# Micro820 Programmable Controllers

Catalog Numbers 2080-LC20-20QWB, 2080-LC20-20QBB, 2080-LC20-20AWB, 2080-LC20-20QWBR, 2080-LC20-20QBBR, 2080-LC20-20AWBR

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.

## Preface

## Chapter 1

## Hardware Overview

## Chapter 2

## About Your Controller

## Chapter 3

## Install Your Controller

## Chapter 4

## Wire Your Controller

## Chapter 5

## Communication Connections

## Chapter 6

## Program Execution in Micro800

## Chapter 7

## Controller Security

## Chapter 8

## Using the Micro800 Remote LCD

## Chapter 9

## Using microSD Cards

## Appendix A

## Modbus Mapping for Micro800 Controllers

## Appendix B

## Troubleshooting

## Appendix C

## Quick Starts

## Appendix D

## PID Function Blocks

## About This Publication

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

This manual is a reference guide for Micro820® controllers. 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.

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.

## 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 template | throughout |
| Added statement on inclusive language | Important User Information |
| Added Daylight Saving section | 22 |
| Added Operation Mode section | 65 |

## 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 Plug-in Modules User Manual, publication 2080-UM004 | Provides information on features, configuration, installation, wiring, and specifications for the Micro800 plug-in modules. |
| Micro800 Programmable Controllers General Instructions Reference Manual, publication 2080-RM001 | Provides nformation on instruction sets for developing programs for use in Micro800 control systems. |
| 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. |
| Micro800 Programmable Controllers Installation Instructions, publication 2080-IN013 | Provides information on mounting and wiring Micro800 Controllers. |
| Micro800 RS-232/RS-485 Isolated Serial Port Plug-in Module Wiring Diagrams, publication 2080-WD002 | Provides nformation on mounting and wiring the Micro800 RS-232/485 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. |

## Additional Resources (Continued)

| Resource | Description |
|---|---|
| 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 | Provides 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 | Provides information on mounting and wiring the Micro800 Non-isolated Thermocouple Plug-in Module. |
| Micro800 Remote LCD Installation instructions, publication 2080-IN010 | Provides information on mounting and wiring the Micro800 remote LCD module. |
| Micro800 6-Channel Trimpot Analog Input Plug-In Module Wiring Diagrams, publication 2080-WD008 | Provides 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 | Provides 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 | Provides information on mounting and wiring the Micro800 Digital Input, Output, and Combination Plug-in Modules. |
| Micro800 High Speed Counter Plug-in Module Wiring Diagrams, publication 2080-WD012 | Provides information on mounting and wiring the High Speed Counter Plug-in module. |
| Micro800 DeviceNet Plug-in Module Wiring Diagrams, publication 2080-WD013 | Provides 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. |
| UL Standards Listing for Industrial Control Products, publication CMPNTS-SR002 | Assists original equipment manufacturers (OEMs) with construction of panels, to help ensure that they conform to the requirements of Underwriters Laboratories. |
| American Standards, Configurations, and Ratings: Introduction to Motor Circuit Design, publication IC-AT001 | Provides an overview of American motor circuit design based on methods that are outlined in the NEC. |
| 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 Selection and Configuration webpage, rok.auto/systemtools | Helps configure complete, valid catalog numbers and build complete quotes based on detailed product information. |
| 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.

Chapter 1

# Hardware Overview

This chapter provides an overview of the Micro820 hardware features.

## Hardware Features

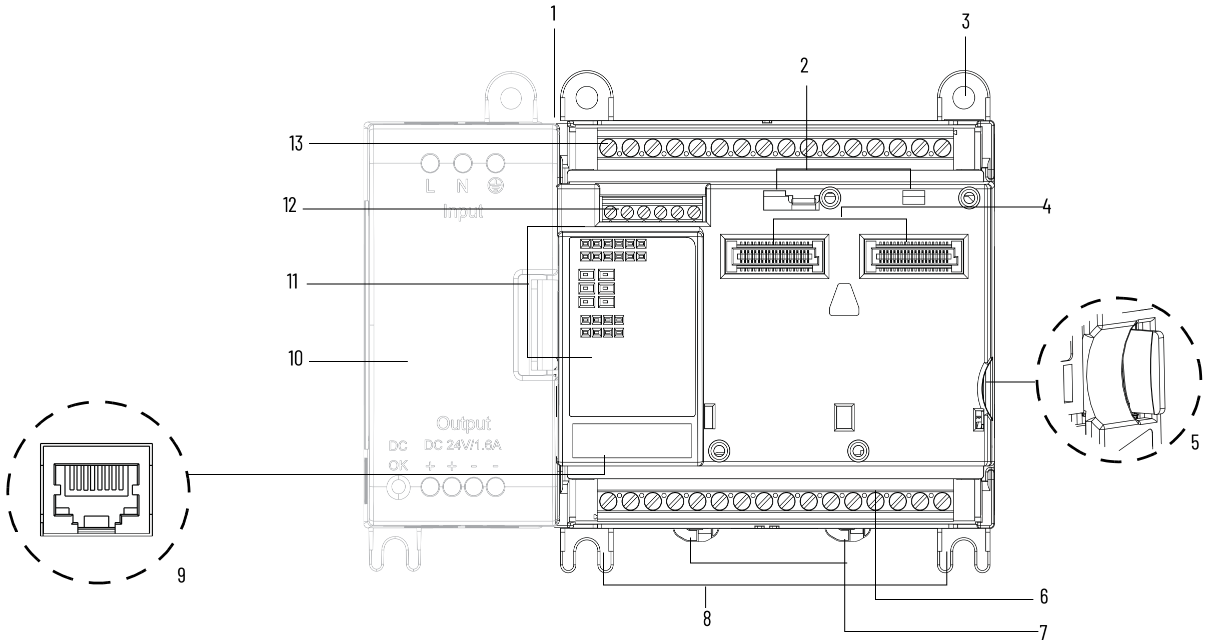
Micro820 controllers are 20-point economical brick style controllers with embedded inputs and outputs. These controllers can accommodate up to two plug-in modules and can connect to a remote LCD (2080-REMLCD) for configuring. The Micro820 controller also has a microSD™ card slot for project backup and restore, and datalog and recipe. Only Allen-Bradley® microSD cards, 2080-SD-2GB are supported.

| | |
|---|---|
| **IMPORTANT** | The Micro820 controller supports all Micro800 plug-in modules, except for 2080-MEMBAK-RTC and 2080-MEMBAK-RTC2.<br>For more information, see the Micro800 Plug-in Modules User Manual, publication 2080-UM004. |

For information on the REMLCD module, see Using the Micro800 Remote LCD on page 69.

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

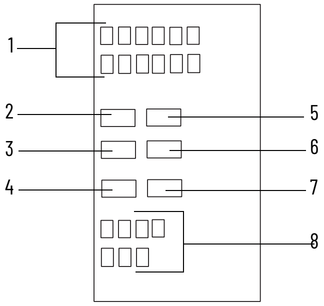
### Micro820 Controllers



### Micro820 Controller Description

| | Description | | Description |
|---|---|---|---|
| 1 | Optional power supply slot | 8 | Mounting feet |
| 2 | Plug-in latch | 9 | RJ45 Ethernet connector port |
| 3 | Mounting screw hole | 10 | Power supply |
| 4 | 40-pin high-speed plug-in connector slot | 11 | Status indicators |
| 5 | microSD (Micro Secure Digital) card slot | 12 | RS-232/RS-485 non-isolated combo Serial ports |
| 6 | Removable/fixed terminal blocks | 13 | Removable/fixed terminal blocks |
| 7 | DIN rail mounting latch | | |

### Status Indicators



### Micro820 Controller Status Indicator Description[1]

| | Description | | Description |
|---|---|---|---|
| 1 | Input status | 5 | Fault status |
| 2 | Run status | 6 | Comm status |
| 3 | Force status | 7 | SD status |
| 4 | ENET status | 8 | Output status |

[1] For detailed description of the different status indicators, see Troubleshooting on page 109.

**ATTENTION:** Removable terminal blocks are available on catalog numbers that end in R (for example, 2080-LC20-20QBBR). Fixed terminal blocks are available on catalog numbers that do not end in R (for example, 2080-LC20-20QBB).
You can order a replacement terminal block, catalog number 2080-RPL24RTB, separately.

## Inputs and Outputs

Table 1 - Micro820 Controllers – Number and Types of Inputs/Outputs

| Catalog Number | Inputs | | | Outputs | | | Analog Out 0…10V DC | Analog In 0…10V (shared with DC In) | PWM Support |
|---|---|---|---|---|---|---|---|---|---|
| | 120V AC | 120/240V AC | 24V DC | Relay | 24V DC Source | 24V DC Sink | | | |
| 2080-LC20-20AWB | 8 | – | 4 | 7 | – | – | 1 | 4 | – |
| 2080-LC20-20AWBR | 8 | – | 4 | 7 | | – | 1 | 4 | – |
| 2080-LC20-20QWB | – | – | 12 | 7 | – | – | 1 | 4 | – |
| 2080-LC20-20QWBR | – | – | 12 | 7 | – | – | 1 | 4 | – |
| 2080-LC20-20QBB | – | – | 12 | | 7 | – | 1 | 4 | 1 |
| 2080-LC20-20QBBR | – | – | 12 | – | 7 | – | 1 | 4 | 1 |

## Embedded microSD Card Slot

Micro820 controllers support Micro Secure Digital (microSD) cards through an embedded microSD card slot. The design supports Class 6 and 10 SDSC and SDHC microSD cards, with FAT32/16 formats. The microSD file system supports only one file partition. Only Allen-Bradley microSD cards, 2080-SD-2GB are supported.

The microSD card is primarily used for project backup and restore, as well as datalog and recipe functions. It can also be used to configure power-up settings (such as controller mode, IP address, and so on) through an optional ConfigMeFirst.txt file.

For more information, see Using microSD Cards on page 79.

To help you troubleshoot microSD card-related errors, see Troubleshooting on page 109.

**WARNING:** The microSD card slot only supports Allen-Bradley microSD card 2080-SD-2GB. Other microSD cards may not work and are not recommended.

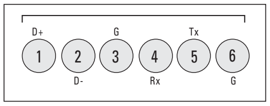
## Embedded RS-232/RS-485 Serial Port Combo

The Micro820 controller supports an embedded non-isolated RS-232/RS-485 combo communications port. Only one port (RS-232 or RS-485) can work at any given time. The communication rate of this port supports up to 38.4 K.

The communication port uses a 6-pin 3.5 mm terminal block with pin definition shown in Table 2.

**IMPORTANT** Serial port cables should not exceed 3 m (9.84 ft.) length.

Table 2 - RS-232/RS-485 Serial Port Pin Definition



| Pin | Definition | RS-485 Example | RS-232 Example |
|---|---|---|---|
| 1 | RS-485+ | RS-485+ | (not used) |
| 2 | RS-485- | RS-485- | (not used) |
| 3 | GND | GND | GND |
| 4 | RS-232 input (receiver) | (not used) | RxD |
| 5 | RS-232 output (driver) | (not used) | TxD |
| 6 | GND | GND | GND |

The communication port (both RS-232 and RS-485) are non-isolated. The signal ground of the port is not isolated to the logic ground of the controller.

The RS-232 port supports connection to the Micro800 Remote LCD module (2080-REMLCD).

### REMLCD to Micro820 Serial Port Terminal Block Wiring

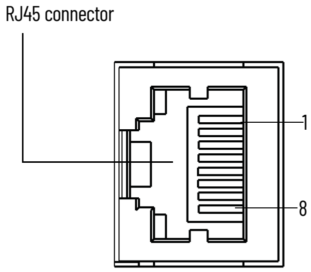
| REMLCD Serial Port Terminal Block | | | Micro820 Serial Port Terminal Block | |
|---|---|---|---|---|
| Signal | Pin number | | Pin number | Signal |
| RS-232 TX | 1 | <--------> | 4 | RX RS-232 |
| RS-232 RX | 2 | <--------> | 5 | TX RS-232 |
| RS-232 G | 3 | <--------> | 6 | G RS-232 |

## Embedded Ethernet Support

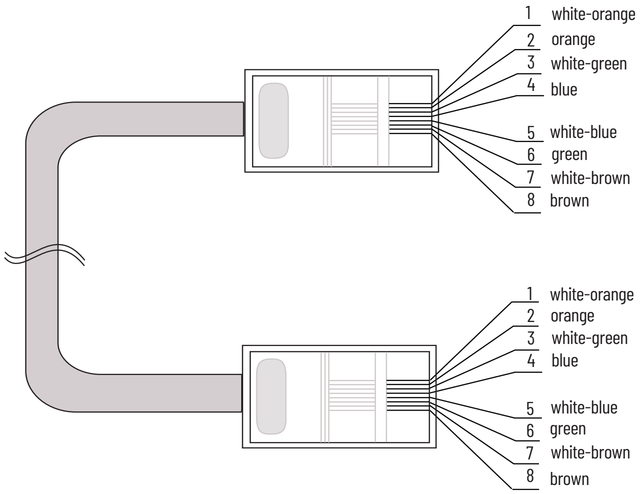
A 10/100 Base-T Port is available for connection to an Ethernet network through any standard RJ45 Ethernet cable.

### RJ45 Ethernet Port Pin Mapping

| Contact Number | Signal | Direction | Primary Function |
|---|---|---|---|
| 1 | TX+ | OUT | Transmit data + |
| 2 | TX- | OUT | Transmit data - |
| 3 | RX+ | IN | Receive data + |
| 4 | – | – | – |
| 5 | – | – | – |
| 6 | RX- | IN | Receive data - |
| 7 | – | – | – |
| 8 | – | – | – |



**Ethernet port pin-to-pin connection**



For descriptions of the ENET status indicator, see Troubleshooting on page 109.

# 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 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, structured text) with user-defined function block (UDFB) 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.

## Controller Changes in Run Mode

Micro820 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 Using Run Mode Change (RMC) on page 13.
- 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 Configuration Change (RMCC) on page 17.

## Using Run Mode Change (RMC)

Run Mode Change (RMC) is a productivity enhancement feature introduced in software version 8 for Micro820 controllers. The feature saves the user time by allowing logic modifications to a running project without going to remote program mode and without disconnecting from the controller. You must use the Connected Components Workbench Developer Edition software version 8 software, or later, to use this feature.

| | |
|---|---|
| **IMPORTANT** | You must use Micro820 controller firmware revision 8.xxx or later 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 in order to prevent a large delay in the program scan. This 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 3 - 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 the RMC button is clicked while connected to the controller and ends when the accept 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 will respond 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 the variable to notify you on the HMI that there are uncommitted changes in the controller.

**Table 4 - Bit Definitions of Global Variable – __SYSVA_PROJ_INCOMPLETE**

| Bit | Definition |
|---|---|
| 0 | Set when the Run Mode Change process starts.<br>Clears once the Run Mode Change is written permanently to the controller (completion of Accept or Undo).<br>Use the bit to warn you that a Run Mode Change is in progress and that there are uncommitted changes in the controller. |
| 1 | Set if an error occurs while saving the changes to flash memory or if an integrity check fails during Run Mode Change.<br>Clears 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 you perform a Test Logic is performed, or undo changes after the Test Logic is completed, any active communication instructions are aborted while the changes download to the controller.

## Uncommitted Changes

Uncommitted changes are changes that are made in RMC that have not been accepted or are 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.

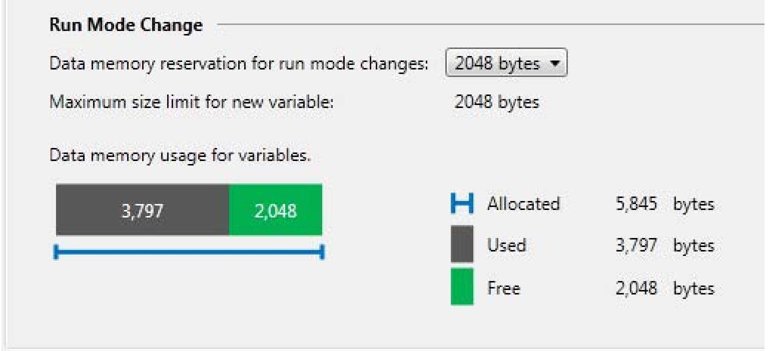
If you choose to upload a project with uncommitted changes from the controller, you cannot enter RMC until you have performed a full download.

## RMC Memory

Use Run Mode Change (RMC) memory to store both the logic and user variable changes made during RMC. The default amount of memory allocated is 2 KB and you can increased it up to 8 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 is reduced by 6 KB to support optimal project settings for the new RMC feature.<br>If you have a project that is developed before software version 8, you may need to reduce the default "Allocated" 8 KB Temporary Variables section from the Memory page in order to compile the project successfully. |

**Figure 1 - Controller Memory Diagnostics Page in Connected Components Workbench Software**



Run Mode Change

Data memory reservation for run mode changes: 2048 bytes

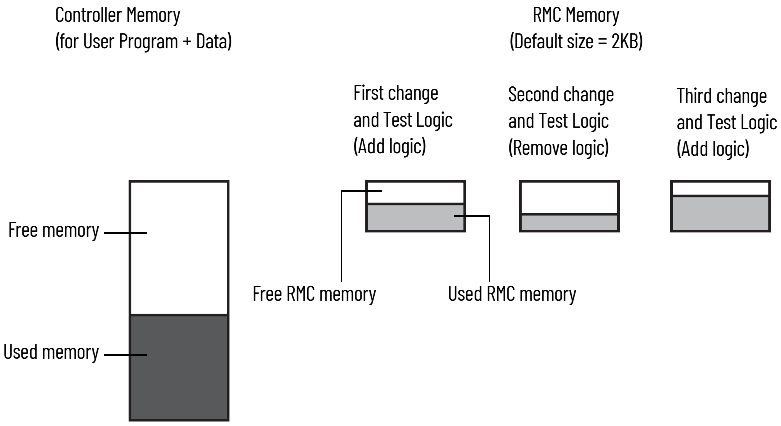
Maximum size limit for new variable: 2048 bytes

Data memory usage for variables.

| | | |
|---|---|---|
| Allocated | 5,845 | bytes |
| Used | 3,797 | bytes |
| Free | 2,048 | bytes |

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**

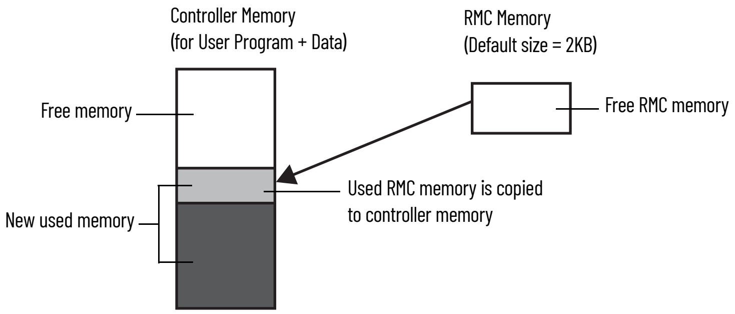


If insufficient RMC memory is available to make more changes (for example, a "not enough memory" error message appears during the RMC build or Test Logic), then you must perform a full download to transfer the incremental changes from the RMC memory to the standard user program and data memory.

*Transfer Contents in RMC Memory to Controller Memory*

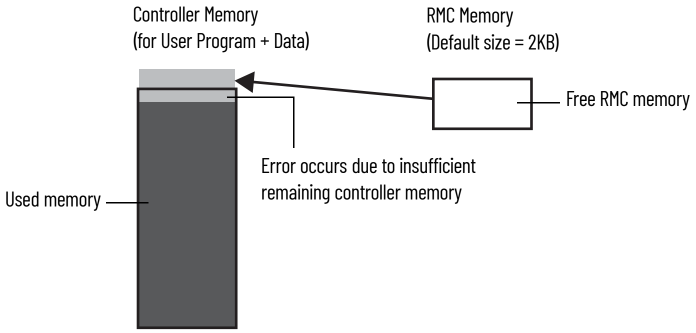
The changes that you made during RMC are stored in the RMC memory and remain there until you perform a full build and download (while the controller is disconnected).

**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 will fail and a "not enough memory" error message will appear. Do not use RMC if you are near the limits of your controller memory.

**Figure 4 - Insufficient Controller Memory Example**


Controller Memory
(for User Program + Data)
RMC Memory
(Default size = 2KB)
Free RMC memory
Error occurs due to insufficient
remaining controller memory
Used memory

## Limitations of RMC

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

- You cannot make configuration changes (for example, change filter times).
- You can add up to 2 KB of logic (approximately 150 boolean instructions) and user variables for each Test Logic.
- You can increase the total memory allocated for RMC (cumulative of all Test Logic Changes) from 2 KB to 8 KB, but the 2 KB limit for logic and user variables per Test Logic remains.
- You can add a total of 20 Program Organizational Units (POU) for each RMC (for example, if you currently have 5 POU, you can add 20 more for a total of 25 POU).
- If you modify a user defined function block (UDFB) that changes the local variables, the local variables are reinitialized or reset to zero and a warning message shows during the build. If you want to reapply the initial value, right-click on the UDFB and select Refactor > Reset Initial Values of Instances.
- RMC is not possible after you perform a Discover Project operation, if a new module is detected, because the configuration has changed.
- You cannot import exchange files when in RMC because an import is considered a configuration change.

- If you make changes to the display configuration (for example, hiding comments), they are treated as logic changes and you must build the project.
- You can add global variables in RMC, but you cannot delete or modify them. To delete or modify a global variable, you must disconnect the controller from the Connected Components Workbench software.
- If you create a global variable in RMC, it does not show in the LCD display.
- When using Common Industrial Protocol (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 timeout after 60 seconds. This behavior applies to Connected Components Workbench software version 9 or earlier projects. For version 10 or later projects you can configure the CIP connection timeout.

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

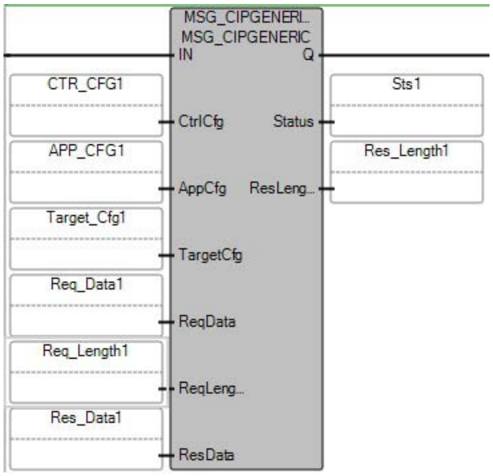
For an example of how to use this feature, see Using Run Mode Change on page 129.

## 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 Micro820 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. You must have Micro820 controller firmware revision 9.011 or later to use this feature.

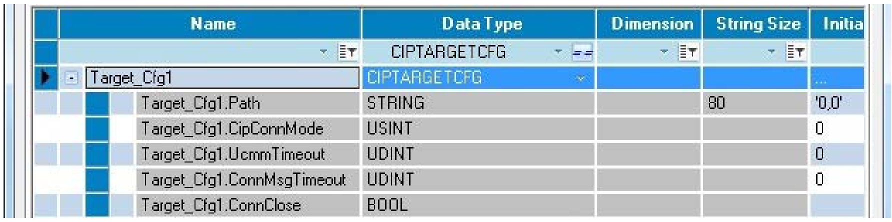
You can use RMCC to change the address configuration of the controller during run mode, when you set the communication protocol to Modbus RTU for the Serial ports, or EtherNet/IP™ for the Ethernet port. RMCC uses a CIP Generic message, which you can only send from within a controller program and not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

**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**



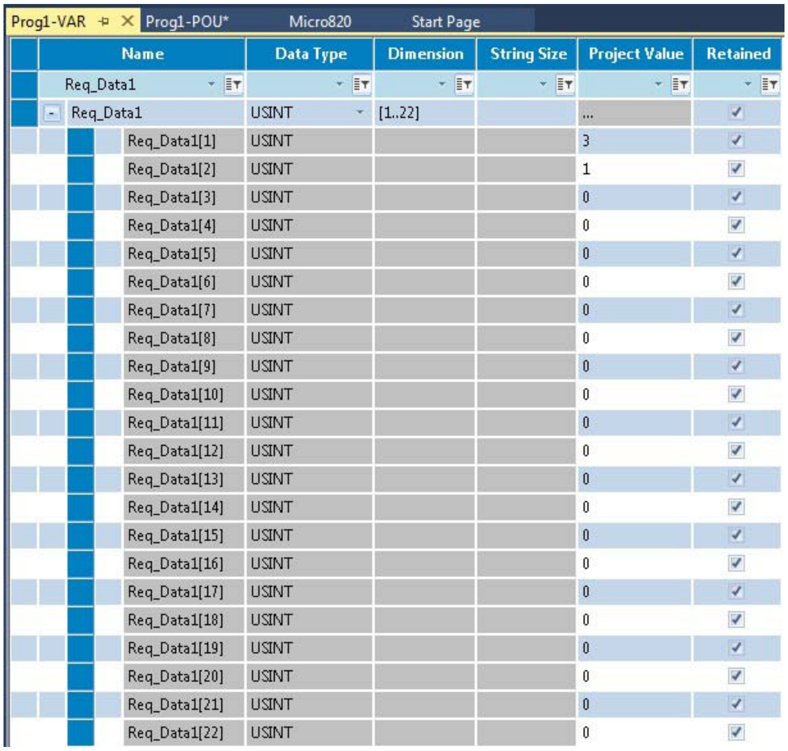
| Name | Data Type | Dimension | String Size | Initia |
|---|---|---|---|---|
| | CIPTARGETCFG | | | |
| Target_Cfg1 | CIPTARGETCFG | | | ... |
| Target_Cfg1.Path | STRING | | 80 | '0,0' |
| Target_Cfg1.CipConnMode | USINT | | | 0 |
| Target_Cfg1.UcmmTimeout | UDINT | | | 0 |
| Target_Cfg1.ConnMsgTimeout | UDINT | | | 0 |
| Target_Cfg1.ConnClose | BOOL | | | |

For Micro820 controllers, the address configuration change is not permanent and is lost when the controller is power cycled (for firmware revision 9 or earlier). If you want the address to change immediately after a power cycle, you must perform this CIP Generic message after every power up. You can do this by doing the following:

- Use the system variable __SYSVA_POWER_UP_BIT.
- Set the Retained flag for the variable containing the new address.

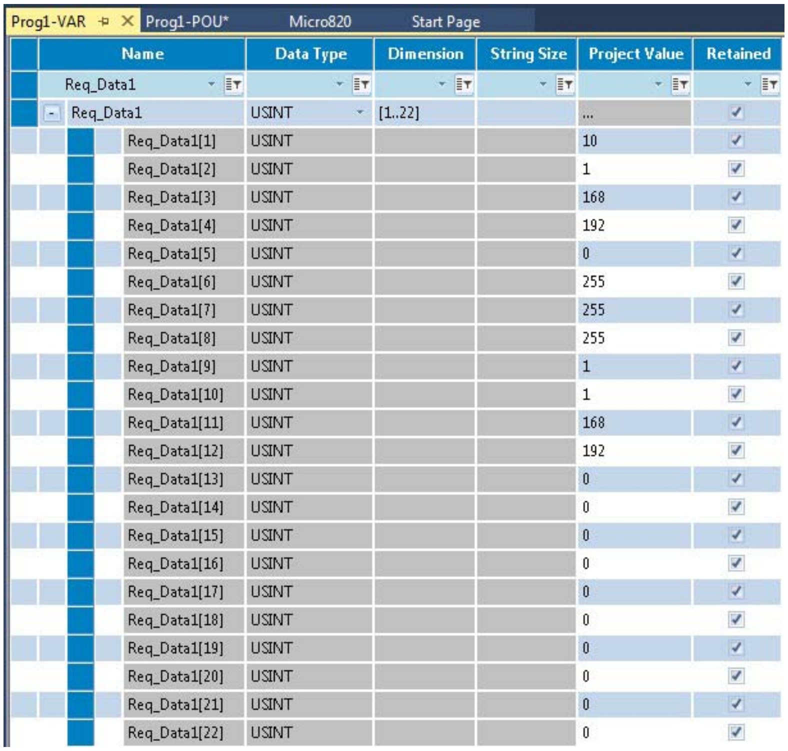
From firmware revision 10 or later, Micro820 controllers now retains the address configuration when you cycle power to the controller.

**Figure 7 - Set Retained Flag – Modbus Address**



| Name | Data Type | Dimension | String Size | Project Value | Retained |
|---|---|---|---|---|---|
| Req_Data1 | | | | | |
| - Req_Data1 | USINT | [1..22] | | ... | ✓ |
| Req_Data1[1] | USINT | | | 3 | ✓ |
| Req_Data1[2] | USINT | | | 1 | ✓ |
| Req_Data1[3] | USINT | | | 0 | ✓ |
| Req_Data1[4] | USINT | | | 0 | ✓ |
| Req_Data1[5] | USINT | | | 0 | ✓ |
| Req_Data1[6] | USINT | | | 0 | ✓ |
| Req_Data1[7] | USINT | | | 0 | ✓ |
| Req_Data1[8] | USINT | | | 0 | ✓ |
| Req_Data1[9] | USINT | | | 0 | ✓ |
| Req_Data1[10] | USINT | | | 0 | ✓ |
| Req_Data1[11] | USINT | | | 0 | ✓ |
| Req_Data1[12] | USINT | | | 0 | ✓ |
| Req_Data1[13] | USINT | | | 0 | ✓ |
| Req_Data1[14] | USINT | | | 0 | ✓ |
| Req_Data1[15] | USINT | | | 0 | ✓ |
| Req_Data1[16] | USINT | | | 0 | ✓ |
| Req_Data1[17] | USINT | | | 0 | ✓ |
| Req_Data1[18] | USINT | | | 0 | ✓ |
| Req_Data1[19] | USINT | | | 0 | ✓ |
| Req_Data1[20] | USINT | | | 0 | ✓ |
| Req_Data1[21] | USINT | | | 0 | ✓ |
| Req_Data1[22] | USINT | | | 0 | ✓ |

**Figure 8 - Set Retained Flag - EtherNet/IP Address**



Prog1-VAR | Prog1-POU* | Micro820 | Start Page

| Name | Data Type | Dimension | String Size | Project Value | Retained |
|---|---|---|---|---|---|
| Req_Data1 | | | | | |
| Req_Data1 | USINT | [1..22] | | ... | ✓ |
| Req_Data1[1] | USINT | | | 10 | ✓ |
| Req_Data1[2] | USINT | | | 1 | ✓ |
| Req_Data1[3] | USINT | | | 168 | ✓ |
| Req_Data1[4] | USINT | | | 192 | ✓ |
| Req_Data1[5] | USINT | | | 0 | ✓ |
| Req_Data1[6] | USINT | | | 255 | ✓ |
| Req_Data1[7] | USINT | | | 255 | ✓ |
| Req_Data1[8] | USINT | | | 255 | ✓ |
| Req_Data1[9] | USINT | | | 1 | ✓ |
| Req_Data1[10] | USINT | | | 1 | ✓ |
| Req_Data1[11] | USINT | | | 168 | ✓ |
| Req_Data1[12] | USINT | | | 192 | ✓ |
| Req_Data1[13] | USINT | | | 0 | ✓ |
| Req_Data1[14] | USINT | | | 0 | ✓ |
| Req_Data1[15] | USINT | | | 0 | ✓ |
| Req_Data1[16] | USINT | | | 0 | ✓ |
| Req_Data1[17] | USINT | | | 0 | ✓ |
| Req_Data1[18] | USINT | | | 0 | ✓ |
| Req_Data1[19] | USINT | | | 0 | ✓ |
| Req_Data1[20] | USINT | | | 0 | ✓ |
| Req_Data1[21] | USINT | | | 0 | ✓ |
| Req_Data1[22] | USINT | | | 0 | ✓ |

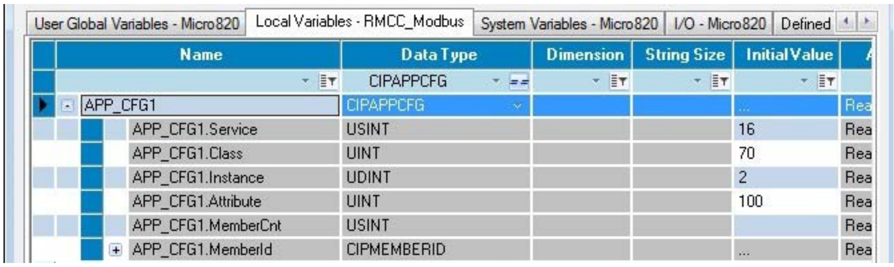
## Using Modbus RTU Communication

To use RMCC with the Modbus RTU communication protocol, you must set the Serial port to the Modbus slave role. A CIP Generic message is sent from within a program with the following parameters.

**Table 5 - CIP Generic Message Parameters for RMCC using Modbus RTU**

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

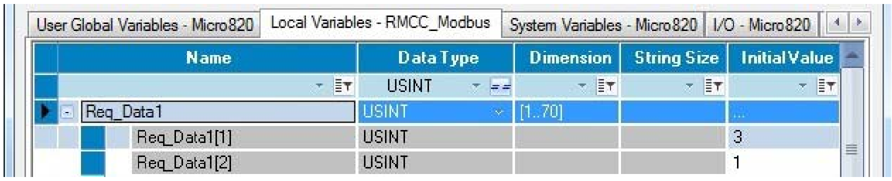
**Figure 9 - RMCC Modbus Example - Set the Parameters**



User Global Variables - Micro820 | Local Variables - RMCC_Modbus | System Variables - Micro820 | I/O - Micro820 | Defined

| Name | Data Type | Dimension | String Size | Initial Value | A |
|---|---|---|---|---|---|
| | CIPAPPCFG | | | | |
| APP_CFG1 | CIPAPPCFG | | | ... | Rea |
| APP_CFG1.Service | USINT | | | 16 | Rea |
| APP_CFG1.Class | UINT | | | 70 | Rea |
| APP_CFG1.Instance | UDINT | | | 2 | Rea |
| APP_CFG1.Attribute | UINT | | | 100 | Rea |
| APP_CFG1.MemberCnt | USINT | | | | Rea |
| APP_CFG1.MemberId | CIPMEMBERID | | | ... | Rea |

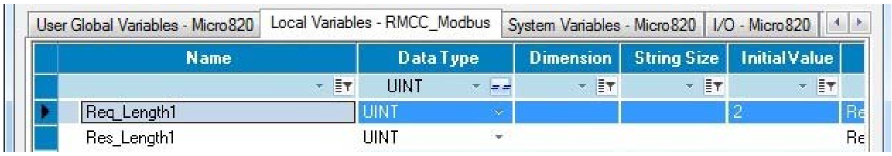
**Figure 10 - RMCC Modbus Example – Set the New Node Address**


User Global Variables - Micro820
Local Variables - RMCC_Modbus
System Variables - Micro820
I/O - Micro820

| Name | Data Type | Dimension | String Size | Initial Value |
|---|---|---|---|---|
| | USINT | | | |
| Req_Data1 | USINT | [1..70] | | ... |
| Req_Data1[1] | USINT | | | 3 |
| Req_Data1[2] | USINT | | | 1 |

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 11 - RMCC Modbus Example – Set the Message Length**


User Global Variables - Micro820
Local Variables - RMCC_Modbus
System Variables - Micro820
I/O - Micro820

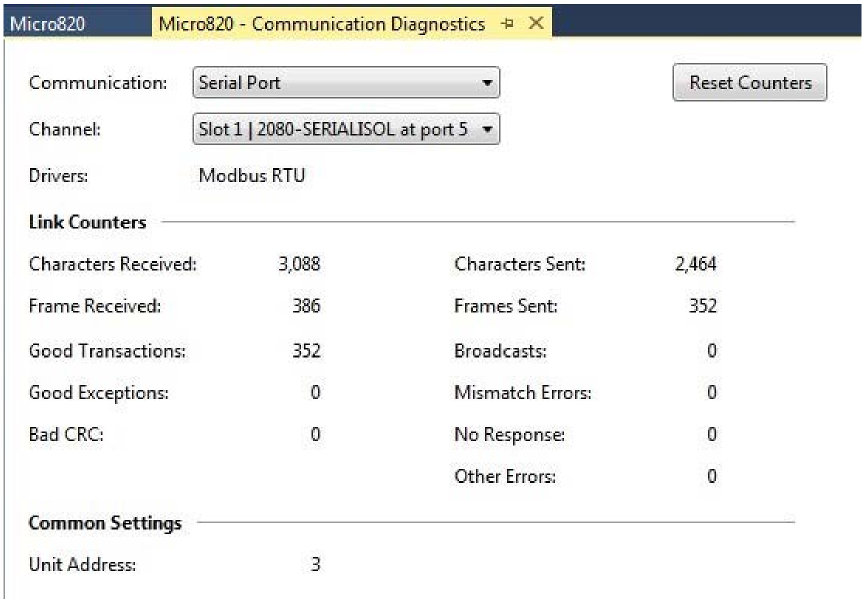
| Name | Data Type | Dimension | String Size | Initial Value | |
|---|---|---|---|---|---|
| | UINT | | | | |
| Req_Length1 | UINT | | | 2 | Re |
| Res_Length1 | UINT | | | | Re |

When the new node address is configured and applied, the port is not restarted.

| IMPORTANT | You must ensure that the new node address being configured is unique as it will not be 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 12 - RMCC Modbus Example – Verify Address Change**



Communication: Serial Port

Channel: Slot 1 | 2080-SERIALISOL at port 5

Drivers: Modbus RTU

Link Counters

| | | | |
|---|---|---|---|
| Characters Received: | 3,088 | Characters Sent: | 2,464 |
| Frame Received: | 386 | Frames Sent: | 352 |
| Good Transactions: | 352 | Broadcasts: | 0 |
| Good Exceptions: | 0 | Mismatch Errors: | 0 |
| Bad CRC: | 0 | No Response: | 0 |
| | | Other Errors: | 0 |

Common Settings

Unit Address: 3

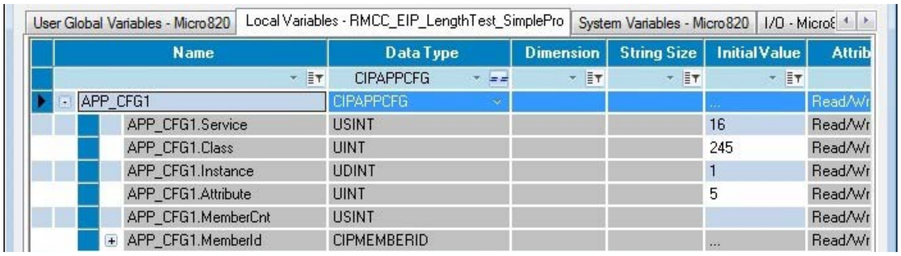
## Using EtherNet/IP Communication

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

Table 6 - CIP Generic Message Parameters for RMCC using EtherNet/IP

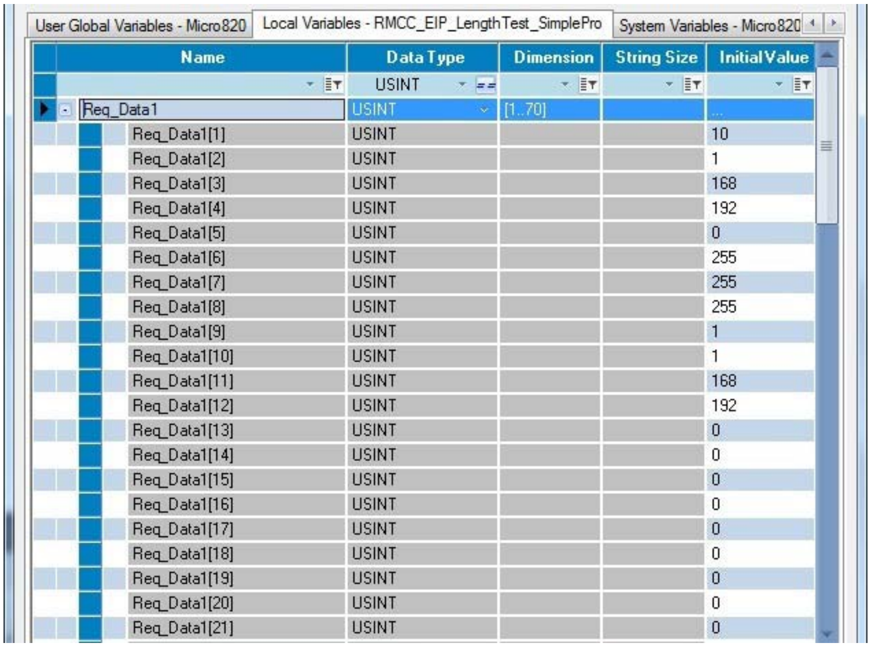
| Parameter | Value |
|---|---|
| Service | 16 |
| Class | 245 |
| Instance | 1 |
| Attribute | 5 |
| ReqData | IP address, Subnet mask, Gateway address |
| ReqLen | 22 bytes |

Figure 13 - RMCC EtherNet/IP Example – Set the Parameters


User Global Variables - Micro820
Local Variables - RMCC_EIP_LengthTest_SimplePro
System Variables - Micro820
I/O - Micro8

| Name | Data Type | Dimension | String Size | Initial Value | Attrib |
|---|---|---|---|---|---|
| | CIPAPPCFG | | | | |
| APP_CFG1 | CIPAPPCFG | | | ... | Read/Wr |
| APP_CFG1.Service | USINT | | | 16 | Read/Wr |
| APP_CFG1.Class | UINT | | | 245 | Read/Wr |
| APP_CFG1.Instance | UDINT | | | 1 | Read/Wr |
| APP_CFG1.Attribute | UINT | | | 5 | Read/Wr |
| APP_CFG1.MemberCnt | USINT | | | | Read/Wr |
| APP_CFG1.MemberId | CIPMEMBERID | | | ... | Read/Wr |

Figure 14 - RMCC EtherNet/IP Example – Set the New IP Address

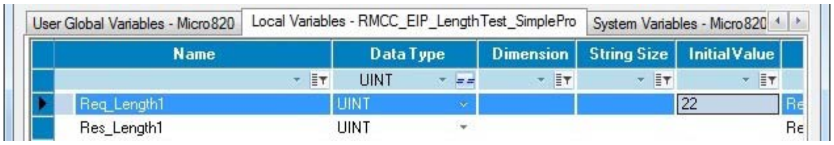

User Global Variables - Micro820
Local Variables - RMCC_EIP_LengthTest_SimplePro
System Variables - Micro820

| Name | Data Type | Dimension | String Size | Initial Value |
|---|---|---|---|---|
| | USINT | | | |
| Req_Data1 | USINT | [1..70] | | ... |
| Req_Data1[1] | USINT | | | 10 |
| Req_Data1[2] | USINT | | | 1 |
| Req_Data1[3] | USINT | | | 168 |
| Req_Data1[4] | USINT | | | 192 |
| Req_Data1[5] | USINT | | | 0 |
| Req_Data1[6] | USINT | | | 255 |
| Req_Data1[7] | USINT | | | 255 |
| Req_Data1[8] | USINT | | | 255 |
| Req_Data1[9] | USINT | | | 1 |
| Req_Data1[10] | USINT | | | 1 |
| Req_Data1[11] | USINT | | | 168 |
| Req_Data1[12] | USINT | | | 192 |
| Req_Data1[13] | USINT | | | 0 |
| Req_Data1[14] | USINT | | | 0 |
| Req_Data1[15] | USINT | | | 0 |
| Req_Data1[16] | USINT | | | 0 |
| Req_Data1[17] | USINT | | | 0 |
| Req_Data1[18] | USINT | | | 0 |
| Req_Data1[19] | USINT | | | 0 |
| Req_Data1[20] | USINT | | | 0 |
| Req_Data1[21] | USINT | | | 0 |

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 15 - RMCC EtherNet/IP Example – Set the Message Length


User Global Variables - Micro820
Local Variables - RMCC_EIP_LengthTest_SimplePro
System Variables - Micro820

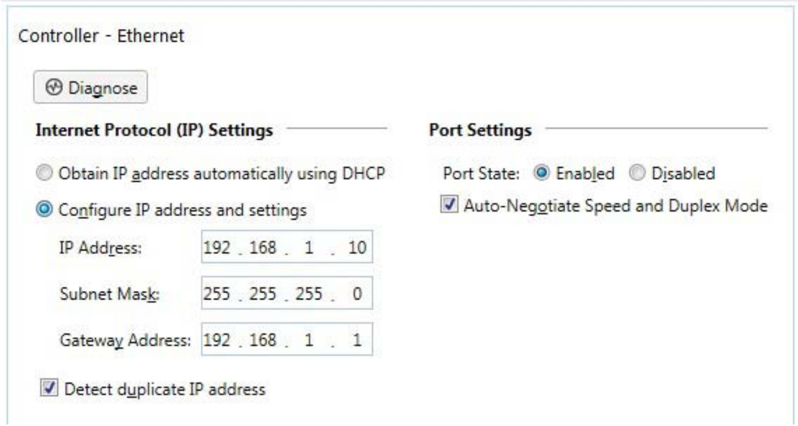
| Name | Data Type | Dimension | String Size | Initial Value | |
|---|---|---|---|---|---|
| | UINT | | | | |
| Req_Length1 | UINT | | | 22 | Re |
| Res_Length1 | UINT | | | | Re |

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

| IMPORTANT | You should not perform IP address changes continuously. Allow an interval of at least six seconds before you perform 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 16 - RMCC EtherNet/IP Example – Verify Address Change



Controller - Ethernet

Internet Protocol (IP) Settings

○ Obtain IP address automatically using DHCP

● Configure IP address and settings

IP Address: 192 . 168 . 1 . 10

Subnet Mask: 255 . 255 . 255 . 0

Gateway Address: 192 . 168 . 1 . 1

☑ Detect duplicate IP address

Port Settings

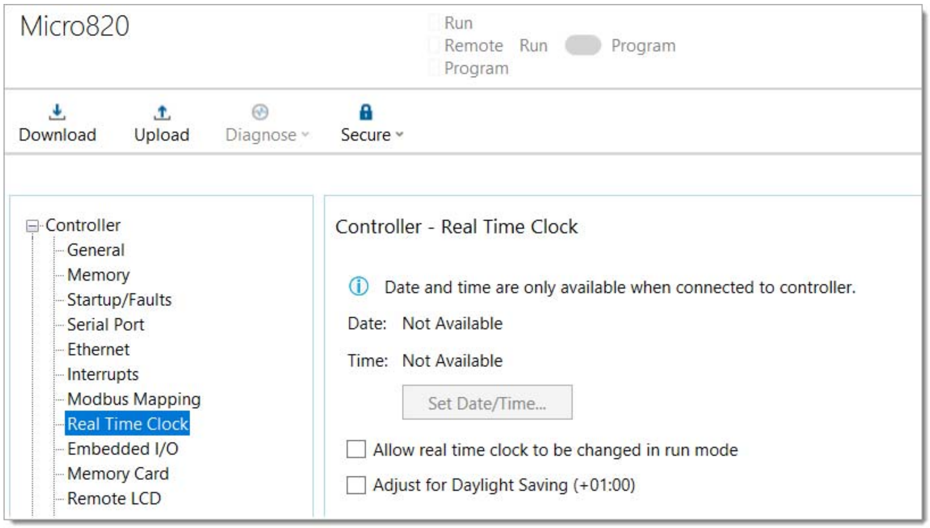
Port State: ● Enabled ○ Disabled

☑ Auto-Negotiate Speed and Duplex Mode

## Daylight Saving

The Adjust for Daylight Saving (+01:00) option allows you to configure the controller to shift the clock by 1 hour during the summer time. The feature is available for Micro820 controllers in the Connected Components Workbench software version 22.00 or later.

Figure 17 - Daylight Saving



Controller - Real Time Clock

Date and time are only available when connected to controller.

Date: Not Available

Time: Not Available

□ Allow real time clock to be changed in run mode

□ Adjust for Daylight Saving (+01:00)

To enable daylight savings, do the following:

1. On the device configuration tree, go to the Controller properties and select Real Time Clock.
2. Select the Adjust for Daylight Saving (+01:00) option.
3. Download the project to the controller.

   You can adjust the setting during runtime.

## Safety Considerations

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

### Disconnecting Main Power

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

The main power disconnect switch should be located where operators and maintenance personnel have quick and easy access to it. In addition to disconnecting electrical power, all other sources of power (pneumatic and hydraulic) should be de-energized before working on a machine or process controlled by a controller.

### Safety Circuits

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

Circuits 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, thereby removing power to the machine. Never alter these circuits to defeat their function. Serious injury or machine damage could result.

### 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 remain 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 assure they will 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 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 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 is that the power supply capacitors charge more slowly. However, consider the effect of a voltage sag on other equipment. For example, a deep voltage sag may reset a computer 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 at the same time, a brief sag in the power source voltage typically will not affect any equipment.

### Loss of Power Source

The optional Micro800 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 program scan hold-up time after loss of power. The duration of the power supply hold-up time depends on power consumption of 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 a period of time. Both of these conditions are considered to be a loss of power for the system.

## Preventing Excessive Heat

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

In some applications, a substantial amount of heat is produced by other equipment inside or outside the enclosure. In this case, place blower fans inside the enclosure to assist in air circulation and to reduce “hot spots” near the controller.

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 could cause improper operation or damage to components. In extreme cases, you may need to 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 Figure 18 and Figure 19.

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

If you are using 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 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 operators and maintenance personnel have quick and easy access to it. 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 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 processor 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 inspecting or installing terminal connections, replacing output fuses, or working 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 18 and Figure 19, input and output circuits are shown with MCR protection. However, in most applications, only output circuits require MCR protection.

The following illustrations show the Master Control Relay wired in a grounded system.

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

Figure 18 - Schematic with IEC Symbols

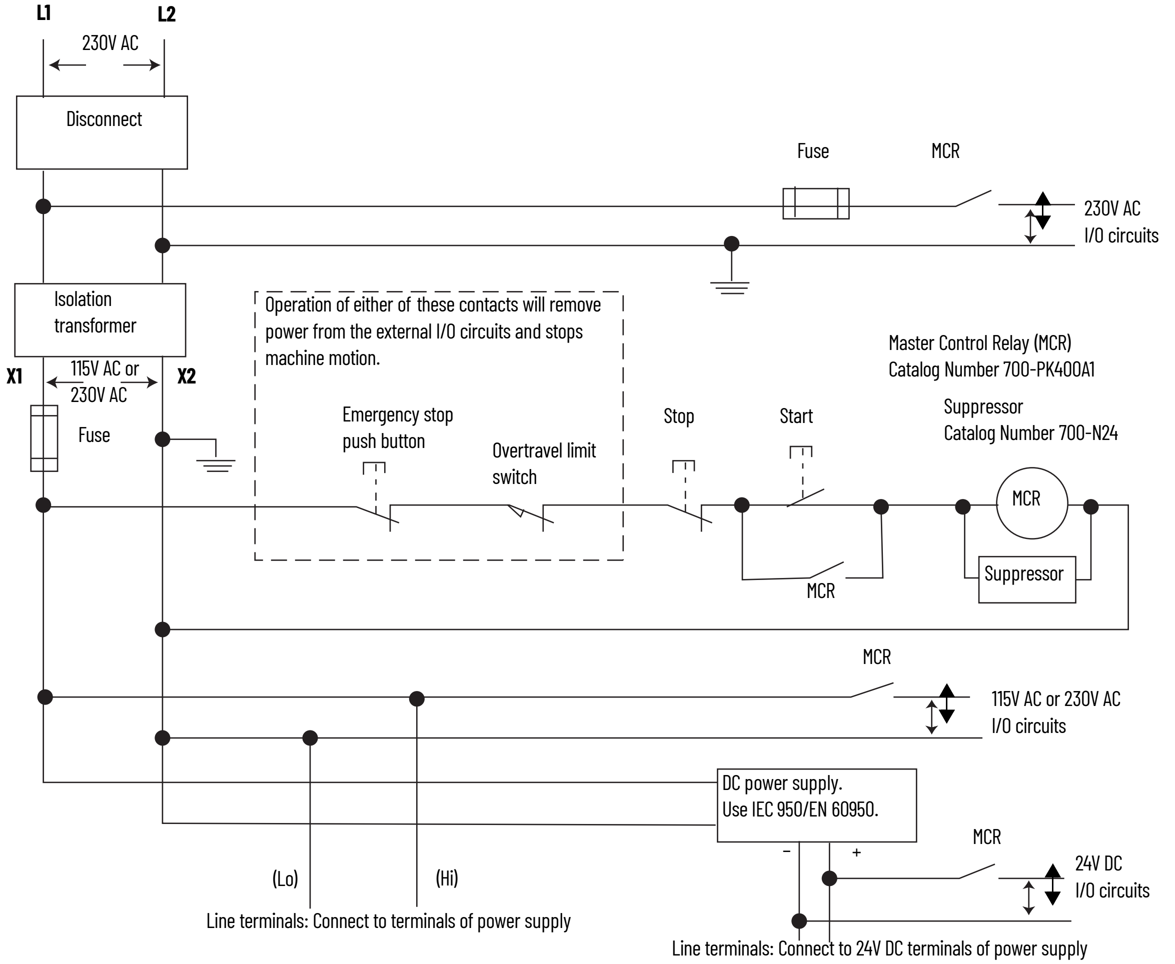
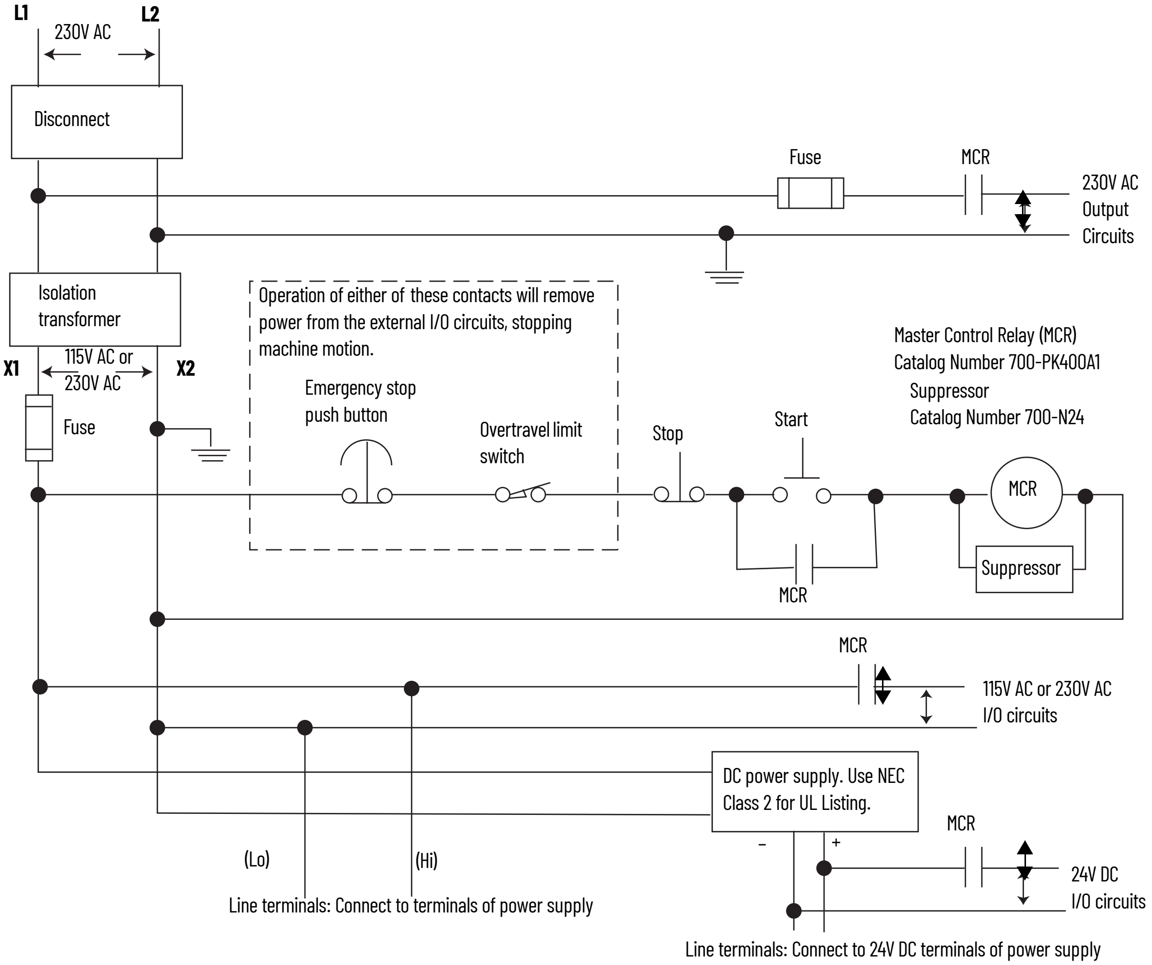


Figure 19 - Schematic with ANSI/CSA Symbols


L1
L2
230V AC
Disconnect
Fuse
MCR
230V AC
Output
Circuits
Isolation
transformer
X1
115V AC or
230V AC
X2
Fuse
Operation of either of these contacts will remove power from the external I/O circuits, stopping machine motion.
Emergency stop
push button
Overtravel limit
switch
Stop
Start
Master Control Relay (MCR)
Catalog Number 700-PK400A1
Suppressor
Catalog Number 700-N24
MCR
Suppressor
MCR
MCR
115V AC or 230V AC
I/O circuits
DC power supply. Use NEC
Class 2 for UL Listing.
–
+
MCR
24V DC
I/O circuits
(Lo)
(Hi)
Line terminals: Connect to terminals of power supply
Line terminals: Connect to 24V DC terminals of power supply

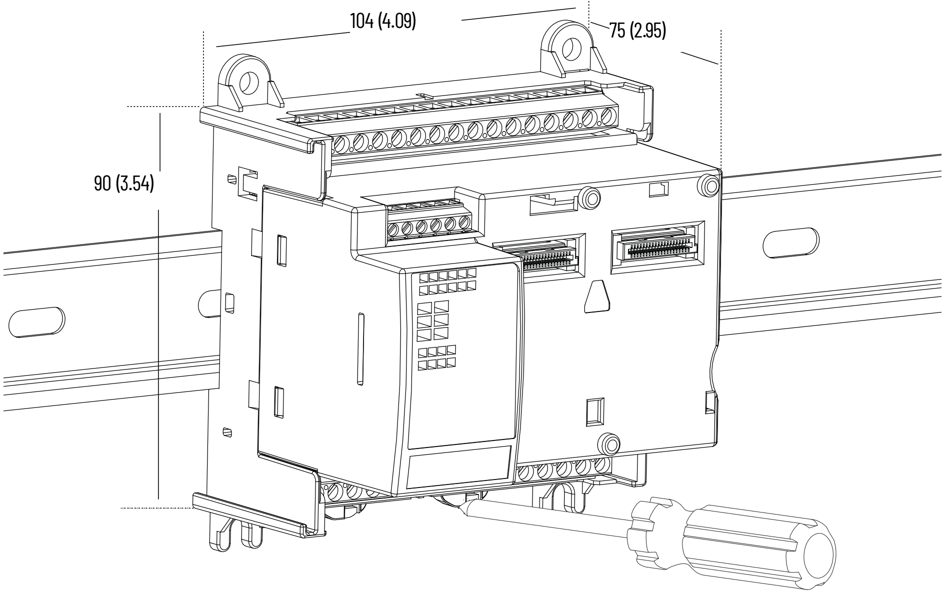
## Notes:

# Install Your Controller

This chapter serves to guide you on how to install the controller.

## Controller Mounting Dimensions

Mounting dimensions do not include mounting feet or DIN rail latches.



**Measurements in mm (in.)**

### Module Spacing

Maintain spacing from enclosure walls, wireways, and adjacent equipment. Allow 50.8 mm (2 in.) of space on all sides. This provides ventilation and electrical isolation. 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 and 35 x 15 mm (EN 50 022 - 35 x 7.5 and EN 50 022 - 35 x 15).

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 flat-blade 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

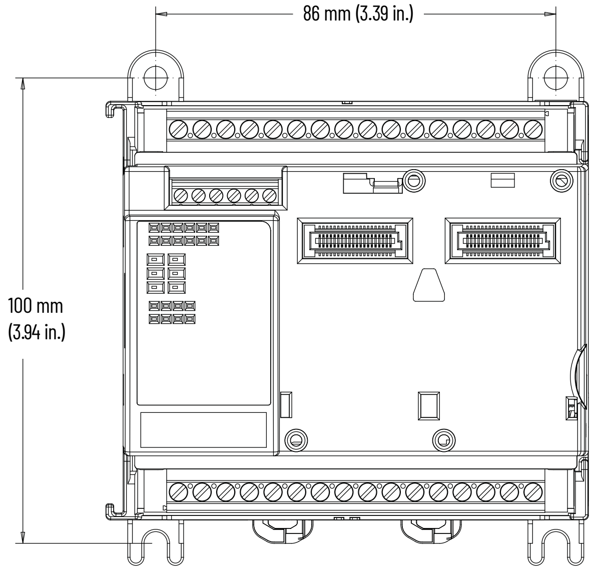
The preferred mounting method is to use four M4 (#8) screws per module. Hole spacing tolerance: ±0.4 mm (0.016 in.).

Follow these steps to install your controller using mounting screws.

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

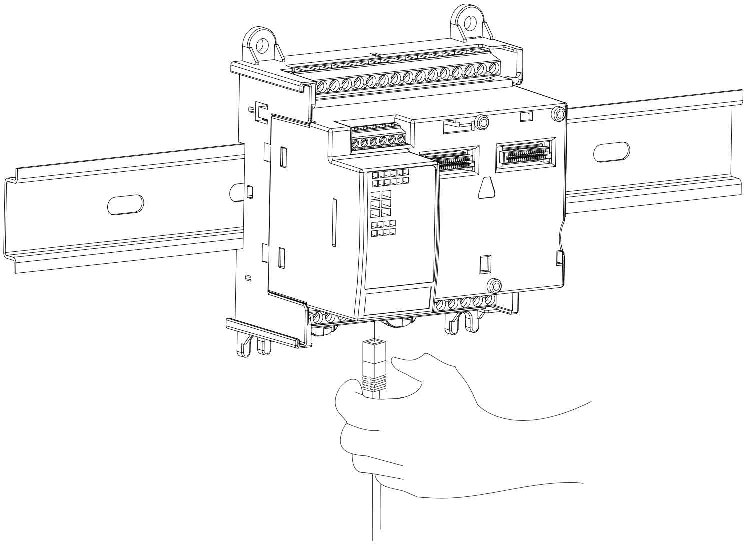
## Panel Mounting Dimensions

**Micro820 20-point controllers**
**2080-LC20-20AWB, 2080-LC20-20QWB, 2080-LC20-20QBB 2080-LC20-20AWBR, 2080-LC20-20QWBR, 2080-LC20-20QBBR**



## Connect the Controller to an EtherNet/IP Network

Connect the RJ45 connector of the Ethernet cable to the Ethernet port on the controller. The port is on the bottom of the controller.

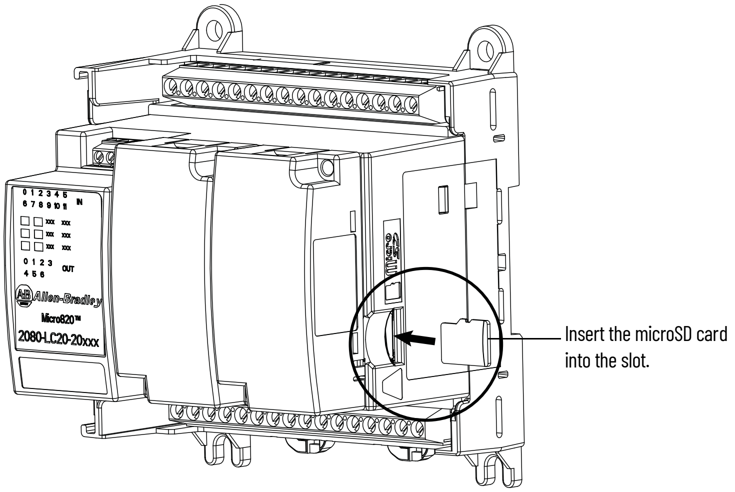


**WARNING:** If you connect or disconnect the communications cable with power applied to this module or any device on the network, an electrical arc can occur. This could cause an explosion in hazardous location installations.
Be sure that power is removed or the area is nonhazardous before proceeding.

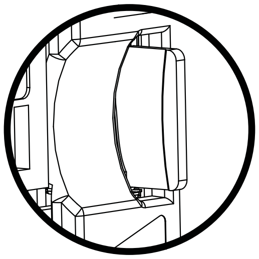
## Install the microSD Card

1. Insert the microSD card into the card slot.

   You can install the microSD card in one orientation only. The beveled corner must be at the bottom. If you feel resistance when you insert the microSD card, pull it out and change the orientation.



2. Gently press the card until it clicks into place.



3. To remove the microSD card from the slot, gently press the card until it clicks back and releases itself from the slot.

## Install the 2080-REMLCD Module

The Micro820 controller supports the 2080-REMLCD module, a simple text display interface for configuring settings such as IP address. It can be mounted through a front panel or on the same DIN rail as the controller.

For information on how the Remote LCD interfaces with the Micro820 controller, see Using the Micro800 Remote LCD on page 69.

To learn about installation, hardware features, and specifications of the 2080-REMLCD module, see the Micro800 Remote LCD Installation Instructions, publication 2080-IN010 in the Literature Library.

# Notes:

# Wire Your Controller

This chapter provides information on the Micro820 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 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.

### Wire Requirements

**Table 7 - Wire Requirements for Fixed Terminal Blocks**

| | Min | Max | |
|---|---|---|---|
| Solid | 0.14 mm$^2$ (26 AWG) | 2.5 mm$^2$ (14 AWG) | Rated @ 90 °C (194 °F) insulation max |
| Stranded | 0.14 mm$^2$ (26 AWG) | 1.5 mm$^2$ (16 AWG) | |

**Table 8 - Wire Requirements for Removable Terminal Blocks**

| | Min | Max | |
|---|---|---|---|
| Solid and stranded | 0.2 mm$^2$ (24 AWG) | 2.5 mm$^2$ (14 AWG) | Rated @ 90 °C (194 °F) insulation max |

**Table 9 - Wire Requirements for RS-232/RS-485 Serial Port Terminal Block**

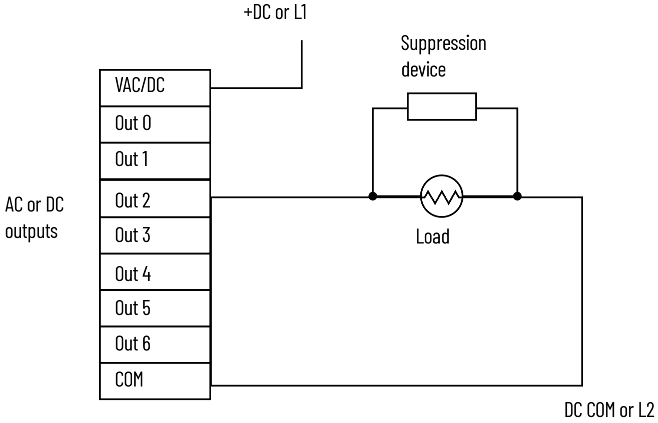
| | Min | Max | |
|---|---|---|---|
| Solid | 0.14 mm$^2$ (26 AWG) | 1.5 mm$^2$ (16 AWG) | Rated @ 90 °C (194 °F) insulation max |
| Stranded | 0.14 mm$^2$ (26 AWG) | 1.0 mm$^2$ (18 AWG) | |

## 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.

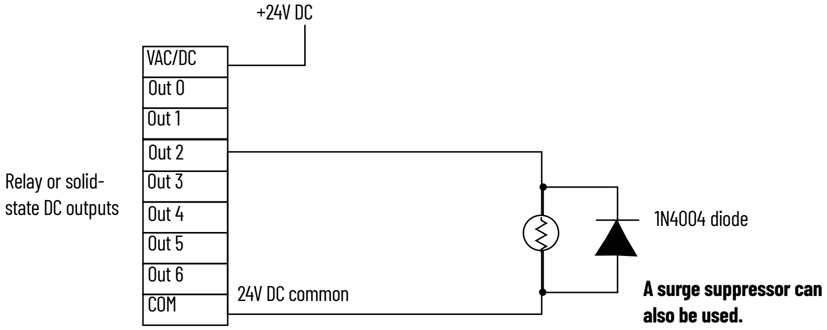
Figure 20 shows an output with a suppression device. We recommend that you locate the suppression device as close as possible to the load device.

**Figure 20 - Output with Suppression Device**



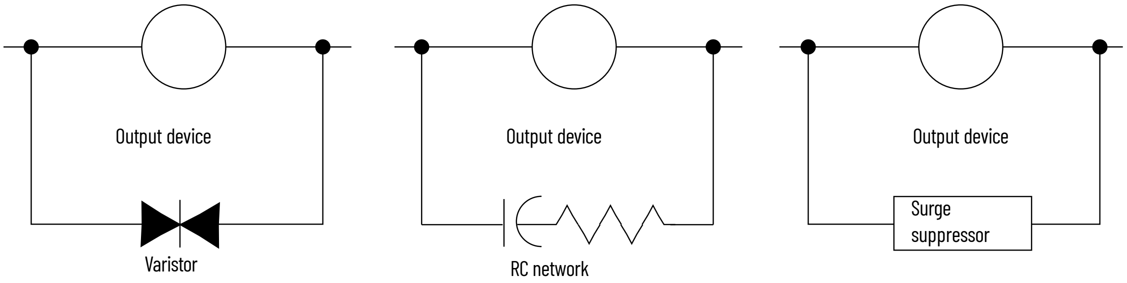
If the outputs are DC, we recommend that you use an 1N4004 diode for surge suppression, as shown in Figure 21. For inductive DC load devices, a diode is suitable. A 1N4004 diode is acceptable for most applications. A surge suppressor can also be used. See Recommended Surge Suppressors on page 35. These surge suppression circuits connect directly across the load device.

**Figure 21 - 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, all shown below. These components must be appropriately rated to suppress the switching transient characteristic of the particular inductive device. See Recommended Surge Suppressors on page 35 for recommended suppressors.

**Figure 22 - Surge Suppression for Inductive AC Load Devices**



## Recommended Surge Suppressors

Use the Allen-Bradley surge suppressors in Figure 10 for use 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…250V 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: Cat. No. 100-FSC48 becomes Cat. No. 100-**CR**FSC48; Cat. No. 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. Refer to the Industrial Automation Wiring and Grounding Guidelines, publication 1770-4.1, for additional information.

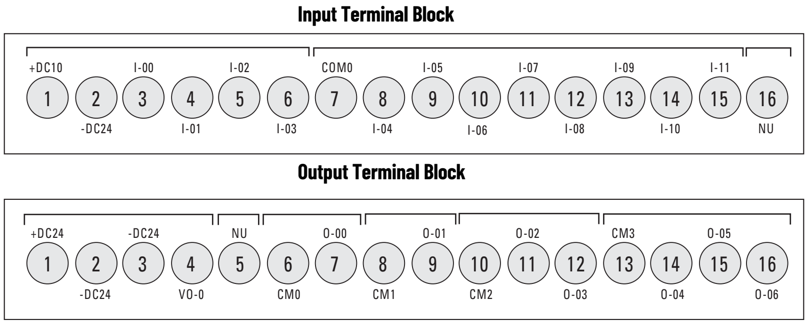
⚠ **WARNING:** All devices connected to the RS-232/RS-485 communication port must be referenced to controller ground, or be floating (not referenced to a potential other than ground). Failure to follow this procedure may result in property damage or personal injury.

## Wiring Diagrams

Figure 23 to Figure 31 show the wiring diagrams for the Micro820 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 ○.

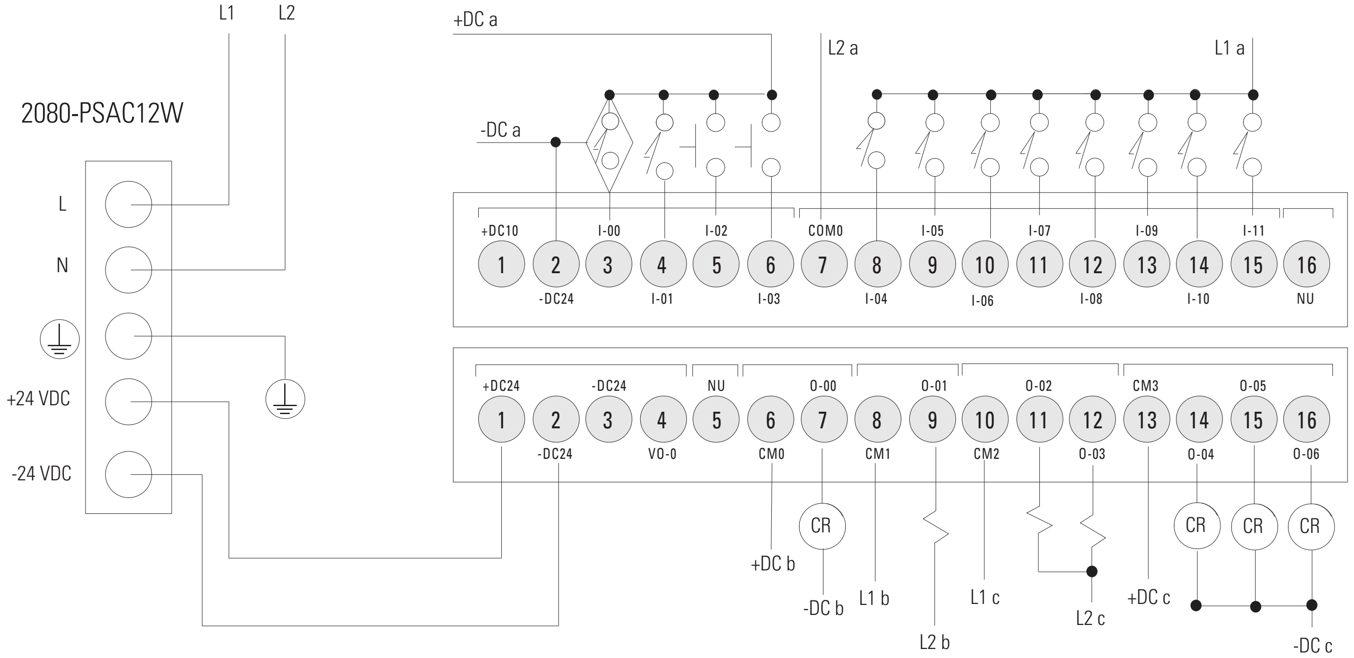
**Figure 23 - 2080-LC20-20AWB, 2080-LC20-20QWB, 2080-LC20-20AWBR, 2080-LC20- 20QWBR**



**IMPORTANT** Note the following:

- The "-DC24" terminals on the Input (terminal 2) and Output (terminals 2 and 3) Terminal Blocks are internally shorted.
- "NU" means that the terminal is not used / no connection.
- Inputs I-00, I-01, I-02, and I-03 are shared between digital and analog inputs.
- Inputs I-00, I-01, I-02, and I-03 can only be used in sinking input configuration.
- I-00 to I-03 are non-isolated input channels, do not connect -DC24 (Input terminal 2) to Earth/Chassis ground.
- Do not connect -DC24 (Output terminal 2) to Earth/Chassis ground.

**Figure 24 - 2080-LC20-20AWB, 2080-LC20-20AWBR — Sinking Input Configuration**



**ATTENTION:** For 2080-LC20-20AWB/R catalogs, inputs 00…03 are limited to 24V DC. All other inputs (04…11) are limited to 120V AC.

Table 11 - Digital Input

| Controller | Terminal Number | Input Common Terminal Label | Terminal Number | Input Terminal Label |
|---|---|---|---|---|
| 2080-LC20-20AWB,<br>2080-LC20-20AWBR | 2 | "-DC24" (24V DC sink only) | 3 | I-00 |
| | | | 4 | I-01 |
| | | | 5 | I-02 |
| | | | 6 | I-03 |
| | 7 | CM0 (120V AC) | 8 | I-04 |
| | | | 9 | I-05 |
| | | | 10 | I-06 |
| | | | 11 | I-07 |
| | | | 12 | I-08 |
| | | | 13 | I-09 |
| | | | 14 | I-10 |
| | | | 15 | I-11 |

Table 12 - Digital Output

| Controller | Terminal Number | Input Common Terminal Label | Terminal Number | Input Terminal Label |
|---|---|---|---|---|
| 2080-LC20-20AWB,<br>2080-LC20-20AWBR | 6 | CM0 (VAC/DC) | 7 | 0-00 |
| | 8 | CM1 (VAC/DC) | 9 | 0-01 |
| | 10 | CM2 (VAC/DC) | 11, 12 | 0-02, 0-03 |
| | 13 | CM3 (VAC/DC) | 14, 15, 16 | 0-04, 0-05, 0-06 |

Figure 25 - 2080-LC20-20QWB, 2080-LC20-20QWBR — DC Sinking Input Configuration - Inputs 00…11

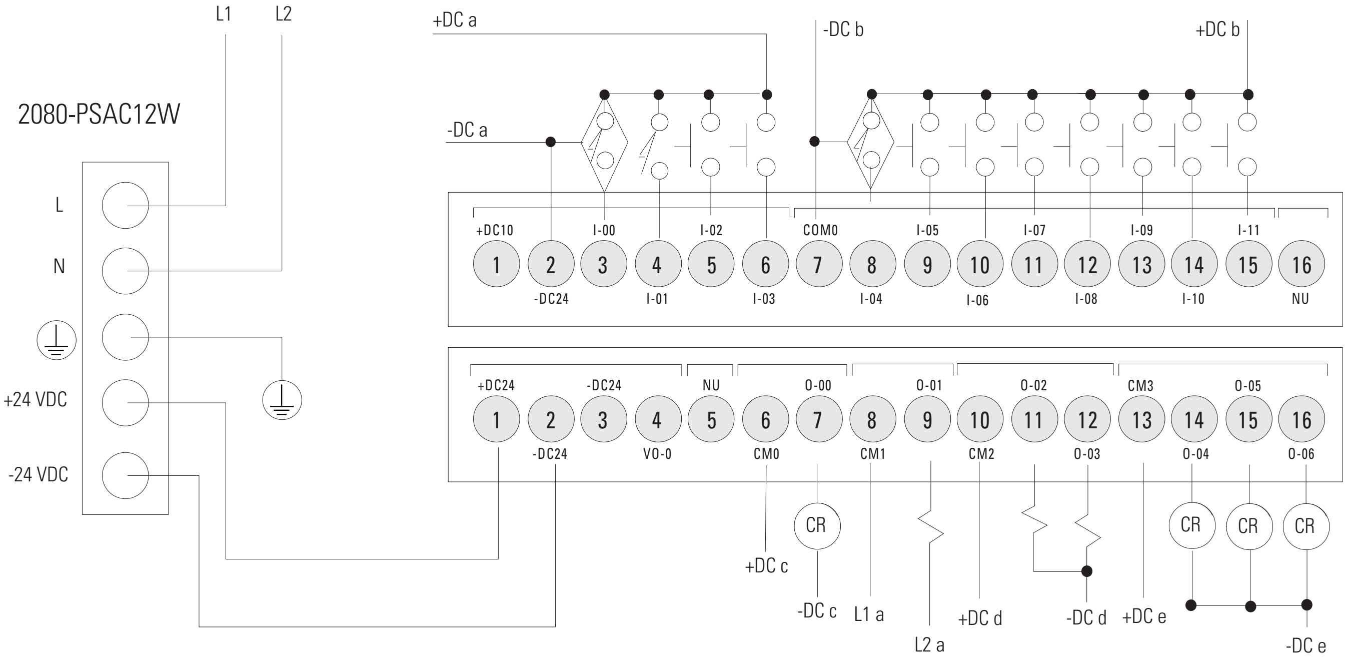


Figure 26 - DC Sourcing Input Configuration – Inputs 4…11

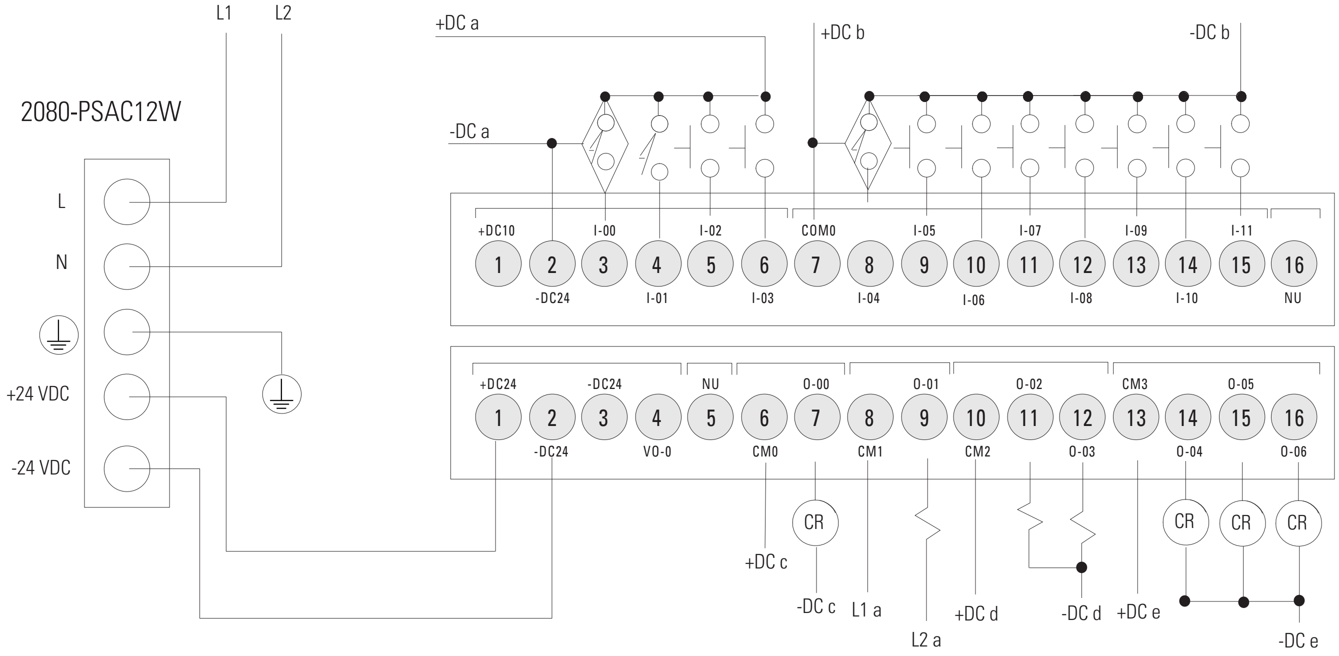


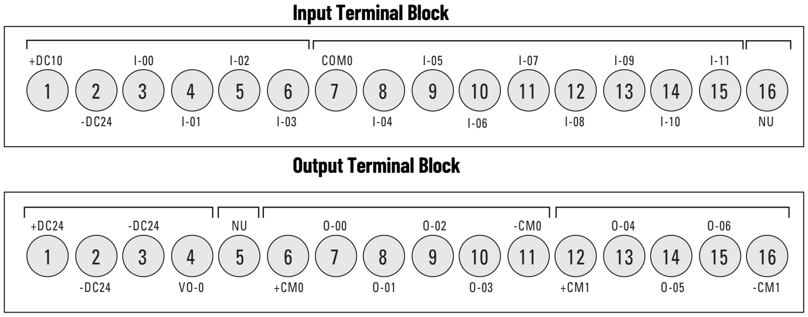
Table 13 - Digital Input

| Controller | Terminal Number | Input Common Terminal Label | Terminal Number | Input Terminal Label |
|---|---|---|---|---|
| 2080-LC20-20QWB,<br>2080-LC20-20QWBR | 2 | "-DC24" (24V DC sink only) | 3 | I-00 |
| | | | 4 | I-01 |
| | | | 5 | I-02 |
| | | | 6 | I-03 |
| | 7 | CM0 (24V DC sink/source) | 8 | I-04 |
| | | | 9 | I-05 |
| | | | 10 | I-06 |
| | | | 11 | I-07 |
| | | | 12 | I-08 |
| | | | 13 | I-09 |
| | | | 14 | I-10 |
| | | | 15 | I-11 |

Table 14 - Digital Output

| Controller | Terminal Number | Input Common Terminal Label | Terminal Number | Input Terminal Label |
|---|---|---|---|---|
| 2080-LC20-20QWB,<br>2080-LC20-20QWBR | 6 | CM0 (VAC/DC) | 7 | 0-00 |
| | 8 | CM1 (VAC/DC) | 9 | 0-01 |
| | 10 | CM2 (VAC/DC) | 11, 12 | 0-02, 0-03 |
| | 13 | CM3 (VAC/DC) | 14, 15, 16 | 0-04, 0-05, 0-06 |

Figure 27 - 2080-LC20-20QBB / 2080-LC20-20QBBR



**IMPORTANT** Note the following:

- The "-DC24" terminals on the Input (terminal 2) and Output (terminals 2 and 3) Terminal Blocks are internally shorted.
- "NU" means that the terminal is not used / no connection.
- Inputs I-00, I-01, I-02, and I-03 are shared between digital and analog inputs.
- Inputs I-00, I-01, I-02, and I-03 can only be used in sinking input configuration.
- I-00 to I-03 are non-isolated input channels, do not connect -DC24 (Input terminal 2) to Earth/Chassis ground.
- Do not connect -DC24 (Output terminal 2) to Earth/Chassis ground.

Figure 28 - DC Sinking Input Configuration – Inputs 00…11

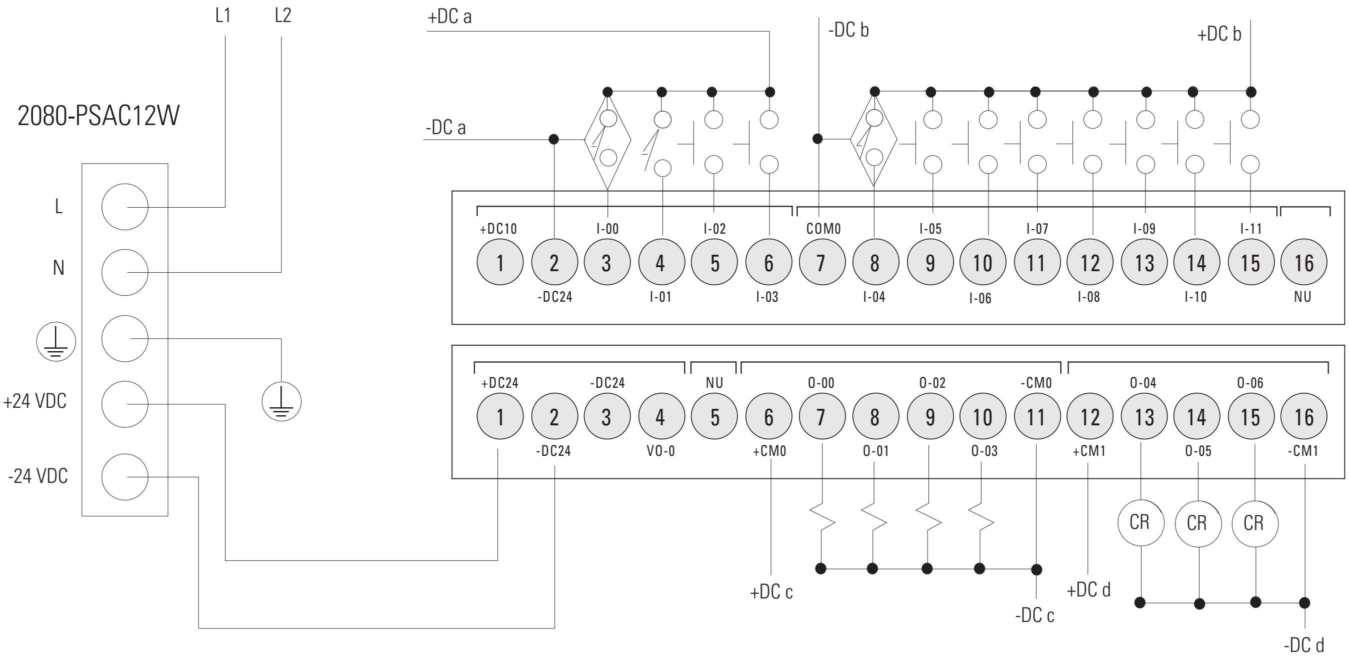


Figure 29 - DC Sourcing Input Configuration - Inputs 4...11

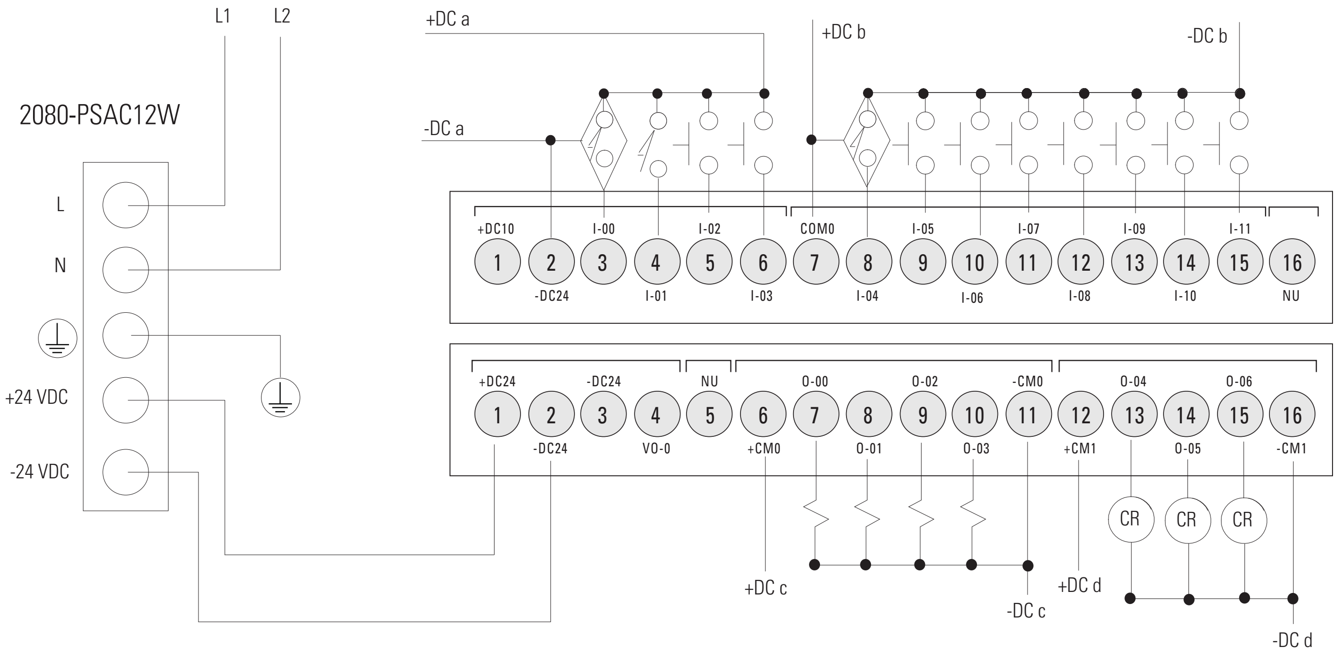


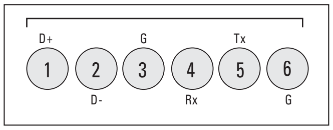
Table 15 - Digital Input

| Controller | Terminal Number | Input Common Terminal Label | Terminal Number | Input Terminal Label |
|---|---|---|---|---|
| 2080-LC20-20QBB,<br>2080-LC20-20QBBR | 2 | "-DC24" (24V DC sink only) | 3 | I-00 |
| | | | 4 | I-01 |
| | | | 5 | I-02 |
| | | | 6 | I-03 |
| | 7 | CM0 (24V DC sink/source) | 8 | I-04 |
| | | | 9 | I-05 |
| | | | 10 | I-06 |
| | | | 11 | I-07 |
| | | | 12 | I-08 |
| | | | 13 | I-09 |
| | | | 14 | I-10 |
| | | | 15 | I-11 |

Table 16 - Digital Output

| Controller | Terminal Number | Input Common Terminal Label | Terminal Number | Input Terminal Label |
|---|---|---|---|---|
| 2080-LC20-20QBB,<br>2080-LC20-20QBBR | 6 | +CM0 (VDC source) | 7, 8, 9, 10 | O-00, O-01, O-02, O-03 |
| | 11 | -CM0 | | |
| | 12 | +CM1 (VDC source) | 13, 14, 15 | O-04, O-05, O-06 |
| | 16 | -CM1 | | |

Figure 30 - Serial Port Terminal Block

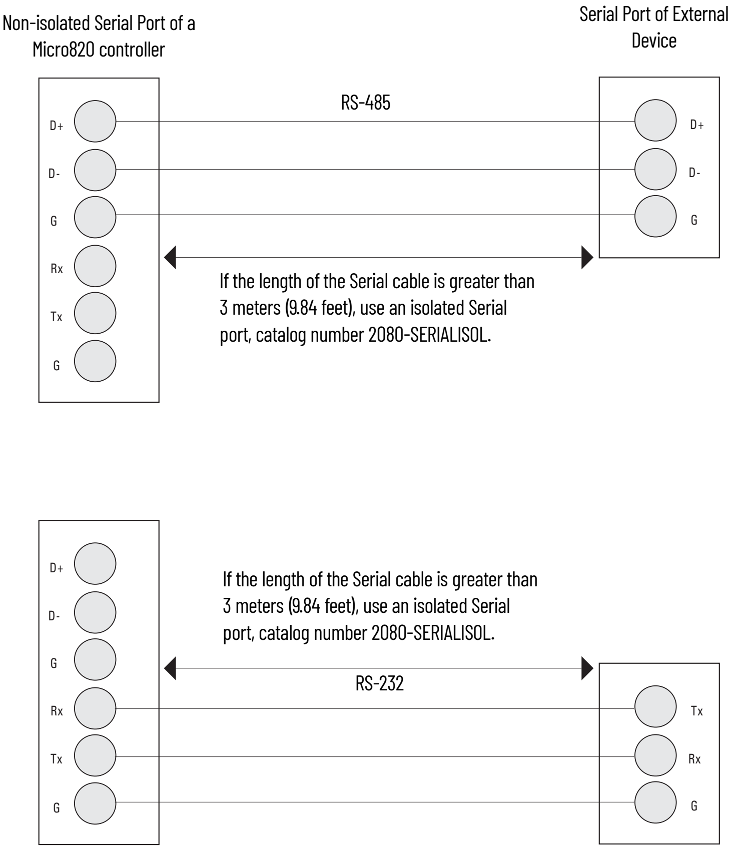


(View into terminal block)

| | | |
|---|---|---|
| Pin 1 | RS-485 | Data + |
| Pin 2 | RS-485 | Data - |
| Pin 3 | RS-485 | Ground[1] |
| Pin 4 | RS-232 | Receive |
| Pin 5 | RS-232 | Transmit |
| Pin 6 | RS-232 | Ground[1] |

[1] Non-isolated.

Figure 31 - Serial Port Wiring


Non-isolated Serial Port of a Micro820 controller
Serial Port of External Device
RS-485
D+
D-
G
Rx
Tx
G
D+
D-
G
If the length of the Serial cable is greater than 3 meters (9.84 feet), use an isolated Serial port, catalog number 2080-SERIALISOL.
D+
D-
G
Rx
Tx
G
If the length of the Serial cable is greater than 3 meters (9.84 feet), use an isolated Serial port, catalog number 2080-SERIALISOL.
RS-232
Tx
Rx
G

**IMPORTANT** Do not connect G terminals of the Serial port to Earth/Chassis ground.

# 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 ensure that all environmental noise will be removed by 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 periodically checking 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 (-DC24) 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 ensure 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

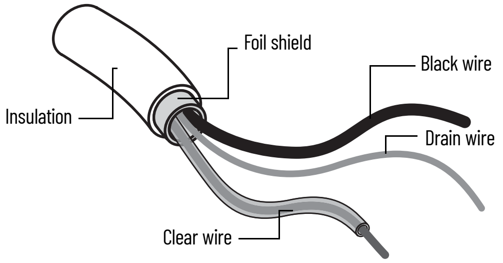
Inputs on analog channels employ digital high-frequency filters that significantly reduce the effects of electrical noise on input signals. However, because of the variety of applications and environments where analog controllers are installed and operated, it is impossible to ensure that all environmental noise will be removed by the 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/IP. 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 earth grounded.
- Route the Belden cable separately from any AC wiring. Additional noise immunity can be obtained by routing the cables in 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 in Figure 32, Figure 33, and Figure 34.

Figure 32 - Sink Input Wiring Example

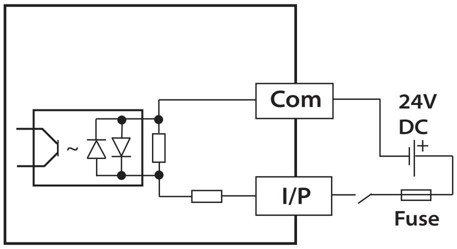
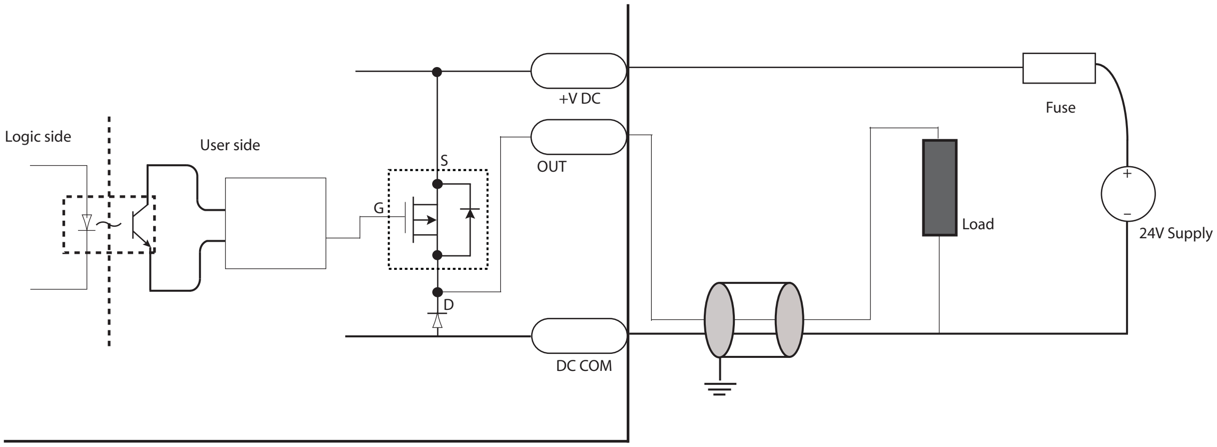
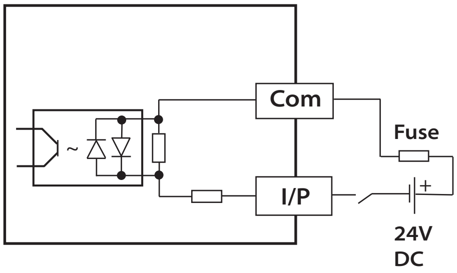


Figure 33 - Source Output Wiring Example



| IMPORTANT | For 2080-LC20-20QBB(R) discrete output 06, shielded cable is required if the output is used as PWM. Otherwise, unshielded cable can be used. |
|---|---|

Figure 34 - Source Input Wiring Example

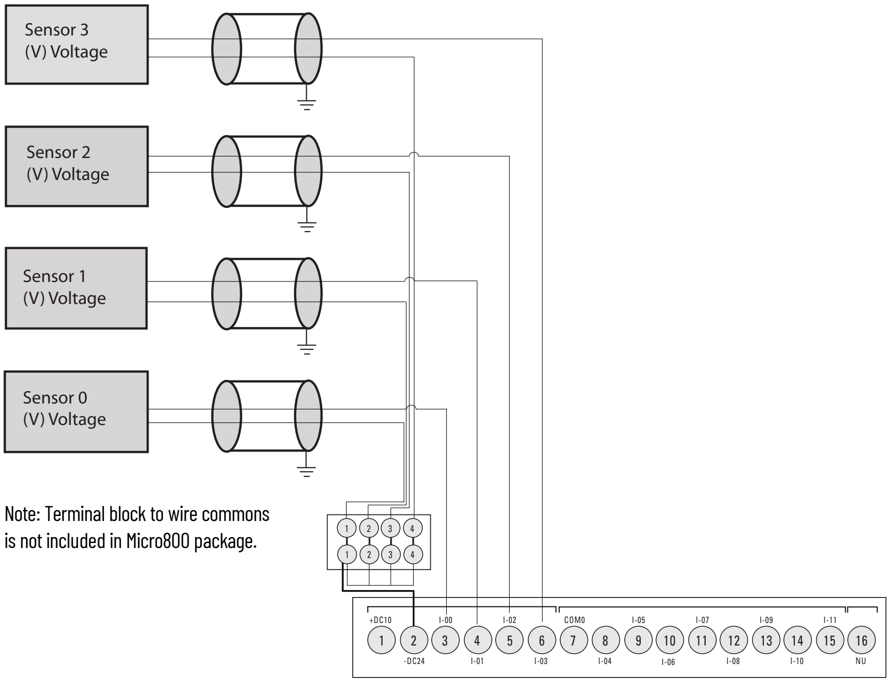


## Wiring Analog Channels

Analog input circuits can monitor voltage signals and convert them to Serial digital data as shown in Figure 35 and Figure 36.

**ATTENTION:** Analog inputs and outputs are not isolated.

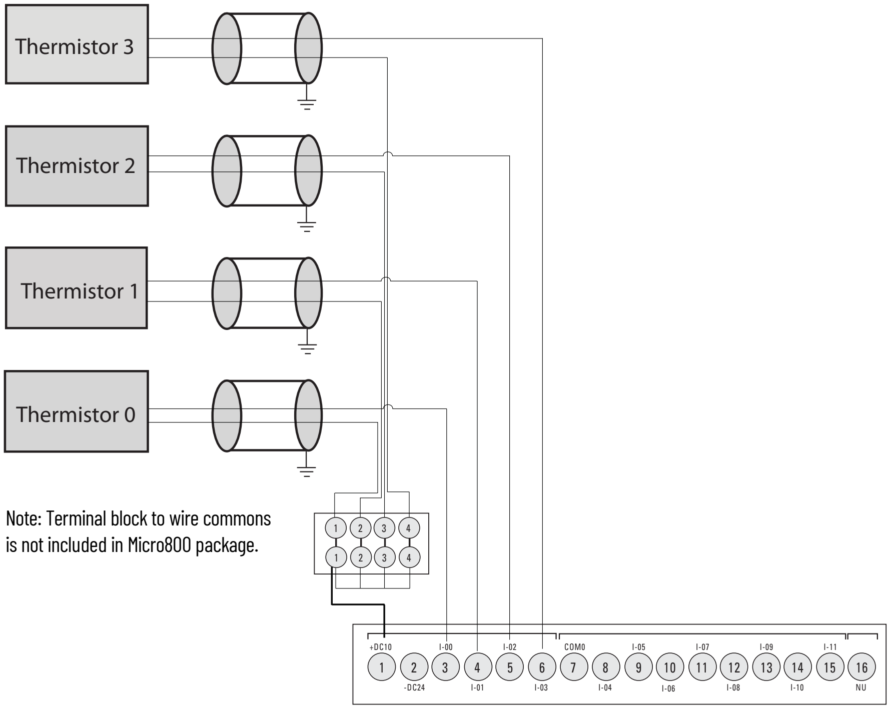
**Figure 35 - Analog Input to Sensors**



Note: Terminal block to wire commons is not included in Micro800 package.

The “-DC24” terminal is the analog ground connection for analog inputs (I-00…I-03).

**Figure 36 - Analog Input to Thermistors**



Note: Terminal block to wire commons is not included in Micro800 package.

The “+DC10” terminal supplies 10V DC power source to the Thermistor inputs (I-00…I-03).

### *Calculate for Thermistor Resistance*

While connecting Analog input to thermistor as shown in previous diagram, calculate input voltage using the following equation:

$$V_i = \frac{R_i}{R_i + R_t} * V_{ref}$$

To calculate for thermistor resistance, use the following equation:

$$R_t = \frac{R_i V_{ref} - V_i R_i}{V_i}$$

*Where*:
$V_i$ = Voltage input (±5% without calibration; ±2% with calibration)
$R_i$ = Resistance input (14.14 kΩ ±2%)
$R_t$ = Thermistor resistance (10 kΩ Thermistor is recommended)
$V_{ref}$ = 10V ±0.5V

**IMPORTANT** Micro820 controllers support 10 kΩ type thermistors.
In order to get the best results, the system must be calibrated.

To convert this resistance into a temperature, use the following Steinhart-Hart equation:

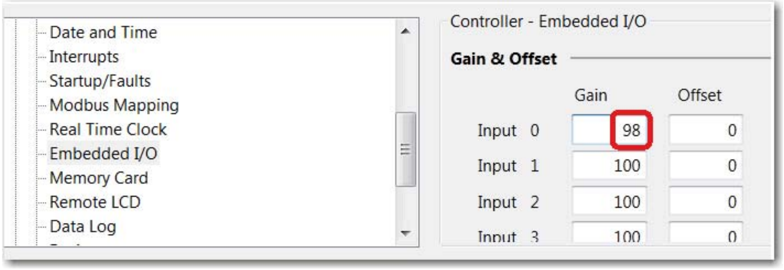
$$\frac{1}{T} = a + b\ln(R) + c[\ln(R)]^3$$

*Where*:
Coefficients a, b, and c are provided by the thermistor manufacturer.

### *Calibrate Thermistor*

1. Connect a resistor (10 kΩ is recommended) across Vref and Analog Input 00 of your Micro820 controller following the diagram, Analog Input to Thermistors on page 44. The resistor is measured as Ri using a precision multimeter.
2. Calculate the ideal counts (C1) for resistor (Ri) following this equation: C1 = 14.14 kΩ / (14.14 kΩ + Ri) * 4095
3. Read the actual counts (C2) of Analog Input 00 from Connected Components Workbench software.
4. Calculate for calibration Gain.
   Gain = C1/C2

   *For example*:
   If Ri is measured as 10.00 kΩ, then
   C1 = 14.14 / (14.14 + 10.00) * 4095 = 2399 counts;
   C2 is read from Connected Components Workbench software as 2440; so
   Gain = 2399/2440 = 98%.
5. In Connected Components Workbench software, go to Embedded I/O configuration page. Change the Gain parameter value to 98 for Input 0.



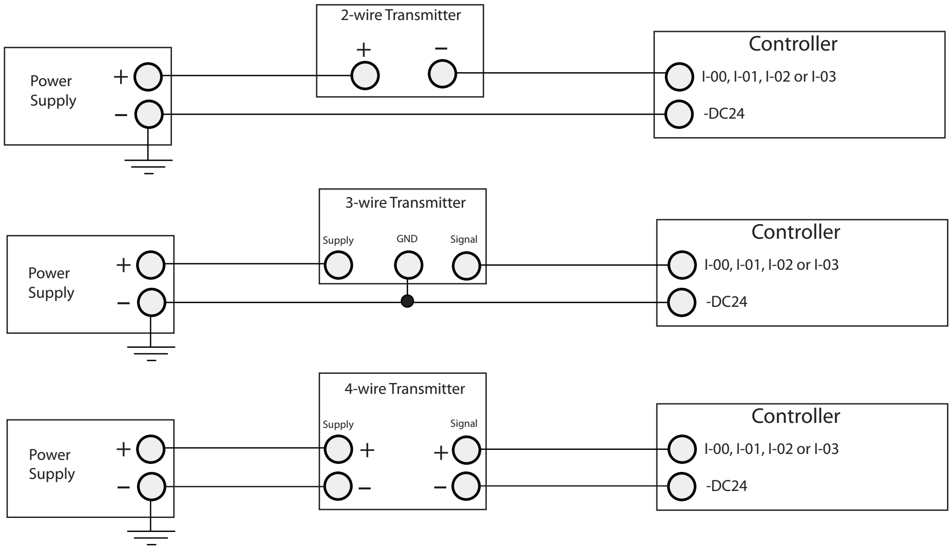
Controller - Embedded I/O

**Gain & Offset**

| | Gain | Offset |
|---|---|---|
| Input 0 | 98 | 0 |
| Input 1 | 100 | 0 |
| Input 2 | 100 | 0 |
| Input 3 | 100 | 0 |

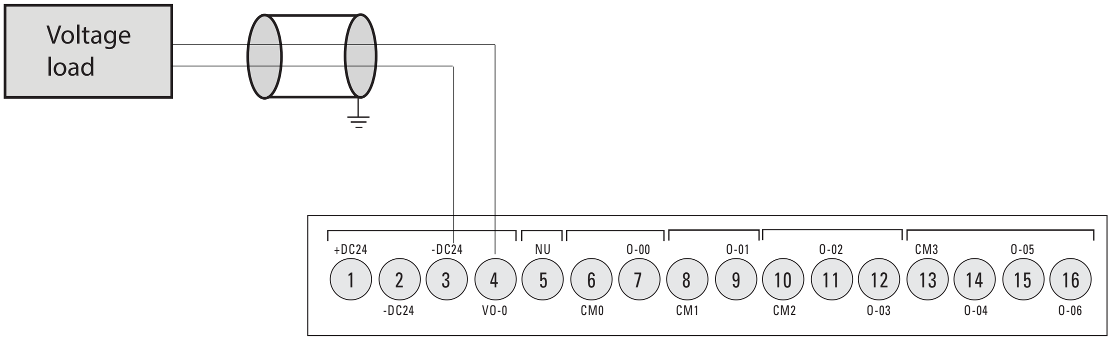
6. Repeat the same steps to calibrate all the other analog input channels.

### Transmitters to Analog Input



### Analog Output

The analog output can support voltage function as shown in the following illustration.



The "-DC24" terminal is the analog ground connection for analog output (VO-0).

# 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 Micro820 controllers have the following embedded communication channels:

- A non-isolated RS-232/RS-485 combo port
- RJ45 Ethernet port

## Supported Communication Protocols

Micro820 controllers support communication through the embedded RS-232/RS-485 Serial port and any installed Serial port plug-in module. In addition, Micro820 controllers also 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.

Micro820 controllers support the following communication protocols:

- Modbus RTU Master and Slave
- CIP Serial Client/Server (RS-232 only)
- ASCII
- EtherNet/IP Client/Server
- Modbus TCP Client/Server
- CIP Symbolic Client/Server
- DHCP Client
- Sockets Client/Server TCP/UDP

**Table 17 - Connection Limits for Micro820 Controllers**

| Description | | Connections |
|---|---|---|
| **CIP Connections** | | |
| Total number of client plus server connections | | 24 |
| Maximum number of client connections for all ports | | 16 |
| Maximum number of server connections for all ports | | 24 |
| Maximum number of EtherNet/IP connections | Client | 16 |
| | Server | 24 |
| Maximum number of USB connections | Client | – |
| | Server | |
| Maximum number of Serial connections | Client | 16 |
| | Server | 24 |
| **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<br>(any combination of UDP plus TCP Client/Server) | | 8 |

The 2080-REMLCD USB uses the CIP Serial server connections for embedded Serial port.

| 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 based on the limits described in Table 17:

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 how to configure your Micro800 controller for Modbus protocol, refer to the Connected Components Workbench Online Help. For more information about the Modbus protocol, see Modbus Protocol Specifications available from www.modbus.org.

For information on Modbus mapping, see Modbus Mapping for Micro800 on page 175.

To configure the Serial port as Modbus RTU, see Configure Modbus RTU on page 54.

Use MSG_MODBUS instruction to send Modbus messages over the Serial port.

## CIP Serial Client/Server – RS-232 only

CIP Serial Client/Server allows CIP protocol to be used over an RS-232 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 bar code readers, weigh scales, Serial printers, and other intelligent devices. You can use ASCII by configuring the embedded or any plug-in Serial RS-232 or 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 Configure ASCII on page 55.

## Modbus TCP Client/Server

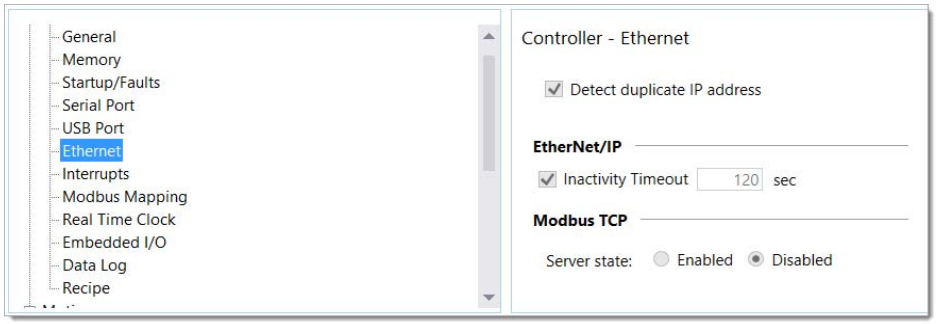
The Modbus TCP Client/Server communication protocol uses the same Modbus mapping features as Modbus RTU, but instead of the Serial port, it is supported over Ethernet. Modbus TCP Server takes on Modbus Slave features on Ethernet.

The Micro820 controller supports up to 16 simultaneous Modbus TCP Client connections and 24 simultaneous Modbus TCP Server connections.

No protocol configuration is required other than configuring the Modbus mapping table. For information on Modbus mapping, see Modbus Mapping for Micro800 on page 175.

Use MSG_MODBUS2 instruction to send Modbus TCP message 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.



Controller - Ethernet

☑ Detect duplicate IP address

EtherNet/IP

☑ Inactivity Timeout 120 sec

Modbus TCP

Server state: ○ Enabled ● Disabled

## 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 easily connect to the Micro820 controller.

CIP Serial, supported on Micro820 controllers, uses the DF1 Full-duplex protocol, which provides point-to-point connection between two devices.

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.

EtherNet/IP, supported on Micro820 controllers, uses the standard Ethernet TCP/IP protocol.

The Micro820 controllers support up to 16 simultaneous EtherNet/IP Client connections and 24 simultaneous EtherNet/IP Server connections.

To configure CIP Serial, see Configure CIP Serial Driver on page 53.

To configure for EtherNet/IP, see Configure Ethernet Settings on page 56.

### *CIP Symbolic Addressing*

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

One-dimension 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. Array of strings are also supported.

#### Supported Data Types in CIP Symbolic

| Data Type[1] | Description |
|---|---|
| BOOL | Logical Boolean with values TRUE(1) and FALSE(0)<br>(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 |

### Supported Data Types in CIP Symbolic

| Data Type[1] | Description |
|---|---|
| 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 HMI 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 quickstart projects to help you use the CIP Client Messaging feature, see Micro800 Programmable Controllers: Getting Started with CIP Client Messaging Quick Start, publication 2080-QS002.

## Sockets Client/Server TCP/UDP

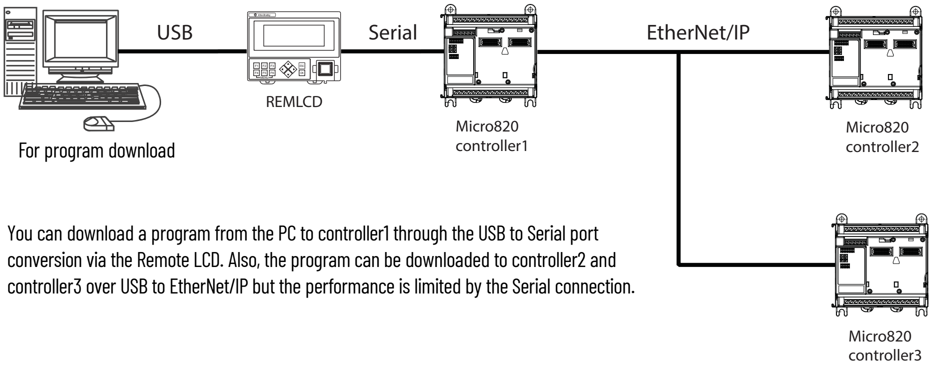
Sockets protocol is used for Ethernet communications to devices that do not support Modbus TCP and EtherNet/IP. Sockets support client and server, and TCP and UDP. Typical applications include communicating to printers, barcode readers, and PCs.

# CIP Communications Pass-thru

The Micro820 controllers support pass-thru on any communications port that supports CIP for applications such as program download. It does not support applications that require dedicated connections such as HMI. Micro820 controllers support a maximum of one hop. A hop is defined as an intermediate connection or communications link between two devices – in Micro800, this is through EtherNet/IP or CIP Serial.

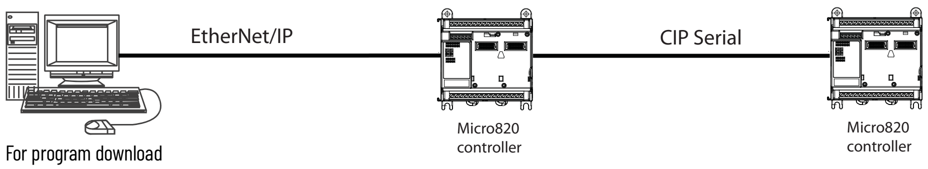
## Examples of Supported Architectures

### CIP Serial to EtherNet/IP

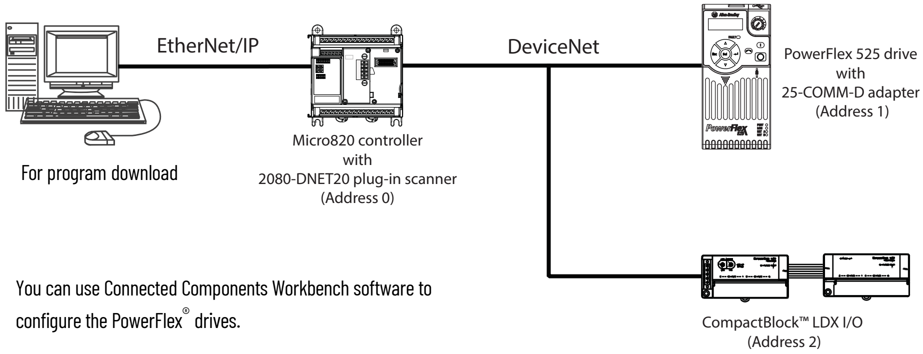


You can download a program from the PC to controller1 through the USB to Serial port conversion via the Remote LCD. Also, the program can be downloaded to controller2 and controller3 over USB to EtherNet/IP but the performance is limited by the Serial connection.

### EtherNet/IP to CIP Serial



### EtherNet/IP to DeviceNet®


EtherNet/IP
DeviceNet
For program download
Micro820 controller with 2080-DNET20 plug-in scanner (Address 0)
PowerFlex 525 drive with 25-COMM-D adapter (Address 1)
CompactBlock™ LDX I/O (Address 2)

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 Micro820 controllers.

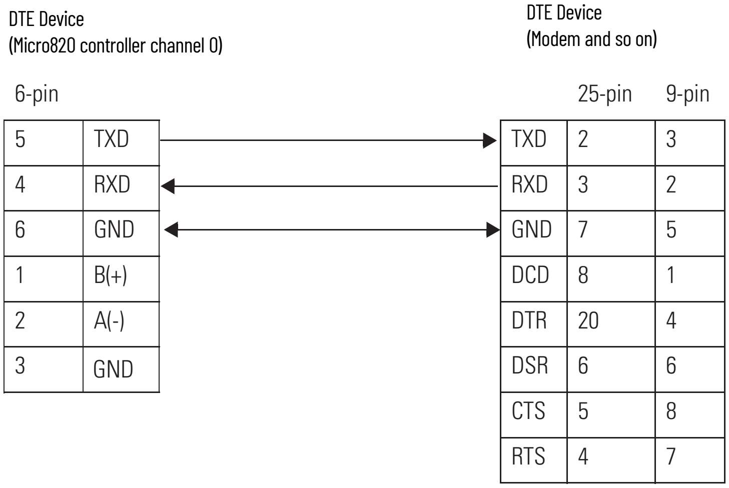
### Making a DF1 Point-to-Point Connection

You can connect the Micro820 controller to your Serial modem. The recommended protocol for this is Modbus RTU.

### Construct Your Own Modem Cable

If you construct your own modem cable, the maximum cable length is 3 m (10 ft) with a 25-pin or 9-pin connector. Figure 37 shows the typical pinout for constructing a straight-through cable.

**Figure 37 - Straight-through Cable Pinout Guide**



DTE Device
(Micro820 controller channel 0)

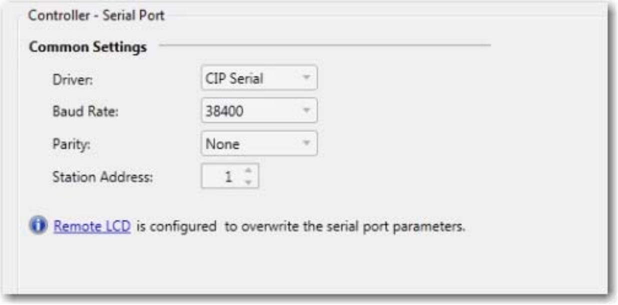
DTE Device
(Modem and so on)

| 6-pin | | | | 25-pin | 9-pin |
|---|---|---|---|---|---|
| 5 | TXD | → | TXD | 2 | 3 |
| 4 | RXD | ← | RXD | 3 | 2 |
| 6 | GND | ↔ | GND | 7 | 5 |
| 1 | B(+) | | DCD | 8 | 1 |
| 2 | A(-) | | DTR | 20 | 4 |
| 3 | GND | | DSR | 6 | 6 |
| | | | CTS | 5 | 8 |
| | | | RTS | 4 | 7 |

## Configure Serial Port

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

By default, when a Micro820 controller is added to the Project Organizer in Connected Components Workbench, Remote LCD parameters are configured to overwrite Serial Port settings.



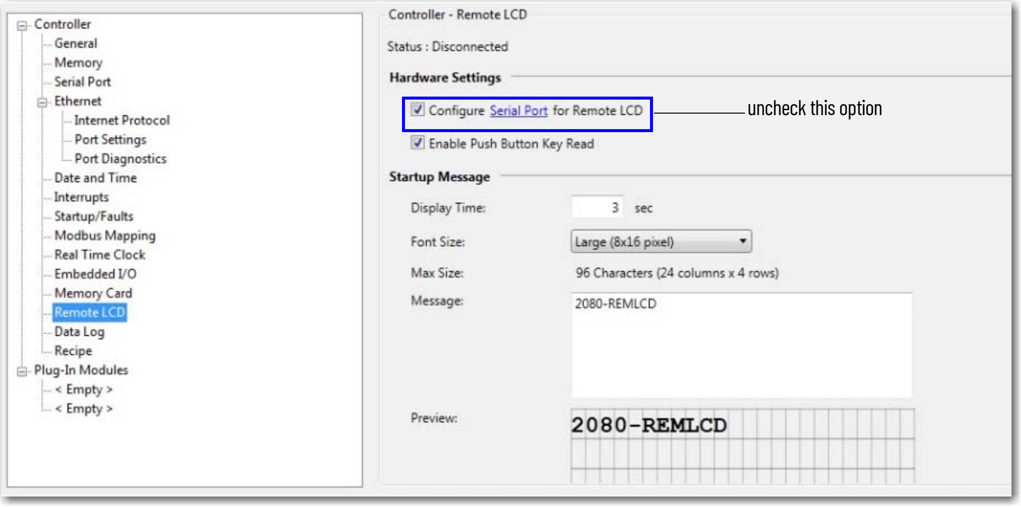
Controller - Serial Port

Common Settings

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

Remote LCD is configured to overwrite the serial port parameters.

To edit Serial Port settings, go to the Remote LCD configuration page and uncheck the Configure Serial Port for Remote LCD option button.



Controller - Remote LCD

Status : Disconnected

Hardware Settings

☑ Configure Serial Port for Remote LCD

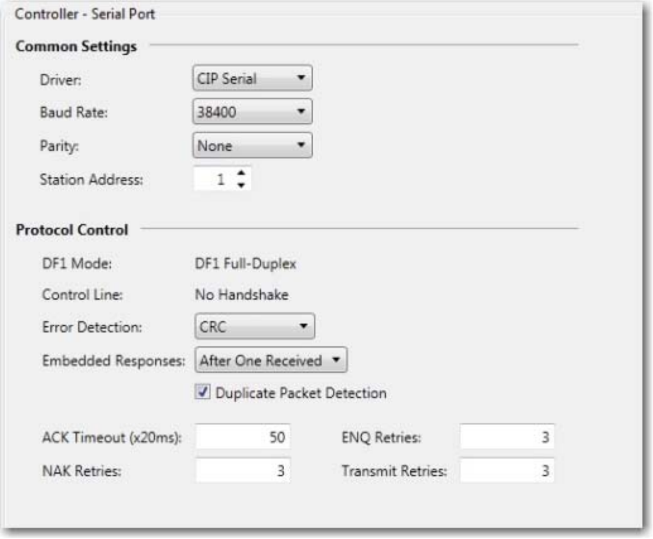
uncheck this option

☑ Enable Push Button Key Read

Startup Message

Display Time: 3 sec
Font Size: Large (8x16 pixel)
Max Size: 96 Characters (24 columns x 4 rows)
Message: 2080-REMLCD
Preview: 2080-REMLCD

When the Remote LCD configuration is unchecked, the Serial Port values are visible and can be edited.



Controller - Serial Port

Common Settings

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

Protocol Control

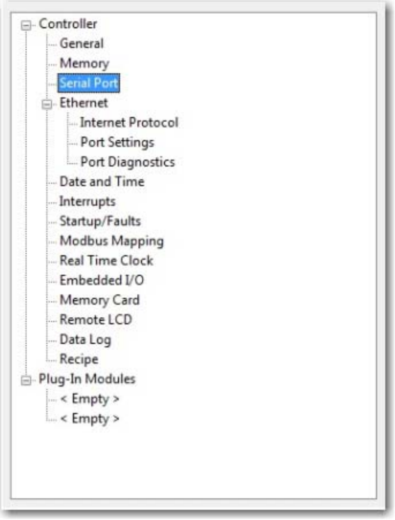
DF1 Mode: DF1 Full-Duplex
Control Line: No Handshake
Error Detection: CRC
Embedded Responses: After One Received
☑ Duplicate Packet Detection

ACK Timeout (x20ms): 50 ENQ Retries: 3
NAK Retries: 3 Transmit Retries: 3

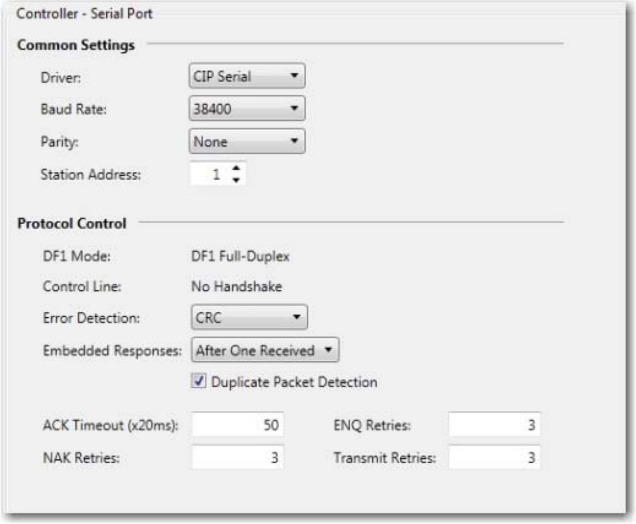
| | |
|---|---|
| **IMPORTANT** | After changing the Serial Port settings on the controller, power cycle the Remote LCD. |

## Configure CIP Serial Driver

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



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: DF1 Full-Duplex
Control Line: No Handshake
Error Detection: CRC
Embedded Responses: After One Received
☑ Duplicate Packet Detection

ACK Timeout (x20ms): 50
ENQ Retries: 3
NAK Retries: 3
Transmit 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 should be left at default settings.
5. Select Advanced Settings and set Advanced parameters. See Table 18 for a description of the CIP Serial parameters.

**Table 18 - CIP Serial Driver Parameters**

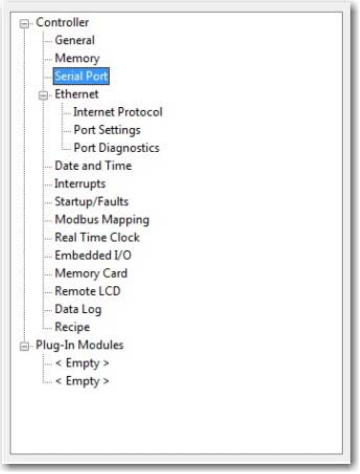
| Parameter | Options | Default |
|---|---|---|
| Baud Rate | Toggles between the communication rate of 1200, 2400, 4800, 9600, 19200, and 38400 | 38400 |
| Parity | Specifies the parity setting for the Serial port. Parity provides additional message-packet error detection. Select Even, Odd, or None. | None |
| Station Address | The station address for the Serial port on the DF1 master. The only valid address is 0…254. | 1 |
| DF1 Mode | DF1 Full-duplex (read only) | Configured as Full-duplex by default |
| Control Line | No Handshake (read only) | Configured as no handshake by default |

**Table 18 - CIP Serial Driver Parameters (Continued)**

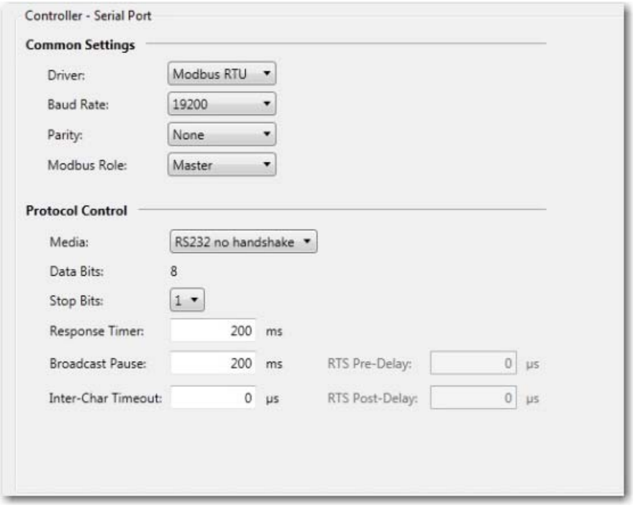
| Parameter | Options | Default |
|---|---|---|
| Duplicate Packet Detection | Detects and eliminates duplicate responses to a message. Duplicate packets may be sent under noisy communication conditions when the sender's retries are not set to 0. Toggles between Enabled and Disabled. | Enabled |
| Error 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.<br>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 will resend a message packet because the processor 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 |

## 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.



Controller - Serial Port

Common Settings

Driver: Modbus RTU

Baud Rate: 19200

Parity: None

Modbus Role: Master

Protocol Control

Media: RS232 no handshake

Data Bits: 8

Stop Bits: 1

Response Timer: 200 ms

Broadcast Pause: 200 ms

RTS Pre-Delay: 0 µs

Inter-Char Timeout: 0 µs

RTS Post-Delay: 0 µs

3. Specify the following parameters:
    - Baud rate
    - Parity
    - Unit address
    - Modbus Role (Master, Slave, Auto)

**Table 19 - 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 |

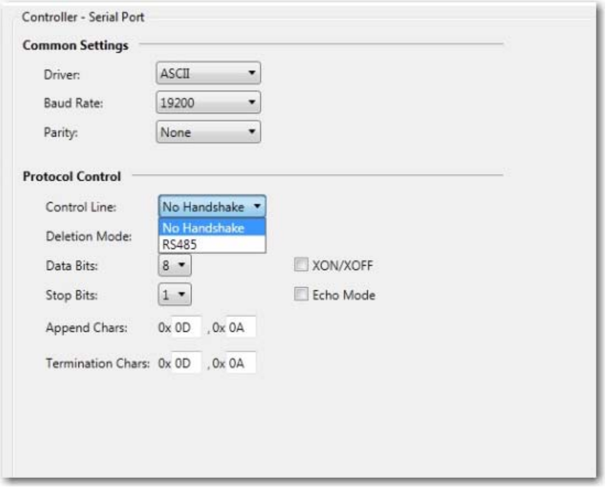
4. You can configure additional parameters under Advanced Settings.

**Table 20 - 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. Select Serial Port.
2. Select ASCII on the Driver field.



Controller - Serial Port

Common Settings

Driver: ASCII

Baud Rate: 19200

Parity: None

Protocol Control

Control Line: No Handshake

Deletion Mode:

Data Bits: 8

Stop Bits: 1

□ XON/XOFF

□ Echo Mode

Append Chars: 0x 0D , 0x 0A

Termination Chars: 0x 0D , 0x 0A

3. Specify the following parameters:
    - Baud Rate
    - Parity

**Table 21 - ASCII Parameters**

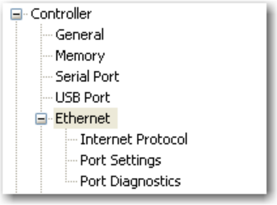
| Parameter | Options | Default |
|---|---|---|
| Baud Rate | 1200, 2400, 4800, 9600, 19200, 38400 | 19200 |
| Parity | None, Odd, Even | None |
| Control Line | RS-485<br>No Handshake | No Handshake |

Table 21 - ASCII Parameters (Continued)

| Parameter | Options | Default |
|---|---|---|
| Deletion Mode | CRT<br>Ignore<br>Printer | Ignore |
| Data Bits | 7, 8 | 8 |
| XON/XOFF | Enabled or Disabled | Disabled |
| Stop Bits | 1, 2 | 1 |
| Echo Mode | Enabled or Disabled | Disabled |
| Append Chars | 0x0D, 0x0A, or user-specified value | 0x0D, 0x0A |
| Termination Chars | 0x0D, 0x0A, or user-specified value | 0x0D, 0x0A |

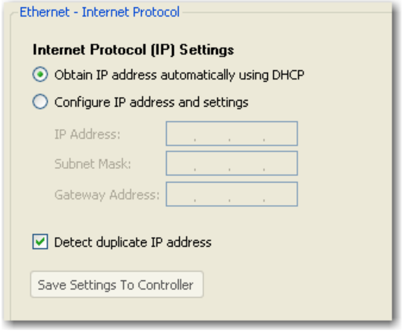
## Configure Ethernet Settings

1. Open your Connected Components Workbench project (for example, Micro820). On the device configuration tree, go to Controller properties. Select Ethernet.



```
⊟ Controller
    General
    Memory
    Serial Port
    USB Port
  ⊟ Ethernet
      Internet Protocol
      Port Settings
      Port Diagnostics
```

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.



Ethernet - Internet Protocol

**Internet Protocol (IP) Settings**

● Obtain IP address automatically using DHCP

○ Configure IP address and settings

IP Address: . . .

Subnet Mask: . . .

Gateway Address: . . .

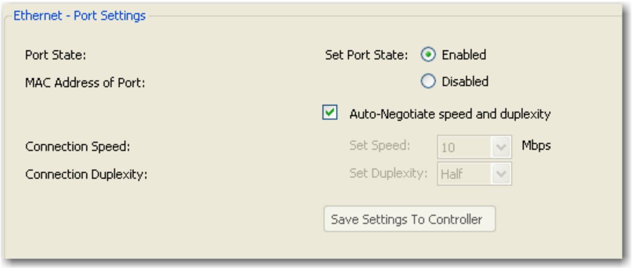
☑ Detect duplicate IP address

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 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.



Ethernet - Port Settings

Port State:

MAC Address of Port:

Set Port State: ● Enabled ○ Disabled

☑ Auto-Negotiate speed and duplexity

Connection Speed:

Connection Duplexity:

Set Speed: 10 Mbps

Set Duplexity: Half

5. Set Port State as Enabled or Disabled.
6. To set connection speed and duplexity manually, clear the checkbox Auto-Negotiate speed and duplexity. Then, set Speed (10 or 100 Mbps) and Duplexity (Half or Full) values.
7. Select Save Settings to Controller if you want 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 be not 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 address exclude:

- Broadcast or zero IP (255.255.255.255 or 0.0.0.0)
- IP address starting with 0 or 127 (0.xxx.xxx.xxx or 127.xxx.xxx.xxx)
- IP address ending with 0 or 255 (xxx.xxx.xxx.0 or xxx.xxx.xxx.255)
- IP addresses in range 169.254.xxx.xxx (169.254.0.0 to 169.254.255.255)
- IP addresses in range 224.0.0.0 to 255.255.255.255

### Ethernet Host Name

Micro800 controllers implement unique host names for each controller, 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: 2080LC20-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.

## OPC Support Using FactoryTalk Linx

Support for Open Platform Communications (OPC) using CIP symbolic is added from firmware revision 7.0 onwards. You can use this instead of Modbus addressing.

FactoryTalk® Linx software version 5.70 (CPR9 SR7) or later and FactoryTalk® Linx Gateway software version 3.70 (CPR9 SPR7) or later are required.

## Notes:

# Program Execution in Micro800

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

| | |
|---|---|
| **IMPORTANT** | This section generally describes program execution in Micro800 controllers. Certain elements may not be applicable or true in 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 General Instructions Reference 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 (datalog, 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 software, 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 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 just 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 set point interval (1…65535 ms).

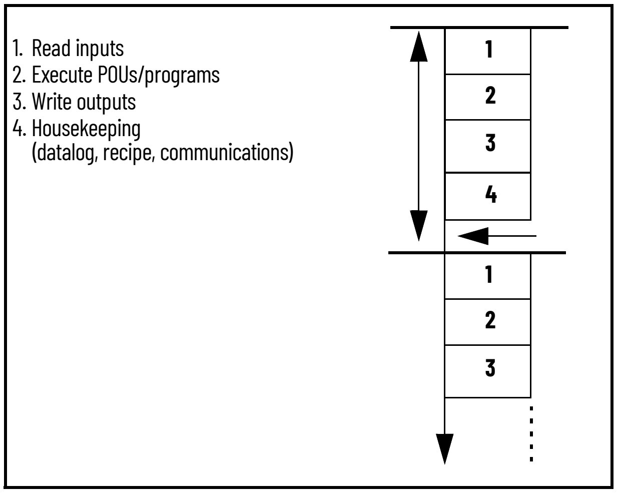
The Global System Variables associated with cycles/scans are:

- __SYSVA_CYCLECNT – Cycle counter
- __SYSVA_TCYCURRENT – Current cycle time
- __SYSVA_TCYMAXIMUM – Maximum cycle time since last start

## Execution Rules

This section illustrates the execution of a program. The execution follows four main steps within a loop. The loop duration is a cycle time for a program.

**Figure 38 - Program Execution Steps**



When you specify a cycle time, 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 you do not specify a cycle time, a resource performs all steps in the loop then restarts a new cycle without waiting.

## Optional Module

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 38) could be interrupted by other controller activities which 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 will be 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.

## Periodic Execution of Programs

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

We do not recommend that you use the system variable __SYSVA_TCYCYCTIME 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 22 - System Variable for Programmed Cycle Time

| Variable | Type | Description |
|---|---|---|
| __SYSVA_TCYCYCTIME | TIME | Programmed cycle time.<br>**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 available.

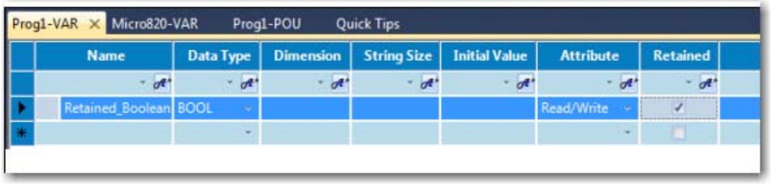
Table 23 - System Variables for Scan and Power-up

| Variable | Type | Description |
|---|---|---|
| _SYSVA_FIRST_SCAN | BOOL | First scan bit.<br>Can be used to initialize or reset variables immediately after every transition from Program to Run mode.<br>**Note**: True only on first scan. After that, it is false. |
| _SYSVA_POWER_UP_BIT | BOOL | Power-up bit.<br>Can be used to initialize or reset variables immediately after download from Connected Components Workbench software or immediately after being loaded from memory backup module (for example, microSD card).<br>**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. Micro820 controllers can only retain a maximum of 400 bytes of user-created variable values.

For example: A user-created variable 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 variables are retained. You can choose which variables to retain by selecting them on the global variable page.



| Name | Data Type | Dimension | String Size | Initial Value | Attribute | Retained |
|---|---|---|---|---|---|---|
| Retained_Boolean | BOOL | | | | Read/Write | ✓ |
| | | | | | | ☐ |

## Memory Allocation

Depending on the base size, available memory on Micro800 controllers is shown in Table 24.

Table 24 - Memory Allocation for Micro800 Controllers

| Attribute | 10/16-point (Micro830®) | 20-point (Micro820) | 24-point and 48-point (Micro830, 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 project is created for Micro800 controllers, 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 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 a lot of initial and project values, it may cause power up time to exceed 30 seconds. After boot up, EtherNet/IP connections may take up to 60 seconds to establish.

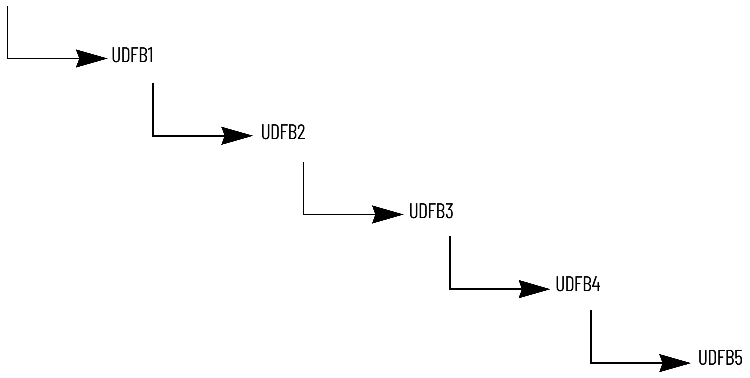
The Micro800 controllers also have project memory, which stores a copy of the entire downloaded project (including comments), as well as configuration memory for storing plug-in setup information, and so on.

## 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. It is recommended that you split large programs into smaller programs to improve code readability, simplify debugging and maintenance tasks.
- A UDF uses significantly less memory than a 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 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 may result in a compile error. This also applies to UDFs.

Example of Five Nested UDFBs



- 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, then 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% less 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:

Chapter 7

# Controller Security

Micro800 security generally has two components:

- **Exclusive Access** that prevents simultaneous configuration of the controller by two users
- **Controller Password Protection** that secures the Intellectual Property contained within the controller and prevents unauthorized access

## Operation Mode

To maintain the secure operation of your Micro800 controllers, operations that can disrupt controller operations are restricted based on the controller operation mode.

**Table 25 - 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 |

[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.

## 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 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. When a Connected Components Workbench user connects to a Micro800 controller, the controller is given exclusive access to that controller.

## Password Protection

By setting a password on the controller, a user effectively restricts access to the programming software connection of the controller to software sessions that can supply the correct password. Essentially, Connected Components Workbench operations such as upload and download are prevented if the controller is secured with a password and the correct password is not provided.

Micro800 controllers are shipped with no password but a password can be set through the Connected Components Workbench software (using controller firmware revision 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 version 10 or later.

The controller password is also backed up to the memory backup module (that is, 2080-MEMBAK-RTC2 module for Micro850® and Micro870® controllers, 2080-LCD module for Micro810® controllers, and microSD card for Micro820 controllers).

For instructions on how to set, change, and clear controller passwords, see Configure Controller Password on page 125.

## Compatibility

The Controller Password feature is supported on:

- Connected Components Workbench software **version 2** and later
- Micro800 controllers with at least **revision 2** firmware

For users with earlier versions of the software and/or hardware, refer to the compatibility scenarios below.

### *Connected Components Workbench software version 1 with Micro800 controller firmware revision 2 and later*

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 will be successful. However, the software will not be able to determine whether the controller is locked or not.

If the controller is not locked, access to the user application will be allowed, provided the controller is not busy with another session. If the controller is locked, access to the user application will fail. Users will need to upgrade to version 2 of the Connected Components Workbench software.

### *Connected Components Workbench software version 2 and later with Micro800 controller firmware revision 1*

Connected Components Workbench software version 2 is capable of "discovering" and connecting to Micro800 controllers with firmware revision earlier than revision 2 (that is, not supporting the Controller Password feature). However, the Controller Password feature will not be available to these controllers. The user will not be able see interfaces associated with the Controller Password feature in the Connected Components Workbench session.

Users are advised to upgrade the firmware. See Update Your Micro800 Controller Firmware on page 117 for instructions.

**ATTENTION:** Connected Components Workbench software version 9 or earlier with Micro800 controller revision 10 or later.
If a Micro800 controller with firmware revision 10 or later is locked using the new password algorithm introduced in Connected Components Workbench software version 10 or later, it cannot be accessed using 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 selecting 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 selecting 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. Select Connect.
3. Select the target controller.
4. When requested, provide the controller password.
5. Build and save the project, if needed.
6. Select Download.
7. Select Disconnect.

**IMPORTANT** If the controller has a password locked version 10 or later project, you cannot access the controller using Connected 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 will be automatically converted to the old algorithm.

**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 will be automatically converted to the new algorithm.

**IMPORTANT** If communication is lost during the download, repeat the download and verify that the controller is password protected.

## Transfer Controller Program and Lock Receiving Controller

In this scenario, the user needs to transfer 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 will be locked.

1. In the Project Organizer, select 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. Select Disconnect.
6. Power down controller1.
7. Swap controller1 hardware with controller2 hardware.
8. Power up controller2.
9. Select Connect.
10. Select target controller2.

11. Select Download.
12. Lock controller2. See Configure Controller Password on page 125.

### Back Up and Restore a Password-Protected Controller

In this workflow, user application will be backed up from a Micro800 controller that is locked to a memory plug-in device.

1. In the Project Organizer, select 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 will fail 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 flash update the controller using the Reset option to clear the password before restoring the project to the controller.

## Configure Controller Password

To set, change, and clear controller password, see the quickstart instructions Configure Controller Password on page 125.

## Recover from a Lost Password

If the controller is secured with a password and the password has been lost, then it is 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 Micro850 and Micro870 controllers, or the 2080-REMLCD module for Micro820 controllers. Then, ControlFLASH™ software 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 will be lost but a new project can be downloaded.

# Using the Micro800 Remote LCD

This chapter provides a description of how you can use the Micro800 Remote LCD with the Micro820 controller.

## Overview

The 2080-REMLCD module serves as a simple IP65 text display that allows the configuration of such controller settings as IP address. It connects to the Micro820 controller through the RS-232 port. The Remote LCD module has a dot matrix LCD with backlight and supports multilingual characters. The display size is 3.5 inches with 192 x 64 pixel resolution.

It also has:

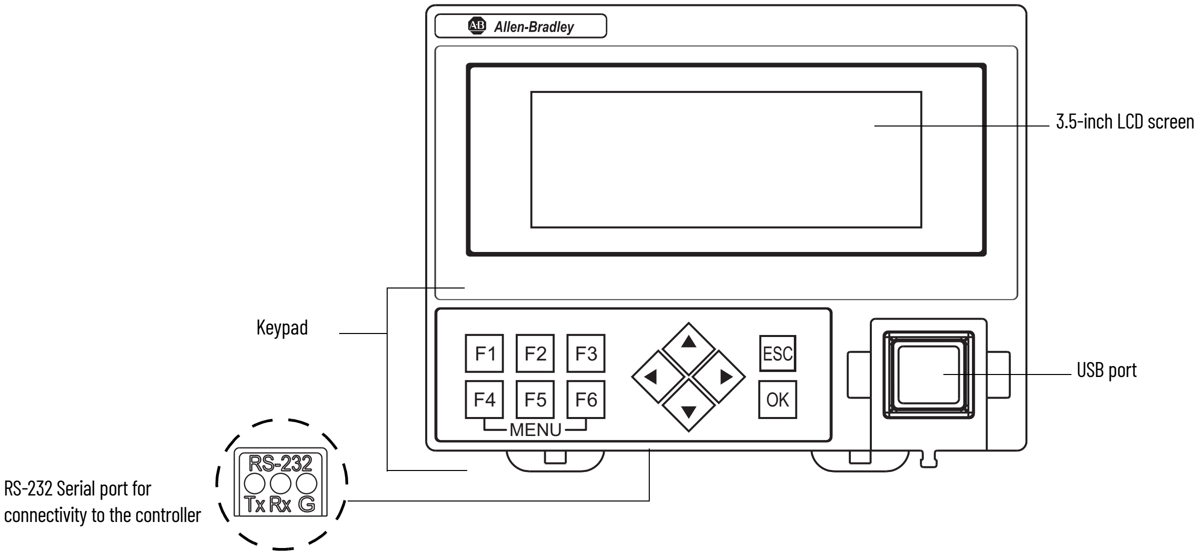
- Four arrow keys
- Six function keys
- ESC key
- OK key
- USB port for Connected Components Workbench connectivity

It supports:

- Small character set: 24 characters by 8 lines
- Large character set: 24 characters by 4 lines
- Extra large character set: displays 12 characters by 4 lines

The Remote LCD module supports English, French, Spanish, Italian, and Simplified Chinese languages for the Main Menu.

**Micro800 Remote LCD**



The Remote LCD module is IP65-rated and can be mounted through the front panel or on the same DIN rail as the Micro820 controller.

It has two modes of operation:

- USB Mode
- Text Display Mode

- I/O Status and Main Menu operations (for example, change to Run mode)
- Optional user-defined screens (using the LCD_REM instructions)

## USB Mode

In USB mode, the Remote LCD module acts as a USB pass-through for Connected Components Workbench software. The Remote LCD module automatically enters USB mode when traffic is detected.

For example:

1. Remote LCD is in text display mode showing the I/O Status screen by default.
2. The user connects a USB cable between the PC and the Remote LCD.
3. Remote LCD is automatically detected by the PC as a USB device and the Remote LCD automatically goes to USB mode.
4. I/O Status screen is no longer shown. The user is now able to download program over USB using Connected Components Workbench software.
5. When the USB cable is disconnected and no traffic is detected for 30 seconds, the Remote LCD automatically goes back to text display mode showing the I/O Status screen.

| IMPORTANT | Using the USB port is convenient when accessing the controller from the front of the cabinet without opening the door and when the IP address is unknown. For larger programs, it is recommended to use USB port through the Remote LCD to set the IP address and then use Ethernet to download. Ethernet is faster due to limitations of the USB to Serial conversion. |
|---|---|

## Text Display Mode

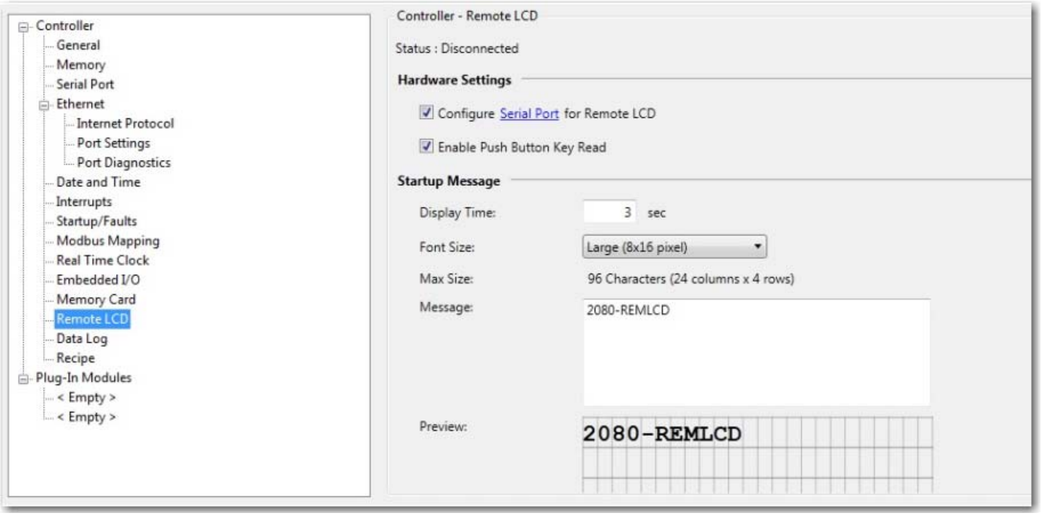
In text display mode, you are either in I/O Status, Main Menu, or executing Remote LCD instructions.

### Startup Screen

Default startup screen

On power-up, the Remote LCD module powers up with a splash screen that displays "Initializing". Then, it displays "Connecting to Controller" until the connection is established. The controller then displays the startup screen for 3 seconds by default or user-defined duration after the connection is established.

You can customize this startup screen in the Connected Components Workbench software. The controller displays the default startup screen at power-up when the customized startup screen is blank.



Controller - Remote LCD

Status : Disconnected

Hardware Settings

☑ Configure Serial Port for Remote LCD

☑ Enable Push Button Key Read

Startup Message

Display Time: 3 sec

Font Size: Large (8x16 pixel)

Max Size: 96 Characters (24 columns x 4 rows)

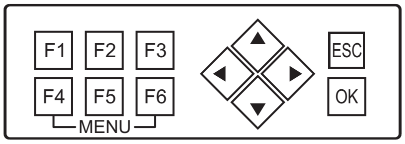
Message: 2080-REMLCD

Preview: 2080-REMLCD

After the startup message, the Remote LCD displays the I/O Status screen, if no LCD_REM instructions are executing.

## Navigate the Remote LCD

In text display mode, you can use the navigation keys (function keys, arrow keys, ESC and OK) to navigate through the menus.



The module has twelve keys with the operations shown in Table 26.

Table 26 - Function Keys Operation

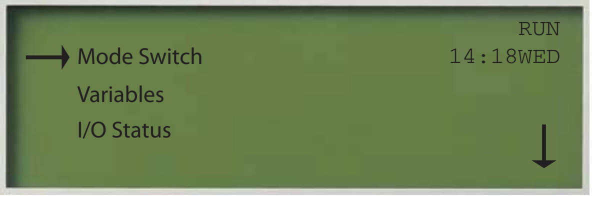
| Button | Function |
|---|---|
| Arrow keys (cursor buttons) | Move cursor |
| | Select menu item |
| | Increment/Decrement number |
| | Choose numbers, values, times, and so on |
| OK | Next menu level, store your entry |
| Esc | Previous menu level, cancel your entry. |
| F1 | Variable (shortcut) |
| F2 | ENET Cfg (shortcut) |
| F3 | Mode Switch (shortcut) |
| F4 | Fault Mode (shortcut) |
| F5 | Security (shortcut) |
| F6 | Backlight (shortcut) |

Shortcut keys jump from the I/O Status screen to the specific main menu operation.

## Main Menu

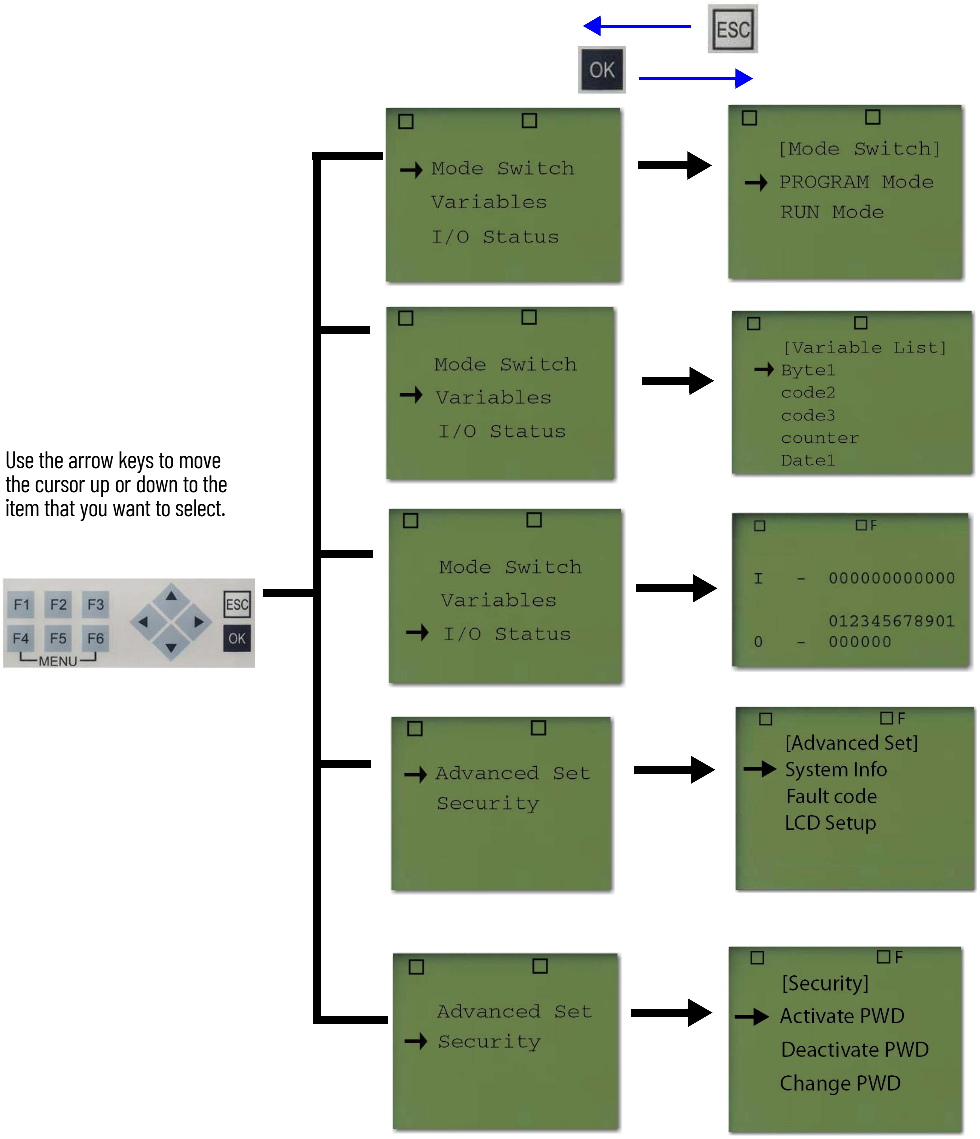
To access the Main Menu and available submenus, press F4 and F6 simultaneously. To exit the Main Menu, press ESC.

The Main Menu shows the following screen:



The structure tree shown in Figure 39 takes you through the different menus available in the Remote LCD module and their general description.

**Figure 39 - 2080-REMLCD Menu Structure Tree**



Use the arrow keys to move the cursor up or down to the item that you want to select.

**Mode Switch**
Set the controller to Program mode or Run mode from this screen.

**Variable**
Monitor or set values for program-defined variables.

**I/O Status**
Monitor the I/O status from this screen.

**Advanced Set**
View:

| | |
|---|---|
| System Info | Analog Calibration |
| Fault Code | PwrUp Behavior |
| LCD Setup | Memory Card |
| Clock Setup | ENET Cfg |
| Language | |

**Security**
Activate, deactivate, and change password.

**Table 27 - Main Menu Items**

| Menu Item | Description |
|---|---|
| I/O Status | Shows the status of the local I/O |
| Mode switch | Change the mode switch selection. |
| Variables | View and change the data value of a variable. Using Connected Components Workbench software, you can specify which variables in the program can be viewed and edited through the 2080-REMLCD module.<br>See View and Edit Variable Values Through the Remote LCD on page 73. |

Table 27 - Main Menu Items (Continued)

| Menu Item | Description | |
|---|---|---|
| Security | Activate, deactivate, and change password protection. | |
| Advanced Set | System Info | View system information such as operating systems series and firmware revision. |
| | Fault Code | View controller fault code information. |
| | LCD Setup | Adjust LCD contrast, backlight color, and push button. |
| | Clock Setup | The real-time clock and daylight saving time |
| | Language | Change menu language to French, Italian, Spanish, and Chinese. |
| | Analog Calibration | Configure calibration parameter of embedded analog inputs. |
| | PwrUp Behavior | Configure controller mode on power-up. |
| | Memory Card | Access the microSD card. |
| | ENET Cfg | View and change the Ethernet port configuration. |

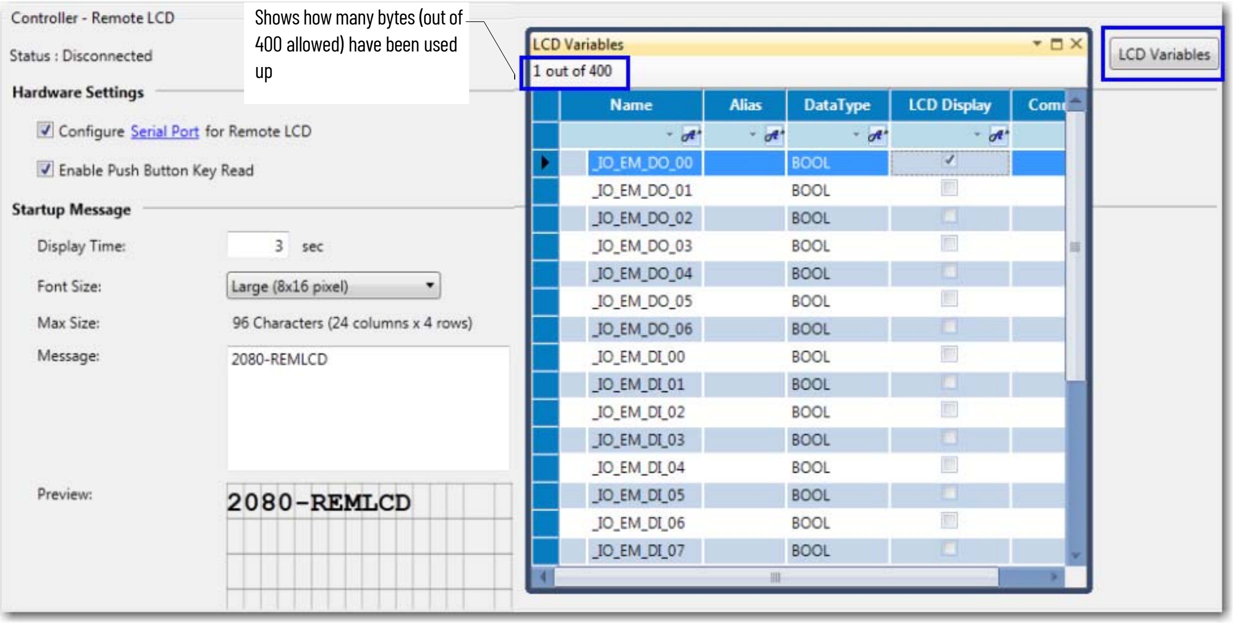
The controller limits certain operations according to controller mode, as shown in Figure 28.

Table 28 - Operational Limit on 2080-REMLCD

| Operation | PROG Mode | Run Mode |
|---|---|---|
| Variable Edit | NO | YES |
| Analog  Calibration | YES | NO |
| Controller > Memory Card | YES | NO |
| Memory Card > Controller | YES | NO |
| Others | YES | YES |

### *View and Edit Variable Values Through the Remote LCD*

Go to the 2080-REMLCD configuration window in the Connected Components Workbench software. Select LCD Variables and select which variables that you want to edit through the Remote LCD.



Shows how many bytes (out of 400 allowed) have been used up

Controller - Remote LCD

Status : Disconnected

Hardware Settings

☑ Configure Serial Port for Remote LCD

☑ Enable Push Button Key Read

Startup Message

Display Time: 3 sec

Font Size: Large (8x16 pixel)

Max Size: 96 Characters (24 columns x 4 rows)

Message: 2080-REMLCD

Preview: 2080-REMLCD

LCD Variables

1 out of 400

| Name | Alias | DataType | LCD Display | Com |
|---|---|---|---|---|
| _IO_EM_DO_00 | | BOOL | ✓ | |
| _IO_EM_DO_01 | | BOOL | ☐ | |
| _IO_EM_DO_02 | | BOOL | ☐ | |
| _IO_EM_DO_03 | | BOOL | ☐ | |
| _IO_EM_DO_04 | | BOOL | ☐ | |
| _IO_EM_DO_05 | | BOOL | ☐ | |
| _IO_EM_DO_06 | | BOOL | ☐ | |
| _IO_EM_DI_00 | | BOOL | ☐ | |
| _IO_EM_DI_01 | | BOOL | ☐ | |
| _IO_EM_DI_02 | | BOOL | ☐ | |
| _IO_EM_DI_03 | | BOOL | ☐ | |
| _IO_EM_DI_04 | | BOOL | ☐ | |
| _IO_EM_DI_05 | | BOOL | ☐ | |
| _IO_EM_DI_06 | | BOOL | ☐ | |
| _IO_EM_DI_07 | | BOOL | ☐ | |

## User-defined Screens

To create user-defined screens through the Connected Components Workbench software, you can program the Remote LCD module using the following function blocks.

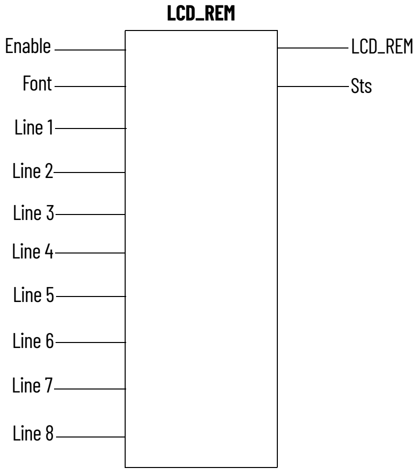
**Table 29 - 2080-REMLCD Function Blocks**

| Function Block Name | Description |
|---|---|
| LCD_REM | Used to display string or numbers on the Remote LCD |
| KEY_READ_REM | Used to read keypad input on the Remote LCD |
| LCD_BKLT_REM | Used to change the backlight color and mode of the Remote LCD screen |

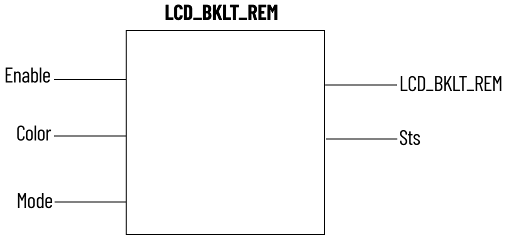
When the instructions are executing, the user-defined screen is shown, but when in the Main Menu, the Remote LCD instructions are disabled. For example, the KEY_READ_REM instruction no longer read keypad input.

### *LCD_REM*

The LCD_REM function block is used to display user strings on the Remote LCD module when the Remote LCD module is present and connected.



### *LCD_BKLT_REM*



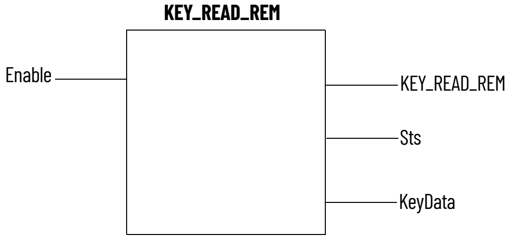
You can use this function block to configure backlight parameters on the Remote LCD module.

Execution of the LCD_BKLT_REM takes precedence over current backlight settings in the Main Menu. When Enable, input becomes False and the instructions stop executing, the last Main Menu setting of the backlight takes effect.

The LCD_BKLT_REM instruction is only effective when displaying user-defined screen or default I/O Status screen. While in the Main Menu, backlight settings configured through the Main Menu take effect.

| | |
|---|---|
| **IMPORTANT** | When in the Main Menu, the LCD_BKLT_REM instruction will be disabled or ineffective. |

*KEY_READ_REM*



You can use this function block to read key status on the Remote LCD module when the user-defined screen is active. When user-defined screen is not active, KEY_READ_REM instruction flags an error.

The KEY_READ_REM instruction will always show key status as False if Push Button Key Read is disabled in Connected Components Workbench software or the Remote LCD module.

## Backup and Restore

To initiate backup and restore through the Remote LCD module, access the memory card by going to the Main Menu > Advanced Set > Memory Card.

For information on how to backup and restore a project on the microSD card, see Using microSD Cards on page 79.

For installation, hardware features, and specifications of the Micro800 Remote LCD module, see the Micro800 Remote LCD Installation Instructions, publication 2080-IN010.

## ASCII Code for Special Characters

Figure 30 lists the special characters supported by the Remote LCD module.

- Small (8 x 8 pixels)
- Medium (8 x 16 pixels)
- Large (16 x 16 pixels)

Table 30 - Special Characters

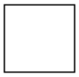
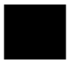
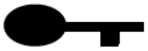
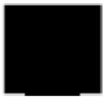
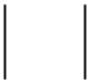
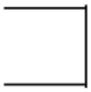
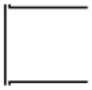
| Character Code (Hex) | Character | Description |
|---|---|---|
| 01H |  | Empty box (with border) |
| 02H |  | Filled box (with border) |
| 08H |  | Key sign (with border) |
| 10H |  | Filled box (without border) |
| 11H | | Horizontal parallel lines (without border) |
| 12H |  | Vertical parallel lines (without border) |
| 13H |  | Horizontal line cap right (without border) |
| 14H |  | Horizontal line cap left (without border) |

Table 30 - Special Characters (Continued)

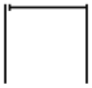
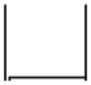
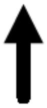
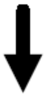
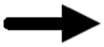
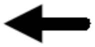
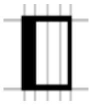
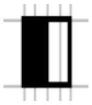
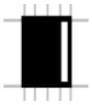
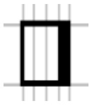
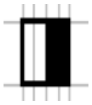
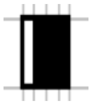
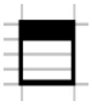
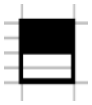
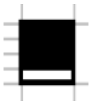
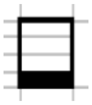
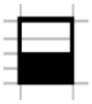
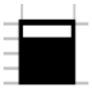
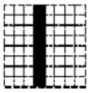
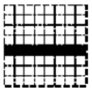
| Character Code (Hex) | Character | Description |
|---|---|---|
| 15H |  | Vertical line cap up (without border) |
| 16H |  | Vertical line cap down (without border) |
| 18H |  | Up arrow (with border) |
| 19H |  | Down arrow (with border) |
| 1AH |  | Right arrow (with border) |
| 1BH |  | Left arrow (with border) |
| 80H |  | 1/4 filled box left (without border) |
| 81H |  | 1/2 filled box left (without border) |
| 82H |  | 3/4 filled box left (without border) |
| 83H |  | 1/4 filled box right (without border) |
| 84H |  | 1/2 filled box right (without border) |
| 85H |  | 3/4 filled box right (without border) |
| 86H |  | 1/4 filled box upside down (without border) |
| 87H |  | 1/2 filled box upside down (without border) |
| 88H |  | 3/4 filled box upside down (without border) |
| 89H |  | 1/4 filled box upside (without border) |

Table 30 - Special Characters (Continued)

| Character Code (Hex) | Character | Description |
|---|---|---|
| 8AH |  | 1/2 filled box upside (without border) |
| 8BH |  | 3/4 filled box upside (without border) |
| 8CH |  | Centered vertical line (without border) |
| 8DH |  | Centered horizontal line (without border) |

# Notes:

Chapter 9

# Using microSD Cards

This chapter provides a description of microSD card support on Micro820 controllers.

The last section provides quickstart projects for the datalog and recipe functions.

## Overview

Micro820 controllers support microSD cards for the following purposes:

- Project backup and restore
- Datalog and Recipe

We recommend to use the Allen-Bradley 2080-SD-2GB microSD card.

| | |
|---|---|
| **IMPORTANT** | For optimum performance, the microSD card should not be more than 90% full. Regularly check available space on your microSD card and ensure that the card is exclusively used for the Micro800 controller and no unnecessary files are present. Regularly delete old datalog files and directories. |

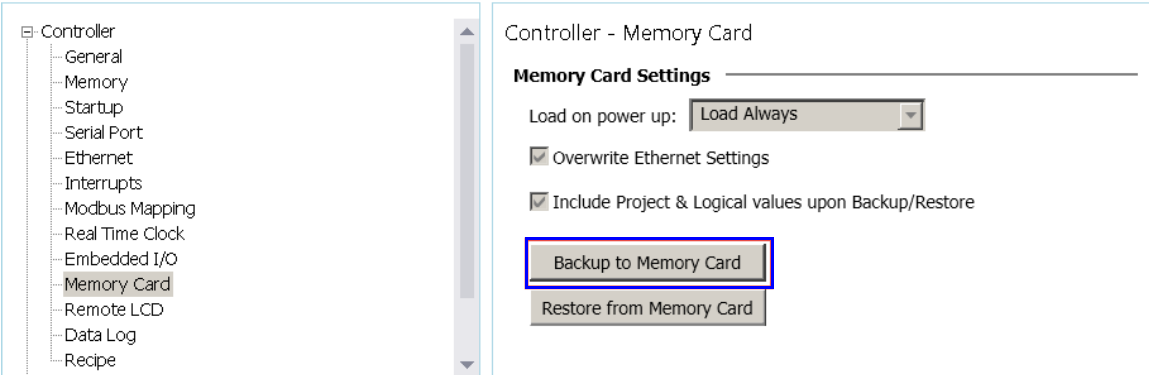
| | |
|---|---|
| **IMPORTANT** | Do not remove the microSD card or power down while operations such as upload, download, delete, search, backup, and restore are ongoing to prevent data loss. A blinking SD status LED indicates that these operations are ongoing.<br>Note the following:<br>• The SD status LED will not blink when updating the firmware from the microSD card.<br>• The SD status LED does not blink continuously for the entire duration of the restore operation. |

| | |
|---|---|
| **IMPORTANT** | To prevent data loss, recipe and datalog function blocks must indicate Idle status before microSD card is removed. |

## Project Backup and Restore

Project backup and restore on Micro820 controllers are mainly supported through the microSD card. Both backup and restore can be initiated or manually triggered and configured through the Connected Components Workbench software, the 2080-REMLCD module, and the ConfigMeFirst.txt file in the microSD card. These backup files are not the same as 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 Connected Components Workbench software.



Controller - Memory Card

Memory Card Settings

Load on power up: Load Always

☑ Overwrite Ethernet Settings

☑ Include Project & Logical values upon Backup/Restore

**IMPORTANT** To learn about restore and backup using the 2080-REMLCD module, see Using the Micro800 Remote LCD on page 69.

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 user project can only be achieved through the POU password protection mechanism in Connected Components Workbench software (Developer Edition) and NOT via Controller Lock feature.

**IMPORTANT** If the Load Always setting is enabled and power is lost when restoring a project from the micoSD 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 will fail and a fault occurs, or the wrong project will be loaded.

The default project name is the name of the controller, for example "Micro820", and the default directory is "Micro820\USERPRJ".

If you change the name of the controller from the default, you must configure the UPD setting in the ConfigMeFirst.txt file.

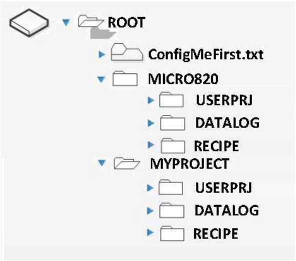
The microSD card stores the controller password in encrypted format. When the password is mismatched, the contents of the microSD card is not restored on the controller.

Backup and restore can be configured to trigger as shown in Table 31.

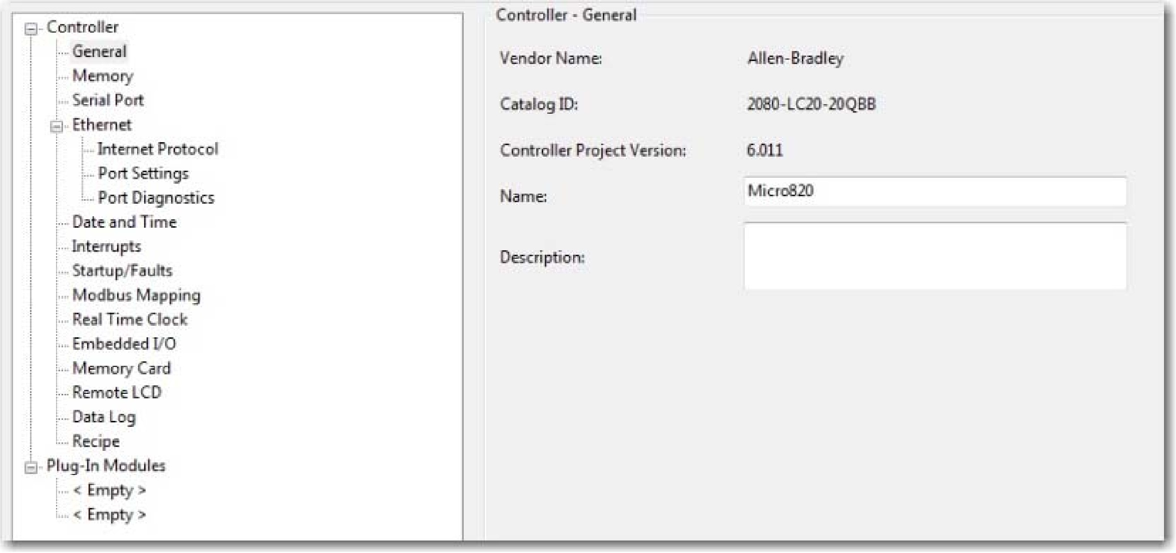
Table 31 - Backup and Restore Methods

| Method | Backup | Restore |
|---|---|---|
| Online with Connected Components Workbench | Yes | Yes |
| 2080-REMLCD | 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<br>(Through the [BKD] command) | Yes<br>(Through the [RSD] command) |

## Backup and Restore Directory Structure



When a user project is backed up, a subdirectory named Micro820\USERPRJ is created on the microSD card. The folder name takes the name of the project specified in the General Page in the Connected Components Workbench software, which is Micro820 by default. However, if the ConfigMeFirst.txt file specifies a different subdirectory (example: MyProject), the project is backed up to that directory. See General Configuration Rules in ConfigMeFirst.txt on page 83.



Controller - General

Vendor Name: Allen-Bradley

Catalog ID: 2080-LC20-20QBB

Controller Project Version: 6.011

Name: Micro820

Description:

Project restore is done from the subdirectory specified in ConfigMeFirst.txt file or the Micro820/USERPRJ default folder, if none is specified in the ConfigMeFirst.txt file. Ensure that the directory is populated with 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 datalog 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 Micro820 controller reads and implements configuration settings that are described in the ConfigMeFirst.txt file. However, the UPD setting also takes effect when you insert the microSD card. The configuration settings for the ConfigMeFirst.txt file are shown in Table 32.

**Table 32 - 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:<br>firmware\<catalog number>\<filename of firmware> |
| [FWDOWN] | Power-up | Sets whether to upgrade or downgrade the controller firmware from the current revision.<br>0 = Upgrade firmware; 1 = Downgrade firmware<br>**IMPORTANT:** Firmware Upgrade will happen if [FWFILE] setting points to a newer version of firmware file compared to current firmware in the controller, irrespective of [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 backup directory, My Proj1\USERPRJ. Requires 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 restore directory MyProj2\USERPRJ into controller. Requires 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. |

Table 32 - ConfigMeFirst.txt Configuration Settings (Continued)

| Setting | Takes effect on... | Description |
|---|---|---|
| [UPD = My Proj] | Power-up and Insertion | For normal usage of backup and restore (that is, through Connected Components Workbench, 2080-REMLCD, 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.<br>This directory is also used by data logging and recipe function. |
| **Network settings** | | |
| [ESFD] | Power-up | Embedded Serial Factory Defaults.<br>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.<br>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 8.0 or later, you can update your Micro820 controller from the microSD card or by using ControlFLASH. See Firmware Update From microSD Card on page 119 for instructions.

- You must place [FWFILE] and [FWDOWN] settings at the beginning of the file.

**IMPORTANT**

**Directory Settings**

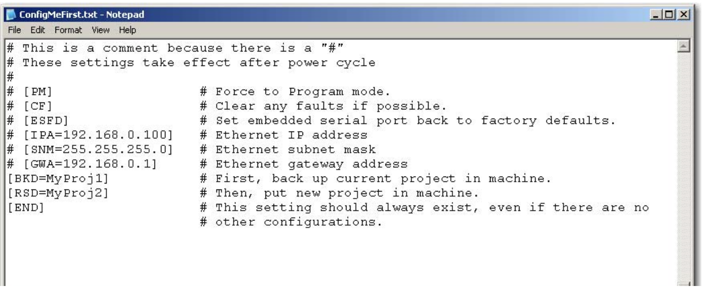
- If you do not specify a directory in the ConfigMeFirst.txt file, then backup and restore occurs in the controller name directory (Micro820/USERPRJ, by default).
- If you configure [UPD] in the ConfigMeFirst.txt file, then backup and restore occurs in the [UPD] directory specified.
- [BKD] setting implements even when the controller is locked or password protected.
- [BKD] directory is created automatically if it does not already exist.

**IMPORTANT**

Power-up Network Parameter Settings

- [IPA], [SNM] and [GWA] follow the general IP configuration rules.
- When you set [IPA] in ConfigMeFirst.txt, you must always configure it with a valid [SNM] and vice versa.
- When you use the optional [GWA] setting, 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.

Figure 40 - Sample ConfigMeFirst.txt File



```
# 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 will be carried out 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 will be 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 shown below.

### *Backup 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 Micro820 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.

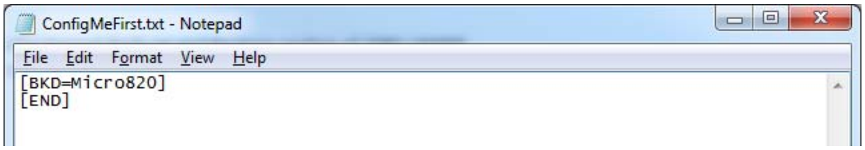
| | |
|---|---|
| **IMPORTANT** | The Backup to Memory Card button is enabled when the controller is in program mode and a microSD card is in the microSD card slot. |

5. After the backup is completed, select OK.

The image files are stored in the default location on the microSD card Micro820\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 41 - Example Configuration for Project Backup**



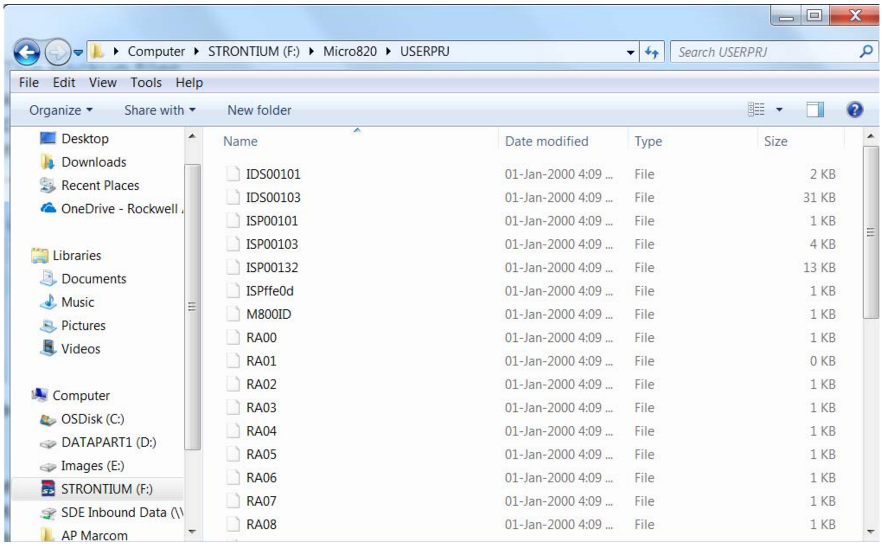
```
[BKD=Micro820]
[END]
```

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 Micro820\USERPRJ).



| Name | Date modified | Type | Size |
|---|---|---|---|
| IDS00101 | 01-Jan-2000 4:09 ... | File | 2 KB |
| IDS00103 | 01-Jan-2000 4:09 ... | File | 31 KB |
| ISP00101 | 01-Jan-2000 4:09 ... | File | 1 KB |
| ISP00103 | 01-Jan-2000 4:09 ... | File | 4 KB |
| ISP00132 | 01-Jan-2000 4:09 ... | File | 13 KB |
| ISPffe0d | 01-Jan-2000 4:09 ... | File | 1 KB |
| M800ID | 01-Jan-2000 4:09 ... | File | 1 KB |
| RA00 | 01-Jan-2000 4:09 ... | File | 1 KB |
| RA01 | 01-Jan-2000 4:09 ... | File | 0 KB |
| RA02 | 01-Jan-2000 4:09 ... | File | 1 KB |
| RA03 | 01-Jan-2000 4:09 ... | File | 1 KB |
| RA04 | 01-Jan-2000 4:09 ... | File | 1 KB |
| RA05 | 01-Jan-2000 4:09 ... | File | 1 KB |
| RA06 | 01-Jan-2000 4:09 ... | File | 1 KB |
| RA07 | 01-Jan-2000 4:09 ... | File | 1 KB |
| RA08 | 01-Jan-2000 4:09 ... | File | 1 KB |

3. Use a compression program to zip these image files and send them to your customer through email.

The customer must unzip these image files into the root directory of their microSD card and verify that the location is identical to the original (default is Micro820\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.

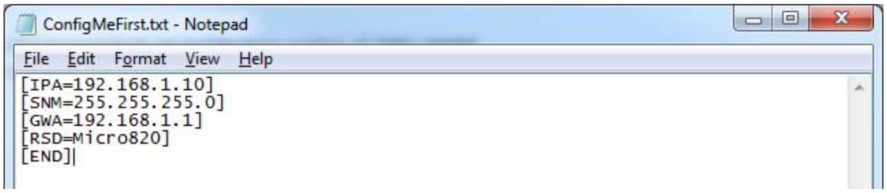
1. Insert the microSD card into the microSD card slot.
2. Cycle power to the controller.
3. When the SD status LED displays a steady green light, the project restore is complete.

Use this method 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.

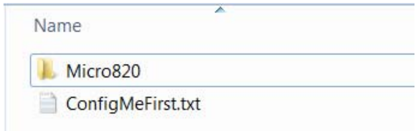
**Figure 42 - Example Configuration for Project Restore**



```
[IPA=192.168.1.10]
[SNM=255.255.255.0]
[GWA=192.168.1.1]
[RSD=Micro820]
[END]
```

In the example shown in Figure 42, 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.

You must place the ConfigMeFirst.txt file in the same root directory as the backup folder in the microSD card.



Name

Micro820

ConfigMeFirst.txt

1. Insert the microSD card into the microSD card slot.
2. Cycle power to the controller.
3. When the SD status LED displays a steady green light, the project restore is complete.

## Datalog

The data logging feature allows you to capture global and local variables with timestamp 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 uploading through the Connected Components Workbench software.

A maximum number of 10 datasets is supported for a Micro820 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 to store datalogs on the microSD card, see the Datalog Directory Structure on page 86.

You can retrieve datalog files from the microSD card using a card reader or by uploading the datalogs through the Connected Components Workbench software.

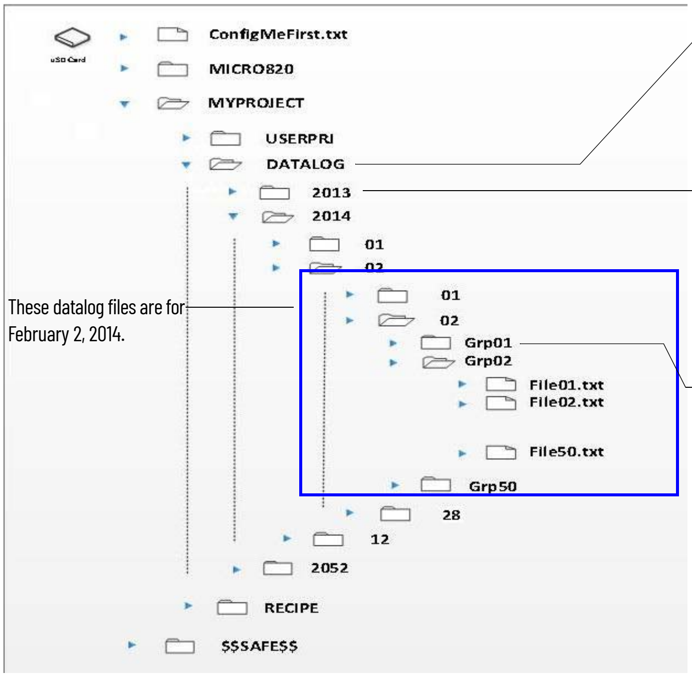
| IMPORTANT | For optimum performance and to prevent file access conflicts, we recommend that you upload datalog files in PROGRAM mode. For example, if the datalog instruction is executing, Connected Components Workbench software does not upload the last datalog file. |
|---|---|

See the sample quick start project to get you started on the Datalog feature, Use the Datalog Feature on page 93.

| IMPORTANT | Datalog execution time depends on your application and its complexity. Do not datalog **faster than every two seconds** for typical applications. Housekeeping takes at least 5 ms per program scan. For more information on program scan and execution rules and sequence, see Program Execution in Micro800 on page 59.<br>See also Datalog - Data Payload vs. Performance Time on page 130. |
|---|---|

| IMPORTANT | In cases where there are simultaneous RCP and DLG function block executions or uploads/downloads/searches, the activities are queued, and the program scan handles them one by one. You can observe a slowdown in performance in these cases. |
|---|---|

## Datalog Directory Structure



These datalog files are for February 2, 2014.

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 81.

Subdirectories are also created following the controller RTC timestamp. This means that if RTC date at the time of function block execution is February 02, 2013, the subfolder 2013 is created under DATALOG. Under the 2013 folder, the subfolder 02 (which stands for 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. You can generate a maximum of 50 Grpxxx folders on the microSD card per day.

Under the current Grpxxx working folder, the datalog file File01.txt is created. Once this file reaches more than 4 KB, another file, File02.txt, is created automatically to store data. The file size is kept small in order 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 created automatically to store the next datalog files for that day. This automatic folder and file generation goes on until the Grpxx folder reaches 50 for that day.
When you insert a microSD card, the DLG function block looks for the last Grpxx folder and filexx.txt file, and proceeds to perform the data logging based on that information.

Table 33 summarizes data logging performance onMicro820 controllers.

Table 33 - Datalog 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 directory is full, a new directory is automatically created in Run mode. |
| Maximum files (Filexx.txt) per day | 50 | When file reaches maximum size, a new file is automatically created in Run mode. |
| Typical data per day | 10 MB | |

[1] Once you reach the datalog limits (that is, 50 Grpxx folders per day) an error (ErrorID 3: DLG_ERR_DATAFILE_ACCESS) is returned.

## Datalog 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.

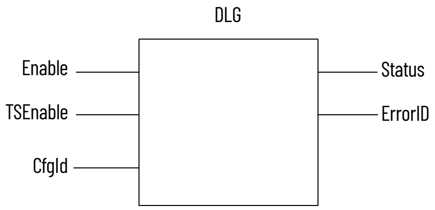


Table 34 - DLG Input and Output Parameters

| Parameter | Parameter Type | Data Type | Description |
|---|---|---|---|
| Enable | INPUT | BOOL | Data logging write function enable.<br>On rising edge (that is, Enable value triggers from low to high), the function block executes. The precondition for execution is that the last operation has completed. |

Table 34 - DLG Input and Output Parameters (Continued)

| Parameter | Parameter Type | Data Type | Description |
|---|---|---|---|
| TSEnable | INPUT | BOOL | Date and timestamp logging enable flag |
| CfgId | INPUT | USINT | Configured dataset (DSET) number (1…10) |
| Status | OUTPUT | USINT | Data logging function block current status |
| ErrorID | OUTPUT | UDINT | Error ID if DLG Write fails |

Table 35 - DLG Function Block Status

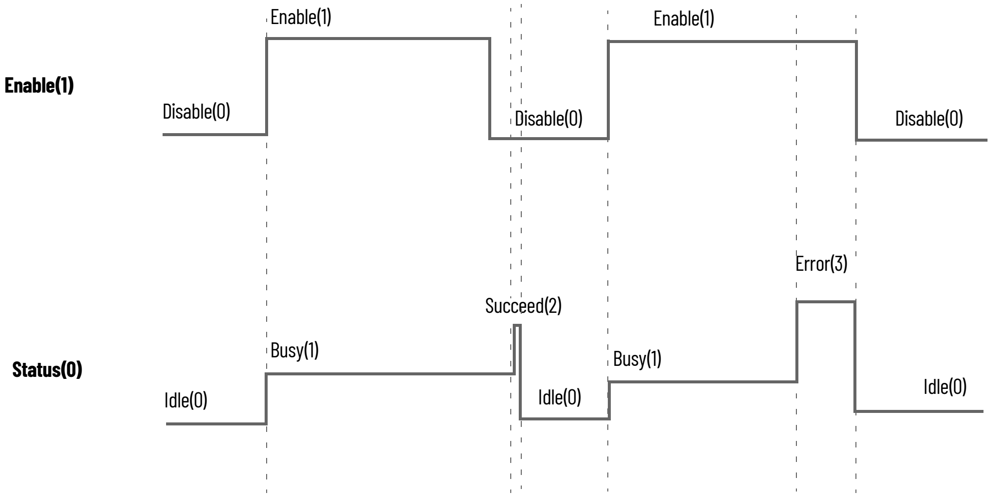
| Status Code | Description |
|---|---|
| 0 | Data logging IDLE status |
| 1 | Data logging BUSY status |
| 2 | Data logging COMPLETE SUCCEED status |
| 3 | Data logging COMPLETE ERROR status |

Table 36 - 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 datalog file in microSD card |
| 4 | DLG_ERR_CFG_ABSENT | Datalog configuration file is absent. |
| 5 | DLG_ERR_CFG_ID | Configuration ID is missing in datalog configuration file. |
| 6 | DLG_ERR_RESOURCE_BUSY | Same Configuration ID is used with other datalog function block call at the same time. |
| 7 | DLG_ERR_CFG_FORMAT | Datalog 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 will be returned during DLG function block execution when card is full.

Figure 43 - Datalog Function Block Timing Diagram



**IMPORTANT** **Datalog Function Block Execution**

- There are three possible states for the Datalog function block: Idle, Busy, and Complete (which includes Complete with Succeed and Complete with Error).
- For one Datalog 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 the input Enable changes to False.
- Only the DLG instruction block can create the datalog file. The Connected Components Workbench software can only upload and delete the datalog file.
- There are separators between every data variable in the data file that is defined during configuration in the Connected Components Workbench software.
  See Supported Data Types for Datalog and Recipe Function Blocks on page 88.
- Data variable values are sampled when data logging function block is in Busy state. However, the data logging file is only created when data logging function block is in the Complete state.

Table 37 - Supported Data Types for Datalog and Recipe Function Blocks

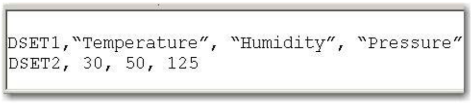
| Data Type | Description | Example Format in Output Datalog File |
|---|---|---|
| BOOL(1) | Logical Boolean with values TRUE and FALSE | 0: FALSE<br>1: TRUe |
| SINT | Signed 8-bit integer value | -128<br>+127 |
| INT | Signed 16-bit integer value | -32,768<br>+32,767 |
| DINT | Signed 32-bit integer value | -2,147,483,648<br>+2,147,483,647 |
| LINT | Signed 64-bit integer value | -9,223,372,036,854,775,808<br>+9,223,372,036,854,775,807 |
| USINT(BYTE) | Unsigned 8-bit integer value | 0<br>255 |
| UINT(WORD) | Unsigned 16-bit integer value | 0<br>65,535 |
| UDINT(DWORD) | Unsigned 32-bit integer value | 0<br>4,294,967,295 |
| ULINT(LWORD) | Unsigned 64-bit integer value | 0<br>18,446,744,073,709,551,615 |
| REAL | 32-bit floating point value | -3.40282347E+38<br>+3.40282347E+38 |
| LREAL | 64-bit floating point value | -1.7976931348623157E+308<br>+1.7976931348623157E+308 |

**Table 37 - Supported Data Types for Datalog and Recipe Function Blocks (Continued)**

| Data Type | Description | Example Format in Output Datalog File |
|---|---|---|
| STRING[2] | character string (1 byte per character) | '"Rotation Speed" |
| DATE[1] | Unsigned 32-bit integer value | 1234567<br>(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<br>(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 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 datalog file.
The example below shows DSET1 using string variables and DSET2 using integers.

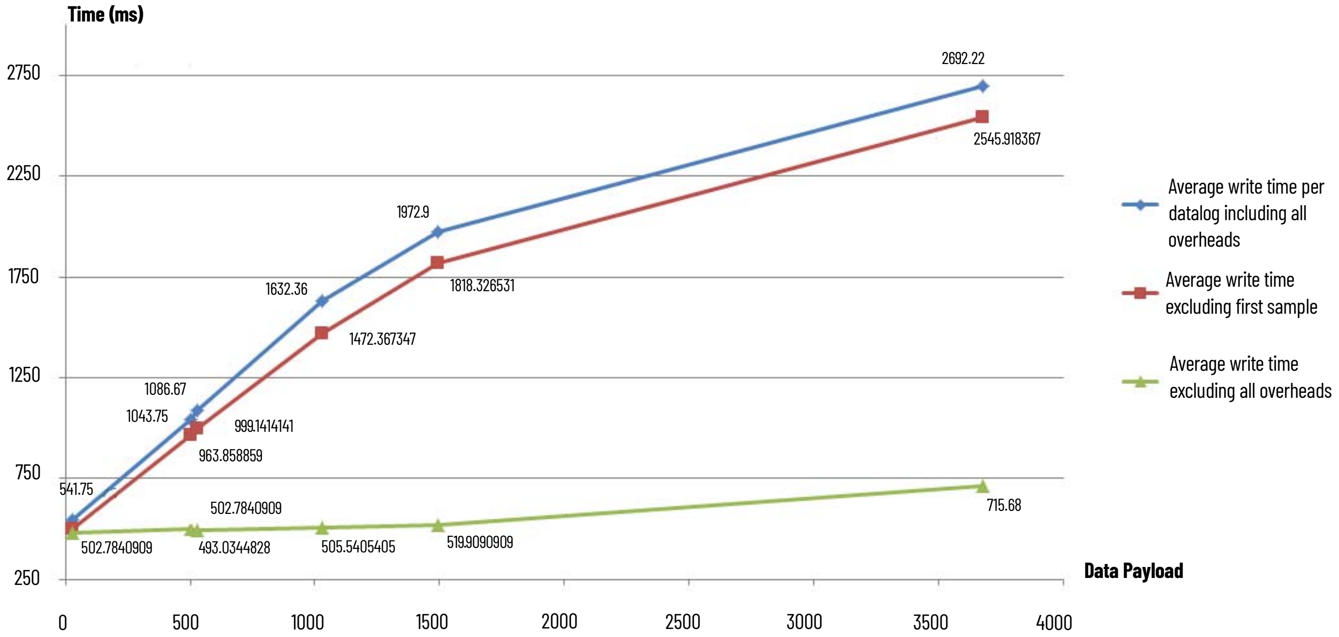


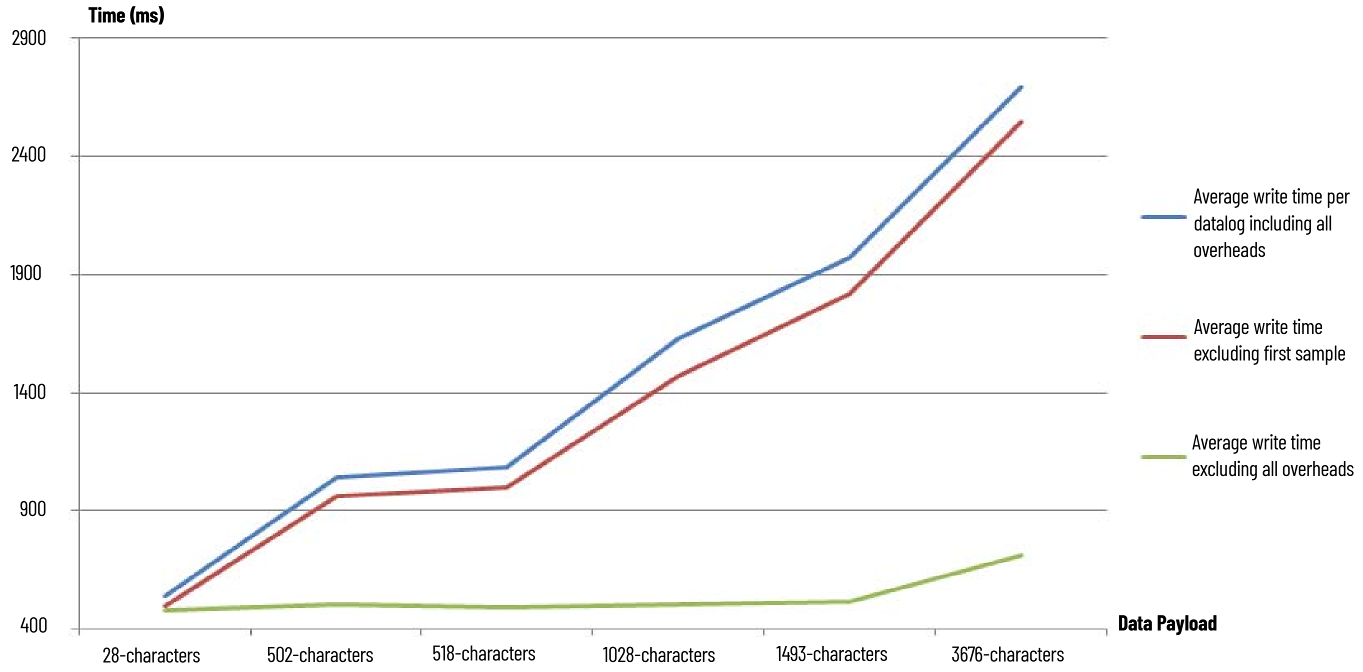
```
DSET1,"Temperature", "Humidity", "Pressure"
DSET2, 30, 50, 125
```

## *Datalog Performance*

**Table 38 - Datalog – Data Payload vs. Performance Time**

| Parameter | Number of Characters | | | | | |
|---|---|---|---|---|---|---|
| | **28** | **502** | **518** | **1028** | **1493** | **3676** |
| Average write time per datalog file including all overheads | 541.77 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 | 1472.36 ms | 1818.33 ms | 2545.92 ms |
| Average write time excluding all overheads | 479.10 ms | 502.78 ms | 493.03 ms | 505.54 ms | 519.91 ms | 715.68 ms |





# Recipe

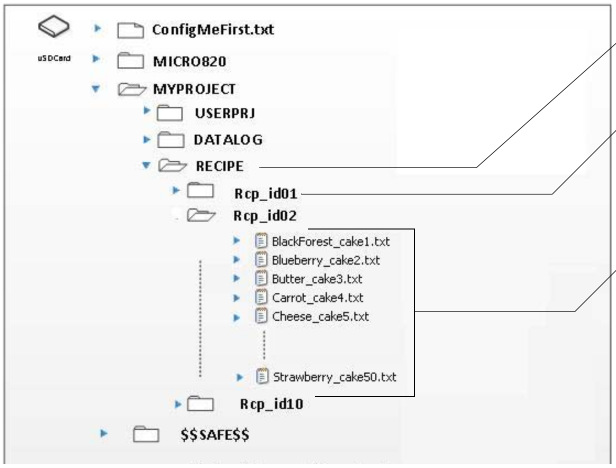
Micro820 controllers support the Recipe feature and allows users to store and load a list of data to and/or from recipe data files using the RCP instruction. It also allows you to download, upload, and delete Recipe data on the microSD card through the Connected Components Workbench software.

A Micro820 program supports a maximum of 10 recipe sets. 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 to store recipes on the microSD card, see the Recipe Directory Structure on page 90.

Table 39 - 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 RcpName parameter for the RCP function block, as configured in Connected Components Workbench. Each Recipe set can contain up to 50 recipe files or variations. Filenames for recipe files should not exceed 30 characters.

*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 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 smaller controller into the recipe data file in the microSD card.

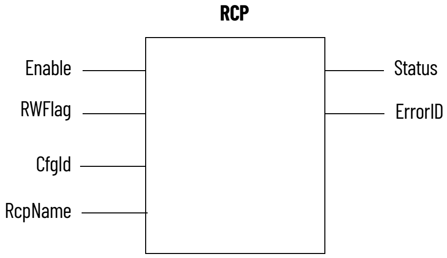


**Table 40 - RCP Input and Output Parameters**

| 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 last operation is completed. |
| RWFlag | INPUT | BOOL | TRUE: Recipe write data variables to recipe files into the microSD card.<br>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 error ID information if RCP read/write fails |

**Table 41 - 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 42 - RCP Function Block Errors**

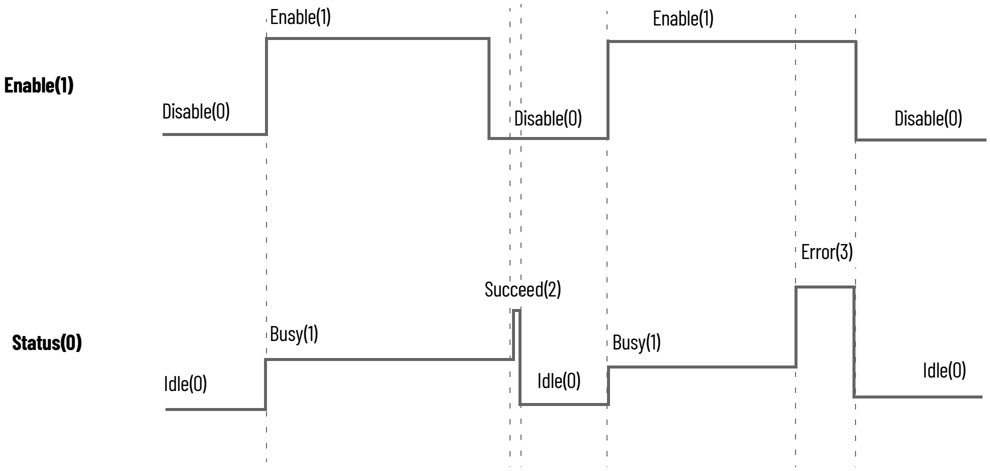
| Error ID | Error Name | Description |
|---|---|---|
| 0 | RCP_ERR_NONE | No error |
| 1 | RCP_ERR_NO_SDCARD | microSD card is absent. |
| 2 | RCP_ERR_DATAFILE_FULL | Recipe files exceed maximum number of files per recipe set folder. |
| 3 | RCP_ERR_DATAFILE_ACCESS | Error to access recipe data file in microSD card |
| 4 | RCP_ERR_CFG_ABSENT | Recipe configuration file is absent. |
| 5 | RCP_ERR_CFG_ID | Configure ID is absent in 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. |

Table 42 - RCP Function Block Errors (Continued)

| Error ID | Error Name | Description |
|---|---|---|
| 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 card is full.

Figure 44 - Recipe Function Block Timing Diagram



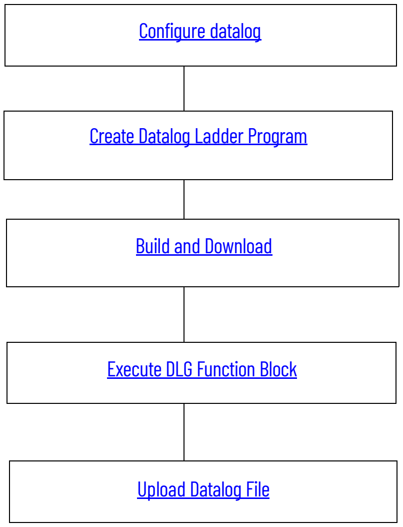
**IMPORTANT** RCP Function Block Execution

- There are three possible states for Recipe function block: Idle, Busy, and 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 needs to go back to Idle first.
- Idle status changes to Busy status only when Enable input signal is in 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 Enable input parameter's rising edge when a new function block execution starts. During function block execution, 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, 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 extension (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 which is defined during configuration in Connected Components Workbench software. Redundant tab, space, carriage return, and line feed characters are strictly not allowed. See Supported Data Types for Datalog and Recipe Function Blocks on page 88.
- Double quotes are not allowed within a string in a recipe file.

## Quick Start Projects for Datalog and Recipe Function Blocks

The sample quick start projects shown in Figure 45 provide step-by-step instructions on how to use the Datalog and Recipe function blocks in Connected Components Workbench software to generate and manage your recipe files and datalogs.

**Figure 45 - Use the Datalog Feature**



Configure datalog

Create Datalog Ladder Program

Build and Download

Execute DLG Function Block

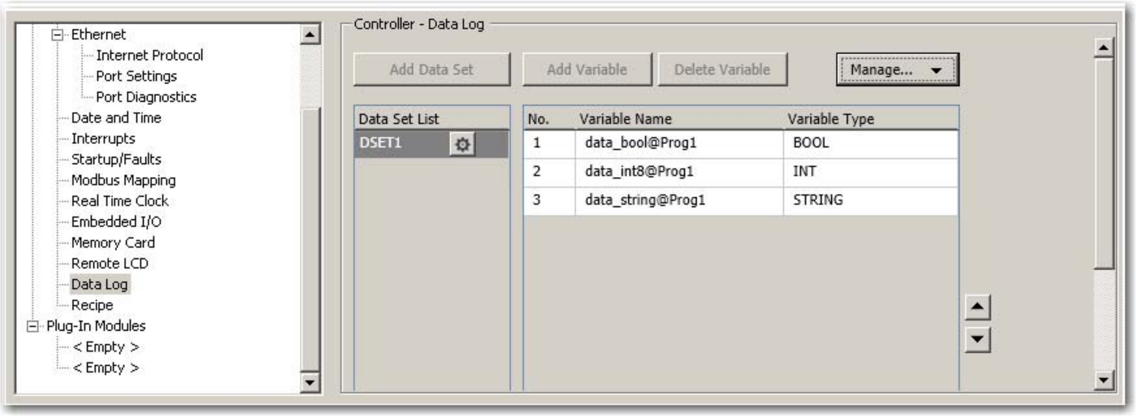
Upload Datalog File

### *Configure datalog*

1. In the Connected Components Workbench software, go to the Properties pane to configure your datalog.
2. Select Data Log. Select Add Data Set to add a dataset. Each dataset is stored in the same file. You can add up to 10 datasets per configuration.
3. Select 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 variables shown in Table 43 that you have previously created to Dataset 1.

**Table 43 - Local Variables**

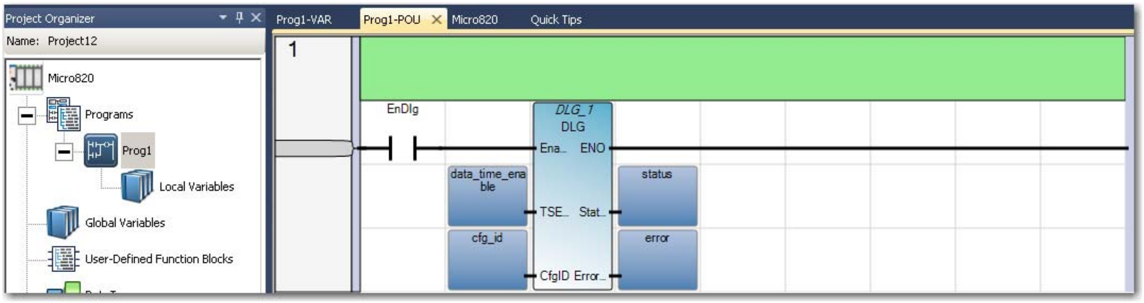
| Variable Name | Data Type |
|---|---|
| data_bool | BOOL |
| data_int8 | INT |
| data_string | STRING |



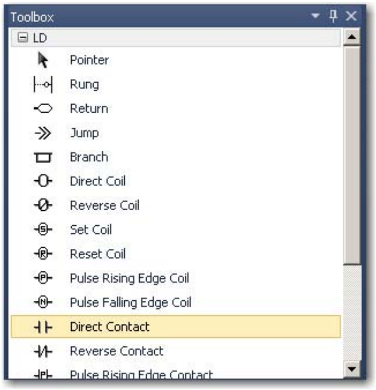
| Data Set List |
|---|
| DSET1 |

| No. | Variable Name | Variable Type |
|---|---|---|
| 1 | data_bool@Prog1 | BOOL |
| 2 | data_int8@Prog1 | INT |
| 3 | data_string@Prog1 | STRING |

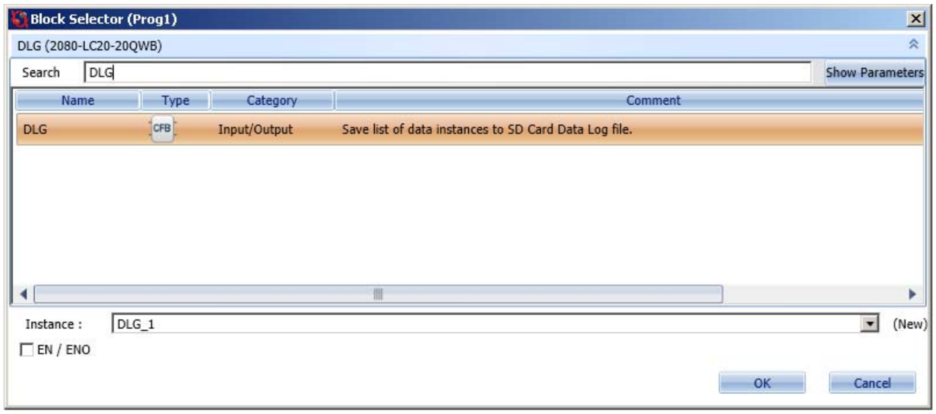
*Create Datalog Ladder Program*



1. Launch the Connected Components Workbench software. Create a user program for your Micro820 controller.
2. Right-click Programs. Select Add New LD: Ladder Diagram. Name the Program (for example, Prog1).
3. From the Toolbox, select 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.


Block Selector (Prog1)

DLG (2080-LC20-20QWB)

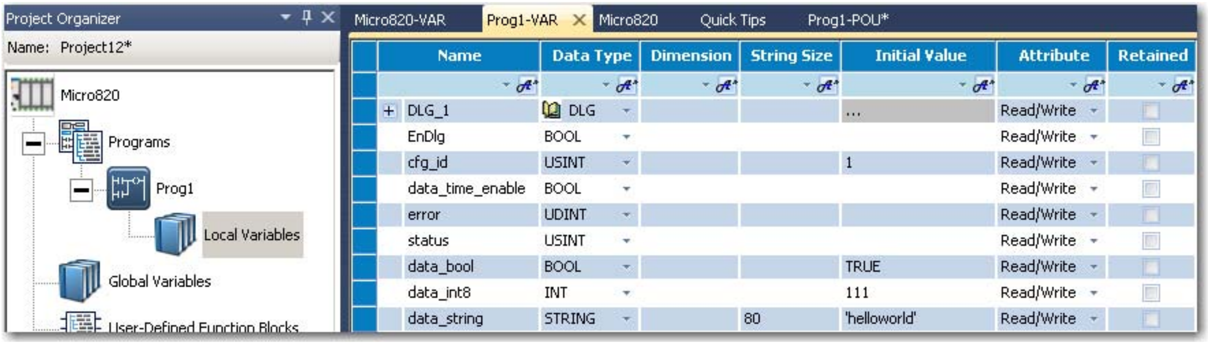
Search: DLG

| Name | Type | Category | Comment |
|---|---|---|---|
| DLG | CFB | Input/Output | Save list of data instances to SD Card Data Log file. |

Instance : DLG_1 (New)

□ EN / ENO

6. Create the local variables shown in Table 44 for your project.

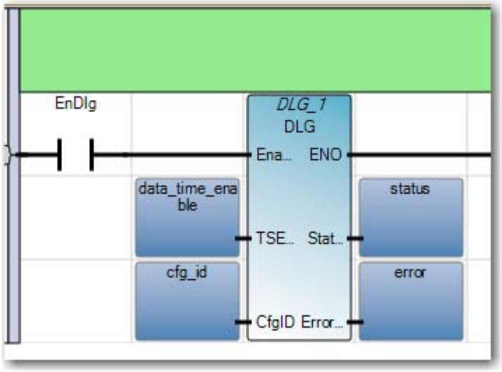


| Name | Data Type | Dimension | String Size | Initial Value | Attribute | Retained |
|---|---|---|---|---|---|---|
| + DLG_1 | DLG | | | ... | Read/Write | □ |
| EnDlg | BOOL | | | | Read/Write | □ |
| cfg_id | USINT | | | 1 | Read/Write | □ |
| data_time_enable | BOOL | | | | Read/Write | □ |
| error | UDINT | | | | Read/Write | □ |
| status | USINT | | | | Read/Write | □ |
| data_bool | BOOL | | | TRUE | Read/Write | □ |
| data_int8 | INT | | | 111 | Read/Write | □ |
| data_string | STRING | | 80 | 'helloworld' | Read/Write | □ |

**Table 44 - Local Variables**

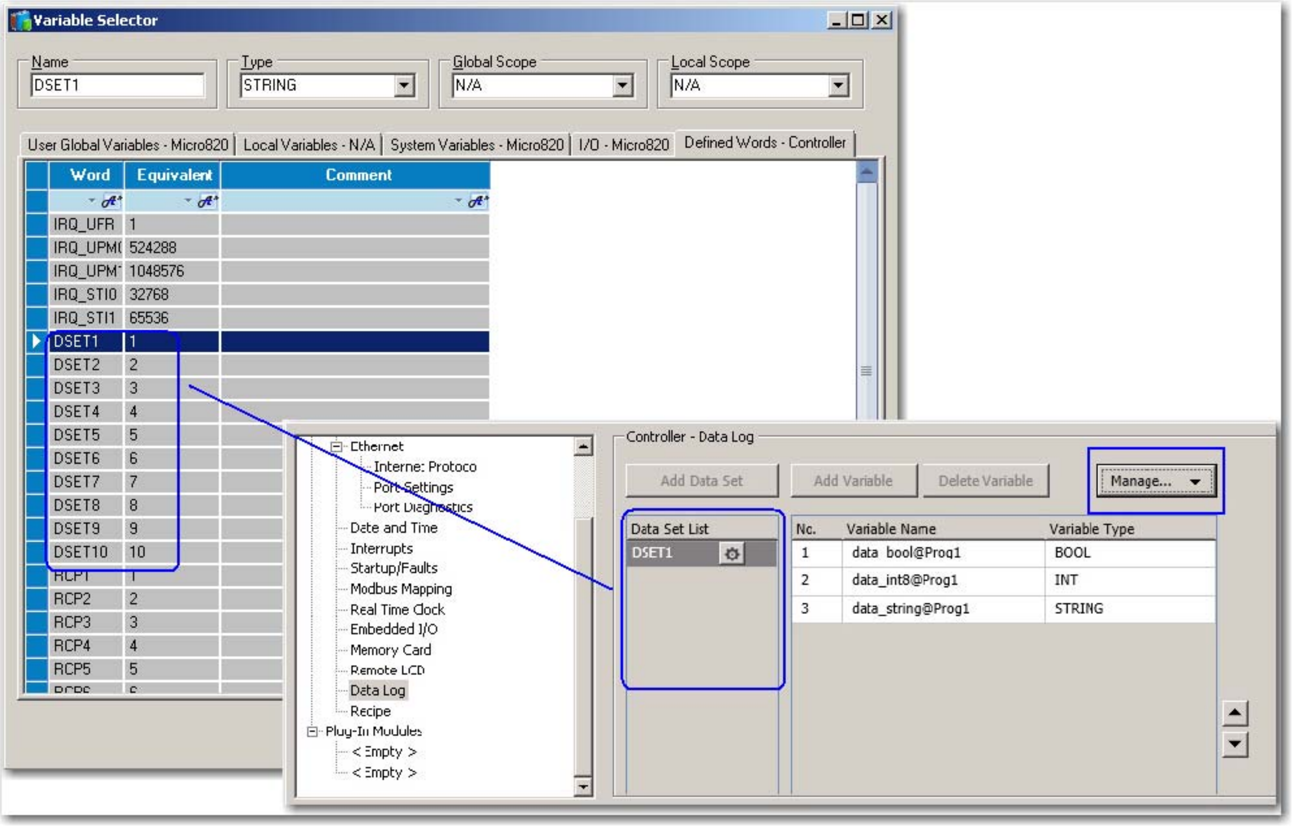
| Variable Name | Data Type |
|---|---|
| EnDlg | BOOL |
| cfg_id | USINT |
| data_time_enable | BOOL |
| error | UDINT |
| status | USINT |
| data_bool | BOOL |
| data_int8 | INT |
| data_string | STRING |

7. Assign the variables to the DLG input and output parameters as follows:



For CfgID input parameter, you can choose a predefined variable by choosing from the Defined Words in 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. See Figure 46.

**Figure 46 - Choose a Predefined Variable**



Name: DSET1
Type: STRING
Global Scope: N/A
Local Scope: N/A

| Word | Equivalent | Comment |
|---|---|---|
| IRQ_UFR | 1 | |
| IRQ_UPM( | 524288 | |
| IRQ_UPM' | 1048576 | |
| IRQ_STI0 | 32768 | |
| IRQ_STI1 | 65536 | |
| DSET1 | 1 | |
| DSET2 | 2 | |
| DSET3 | 3 | |
| DSET4 | 4 | |
| DSET5 | 5 | |
| DSET6 | 6 | |
| DSET7 | 7 | |
| DSET8 | 8 | |
| DSET9 | 9 | |
| DSET10 | 10 | |
| RCP1 | 1 | |
| RCP2 | 2 | |
| RCP3 | 3 | |
| RCP4 | 4 | |
| RCP5 | 5 | |

Data Set List: DSET1

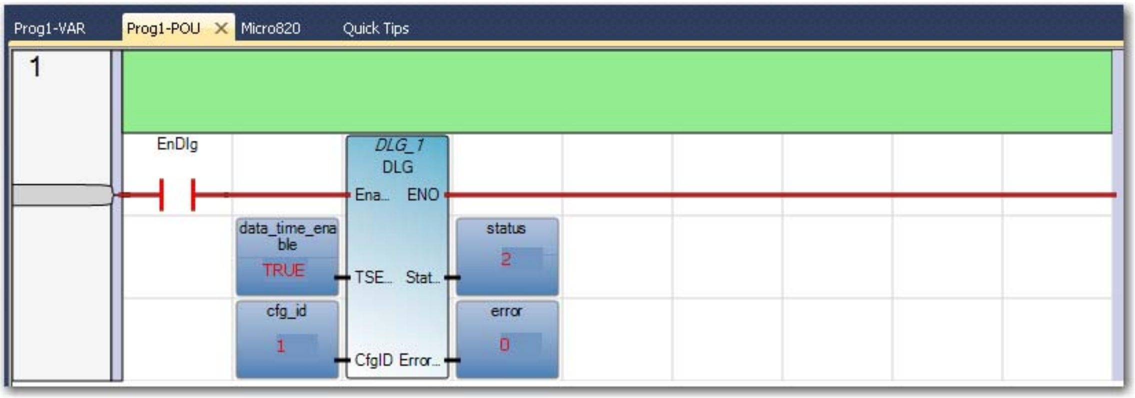
| No. | Variable Name | Variable Type |
|---|---|---|
| 1 | data_bool@Prog1 | BOOL |
| 2 | data_int8@Prog1 | INT |
| 3 | data_string@Prog1 | STRING |

### *Build and Download*

After configuring datalog properties, build the program and download to the controller.

### *Execute DLG Function Block*

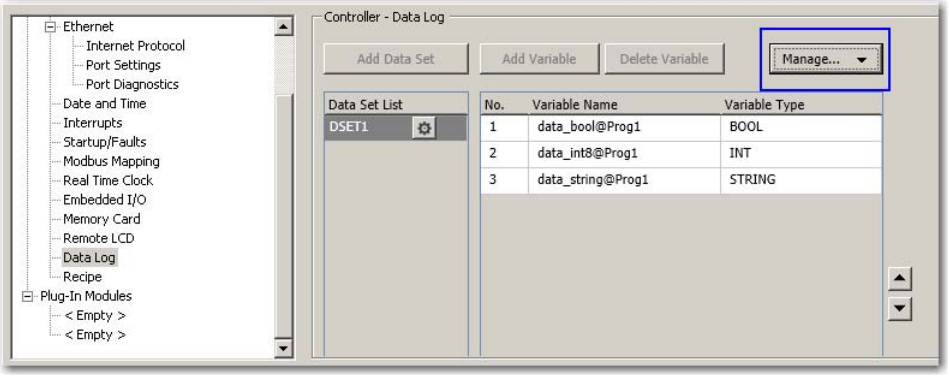
Execute the DLG function block. Notice the Status output go from 0 (Idle) to 1 (Enable), and then 2 (Succeed).



### *Upload Datalog File*

You can retrieve datalog files from the microSD card with a card reader or by uploading the datalogs through Connected Components Workbench software.

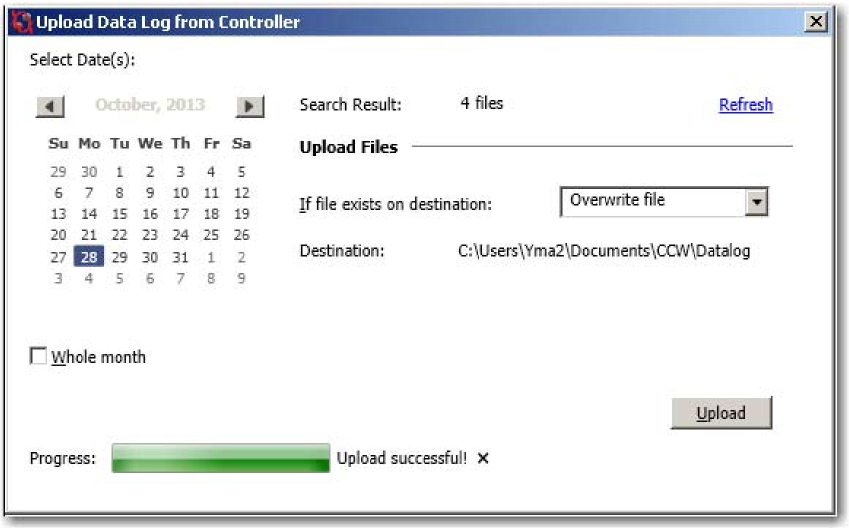
1. To use the Upload feature, go to the Properties section of your project in the Connected Components Workbench software.
2. Select Data Log. Select Manage and then choose Upload.



| No. | Variable Name | Variable Type |
|---|---|---|
| 1 | data_bool@Prog1 | BOOL |
| 2 | data_int8@Prog1 | INT |
| 3 | data_string@Prog1 | STRING |

**IMPORTANT** The Manage button is not available in DEBUG mode. You need to stop DEBUG mode to use the Manage button to upload datalog files. Uploading datalog files in PROGRAM mode is recommended for performance and file locking reasons.

3. From the Upload window that appears, select the date of the datalog files that you want to upload. You can upload datalogs for the entire month by selecting the Whole Month option.



Upload Data Log from Controller

Select Date(s):

October, 2013

| Su | Mo | Tu | We | Th | Fr | Sa |
|---|---|---|---|---|---|---|
| 29 | 30 | 1 | 2 | 3 | 4 | 5 |
| 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| 13 | 14 | 15 | 16 | 17 | 18 | 19 |
| 20 | 21 | 22 | 23 | 24 | 25 | 26 |
| 27 | 28 | 29 | 30 | 31 | 1 | 2 |
| 3 | 4 | 5 | 6 | 7 | 8 | 9 |

□ Whole month

Search Result: 4 files

Upload Files

If file exists on destination: Overwrite file

Destination: C:\Users\Yma2\Documents\CCW\Datalog

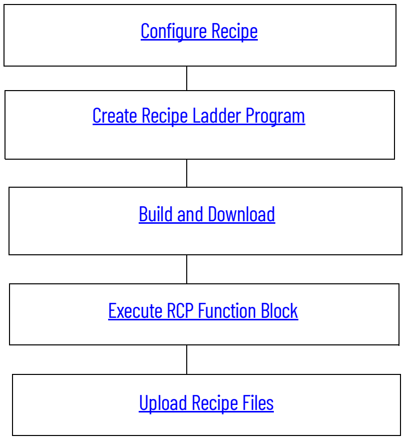
Progress: Upload successful!

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 indicates 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.

**IMPORTANT** For better datalog file management, you can use a third-party tool or DOS CMD to merge all your datalog files into a single 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

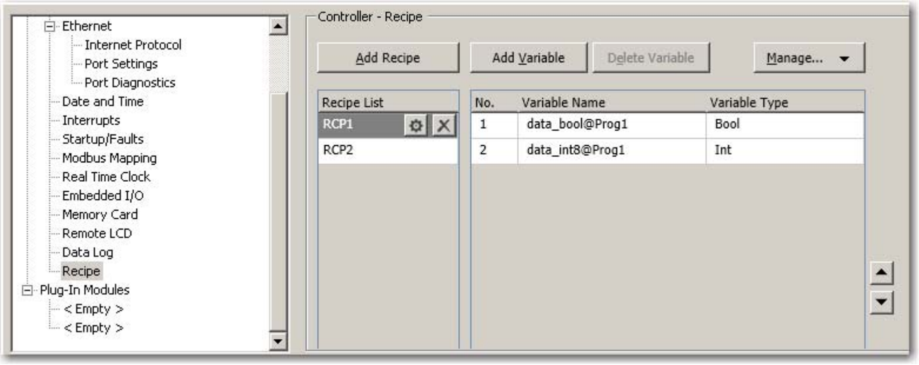
### *Configure Recipe*

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 variables shown in Figure 45 that you have previously created to RCP 1:

Table 45 - Local Variables

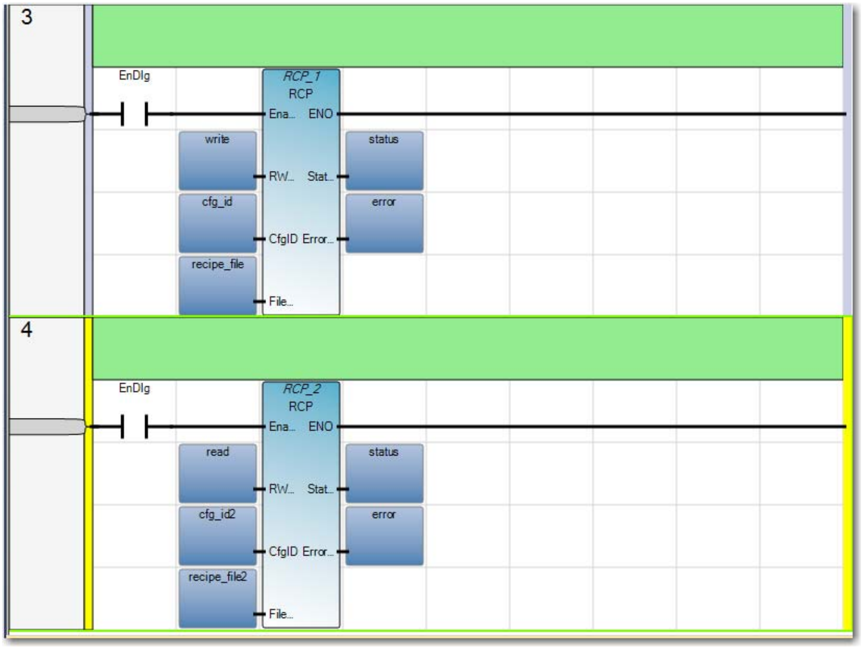
| Variable Name | Data Type |
|---|---|
| data_bool | BOOL |
| data_int8 | INT |



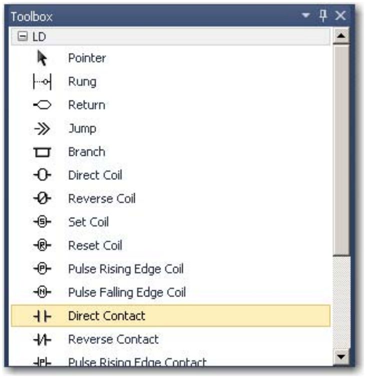
| Recipe List |
|---|
| RCP1 |
| RCP2 |

| No. | Variable Name | Variable Type |
|---|---|---|
| 1 | data_bool@Prog1 | Bool |
| 2 | data_int8@Prog1 | Int |

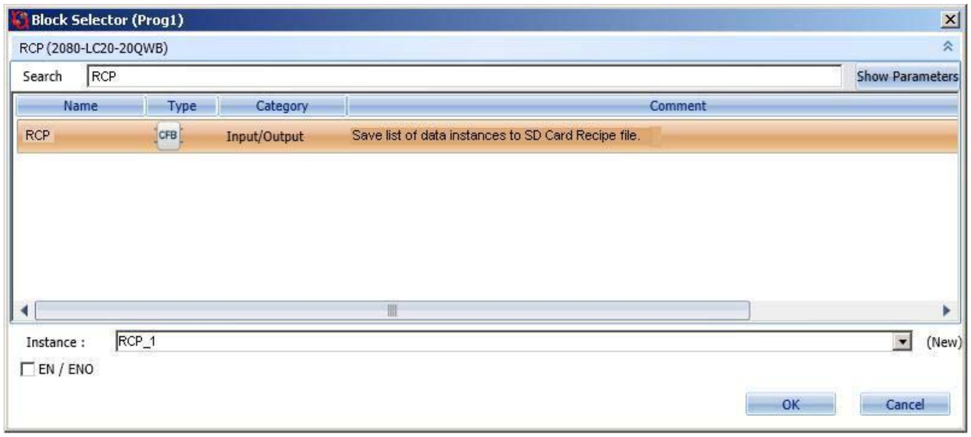
*Create Recipe Ladder Program*



4. Launch the Connected Components Workbench software. Create a user program for your Micro820 controller.
5. Right-click Programs. Select Add New LD: Ladder Diagram. Name the Program (for example, Prog2).
6. From the Toolbox, select Direct Contact to add it to the first rung.



7. From the Toolbox, select Block to add it to the rung.
8. On the Block Selector window that appears, type RCP to filter the Recipe function block from the list of available function blocks. Select OK.



Block Selector (Prog1)

RCP (2080-LC20-20QWB)

Search RCP Show Parameters

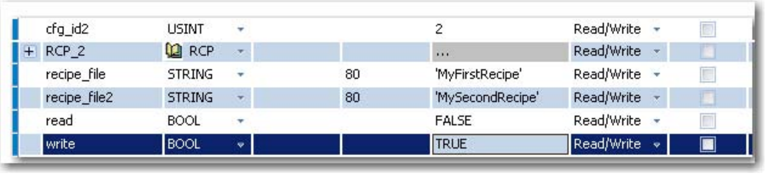
| Name | Type | Category | Comment |
|---|---|---|---|
| RCP | CFB | Input/Output | Save list of data instances to SD Card Recipe file. |

Instance : RCP_1 (New)

□ EN / ENO

OK Cancel

9. From the Toolbox, select rung to add another rung.
10. Add a Direct Contact and RCP function block to this second rung by following steps 3…5.
11. Create the following local variables shown in Figure 46 for your program, in addition to the ones that you have already created for datalog.



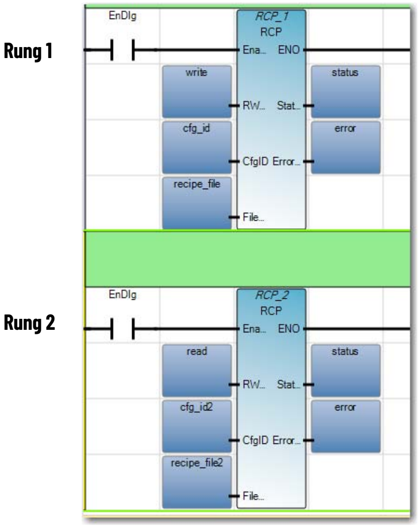
| | | | | | | |
|---|---|---|---|---|---|---|
| cfg_id2 | USINT | | | 2 | Read/Write | □ |
| + RCP_2 | RCP | | | ... | Read/Write | □ |
| recipe_file | STRING | | 80 | 'MyFirstRecipe' | Read/Write | □ |
| recipe_file2 | STRING | | 80 | 'MySecondRecipe' | Read/Write | □ |
| read | BOOL | | | FALSE | Read/Write | □ |
| write | BOOL | | | TRUE | Read/Write | □ |

Table 46 - Local Variables

| Variable Name | Data Type |
|---|---|
| recipe_file | STRING |
| recipe_file2 | STRING |
| cfg_id2 | USINT |
| read | BOOL |
| write | BOOL |

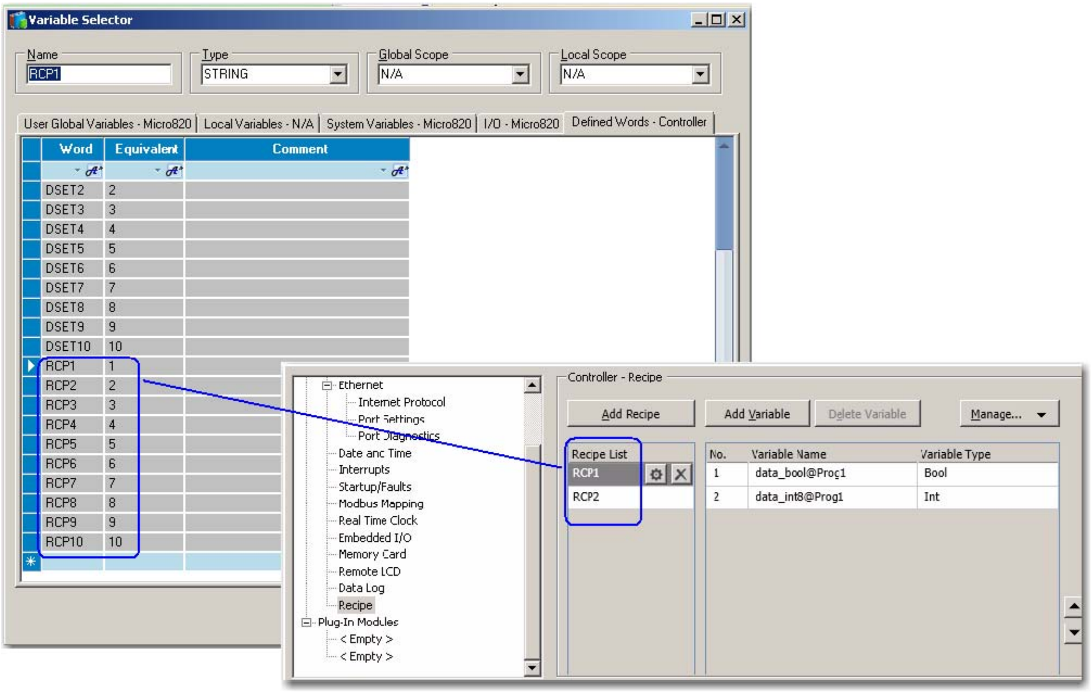
12. Assign the variables to the RCP input and output parameters as shown in Figure 47:

Figure 47 - Assign Variables



For CfgID input parameter, you can choose a predefined variable 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 48.

**Figure 48 - Defined Words**



Name: RCP1
Type: STRING
Global Scope: N/A
Local Scope: N/A

| Word | Equivalent | Comment |
|---|---|---|
| DSET2 | 2 | |
| DSET3 | 3 | |
| DSET4 | 4 | |
| DSET5 | 5 | |
| DSET6 | 6 | |
| DSET7 | 7 | |
| DSET8 | 8 | |
| DSET9 | 9 | |
| DSET10 | 10 | |
| RCP1 | 1 | |
| RCP2 | 2 | |
| RCP3 | 3 | |
| RCP4 | 4 | |
| RCP5 | 5 | |
| RCP6 | 6 | |
| RCP7 | 7 | |
| RCP8 | 8 | |
| RCP9 | 9 | |
| RCP10 | 10 | |

| Recipe List |
|---|
| RCP1 |
| RCP2 |

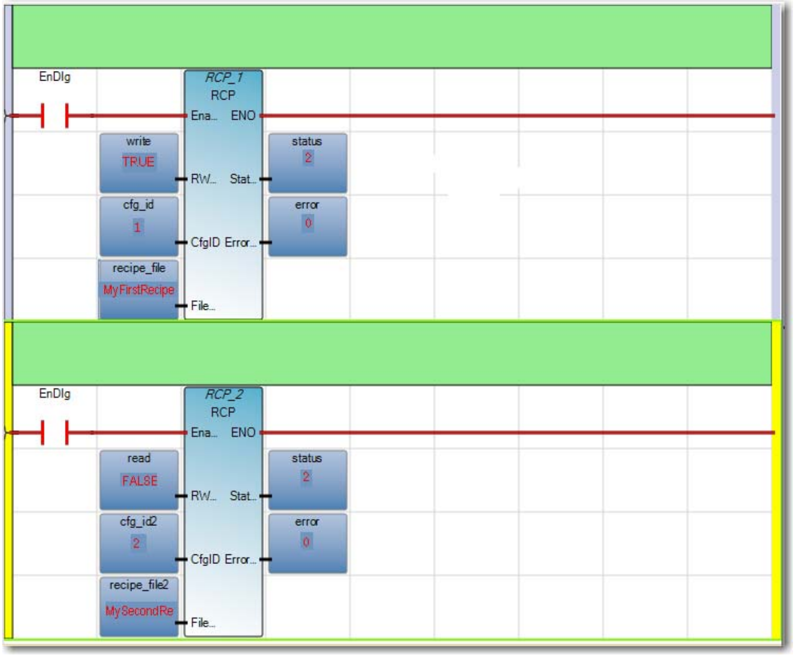
| No. | Variable Name | Variable Type |
|---|---|---|
| 1 | data_bool@Prog1 | Bool |
| 2 | data_int8@Prog1 | Int |

*Build and Download*

After configuring Recipe, build the program and download to the controller.

*Execute RCP Function Block*

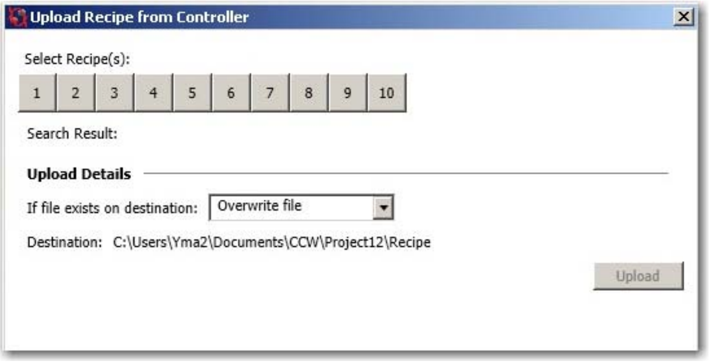
Execute the RCP function block. Notice the Status output go from 0 (Idle) to 1 (Enable), and then 2 (Succeed).



### *Upload Recipe Files*

You can retrieve recipe files from the microSD card with a card reader or by uploading the recipe files through 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 choose 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 want to upload.



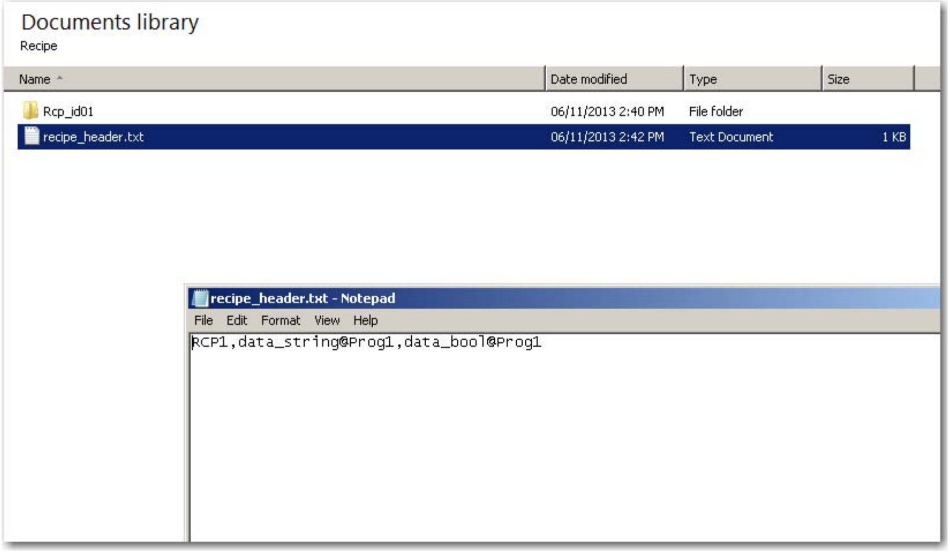
If file exists on destination: Overwrite file

Destination: C:\Users\Yma2\Documents\CCW\Project12\Recipe

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 indicates 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.



| 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 |

```
RCP1,data_string@Prog1,data_bool@Prog1
```

## Notes:

# Modbus Mapping for Micro800 Controllers

## Modbus Mapping

Micro820 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.

### 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 47 - 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 |
| 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.

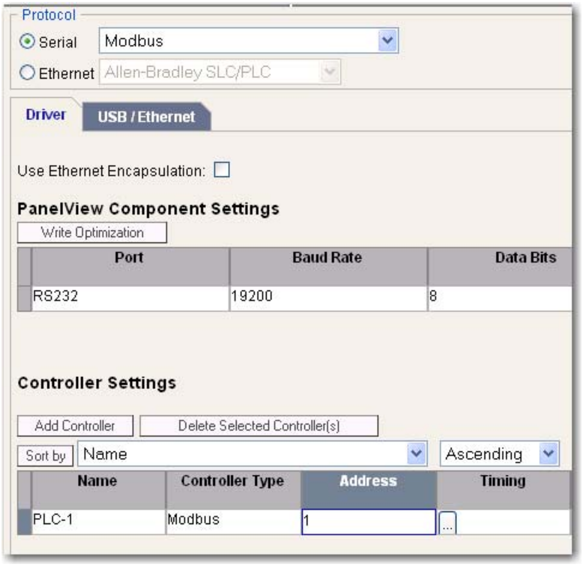
## Example 1, PanelView Component HMI (Master) to Micro800 (Slave)

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 in order to set up communications from PV800 to Micro800.

1. Change the Protocol from DF1 to Modbus.



Protocol
● Serial: Modbus
○ Ethernet: Allen-Bradley SLC/PLC

Use Ethernet Encapsulation: □

PanelView Component Settings

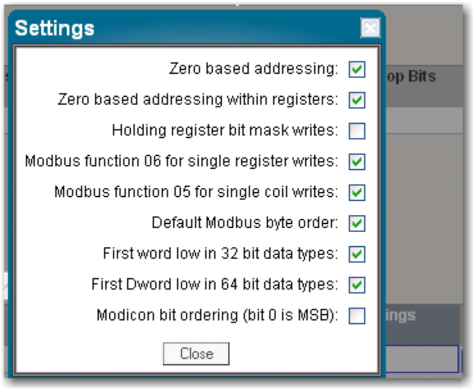
| Port | Baud Rate | Data Bits |
|---|---|---|
| RS232 | 19200 | 8 |

Controller Settings

Sort by: Name, Ascending

| Name | Controller Type | Address | Timing |
|---|---|---|---|
| PLC-1 | Modbus | 1 | ... |

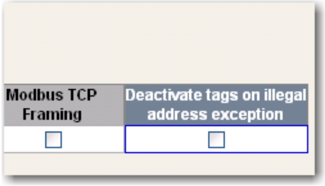
2. Set the Address of Micro800 slave to match the Serial port configuration for the controller.



Settings

Zero based addressing: ☑
Zero based addressing within registers: ☑
Holding register bit mask writes: □
Modbus function 06 for single register writes: ☑
Modbus function 05 for single coil writes: ☑
Default Modbus byte order: ☑
First word low in 32 bit data types: ☑
First Dword low in 64 bit data types: ☑
Modicon bit ordering (bit 0 is MSB): □

3. Deactivate Tags on Error. This is to prevent the requirement of power cycling PanelView 800 when new Modbus Mappings are downloaded from the Connected Components Workbench software to the Micro800 controller.



| Modbus TCP Framing | Deactivate tags on illegal address exception |
|---|---|
| □ | □ |

## Example 2, Micro800 (Master) to PowerFlex 4M Drive (Slave)

The following is the overview of the steps to take to configure a PowerFlex 4M drive. Parameter numbers listed in this section are for a PowerFlex 4M drive and will be different if you are using another PowerFlex 4-class drive.

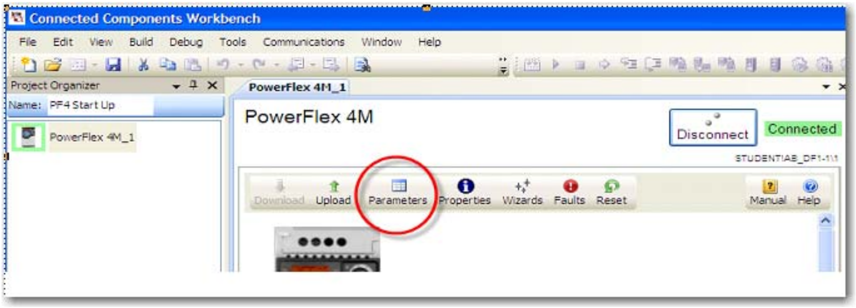
**Table 48 - Parameters in PowerFlex 4-class Drives**

| 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 | | |

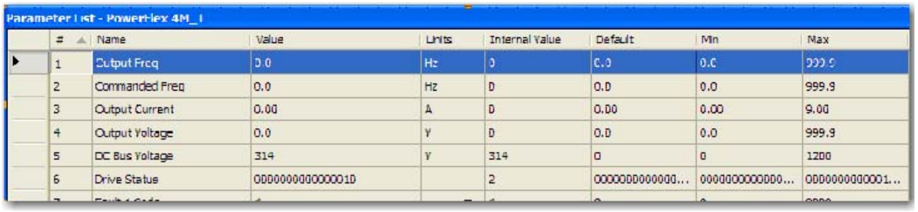
- Connect the 1203-USB to the PowerFlex drive and to the computer.
- Launch Connected Components Workbench software, connect to the drive, and set parameters.

To configure a PowerFlex 4M drive, perform the following steps:

1. Double-click the PowerFlex 4M drive if it is not already open in 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).
   - Accept 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 data values of Parameters.


Parameter List - PowerFlex 4M_1

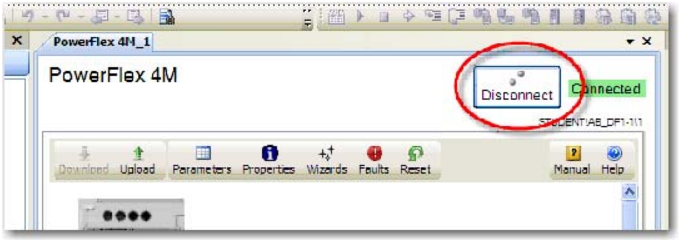
| # | Name | Value | Units | Internal Value | Default | Min | Max |
|---|---|---|---|---|---|---|---|
| 1 | Output Freq | 0.0 | Hz | 0 | 0.0 | 0.0 | 999.9 |
| 2 | Commanded Freq | 0.0 | Hz | 0 | 0.0 | 0.0 | 999.9 |
| 3 | Output Current | 0.00 | A | 0 | 0.00 | 0.00 | 9.00 |
| 4 | Output Voltage | 0.0 | V | 0 | 0.0 | 0.0 | 999.9 |
| 5 | DC Bus Voltage | 314 | V | 314 | 0 | 0 | 1200 |
| 6 | Drive Status | 0000000000000010 | | 2 | 0000000000000... | 0000000000000... | 000000000001... |

7. From the Parameter window, change the following Parameters to set the communications for Modbus RTU so that the PowerFlex 4M drive will communicate with Micro820 via Modbus RTU communication.

**Table 49 - Modbus RTU Parameters**

| Parameter | Description | Setting |
|---|---|---|
| C302 | Communication Data Rate (Baud Rate) 4 = 19200 bps | 4 |
| C303 | Communication Node Address (Address range is 1…127) | 2 |
| C304 | Communication Loss Action (Action taken when loss communication) 0 = Fault with coast stop | 0 |
| C305 | Communication Loss Time (Time remain in communication before taking action set in C304) 5 sec (max 60) | 5 |
| C306 | Communication 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 drive.

   The drive is now ready to be controlled by Modbus RTU communication commands initiated from the Micro820 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 family controllers, the register addresses listed in the PowerFlex 4M User Manual need to 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)**

**EXAMPLE:**

| | |
|---|---|
| 8193 | Logic Command word (Stop, Start, Jog, and so on.) |
| 8194 | Speed Reference word |
| xxx.x format for 4/4M/40, where "123" = 12.3 Hz | |
| xxx.xx format for 40P/400/400N/400P, 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 'n' |

- If the respective PowerFlex drive supports Modbus Function Code 16 Preset (Write) Multiple Registers, use a single write message with a length of "2" to write the Logic Command (8193) and Speed Reference (8194) at the same time.
- Use a single Function Code 03 Read Holding Registers with a length of "4" to read the Logic status (8449), Error Code (8450), and Speed Feedback (8452) at the same time.

See the PowerFlex 4M Adjustable Frequency AC Drive User Manual FRN 1.xx - 2.xx, publication 22F-UM001 for additional information about Modbus addressing. (See the appendix – Modbus RTU Protocol, in PowerFlex 400 Adjustable Frequency AC Drive for Fan & Pump Applications User Manual, publication 22C-UM001).

## Performance

The performance of MSG_MODBUS (Micro800 is 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, baud rate, and slave response time.

The performance of Micro800 when receiving Modbus request messages (Micro800 is slave) is also affected by the Program Scan. Each Serial port is serviced only once per program scan.

# Notes:

# Troubleshooting

## Status Indicators on the Controller

Status indication on the Micro820 controller is as shown in Figure 49.

Figure 49 - Status Indicators

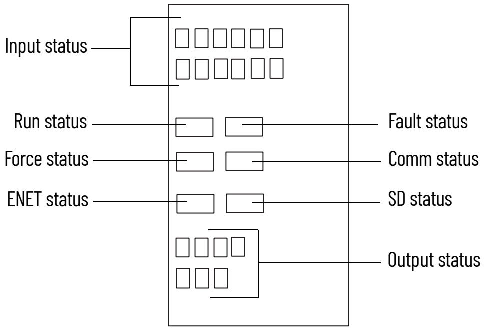


Table 50 - Status Indicator Description

| | Description | State | Indicates |
|---|---|---|---|
| 1 | Input status | Off | Input is low. |
| | | On | Input is energized (terminal status). |
| 2 | Fault status | Off | No fault detected |
| | | Red | Non-recoverable fault that requires a power cycle |
| | | Flashing red | Recoverable fault |
| 3 | Run status | Green | Executing the user program in run mode |
| | | Flashing green (1 Hz) | The controller is in program mode or performing a restore operation. |
| 4 | Serial comm status | Off | No traffic for RS-232/RS-485 |
| | | Green | Traffic through RS-232/RS-485.<br>The indicator only blinks when transmitting data. It does not blink when receiving data. |
| 5 | Force status | Off | No force conditions are active. |
| | | Amber | Force conditions are active. |
| 6 | SD status | Off<br>Uninitialized state | • microSD card is not inserted.<br>• microSD card is inserted but medium is bad.<br>• microSD card is inserted but file system is bad. |
| | | Off<br>Error state | • microSD card read/write failure<br>• Failure to read Configmefirst.txt in the root directory<br>• Errors are detected in ConfigMeFirst.txt. See ConfigMeFirst.txt Errors on page 83 for list of errors. |
| | | On<br>Idle state | • microSD card is initialized completely without read/write on SD card.<br>• microSD card read/write is complete. |
| | | Blinking<br>Operating state | microSD card is being read/written.<br>The status indicator does not blink continuously for the entire duration of the restore operation. |

Table 50 - Status Indicator Description (Continued)

| | Description | State | Indicates |
|---|---|---|---|
| 7 | ENET status | Steady off | Not powered, no connection.<br>The device is powered off, or is powered on but no Ethernet link established. |
| | | Flashing green | No IP address.<br>The device is powered on with Ethernet link established but no IP address is assigned yet.<br>Duplicate IP.<br>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. |
| | | Steady green | Operational.<br>Ethernet ink is active and the device has valid IP address. |
| 8 | Output status | Off | Output is not energized. |
| | | On | Output is energized (logic status). |

## Normal Operation

The Run status indicator is on or flashing. If a force condition is active, the Force status indicator turns on and remains on until all forces are removed.

# Error Codes

This section lists possible error codes for your controller, as well as 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 non-recoverable 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 — You can clear a recoverable fault without having to power cycle the controller. The fault status indicator flashes red when a recoverable fault occurs.
- Non-recoverable — You must power cycle the controller to clear a non-recoverable fault. After you power cycle or reset the controller, check the fault log in the Diagnostic page of the Connected Components Workbench software, then clear the fault. The fault status indicator is solid red when a non-recoverable fault occurs.

Table 51 - 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.<br>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:<br>• See Corrective Actions for Recoverable Faults on page 115.<br>• Check wiring to eliminate any noise, see Wiring Requirements and Recommendation on page 33. |
| 0xF001 | Recoverable | The controller program has been cleared. This happened because:<br>• A power-down occurred during program download or data transfer from the memory module.<br>• The cable was removed from the controller during program download.<br>• The RAM integrity test failed. | Perform one of the following:<br>• See Corrective Actions for Recoverable Faults on page 115.<br>• Transfer the program using the memory module restore utility. |

Table 51 - List of Error Codes for Micro800 controllers (Continued)

| Error Code | Fault Type | Description | Recommended Action |
|---|---|---|---|
| 0xF002 | Non-recoverable | The controller hardware watchdog was activated. The controller hardware watchdog timeout happens if program scan is more than three seconds.<br>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 Non-recoverable Faults on page 115. |
| 0xF003 | Recoverable | One of the following occurred:<br>• The memory module hardware faulted.<br>• The memory module connection faulted.<br>• The memory module was incompatible with the Micro800 controller's firmware revision. | Perform one of the following:<br>• Remove the memory module and plug it in again.<br>• Obtain a new memory module.<br>• See Corrective Actions for Recoverable Faults on page 115.<br>• 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:<br>• See Corrective Actions for Recoverable Faults on page 115.<br>• Attempt the data transfer again. If the error persists, replace the memory module.<br>• 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:<br>• See Corrective Actions for Recoverable Faults on page 115.<br>• See Wire Your Controller on page 33. |
| 0xF006 | Recoverable | The user program is incompatible with the Micro800 controller's firmware revision. | Perform one of the following:<br>• See Corrective Actions for Recoverable Faults on page 115.<br>• 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:<br>• See Corrective Actions for Recoverable Faults on page 115.<br>• Contact your local Rockwell Automation technical support representative. |
| 0xF014 | Recoverable | A memory module memory error occurred. | • See Corrective Actions for Recoverable Faults on page 115.<br>• Reprogram the memory module. If the error persists, replace the memory module. |
| 0xF015 | Non-recoverable | An unexpected software error occurred. | Perform one of the following:<br>• See Corrective Actions for Non-recoverable Faults on page 115.<br>• Refer to Wiring Requirements and Recommendation on page 33. |
| 0xF016 | Non-recoverable | An unexpected hardware error occurred. | Perform one of the following:<br>• See Corrective Actions for Non-recoverable Faults on page 115.<br>• Refer to Wiring Requirements and Recommendation on page 33. |
| 0xF017 | Non-recoverable | An unexpected software error occurred due to unexpected hardware interrupt.<br>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:<br>• See Corrective Actions for Non-recoverable Faults on page 115.<br>• See Wiring Requirements and Recommendation on page 33. |
| 0xF018 | Non-recoverable | An unexpected software error occurred due to SPI communication failure.<br>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:<br>• See Corrective Actions for Non-recoverable Faults on page 115.<br>• See Wiring Requirements and Recommendation on page 33. |
| 0xF019 | Non-recoverable | An unexpected software error occurred due to memory or other controller resource issue. | See Corrective Actions for Non-recoverable Faults on page 115. |
| 0xF01A | Recoverable | The controller was unexpectedly reset during Run Mode Change (RMC) due to a noisy environment or an internal hardware failure.<br>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 115. |
| 0xF020 | Recoverable | The base hardware faulted or is incompatible with the Micro800 controller's firmware revision. | See Corrective Actions for Recoverable Faults on page 115. |
| 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 115. |
| 0xF022 | Recoverable | The user program in the memory module is incompatible with the Micro800 controller's firmware revision. | Perform one of the following:<br>• See Corrective Actions for Recoverable Faults on page 115.<br>• Replace the memory module. |

Table 51 - List of Error Codes for Micro800 controllers (Continued)

| Error Code | Fault Type | Description | Recommended Action |
|---|---|---|---|
| 0xF023 | Non-recoverable | The controller program has been cleared. This happened because:<br>• A power down occurred during program download or transfer from the memory module.<br>• The Flash Integrity Test failed (Micro810 only). | Perform one of the following:<br>• See Corrective Actions for Non-recoverable Faults on page 115.<br>• Download or transfer the program. |
| 0xF030<br>0xF031<br>0xF032<br>0xF033 | Recoverable | Power down information in persistent memory may not be written properly due to a noisy environment or an internal hardware failure.<br>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 115. |
| 0xF050 | Recoverable | The embedded I/O configuration in the user program is invalid. | See Corrective Actions for Recoverable Faults on page 115. |
| 0xF100 | Recoverable | There is general configuration error detected in the motion configuration downloaded from the Connected Components Workbench software, such as number of axis, or motion execution interval being configured out of range. | Perform the following:<br>• See Corrective Actions for Recoverable Faults on page 115.<br>• 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:<br>• See Corrective Actions for Recoverable Faults on page 115.<br>• 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:<br>• See Corrective Actions for Recoverable Faults on page 115.<br>• Remove all axes and re-configure motion with the guidance from the User Manual. |
| 0xF15z[1] | Recoverable | There is a motion engine logic error (firmware logic issue or memory crash) for one axis detected during motion engine cyclic operation. One possible reason can be motion engine data/memory crash. | See Corrective Actions for Recoverable Faults on page 115. |
| 0xF210 | Recoverable | The expansion I/O terminator is missing. | Perform the following:<br>1. Power off the controller.<br>2. Attach the expansion I/O terminator on the last expansion I/O module on the system.<br>3. Power on the controller.<br>4. See Corrective Actions for Recoverable Faults on page 115. |
| 0xF230 | Recoverable | The maximum number of expansion I/O modules has been exceeded. | Perform the following:<br>1. Power off the controller.<br>2. Verify that the number of expansion I/O modules is not more than four.<br>3. Power on the controller.<br>4. See Corrective Actions for Recoverable Faults on page 115. |
| 0xF250 | Recoverable | There is a non-recoverable error and the expansion I/O module(s) could not be detected. | See Corrective Actions for Recoverable Faults on page 115. |
| 0xF26z[2] | Recoverable | An expansion I/O master fault is detected on the system. | See Corrective Actions for Recoverable Faults on page 115. |
| 0xF27z[2] | Recoverable | A non-recoverable communication fault has occurred on the expansion I/O module. | Perform one of the following:<br>• See Corrective Actions for Recoverable Faults on page 115.<br>• Replace the slot number z module. |
| 0xF28z[2] | Recoverable | Expansion I/O communication rate error | Perform one of the following:<br>• See Corrective Actions for Recoverable Faults on page 115.<br>• 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:<br>• See Corrective Actions for Recoverable Faults on page 115.<br>• Replace the slot number z module. |
| 0xF2Az[2] | Recoverable | Expansion I/O power failure | Perform one of the following:<br>• See Corrective Actions for Recoverable Faults on page 115.<br>• Replace the slot number z module. |
| 0xF2Bz[2] | Recoverable | Expansion I/O configuration fault | Perform one of the following:<br>• See Corrective Actions for Recoverable Faults on page 115.<br>• Correct the expansion I/O module configuration in the user program to match that of the actual hardware configuration.<br>• Check the expansion I/O module operation and condition.<br>• Replace the expansion I/O module. |

Table 51 - List of Error Codes for Micro800 controllers (Continued)

| Error Code | Fault Type | Description | Recommended Action |
|---|---|---|---|
| 0xF300 | Recoverable | The memory module is present but memory module is empty and restore operation is requested. | Perform the following:<br>• See Corrective Actions for Recoverable Faults on page 115.<br>• Check to make sure there is a valid project in the memory module.<br>• 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:<br>• See Corrective Actions for Recoverable Faults on page 115.<br>• Check to make sure there is a user program with a controller that has the correct controller catalog configured.<br>• 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 Micro820 controller when Remote LCD performs the restore operation.<br>This fault does not apply to Micro800 controller firmware revision 10 or later. | Perform one of the following:<br>• See Corrective Actions for Recoverable Faults on page 115.<br>• Check to make sure that the user program in the memory module has the correct password.<br>• Download a user program with a password and use the backup function to the memory module.<br>• 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. | Verify that the memory module is present. |
| 0xF0Az[3] | Recoverable | The plug-in I/O module experienced an error during operation. | Perform the following:<br>• Check the condition and operation of the plug-in I/O module.<br>• See Corrective Actions for Recoverable Faults on page 115. |
| 0xF0Bz[3] | Recoverable | The plug-in I/O module configuration does not match the actual I/O configuration detected. | Perform one of the following:<br>• See Corrective Actions for Recoverable Faults on page 115.<br>• Correct the plug-in I/O module configuration in the user program to match that of the actual hardware configuration.<br>• Check the condition and operation of the plug-in I/O module.<br>• 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:<br>1. See Corrective Actions for Recoverable Faults on page 115.<br>2. Correct the plug-in I/O module configuration in the user program.<br>3. Build and download the program using Connected Components Workbench software.<br>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:<br>1. See Corrective Actions for Recoverable Faults on page 115.<br>2. Correct the plug-in I/O module configuration in the user program.<br>3. Build and download the program using Connected Components Workbench software.<br>4. Put the Micro800 controller into Run mode. |
| 0xF830 | Recoverable | An error occurred in the EII configuration. | Perform the following:<br>• See Corrective Actions for Recoverable Faults on page 115.<br>• Review and change the EII configuration in the Micro800 controller properties. |
| 0xF840 | Recoverable | An error occurred in the HSC configuration. | Perform the following:<br>• See Corrective Actions for Recoverable Faults on page 115.<br>• Review and change the EII configuration in the Micro800 controller properties. |
| 0xF850 | Recoverable | An error occurred in the STI configuration. | Perform the following:<br>• See Corrective Actions for Recoverable Faults on page 115.<br>• Review and change the EII configuration in the Micro800 controller properties. |
| 0xF860 | Recoverable | A data overflow occurred.<br>A data overflow error is generated when the ladder, structured text or function block diagram execution encounters a divide-by-zero. | Perform the following:<br>1. See Corrective Actions for Recoverable Faults on page 115.<br>2. Correct the program to ensure that there is no data overflow.<br>3. Build and download the program using Connected Components Workbench software.<br>4. Put the Micro800 controller into Run mode. |

Table 51 - List of Error Codes for Micro800 controllers (Continued)

| Error Code | Fault Type | Description | Recommended Action |
|---|---|---|---|
| 0xF870 | Recoverable | An index address was out of data space. | Perform the following:<br>1. See Corrective Actions for Recoverable Faults on page 115.<br>2. Correct the program to ensure that there is no index used to access an array element beyond the array boundaries.<br>3. Build and download the program using Connected Components Workbench software.<br>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:<br>1. See Corrective Actions for Recoverable Faults on page 115.<br>2. Correct the program to ensure that there is no index used to access a bit beyond the boundaries of the data type.<br>3. Build and download the program using Connected Components Workbench software.<br>4. Put the Micro800 controller into Run mode. |
| 0xF880 | Recoverable | A data conversion error occurred. | Perform the following:<br>1. See Corrective Actions for Recoverable Faults on page 115.<br>2. Correct the program to ensure that there is no data conversion error.<br>3. Build and download the program using Connected Components Workbench software.<br>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:<br>• See Corrective Actions for Recoverable Faults on page 115.<br>• 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:<br>• See Corrective Actions for Recoverable Faults on page 115.<br>• Correct the user interrupt configuration for plug-in I/O module in the user program to match that of the actual hardware configuration. |
| 0xF8A0 | Recoverable | The TOW parameters are invalid. | Perform the following:<br>1. See Corrective Actions for Recoverable Faults on page 115.<br>2. Correct the program to ensure that there are no invalid parameters.<br>3. Build and download the program using Connected Components Workbench software.<br>4. Put the Micro800 controller into Run mode. |
| 0xF8A1 | Recoverable | The DOY parameters are invalid. | Perform the following:<br>1. See Corrective Actions for Recoverable Faults on page 115.<br>2. Correct the program to ensure that there are no invalid parameters.<br>3. Build and download the program using Connected Components Workbench software.<br>4. Put the Micro800 controller into Run mode. |
| 0xFFzz[4] | Recoverable | A user-created fault from the Connected Components Workbench software has occurred. | See Corrective Actions for Recoverable Faults on page 115. |
| 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 115. |
| 0xD011 | Recoverable | The program scan time exceeded the watchdog timeout value. | Perform the following:<br>• See Corrective Actions for Recoverable Faults on page 115.<br>• Determine if the program is caught in a loop and correct the problem.<br>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 65,535.<br>To correct the fault, perform the following:<br>1. Correct the program to ensure that the upper limit is not reached. One method is to use a data type with a larger maximum value.<br>2. Build and download the program using Connected Components Workbench.<br>3. Put the Micro800 controller into Run mode.<br>4. If your program is designed to have a scan time of longer than three 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 display. For program numbers 01x00 to 0xFFFF, only the last byte displays.

## Corrective Action for Recoverable and Non-recoverable Faults

*Corrective Actions for Recoverable Faults*

Perform the following:

1. Optionally save the fault log from Connected Components Workbench software. For more information, see Retrieve a Fault Log on page 115.
2. Clear the recoverable fault using Connected Components Workbench software.
3. If problem persists, contact technical support with the fault log.

*Corrective Actions for Non-recoverable Faults*

Perform the following:

1. Power cycle your Micro800 controller.
2. Controller will go 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 program is lost, build and download your program using Connected Components Workbench software.
5. If 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:

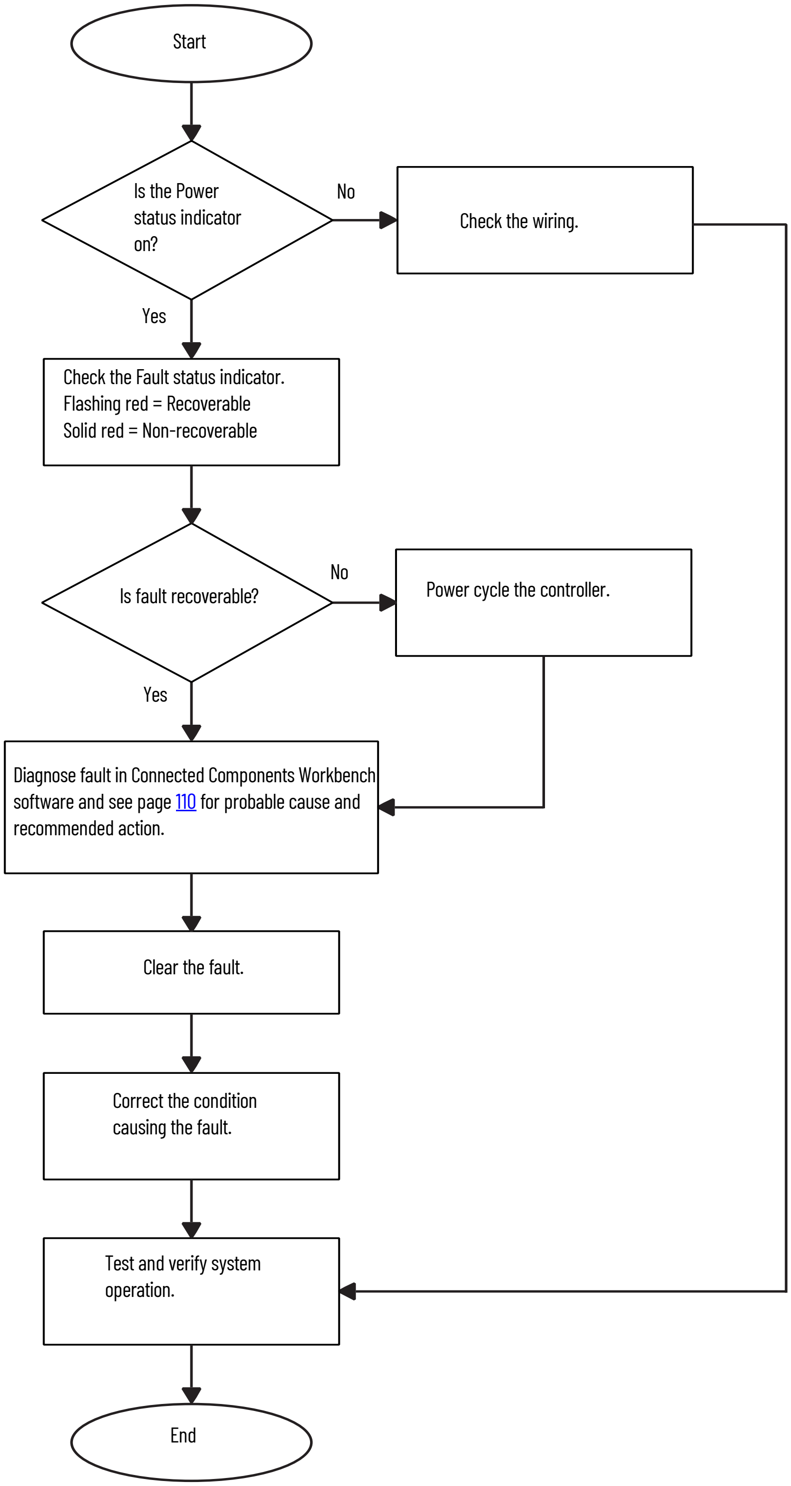
1. Launch the Connected Component Workbench software.
2. Connect to your Micro800 controller.
3. In Project Organizer, right-click the Micro800 controller.
4. Select Diagnose > Fault.

   The Fault Diagnostics tab displays.
5. Select the Get Fault Log button.
6. Save the fault log (.txt) file.

## Controller Error Recovery Model

Use the error recovery model shown in Figure 50 to help you diagnose software and hardware problems in the micro controller. The model provides common questions you might ask to help troubleshoot your system. See the recommended pages within the model for further help.

Figure 50 - Error Recovery Model


Start
Is the Power status indicator on?
No
Check the wiring.
Yes
Check the Fault status indicator.
Flashing red = Recoverable
Solid red = Non-recoverable
Is fault recoverable?
No
Power cycle the controller.
Yes
Diagnose fault in Connected Components Workbench software and see page 110 for probable cause and recommended action.
Clear the fault.
Correct the condition causing the fault.
Test and verify system operation.
End

## 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

# Quick Starts

This chapter covers some common tasks and quickstart instructions that are aimed to make you familiar with the Connected Component 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 to update 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.

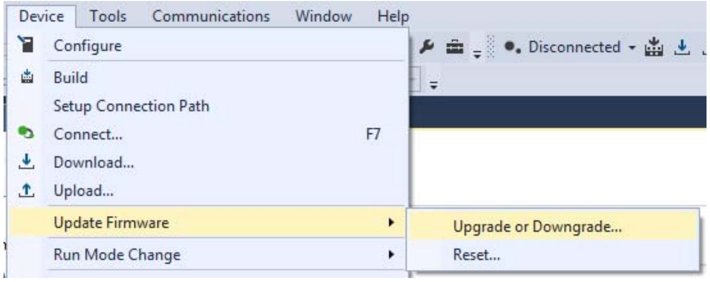
**IMPORTANT** If you have forgotten the password for the controller, use the Reset option to clear the password.

On Micro820 controllers, you can update your controllers through the Ethernet port and the USB of the 2080-REMLCD plug-in module.

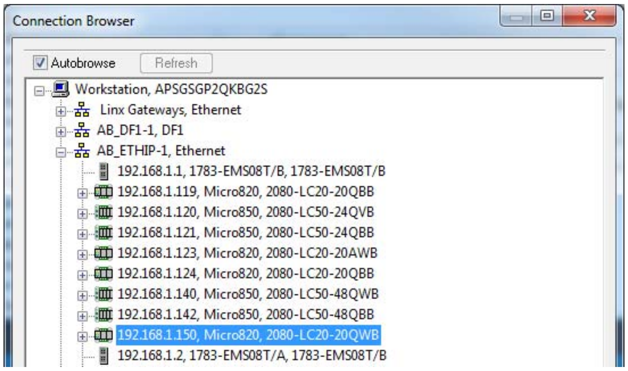
**IMPORTANT** To update your controller over USB successfully, connect only one controller to your computer, and do not perform the update in a virtual machine such as VMware.

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.

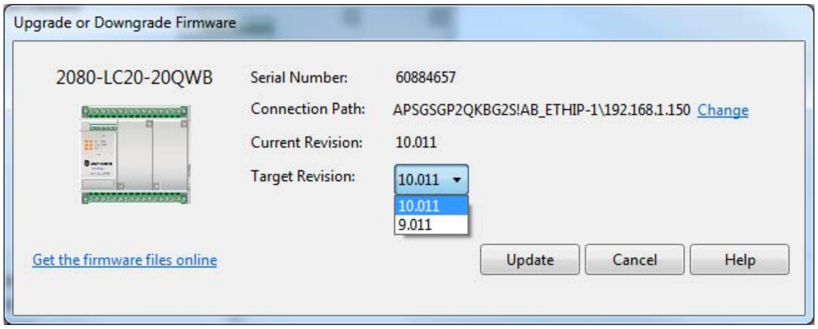


Connection Browser

☑ Autobrowse  Refresh

- Workstation, APSGSGP2QKBG2S
  - Linx Gateways, Ethernet
  - AB_DF1-1, DF1
  - AB_ETHIP-1, Ethernet
    - 192.168.1.1, 1783-EMS08T/B, 1783-EMS08T/B
    - 192.168.1.119, Micro820, 2080-LC20-20QBB
    - 192.168.1.120, Micro850, 2080-LC50-24QVB
    - 192.168.1.121, Micro850, 2080-LC50-24QBB
    - 192.168.1.123, Micro820, 2080-LC20-20AWB
    - 192.168.1.124, Micro820, 2080-LC20-20QBB
    - 192.168.1.140, Micro850, 2080-LC50-48QWB
    - 192.168.1.142, Micro850, 2080-LC50-48QBB
    - 192.168.1.150, Micro820, 2080-LC20-20QWB
    - 192.168.1.2, 1783-EMS08T/A, 1783-EMS08T/B

3. In the Upgrade or Downgrade Firmware dialog box, select the desired Target Revision to update the controller.



Upgrade or Downgrade Firmware

2080-LC20-20QWB

Serial Number: 60884657

Connection Path: APSGSGP2QKBG2S!AB_ETHIP-1\192.168.1.150 Change

Current Revision: 10.011

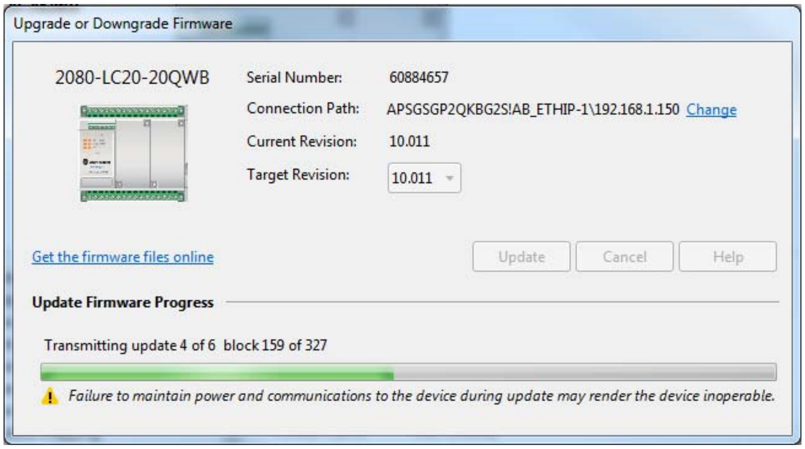
Target Revision: 10.011 (10.011, 9.011)

Get the firmware files online

If you do not see the desired firmware revision in the drop-down list, you can download that firmware revision by selecting 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.



Upgrade or Downgrade Firmware

2080-LC20-20QWB

Serial Number: 60884657

Connection Path: APSGSGP2QKBG2S!AB_ETHIP-1\192.168.1.150 Change

Current Revision: 10.011

Target Revision: 10.011

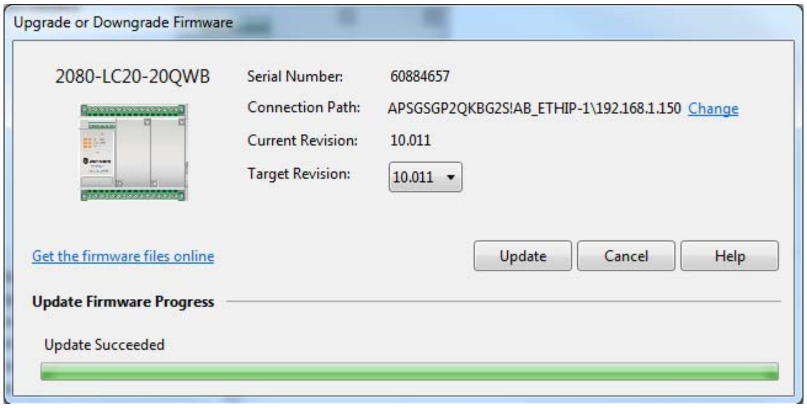
Get the firmware files online

Update Firmware Progress

Transmitting update 4 of 6 block 159 of 327

⚠ *Failure to maintain power and communications to the device during update may render the device inoperable.*

5. After the update is completed, the status is shown in the dialog box.



Upgrade or Downgrade Firmware

2080-LC20-20QWB

Serial Number: 60884657

Connection Path: APSGSGP2QKBG2S!AB_ETHIP-1\192.168.1.150 Change

Current Revision: 10.011

Target Revision: 10.011

Get the firmware files online

Update Firmware Progress

Update Succeeded

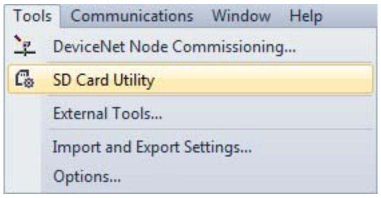
| IMPORTANT | After updating the controller, some microSD cards may not be detected. Remove and insert the microSD card, or power cycle the controller if you encounter this issue. |
|---|---|

## Firmware Update From microSD Card

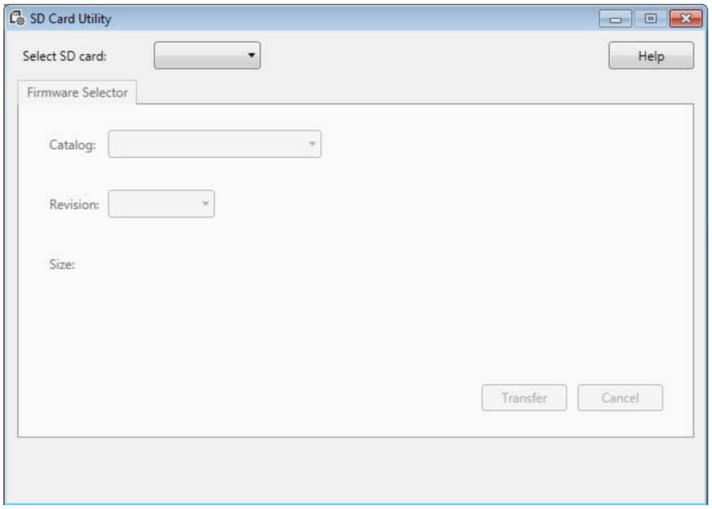
With Connected Components Workbench software version 8.0 or later, you can update your Micro820 controller from the microSD card and with ControlFLASH. This is two-step process: First you transfer the firmware to the microSD card using the SD Card Utility, then you edit the ConfigMeFirst.txt file to initiate the update process. See the following instructions for performing the firmware update from the microSD card.

### *Step 1 – Transfer the Firmware to the microSD Card*

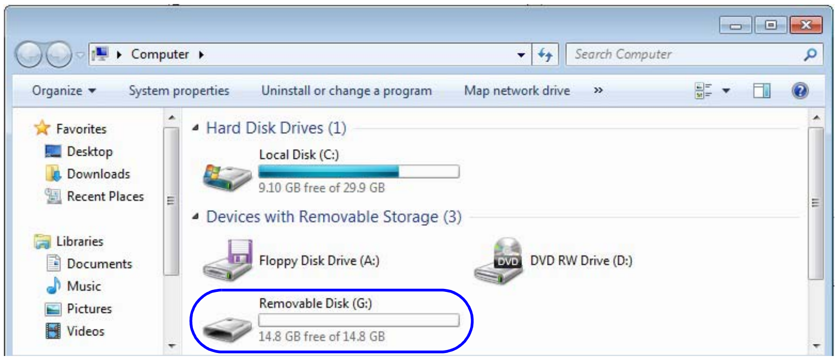
1. Launch the Connected Components Workbench software.
2. Select Tools -> SD Card Utility.



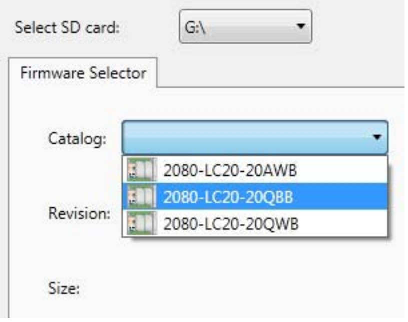
The SD Card Utility window appears.



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 is using the drive letter "G".



4. Select the catalog number of your Micro820 controller.



Select SD card: G:\

Firmware Selector

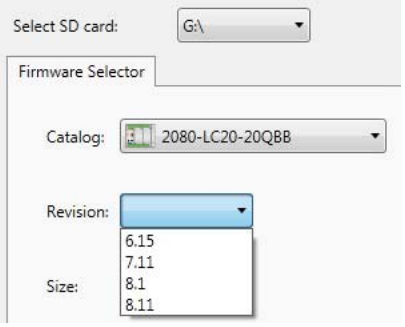
Catalog:

2080-LC20-20AWB
2080-LC20-20QBB
2080-LC20-20QWB

Revision:

Size:

5. Select the firmware revision you want to update your Micro820 controller with.



Select SD card: G:\

Firmware Selector

Catalog: 2080-LC20-20QBB

Revision:

6.15
7.11
8.1
8.11

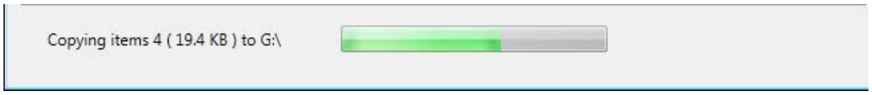
Size:

The list of firmware revisions are 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.

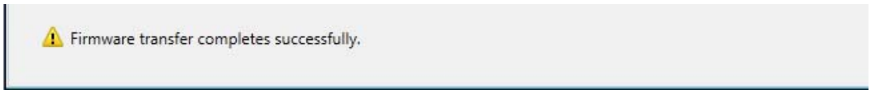
| IMPORTANT | You must sign in to the Rockwell Automation website before downloading a firmware revision. |
|---|---|

Close and relaunch the Connected Components Workbench software, then open the SD Card Utility again. The revision should now appear in the list.

6. Select 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.

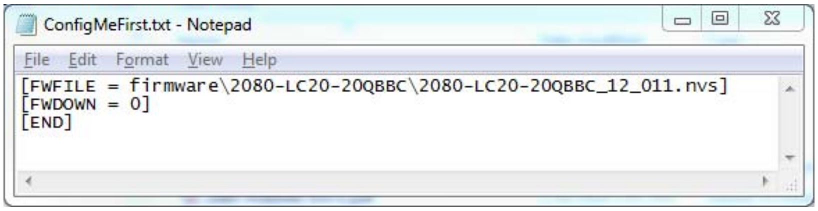
### *Step 2 – Edit the ConfigMeFirst.txt File*

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 in Table 52. You must add these settings at the beginning of the file.

**Table 52 - New ConfigMeFirst.txt Configuration Settings for Flash Upgrade**

| Setting | Takes Effect On... | Description |
|---|---|---|
| **Firmware update settings** | | |
| [FWFILE] | Power-up | File path location of the firmware revision on the microSD card.<br>The default location is in the following format:<br>firmware\<catalog number>\<filename of firmware> |
| [FWDOWN] | Power-up | Sets whether to upgrade or downgrade the controller firmware from the current revision.<br>0 = Upgrade firmware; 1 = Downgrade firmware<br>**IMPORTANT:** Firmware Upgrade occurs if the [FWFILE] setting points to a newer version of firmware file compared to the current firmware in the controller, irrespective of [FWDOWN] setting. |

**Figure 51 - Example of ConfigMeFirst.txt File for Flash Upgrade**



```
[FWFILE = firmware\2080-LC20-20QBBC\2080-LC20-20QBBC_12_011.nvs]
[FWDOWN = 0]
[END]
```

After you have edited the file, insert the microSD card into the controller. Power cycle 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 upgrading a Micro820 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.

#### Fault Status Indicator Description

| State | Indicates |
|---|---|
| Steady Red | Fault |
| Flashing Green | Run |

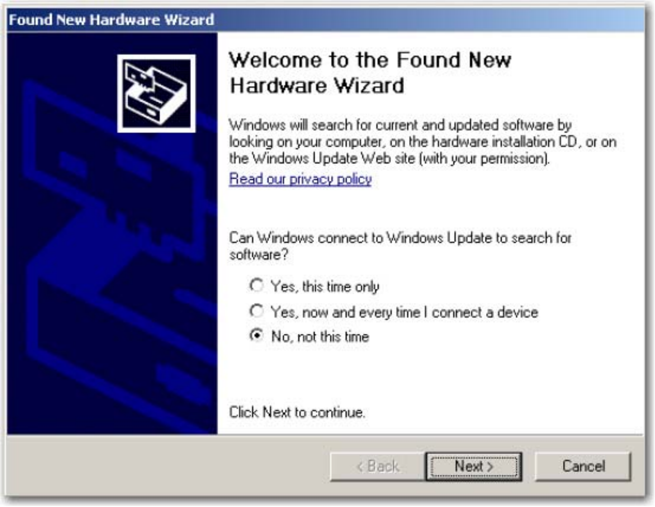
For a list of firmware and series compatibility, see the release notes for firmware revision 11.011 or later, on the Product Compatibility and Download Center (PCDC) at rok.auto/pcdc.

## Establish Communications between RSLinx and a Micro820 Controller Through USB

This quick start shows you how to get RSLinx® RSWho to communicate with a Micro820 controller through USB. Micro820 controller uses the 2080_REMLCD_xxxx driver.

RSLinx® Classic is installed as part of the Connected Components Workbench software installation process. The minimum version of RSLinx Classic with full Micro820 controller support is 3.60.01 (released on December 2013).

1. Power up the Micro820 controller.
2. Connect the USB A/B cable directly between your PC and the USB port on the 2080-REMLCD plug-in module.
3. Windows should discover the new hardware. Select No, not this time and then select Next.



Welcome to the Found New Hardware Wizard

Windows will search for current and updated software by looking on your computer, on the hardware installation CD, or on the Windows Update Web site (with your permission).

Read our privacy policy

Can Windows connect to Windows Update to search for software?

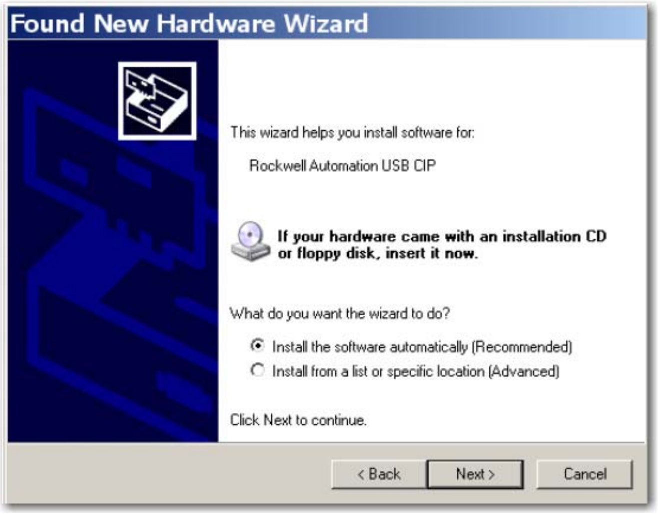
○ Yes, this time only

○ Yes, now and every time I connect a device

● No, not this time

Click Next to continue.

4. Select Install the software automatically (Recommended), and then select Next.



This wizard helps you install software for:

Rockwell Automation USB CIP

**If your hardware came with an installation CD or floppy disk, insert it now.**

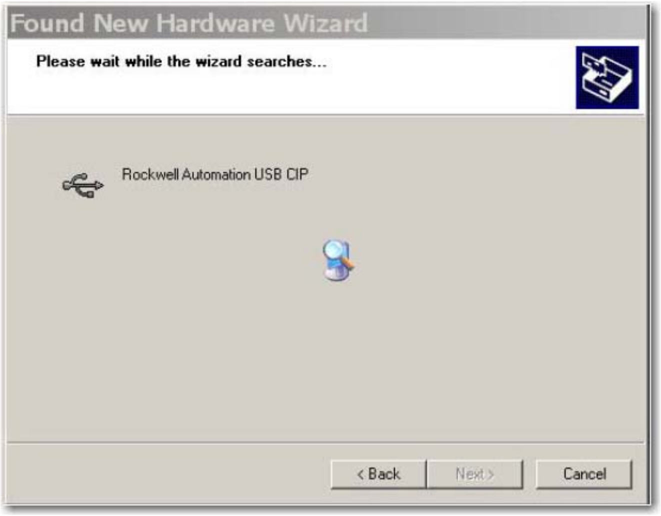
What do you want the wizard to do?

● Install the software automatically (Recommended)

○ Install from a list or specific location (Advanced)

Click Next to continue.

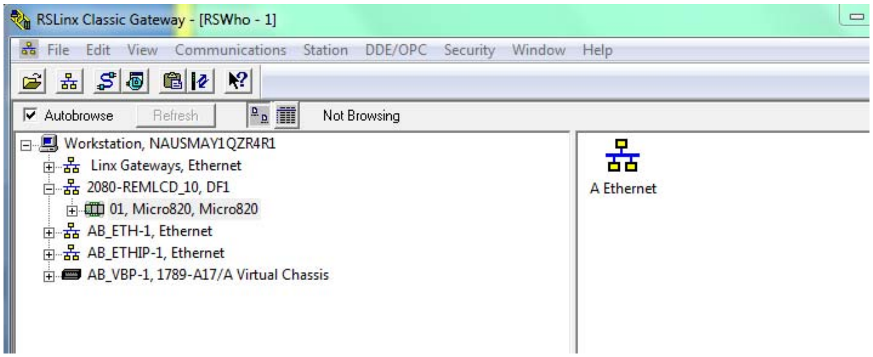
The Wizard searches for new hardware.



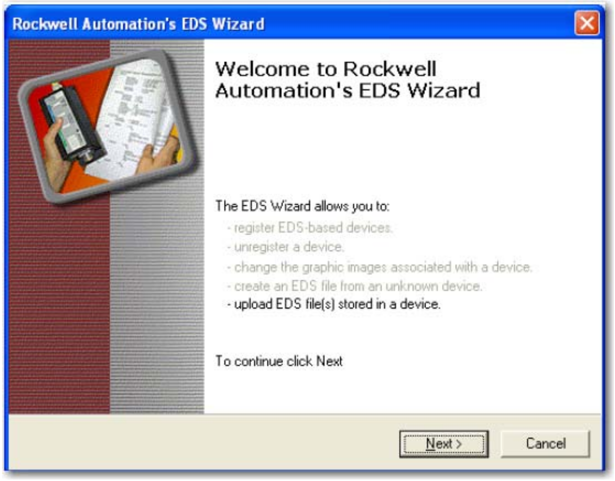
**Please wait while the wizard searches...**

Rockwell Automation USB CIP

5. Open RSLinx Classic and run RSWho by selecting the icon.


RSLinx Classic Gateway - [RSWho - 1]
File Edit View Communications Station DDE/OPC Security Window Help
Autobrowse
Refresh
Not Browsing
Workstation, NAUSMAY1QZR4R1
Linx Gateways, Ethernet
2080-REMLCD_10, DF1
01, Micro820, Micro820
AB_ETH-1, Ethernet
AB_ETHIP-1, Ethernet
AB_VBP-1, 1789-A17/A Virtual Chassis
A Ethernet

6. On the EDS Wizard that appears, select Next to continue.

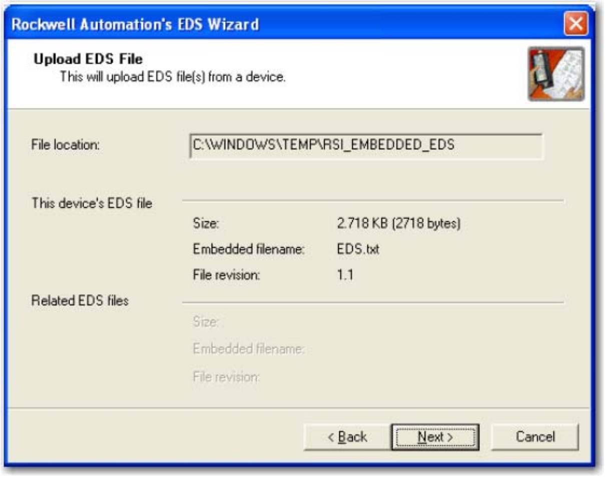


Welcome to Rockwell Automation's EDS Wizard

The EDS Wizard allows you to:
- register EDS-based devices.
- unregister a device.
- change the graphic images associated with a device.
- create an EDS file from an unknown device.
- upload EDS file(s) stored in a device.

To continue click Next

7. Follow the prompts to upload and install the EDS file.

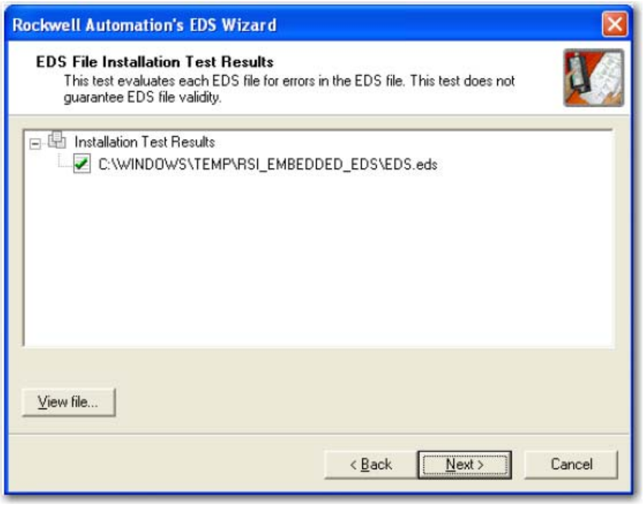


**Upload EDS File**
This will upload EDS file(s) from a device.

File location: C:\WINDOWS\TEMP\RSI_EMBEDDED_EDS

This device's EDS file
Size: 2.718 KB (2718 bytes)
Embedded filename: EDS.txt
File revision: 1.1

Related EDS files
Size:
Embedded filename:
File revision:



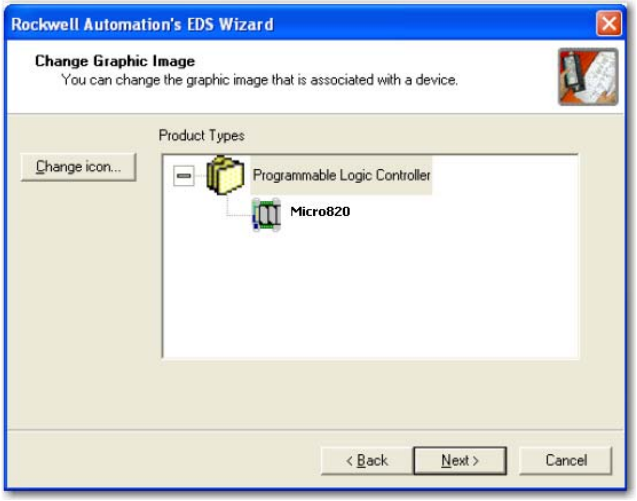
Rockwell Automation's EDS Wizard

**EDS File Installation Test Results**

This test evaluates each EDS file for errors in the EDS file. This test does not guarantee EDS file validity.

Installation Test Results

☑ C:\WINDOWS\TEMP\RSI_EMBEDDED_EDS\EDS.eds



Rockwell Automation's EDS Wizard

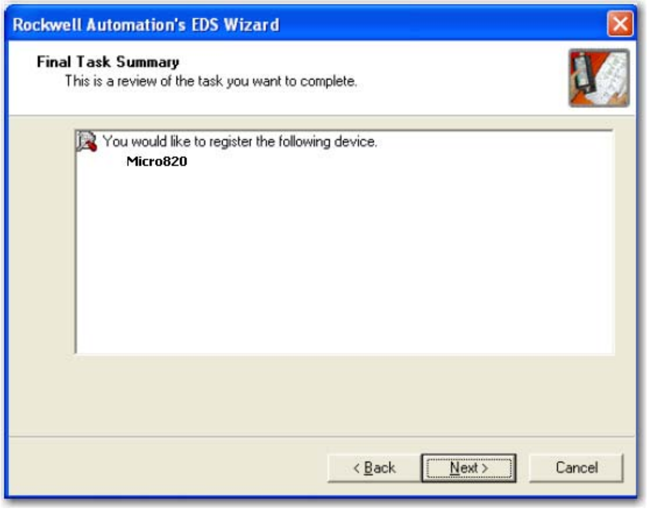
**Change Graphic Image**

You can change the graphic image that is associated with a device.

Product Types

Programmable Logic Controller

Micro820



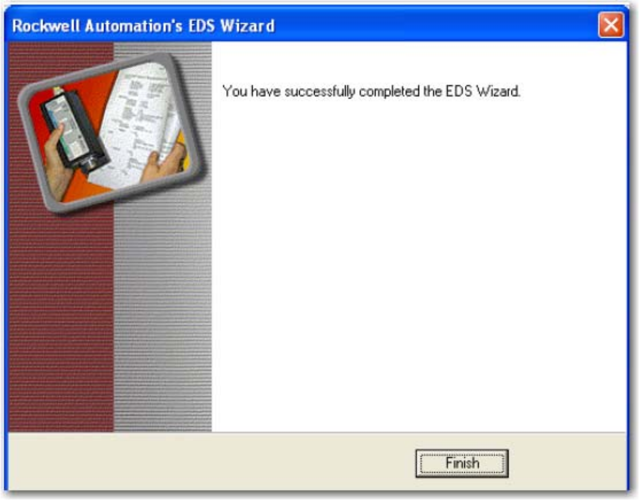
Rockwell Automation's EDS Wizard

**Final Task Summary**

This is a review of the task you want to complete.

You would like to register the following device.

Micro820


Rockwell Automation's EDS Wizard

You have successfully completed the EDS Wizard.

8. Select Finish to complete.

## Configure Controller Password

Set, change, and clear the password on a target controller through the Connected Components Workbench software.

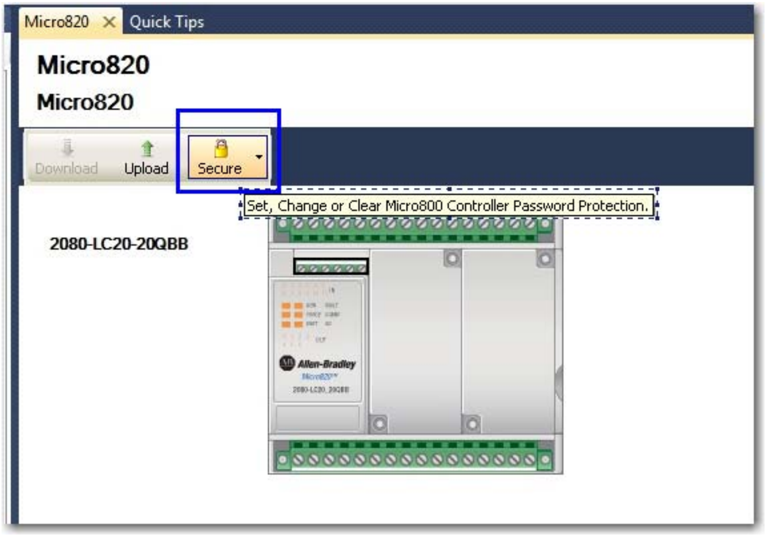
| | |
|---|---|
| **IMPORTANT** | The following instructions are supported in Connected Components Workbench software version 2 and Micro800 controllers with firmware revision 2.<br>For more information about the controller password feature on Micro800 controllers, see Controller Security on page 65. |

### 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. |

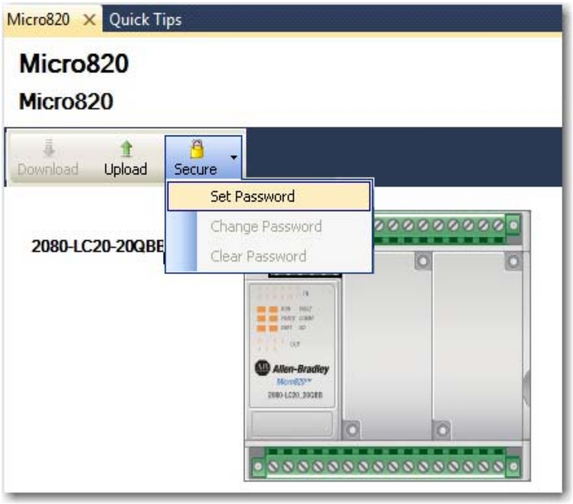
In the following instructions, the Connected Components Workbench software is connected to the Micro800 controller.

1. In 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.

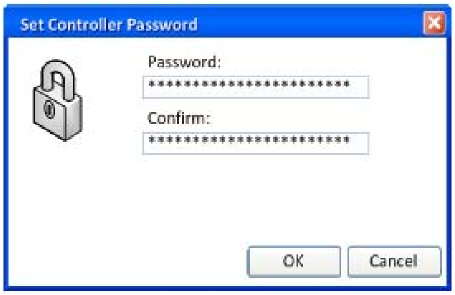


Set, Change or Clear Micro800 Controller Password Protection.

3. Select Secure. Select Set Password.



4. The Set Controller Password dialog appears. Provide a password. Confirm the password by providing it again in the Confirm field.


Set Controller Password

Password: ***********************

Confirm: ***********************

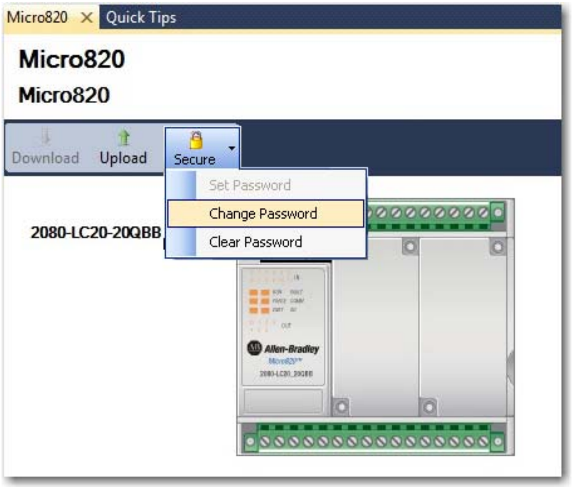
Passwords must have at least eight characters to be valid.

5. Select OK.
   Once a password is created, any new sessions that try to connect to the controller must supply the password to gain exclusive access to the target controller.

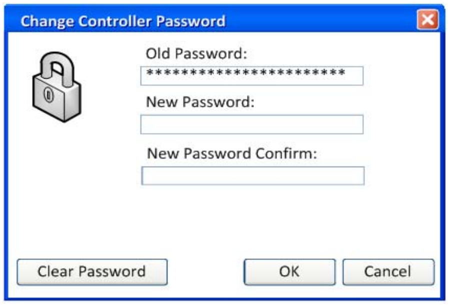
## Change Password

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.



Change Controller Password

Old Password: ************************

New Password:

New Password Confirm:

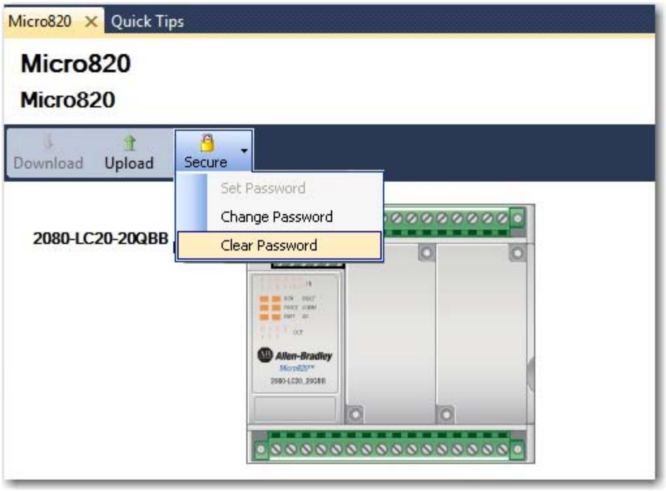
3. Select OK.

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 button. Select Clear Password.



2. The Clear Password dialog appears. Enter Password.
3. Select OK to clear the password.

The controller requires no password on any new session.

# Forcing I/Os

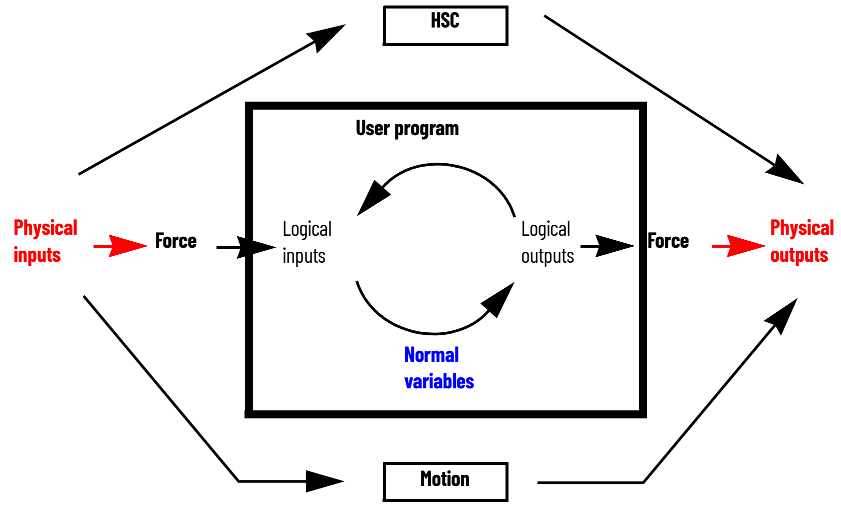
| IMPORTANT | This section generally talks about forcing I/O in Micro800 controllers. Some elements may not apply to certain models (for example, Micro810 and Micro820 controllers do not support PTO motion). |
|---|---|

Inputs are logically forced. LED 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 which execute independently from the User Program scan. For example, for motion, Drive Ready input cannot be forced.

Unlike inputs, outputs are physically forced. LED status indicators do show forced values and the user program does not use forced values.

The following diagram illustrates forcing behavior.

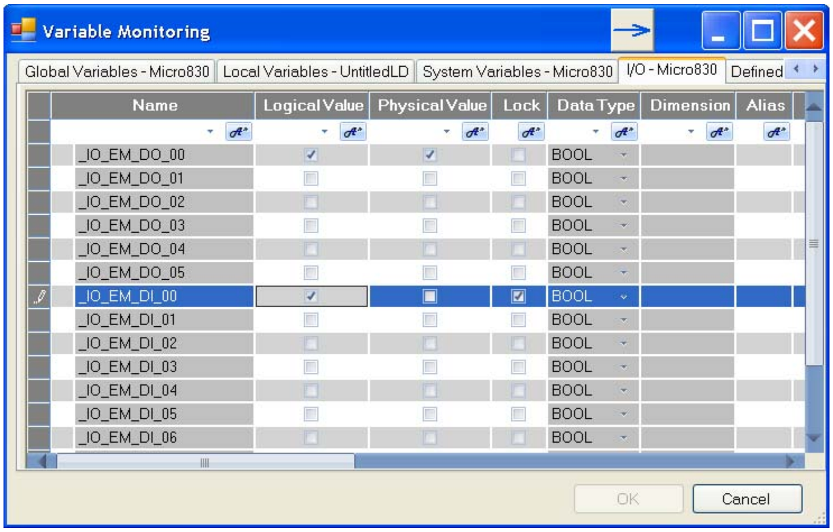


- Status indicators always match the physical value of I/O.
- Normal, non-physical internal variables cannot be forced.
- Special functions such as HSC and Motion cannot be forced.

**ATTENTION:** Forcing variable 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 you cannot force a Physical Input and cannot force a Logical Output.


Variable Monitoring
Global Variables - Micro830
Local Variables - UntitledLD
System Variables - Micro830
I/O - Micro830
Defined
OK
Cancel

| Name | Logical Value | Physical Value | Lock | Data Type | Dimension | Alias |
|---|---|---|---|---|---|---|
| _IO_EM_DO_00 | ☑ | ☑ | □ | BOOL | | |
| _IO_EM_DO_01 | □ | □ | □ | BOOL | | |
| _IO_EM_DO_02 | □ | □ | □ | BOOL | | |
| _IO_EM_DO_03 | □ | □ | □ | BOOL | | |
| _IO_EM_DO_04 | □ | □ | □ | BOOL | | |
| _IO_EM_DO_05 | □ | □ | □ | BOOL | | |
| _IO_EM_DI_00 | ☑ | □ | ☑ | BOOL | | |
| _IO_EM_DI_01 | □ | □ | □ | BOOL | | |
| _IO_EM_DI_02 | □ | □ | □ | BOOL | | |
| _IO_EM_DI_03 | □ | □ | □ | BOOL | | |
| _IO_EM_DI_04 | □ | □ | □ | BOOL | | |
| _IO_EM_DI_05 | □ | □ | □ | BOOL | | |
| _IO_EM_DI_06 | □ | □ | □ | BOOL | | |

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. The following is an example program in Structured Text.

```
(* Read System Information including Force Enable bit *)
SYS_INFO_1(TRUE);

(* Turn on Warning Light if Forces are Enabled *)
If SYS_INFO_1.Sts.ForcesInstall = TRUE THEN
   _IO_EM_DO_05 := TRUE;
ELSE
   _IO_EM_DO_05 := FALSE;
END_IF;
```

### I/O Forces After a Power Cycle

After a controller is power cycled, all I/O forces are cleared from memory.

## Using Run Mode Change

Run Mode Change allows the user 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.

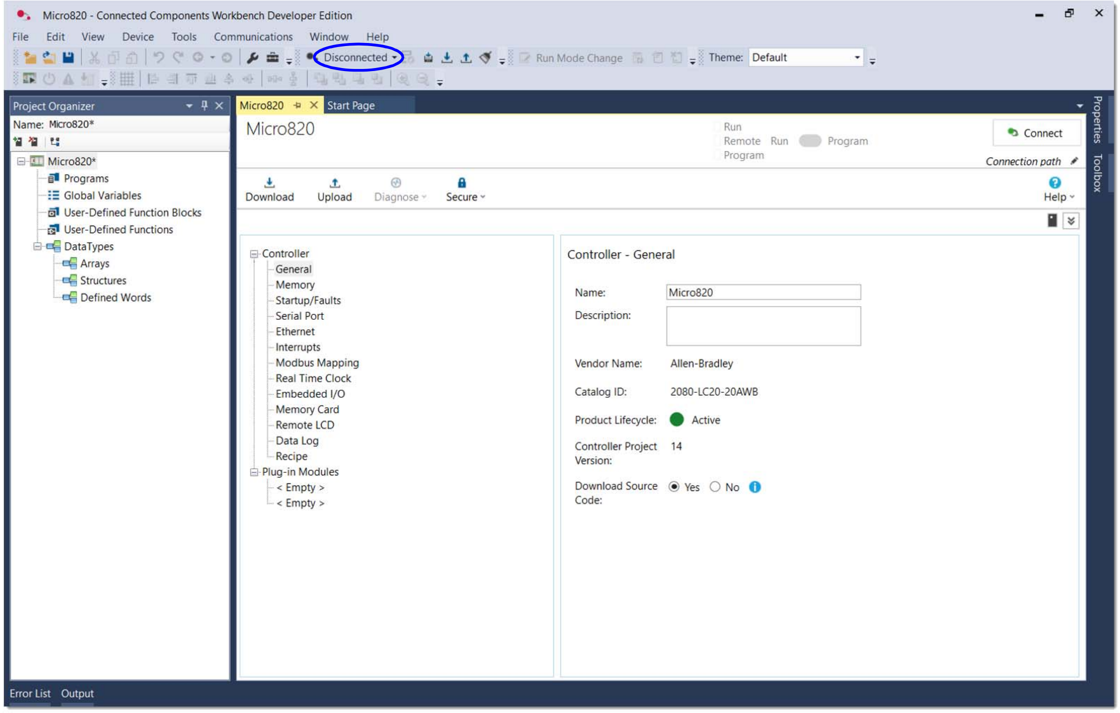
**IMPORTANT** The following requirements must be met to use Run Mode Change:

- Micro820 controller firmware revision 8.0 or later, and
- Connected Components Workbench Developer Edition software, version 8.0 or later.

The following sample project guides you through the creation of a simple application for a Micro820 controller without any plug-in modules, and how to use the Run Mode Change feature.

### Create the Project

1. Create a new project for a Micro820 controller without any plug-ins. Observe that the controller is disconnected.



Controller - General

Name: Micro820

Description:

Vendor Name: Allen-Bradley

Catalog ID: 2080-LC20-20AWB

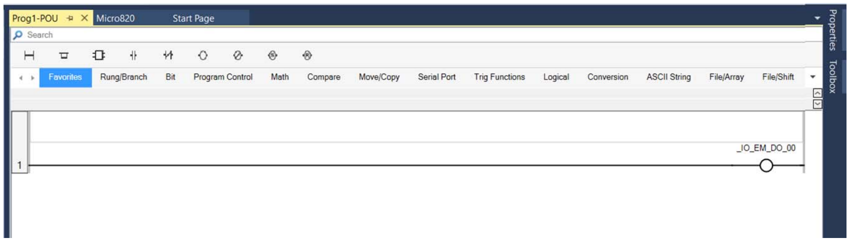
Product Lifecycle: ● Active

Controller Project Version: 14

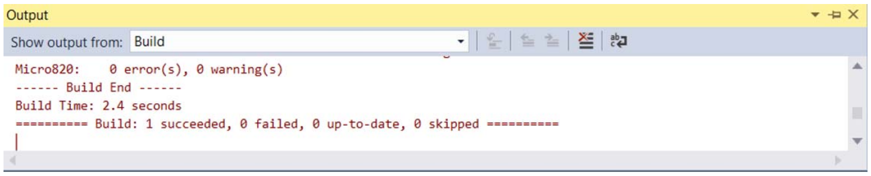
Download Source Code: ● Yes ○ No

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".

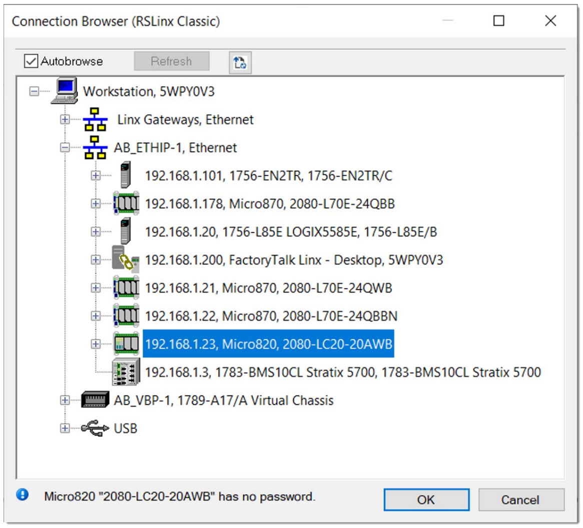


5. Build the project.



```
Micro820:    0 error(s), 0 warning(s)
------ Build End ------
Build Time: 2.4 seconds
========== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped ==========
```

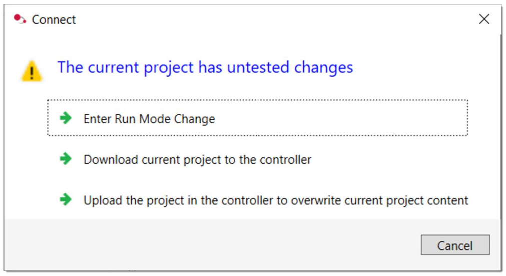
6. Download the project to the controller.
In the Connection Browser dialog, select the Micro820 controller.



- Workstation, 5WPY0V3
  - Linx Gateways, Ethernet
  - AB_ETHIP-1, Ethernet
    - 192.168.1.101, 1756-EN2TR, 1756-EN2TR/C
    - 192.168.1.178, Micro870, 2080-L70E-24QBB
    - 192.168.1.20, 1756-L85E LOGIX5585E, 1756-L85E/B
    - 192.168.1.200, FactoryTalk Linx - Desktop, 5WPY0V3
    - 192.168.1.21, Micro870, 2080-L70E-24QWB
    - 192.168.1.22, Micro870, 2080-L70E-24QBBN
    - 192.168.1.23, Micro820, 2080-LC20-20AWB
    - 192.168.1.3, 1783-BMS10CL Stratix 5700, 1783-BMS10CL Stratix 5700
  - AB_VBP-1, 1789-A17/A Virtual Chassis
  - USB

Micro820 "2080-LC20-20AWB" has no password.

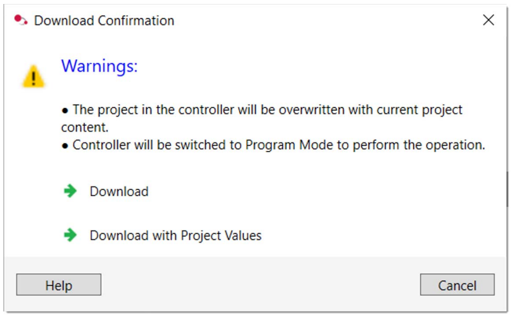
7. Select Download current project to the controller.



The current project has untested changes

- Enter Run Mode Change
- Download current project to the controller
- Upload the project in the controller to overwrite current project content

8. Select Download to confirm.

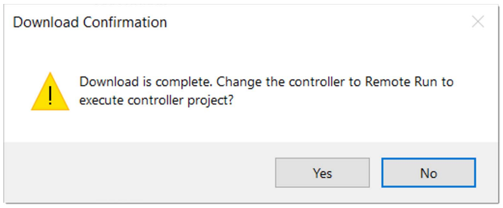


Warnings:

- The project in the controller will be overwritten with current project content.
- Controller will be switched to Program Mode to perform the operation.

- Download
- Download with Project Values

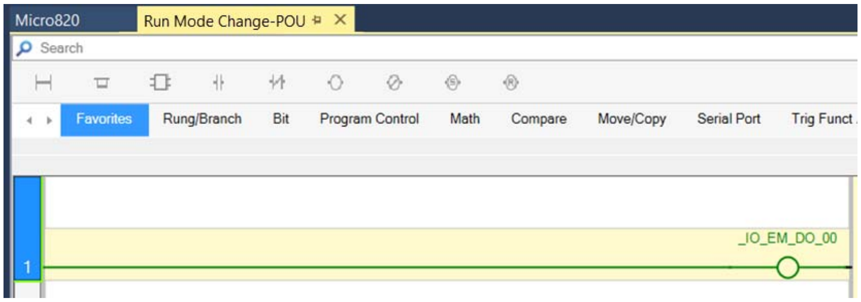
9. When the project is downloaded to the controller, a prompt asking to change the controller to Remote Run mode appears. Select Yes.



Download Confirmation

Download is complete. Change the controller to Remote Run to execute controller project?

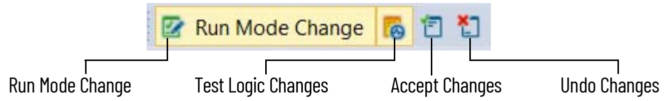
10. Observe that the controller is now in Debug mode.



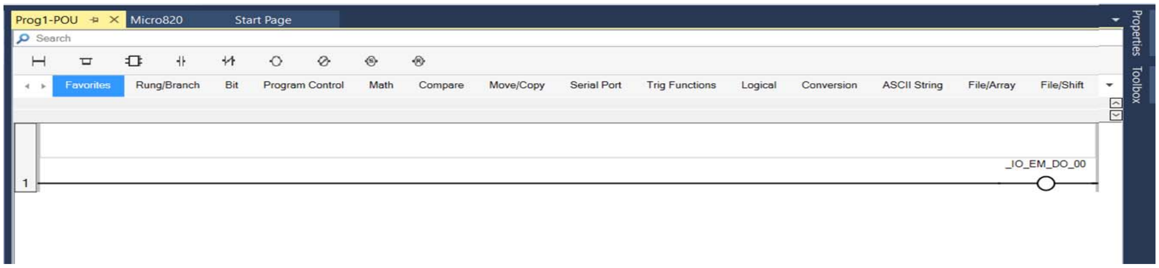
| IMPORTANT | From Connected Components Workbench version 8.0 onwards, selecting “Yes” to change the controller to Remote Run mode after a downloading a project automatically switches it to Debug mode. |
|---|---|

## Edit the Project Using Run Mode Change

### Run Mode Change Toolbar



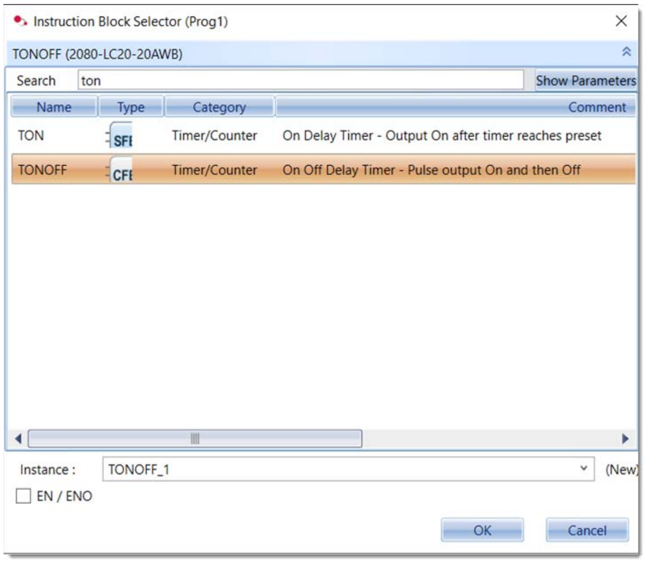
1. Select the Run Mode Change icon.
Observe that the controller goes into Edit mode and is still connected.



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 choose 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).



Instruction Block Selector (Prog1)

TONOFF (2080-LC20-20AWB)

Search: ton

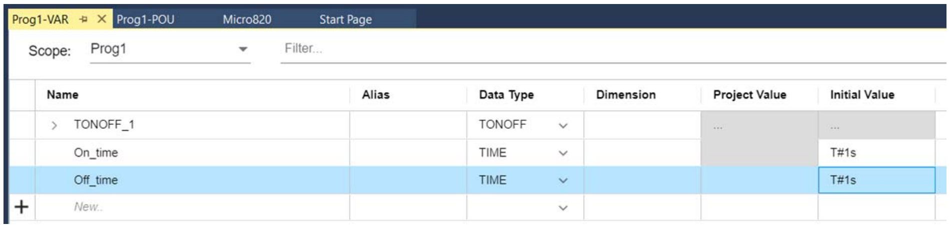
Show Parameters

| Name | Type | Category | Comment |
|---|---|---|---|
| TON | SFB | Timer/Counter | On Delay Timer - Output On after timer reaches preset |
| TONOFF | CFB | Timer/Counter | On Off Delay Timer - Pulse output On and then Off |

Instance : TONOFF_1 (New)

□ EN / ENO

Configure the Instruction Block to trigger every one second.

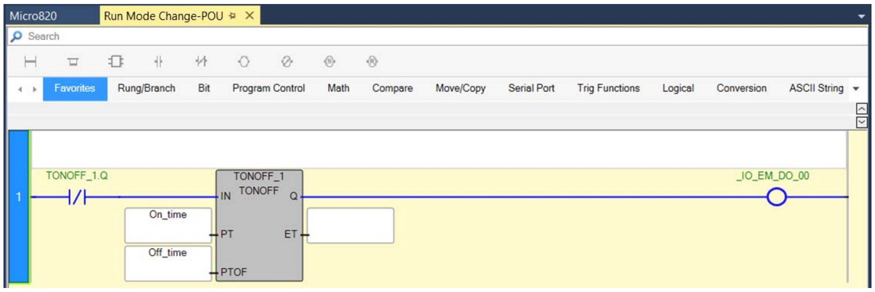


Scope: Prog1

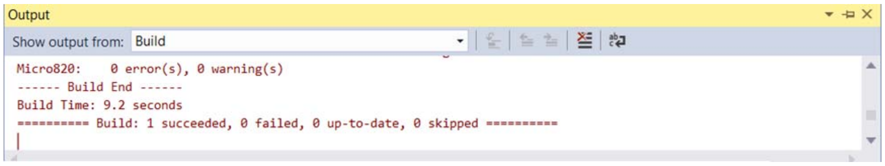
Filter...

| Name | Alias | Data Type | Dimension | Project Value | Initial Value |
|---|---|---|---|---|---|
| TONOFF_1 | | TONOFF | | ... | ... |
| On_time | | TIME | | | T#1s |
| Off_time | | TIME | | | T#1s |
| New.. | | | | | |

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 left of the recently added Instruction Block.



5. Click the Test Logic Changes icon to build the project and download it to the controller.



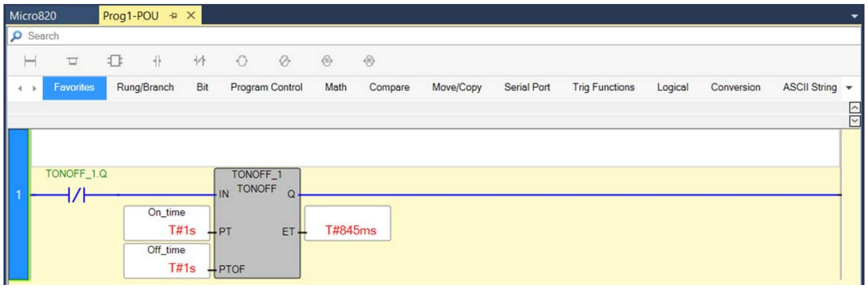
Output

Show output from: Build

```
Micro820:    0 error(s), 0 warning(s)
------ Build End ------
Build Time: 9.2 seconds
========== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped ==========
```

**IMPORTANT** When you perform a Test Logic, or undo changes after the Test Logic is completed, any active communication instructions are aborted while the changes download to the controller.

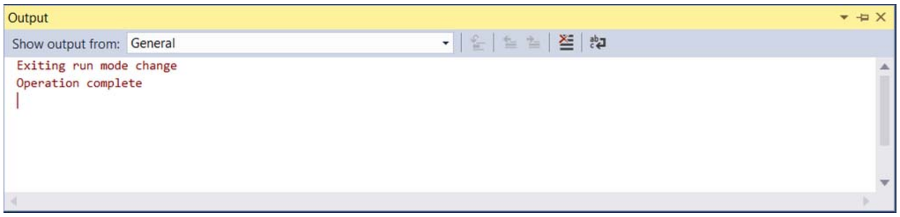
6. The controller automatically goes into Debug mode and display the updated project.



7. You can now choose either to Undo or Accept the changes to the project.

### *To Undo the Changes*

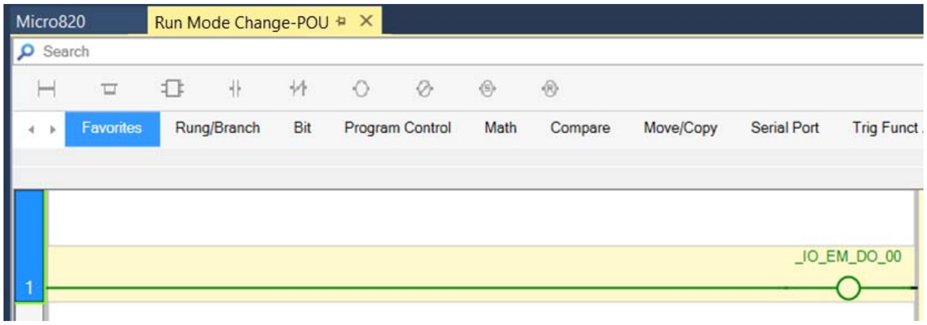
1. Select the Undo Changes icon.
2. The changes are discarded and the original project is restored to the controller.



```
Exiting run mode change
Operation complete
```

| **IMPORTANT** | When you perform a Test Logic, or undo changes after the Test Logic is completed, any active communication instructions are aborted while the changes download to the controller. |
|---|---|

Observe that original project is shown and the controller is in Debug mode.



### *To Accept the Changes*

1. Click the Accept Changes icon.
2. Observe that only the Run Mode Change icon is now enabled and the controller remains in Debug mode.

## Notes:

# PID Function Blocks

The PID function block has parameter naming similar to RSLogix 500® and is recommended if you are already familiar with programming in RSLogix 500. The IPIDCONTROLLER function block has the advantage of supporting auto tune.

Table 53 - 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<br>FALSE = Manual operation using CVManual |
| | CVManual | CV when in manual mode |
| | Enable | TRUE = Start execution with current input parameters.<br>FALSE = CV equals zero. |
| | Active | TRUE = PID state is running.<br>FALSE = PID state is stopped. |
| | Error | TRUE = PID has an error.<br>FALSE = PID has no errors. |
| | ErrorID | PID Error ID |
| **IPIDCONTROLLER specific parameters** | | |
| Auto | – | TRUE = Normal operation of PID<br>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 is applied. |
| AutoTune | | TRUE = Auto tune<br>FALSE = No Auto tune |
| ATParameters | – | Auto tune parameters |
| ATWarning | | Auto tune warning |
| OutGains | | Gains from Auto tune |
| Initialize | | Used for Auto tune |

## PID Function Block

This function block diagram shows the arguments in the PID function block.

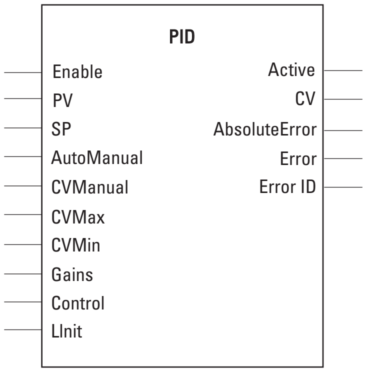


Table 54 explains the arguments that are used in this function block.

**Table 54 - PID Arguments**

| Parameter | Parameter Type | Data Type | Description |
|---|---|---|---|
| Enable | Input | BOOL | Enable instruction<br>TRUE = Start execution with current input parameters.<br>FALSE = CV equals zero. |
| PV | Input | REAL | Process Value. This value is typically read from an analog input module.<br>The SI unit must be the same as Setpoint. |
| SP | Input | REAL | The set point value for the process |
| AutoManual | Input | BOOL | Auto or manual mode selection:<br>TRUE = Normal operation of PID<br>FALSE = Manual operation using CVManual |
| CVManual | Input | REAL | Control value input defined for manual mode operation. The valid range for CVManual is:<br>CVMin < CVManual < CVMax |
| CVMin | Input | REAL | Control value minimum limit.<br>If CV < CVMin, then CV = CVMin.<br>If CVMin > CVMax, an error occurs. |
| CVMax | Input | REAL | Control value maximum limit.<br>If CV > CVMax, then CV = CVMax.<br>If CVMax < CVMin, an error occurs. |
| Gains | Input | PID_GAINS | Gains of PID for controller<br>Use the PID_GAINS data type to configure the Gains parameter. |
| Control | Input | BOOL | Control direction of the process:<br>TRUE = Direct acting, such as Cooling<br>FALSE = Reverse acting, such as Heating |
| LInit | Input | BOOL | Reserved for future use |
| Active | Output | BOOL | Status of the PID controller:<br>TRUE = PID state is running.<br>FALSE = PID state is stopped. |
| CV | Output | REAL | The control value output.<br>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<br>TRUE = PID has an error.<br>FALSE = PID has no errors. |
| ErrorID | Output | USINT | A unique numeric that identifies the error. The errors are defined in PID error codes. |

GAIN_PID Data Type

| Parameter | Parameter Type | Data Type | Description |
|---|---|---|---|
| Kc | Input | REAL | Controller gain for PID.<br>Proportional and Integral are dependent on this gain (≥ 0.0001).<br>Increasing Kc improves response time but also increases overshoot and oscillation of the PID.<br>If Kc is invalid, an error occurs. |
| Ti | Input | REAL | Time integral constant in seconds (≥ 0.0001).<br>Increasing Ti decreases overshoot and oscillation of the PID.<br>If Ti is invalid, an error occurs. |
| Td | Input | REAL | Time derivative constant in seconds (≥ 0.0).<br>When Td equals 0, then there is no derivative action and PID becomes a PI controller.<br>Increasing Td reduces the overshot and removes the oscillation of the PID controller.<br>If Td is invalid, an error occurs. |
| FC | Input | REAL | Filter constant (≥ 0.0).<br>Recommended range for FC is 0…20.<br>Increasing FC smooths the response of the PID controller.<br>If FC is invalid, an error occurs. |

Table 55 - 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<br>CVManaul is invalid. |
| 7 | CVManual > CVMax<br>CVManual is invalid. |

## IPIDCONTROLLER Function Block

This function block diagram shows the arguments in the IPIDCONTROLLER function block.

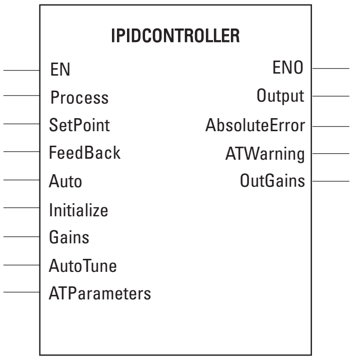


Table 56 explains the arguments used in this function block.

Table 56 - IPIDCONTROLLER Arguments

| Parameter | Parameter Type | Data Type | Description |
|---|---|---|---|
| EN | Input | BOOL | Function block enable.<br>TRUE = Execute function.<br>FALSE = Do not execute function.<br>Applicable to Ladder Diagram programs. |
| Process | Input | REAL | Process value, which is the value measured from the process output. |

Table 56 - IPIDCONTROLLER Arguments (Continued)

| Parameter | Parameter Type | Data Type | Description |
|---|---|---|---|
| SetPoint | Input | REAL | The set point value for the process |
| Feedback | Input | REAL | Feedback signal, which is the value of the control variable applied to the process.<br>For example, the feedback can be IPIDCONTROLLER output. |
| Auto | Input | BOOL | Operating modes of PID controller:<br>TRUE = Normal operation of PID<br>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 Auto tune sequences. |
| Gains | Input | GAIN_PID | Gains PID for IPIDCONTROLLER.<br>Use the GAIN_PID data type to define the parameters for the Gains input. |
| AutoTune | Input | BOOL | TRUE = When AutoTune is TRUE, and Auto and Initialize are FALSE, the AutoTune sequence is started.<br>FALSE = Do not start AutoTune. |
| ATParameters | Input | AT_Param | Auto tune parameters<br>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 value are:<br>0 = No auto tune done<br>1 = In auto tune mode<br>2 = Auto tune done<br>-1 = Error 1: Input automatically set to TRUE, no auto tune possible.<br>-2 = Error 2: Auto tune error, the ATDynamSet expired. |
| OutGains | Output | GAIN_PID | Gains calculated from AutoTune Sequences.<br>Use GAIN_PID data type to define the OutGains output. |
| ENO | Output | BOOL | Enable out.<br>Applicable to Ladder Diagram programs. |

Table 57 - GAIN_PID Data Type

| Parameter | Type | Description |
|---|---|---|
| DirectActing | BOOL | Types of acting:<br>TRUE = Direct acting, output moves 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.<br>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)<br>**Proportional gain for PID (P_Gain)**<br>A higher proportional gain causes a larger change in the output based upon the difference between the PV (measured process value) and SV (set point 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)<br>**Time integral value for PID**<br>A smaller integral time constant causes a faster change in the output based upon the difference between the PV (measured process value) and SV (set point 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 make it more stable, but PV approaches the SV at a slower rate. |

Table 57 - GAIN_PID Data Type (Continued)

| Parameter | Type | Description |
|---|---|---|
| TimeDerivative | REAL | Time derivative value for PID (> 0.0)<br>**Time derivative value for PID (Td)**<br>A smaller derivative time constant causes a faster change in the output based upon the rate of change of the difference between PV (measured process value) and SV (set point 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 taken into consideration. |
| DerivativeGain | REAL | Derivative gain for PID (>= 0.0)<br>**Derivative gain for PID (D_Gain)**<br>A higher derivative gain causes a larger change in the output based upon the rate of change of the difference between the PV (measured process value) and SV (set point 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.<br>**If derivative gain is set to zero, it disables the derivative portion of the PID.** |

Table 58 - AT_Param Data Type

| Parameter | Type | Description |
|---|---|---|
| Load | REAL | Load parameter for auto tuning. This is the output value when starting AutoTune. |
| Deviation | REAL | Deviation for auto tuning. This is the standard deviation 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:<br>TRUE = Resets output to zero<br>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 (72 °F), and a fluctuation of 21.7…22.5 °C (71…72.5 °F) is observed, the value of ATParams.Deviation will be (22.5-21.7)/2=0.4.

## How to Auto Tune

Before you auto tune, you need to:

- Verify that your system is constant when there is no control. For example, for temperature control, 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 shown in Table 59:

Table 59 - GAIN Parameter Values

| GAIN Parameter | Value |
|---|---|
| DirectActing | According to operation:<br>TRUE (for example, Cooling), or<br>FALSE (for example, Heating) |
| DerivativeGain | 0.5 |
| ProportionalGain | 0.0001 |
| TimeIntegral | 0.0001 |
| TimeDerivative | 0.0 |

- Set the AT_Parameter as shown in Table 60:

Table 60 - 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 you want.<br>IMPORTANT: If a load of 40 gives you a process value of 30 °C (86 °F) over a period of time, and you want to tune your system to 30 °C (86 °F), you should set the load to 40. |
| Deviation | This parameter plays a significant role in the auto tune process. The method of deriving this value is explained later in this section. It is not necessary to set this parameter prior to auto tuning. 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 auto tuning. 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 auto tune 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 auto tune process completes. Set this parameter to FALSE to leave the output at load value after the auto tune process completes. |

To auto tune, 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 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 auto tune process is successful.
9. Get the tuned value from the 'OutGains'.

## How Auto Tune Works

The auto tune process begins when the '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 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 second peak, calculations commence and a set of gain will be generated to parameter OutGains.

## Troubleshooting an Auto Tune Process

You can tell what is going on behind the auto tune process from the sequences of control output. Here are some known sequences of control output and what it means if auto tune fails. For the ease of illustrating the sequence of control output, we define:

Load: 50
Step: 20

**Table 61 - 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 |

**Table 62 - 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 |

**Table 63 - 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 |

**Table 64 - 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 52 - Example of a PID Application**

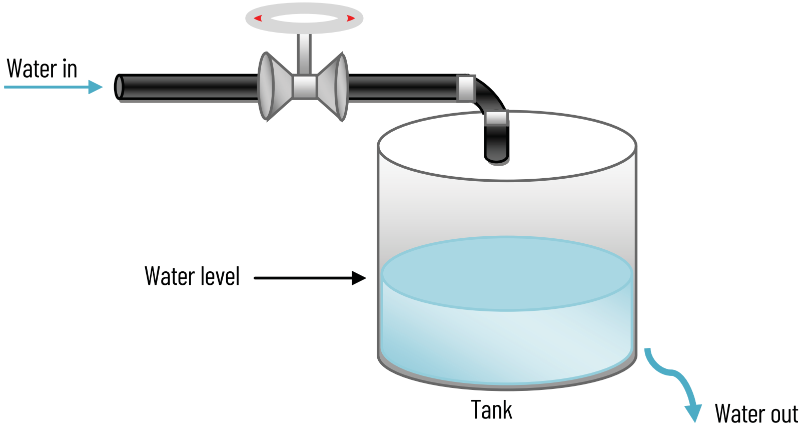


Figure 52 shows a basic water level control system, to maintain a preset water level in the tank. A solenoid valve is used to control incoming water, filling the tank at a preset rate. Similarly, outflowing water is controlled at a measurable rate.

### *IPID Auto Tuning for First and Second Order Systems*

Auto tune of IPID can only work on first and second order systems.

A first order system can be described by a single 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 element 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 $\tau$ 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 will be RC, the forcing function will be the ambient temperature and the system state variable will be the fluid temperature.

A second order system can be described by two independent energy storage elements which exchange stored energy. Examples of second order systems are a motor driving a disk flywheel with the motor 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.

**Figure 53 - PID Code Sample**

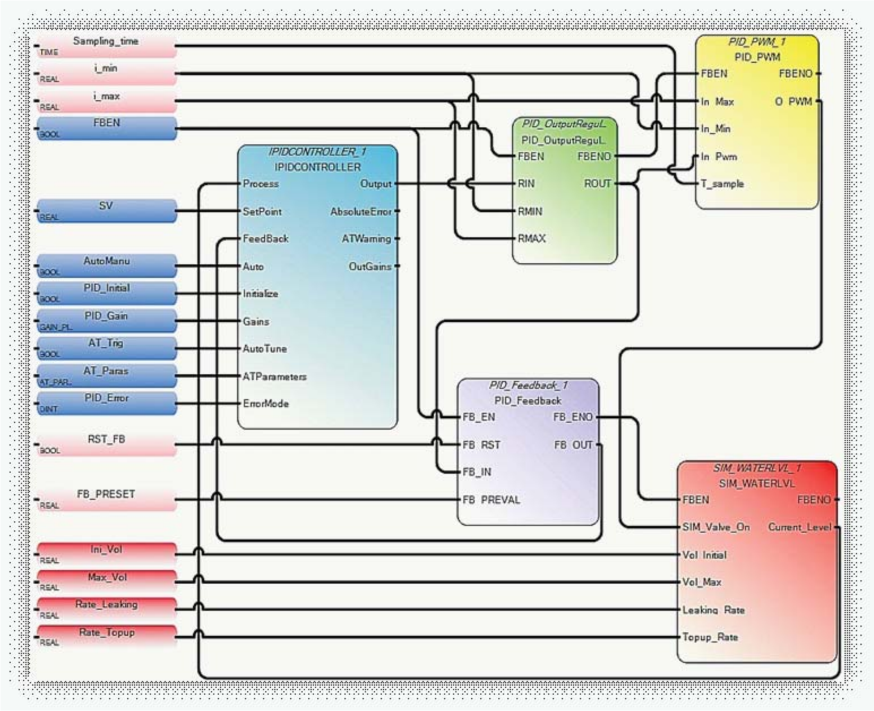


Figure 53 shows sample code for controlling the PID application example (Figure 52). The sample code, developed using Function Block Diagrams, consists of a pre-defined function block, IPIDCONTROLLER, and four user-defined function blocks. The four user-defined function blocks are:

- PID_OutputRegulator
  Regulates the output of IPIDCONTROLLER within a safe range to verify that there is no damage to the hardware 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, 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 52).

| | |
|---|---|
| **IMPORTANT** | User Program Scan Time<br>The auto tuning 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, or the sampling theorem.<br>Also, you must trigger the function block at a relatively constant time interval. |

## 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.

Rockwell Otomasyon Ticaret A.Ş. Kar Plaza İş Merkezi E Blok Kat:6 34752, İçerenköy, İstanbul, Tel: +90 (216) 5698400 EEE Yönetmeliğine Uygundur

Connect with us.

rockwellautomation.com — expanding **human possibility**®

AMERICAS: Rockwell Automation, 1201 South Second Street, Milwaukee, WI 53204-2496 USA, Tel: (1) 414.382.2000
EUROPE/MIDDLE EAST/AFRICA: Rockwell Automation NV, Pegasus Park, De Kleetlaan 12a, 1831 Diegem, Belgium, Tel: (32) 2663 0600
ASIA PACIFIC: Rockwell Automation SEA Pte Ltd, 2 Corporation Road, #04-05, Main Lobby, Corporation Place, Singapore 618494, Tel: (65) 6510 6608
UNITED KINGDOM: Rockwell Automation Ltd., Pitfield, Kiln Farm, Milton Keynes, MK11 3DR, United Kingdom, Tel: (44)(1908) 838-800

Publication 2080-UM005G-EN-E - December 2023
Supersedes Publication 2080-UM005F-EN-E - March 2019