSF is executing
- 105 - Drive is disabled
- 107 - raC_Opr_K5100_MAS is executing
- 108 - Other motion UDFB is sending the command
- 111 - Speed limit is > 80000 or < -80000
- 116 - TorqueReference is out of range
- 125 - TorqueRampTime is out of range
- 129 - Motor is not connected
- 140 - Operation is not supported when the device is virtual
- 141 - Motor type not supported (Linear)

Error Codes

Motion Error Codes (Sts_ERR) describes the error for UDFBs. Table 163 lists the error codes for the Connected Components Workbench application motion instructions for the Kinetix 5100 drive.

Table 163 - Kinetix 5100 Drive UDFB Error Codes

Error Code	Description	MSO	MSF	MAFR	MAS	MAJ	MAT	MAI	MAM	MAH	MAG
100	Drive is not ready.	X	X	X	X	X	X	X	X	X	X
101	Drive is faulted.	X	X		X	X	X	X	X	X	X
102	raC_Opr_K5100_MSO is executing.	X									
103	raC_Opr_K5100_MSF is executing.	X			X	X	X	X	X	X	X
104	Another raC_Opr_K5100_MSF is executing.		X								
105	Drive is disabled.				X	X	X	X	X	X	X
106	Another raC_Opr_K5100_MAFR message is executing.			X							
107	raC_Opr_K5100_MAS is executing.					X	X	X	X	X	X
108	Another RA motion UDFB is sending the command.					X	X	X	X	X	X
111	SpeedReference is out of range.					X	X		X	X	
112	AccelReference is out of range.					X			X	X	
113	DecelReference is out of range.				X	X			X	X	
115	StartingIndex is higher than 99.							X			
116	Torque is out of range.						X				
119	MoveType is out of range.								X		
122	HomingMethod is out of range.									X	
125	TorqueRampTime is out of range.						X				
126	Homing is not completed.								X		
129	Motor is not connected.	X	X		X	X	X	X	X	X	X
131	Gear slave counts are out of range.										X
132	Gear master count is out of range.										X
133	Gear ratio is out of range.										X
140	Operation is not supported when the device is virtual.				X	X	X	X	X	X	X
141	Motor type not supported (Linear).				X	X	X		X	X	X
150	PositionReference in counts exceeds maximum allowed value (2147481984)								X	X	

Use PCCC Commands and MicroLogix Mapping

Use PCCC Commands

Firmware revision 22.011 for Micro 870 (2080-L70E) controllers adds support for Programmable Controller Communication Command (PCCC) and enables the controller to respond over either the DF1 serial port or the Ethernet port.

This option provides backward compatibility for legacy controllers that do not understand the CIP protocol to access the data table mappings inside the Micro870 controller. CIP is the fundamental language of a Micro870 controller.

With proper mapping of Micro870 variables to data files, you can access information in the Micro870 controller using other applications that worked with our legacy controllers over the serial or Ethernet port.

IMPORTANT	For information on mapping of variables to MicroLogix data files, see Map Variables to MicroLogix Files on page 384. To avoid data mismatch, use an array variable of the same data type as the MicroLogix file.

CIP messaging is the preferred method of interacting with Micro870 controllers, but PCCC messaging is serviceable for many applications, especially where the legacy communications product is not able to be modified, and where you are willing to do the extra configuration of data table mappings in the Micro870 controller. Remote applications that depend on DF1 serial communication over a modem or serial radio link can also use this method.

PCCC commands can arrive at the controller in these ways:
- Through the RS-232 serial port
- Encapsulated inside a EtherNet/IP message

The following section identifies the PCCC commands that Micro870 (2080-L70E) controllers support and the formatting that is required. You do not require a license from ODVA to use the PCCC commands that are described as follows.

Supported Subset of PCCC Commands

Micro870 (2080-L70E) controllers support the SLC™ communication commands format to access MicroLogix data files, or to respond automatically to a MicroLogix controller.

PCCC commands that Micro870 (2080-L70E) controllers support in fundamental PCCC instruction:
- SLC Protected Typed Logical Read with 3 Address Fields (CMD=0F,4F; FNC=A2)
- SLC Protected Typed Logical Write with 3 Address Fields (CMD=0F,4F; FNC=AA)

PCCC commands that Micro870 (2080-L70E) controllers understand to auto respond to queries from MicroLogix controllers:
- SLC Protected Typed Logical Read with 3 Address Fields (CMD=0F,4F; FNC=A2)
- SLC Protected Typed Logical Write with 3 Address Fields (CMD=0F,4F; FNC=AA)
- SLC Protected Typed Logical Read with 2 Address Fields (CMD=0F,4F; FNC=A1)
- SLC Protected Typed Logical Write with 2 Address Fields (CMD=0F,4F; FNC=A9)

• SLC Protected Typed Logical Write with 4 Address Fields (CMD=OF,4F; FNC=AB)

These commands use strictly a logical form of addressing (for example, file/element/sub-element). For information on mapping the MicroLogix files to Micro870 variables, see How to Map a MicroLogix Address in Connected Components Workbench Software on page 385.

Micro870 controllers handle protected and unprotected commands the same way, whether the access for the data is set to read/write, read-only, or none.

IMPORTANT	For the SLC Protected Typed Logical Read and SLC Protected Typed Logical Write commands, map the data files to Micro870 variables of types INT, DINT, REAL, or STRING only. Use a MicroLogix file data type that matches the data type of the Micro870 variable. • For INT variables, use file type 85hex (Binary) or 89hex (Integer). • For DINT variables, use file type 91hex (Long). • For REAL variables, use file type 8Ahex (Float). • For STRING variable, use file type 8Dhex (String). For more information on MicroLogix file types, see the DF1 Protocol and Command Set Reference Manual, publication 1770-6.5.16.

MicroLogix logical addressing has a limited number of logical address levels, so there are some special considerations.

In a	The element number is used as the
One-dimensional array	Index of the dimension (data[elem])
Two-dimensional array	Index of the second dimension and the first dimension index is 0 (data[0][elem])
Three-dimensional array	Index of the third dimension and the first and second dimension indices are both 0 (data[0][0][elem])

Micro870 (L70E) controllers only support the one-dimensional array.

For more information on how to use the new PCCC instruction to send PCCC commands, see the Micro800 Programmable Controllers General Instructions Reference Manual, publication 2080-RM001.

Map Variables to MicroLogix Files

This feature is only supported in Micro870 (2080-L70E) controllers with Connected Components Workbench software, version 22 or later.

A Micro870 (2080-L70E) controller stores variable names in the controller so that other devices can read or write data without having to know the physical memory locations. Many products can only understand MicroLogix data files, so the Micro870 controller offers a function to map Micro870 variable names to MicroLogix files.

• You only have to map the file numbers that are used in messages; you do not need to map the other file numbers.

• The mapping table is loaded into the controller and is used whenever a logical address accesses data.

• You can only access controller variables (global data).

Follow these guidelines when you map variables.

• Do not use file numbers 0, 1, and 2. These files are reserved for Output, Input, and Status files respectively in a MicroLogix processor.

• Use MicroLogix mapping only for variable arrays of data type INT, DINT, STRING, or REAL. Attempting to map elements of system structures can produce undesirable effects.

• Use these file types and identifiers.

For this Micro870 Array Type	Use this MicroLogix File Identifier
INT array	N or B

For this Micro870 Array Type	Use this MicroLogix File Identifier
DINT array	L
REAL array	F
STRING array	ST

How to Map a MicroLogix Address in Connected Components Workbench Software
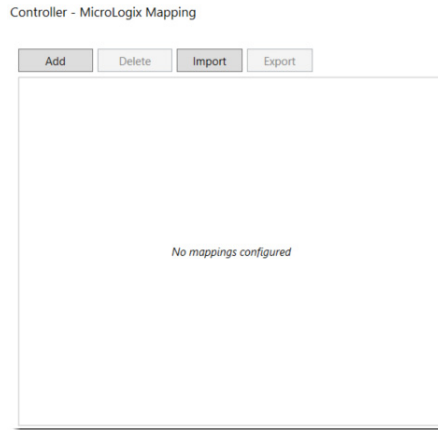
You can add the MicroLogix address in two ways:
- Use the ADD button on the MicroLogix mapping configuration page.
- Use the IMPORT button on the MicroLogix mapping configuration page.

Use the ADD button

To add a MicroLogix mapping, do the following:
1. In the Project or Controller Organizer, double-click the controller to open the controller workspace.
2. In the Controller tree, select MicroLogix Mapping.
3. On the Controller - MicroLogix Mapping configuration page, select Add.
   a. In the Variable Selector dialog, locate and select the variable, then select OK.
   b. (Optional) Change the File Number to define the MicroLogix File Type to map to the Micro800 variable. This step is only required for file types N and B.

IMPORTANT	To read multiple MicroLogix addresses, you must set the Micro800 variable as an array.
	For example:
	• To read the MicroLogix addresses N7:0…N7:10, you must verify that the Micro800 variable that is mapped to Data File 7 is an array of 0…10, or any other arrays whose length is 11.
	• To read the MicroLogix address N7:5, you must verify that the Micro800 variable that is mapped to Data File 7 is an array of 0…5, or any other arrays whose length is 6.

4. Verify that the MicroLogix file type is correct.

To delete a MicroLogix mapping, do the following:
1. In the Project or Controller Organizer, double-click the controller to open the controller workspace.
2. In the Controller tree, select MicroLogix Mapping.
3. On the Controller - MicroLogix Mapping configuration page, select the variable, then select Delete.

Use the IMPORT button

Export, modify, and add MicroLogix mapping information in bulk by importing or exporting CSV files. You can import a maximum of 1000 pieces of MicroLogix mapping information. Figure 89 shows the format of MicroLogix address associations in a CSV file. The format includes the file number, variable name, data type, and MicroLogix file type.

Figure 89 - MicroLogix Address Association in CSV File

	A	B	C	D
1	FileNumber	VariableName	DataType	FileType
2	7	Value1	Int[0..10]	N
3	8	Value2	Real	F
4				
5				

To import a CSV file for MicroLogix mapping, do the following:

1. In Project or Controller Organizer, double-click the controller to open the controller workspace.
2. In the Controller tree, select MicroLogix Mapping.
3. On the Controller - MicroLogix Mapping configuration page, select Import.
4. In the Open dialog box, select the CSV file that you want to import, then select Open.

> **IMPORTANT**   MicroLogix mapping information that is incorrect is not imported and an error message is shown in the Output window.

To export a CSV file for MicroLogix mapping, do the following:

1. In Project or Controller Organizer, double-click the controller to open the controller workspace.
2. In the Controller tree, select MicroLogix Mapping.
3. On the Controller - MicroLogix Mapping configuration page, select Export.
4. In the Save As dialog box, enter the filename, then select the location to save the exported file.
5. Select Save.

Notes:

Rockwell Automation Support

Use these resources to access support information.

Technical Support Center	Find help with how-to videos, FAQs, chat, user forums, Knowledgebase, and product notification updates.	rok.auto/support
Local Technical Support Phone Numbers	Locate the telephone number for your country.	rok.auto/phonesupport
Technical Documentation Center	Quickly access and download technical specifications, installation instructions, and user manuals.	rok.auto/techdocs
Literature Library	Find installation instructions, manuals, brochures, and technical data publications.	rok.auto/literature
Product Compatibility and Download Center (PCDC)	Download firmware, associated files (such as AOP, EDS, and DTM), and access product release notes.	rok.auto/pcdc

Documentation Feedback

Your comments help us serve your documentation needs better. If you have any suggestions on how to improve our content, complete the form at rok.auto/docfeedback.

Waste Electrical and Electronic Equipment (WEEE)

At the end of life, this equipment should be collected separately from any unsorted municipal waste.

Rockwell Automation maintains current product environmental compliance information on its website at rok.auto/pec.

Connect with us.