USER MANUAL

SPECTRUM CONTROLS

An Allient Company

User's Manual Pub. 0300274-05 Rev. A0

Micro800^{TM} BACnet Communication Module

Catalog Number: 2080sc-BAC

SILVER Technology Partner

A ROCKWELL AUTOMATION PARTNER

Important Notes

1. Please read all the information in this owner’s guide before installing the product.
2. The information in this owner's guide applies to hardware Series A and firmware version 1.1 or later.
3. This guide assumes that the reader has a full working knowledge of the relevant processor.

Notice

Limited Warranty

Spectrum Controls, Inc. warrants that its products are free from defects in material and workmanship under normal use and service, as described in Spectrum Controls, Inc. literature covering this product, for a period of 1 year. The obligations of Spectrum Controls, Inc. under this warranty are limited to replacing or repairing, at its option, at its factory or facility, any product which shall, in the applicable period after shipment, be returned to the Spectrum Controls, Inc. facility, transportation charges prepaid, and which after examination is determined, to the satisfaction of Spectrum Controls, Inc., to be thus defective.

This warranty shall not apply to any such equipment which shall have been repaired or altered except by Spectrum Controls or which shall have been subject to misuse, neglect, or accident. In no case shall the liability of Spectrum Controls, Inc. exceed the purchase price. The aforementioned provisions do not extend the original warranty period of any product which has either been repaired or replaced by Spectrum Controls, Inc.

Table of Contents

IMPORTANT NOTES .... **II**

NOTICE .... **II**

LIMITED WARRANTY .... **II**

TABLE OF CONTENTS .... **III**

PREFACE .... **IV**

CHAPTER 1 MODULE OVERVIEW **1-1**

SECTION 1.1 GENERAL DESCRIPTION 1-1
SECTION 1.2 ENVIRONMENT AND ENCLOSURE 1-3
SECTION 1.3 PREVENT ELECTROSTATIC DISCHARGE 1-5
SECTION 1.4 PARTS LIST 1-6
SECTION 1.5 HARDWARE FEATURES 1-7
1.5.1 I/O MS/TP Serial I/O Connector 1-7
1.5.2 Ethernet Connector 1-7
SECTION 1.6 LED INDICATOR 1-8
SECTION 1.7 MODULE DC POWER SPECIFICATIONS 1-8
SECTION 1.8 MODULE CHASSIS EARTH GROUND 1-8

CHAPTER 2 INSTALLATION AND WIRING **2-1**

SECTION 2.1 INSERT MODULE INTO CONTROLLER 2-1
SECTION 2.2 MS/TP WIRING BEST PRACTICES 2-2

CHAPTER 3 CONFIGURING THE MODULE USING SOFTWARE **3-1**

SECTION 3.1 CONFIGURING THE SYSTEM 3-4
SECTION 3.2 SYSTEM CONFIGURATION SAVE AND RELOAD BUTTONS 3-10
SECTION 3.3 SETTING UP TAGS 3-10
SECTION 3.4 EDITING XML TAG INFORMATION 3-13
SECTION 3.5 SAVING TAG SETUP AND RELOADING USING XML 3-14
SECTION 3.6 GENERATING STRUCTURED TEXT 3-15
SECTION 3.7 USING THE CCW STRUCTURED TEXT EXAMPLE 3-16
3.7.1 Program BAC_Slot1 3-17
3.7.2 BACnet_Convert Function Block 3-23
3.7.3 CCW Export and Import of Function Blocks 3-28
3.7.4 IP Address and User Interface Password Recovery 3-28
3.7.5 Upgrading the Software from Version 1.n to Version 2.n 3-29
3.7.6 Memory Mapping 3-29
3.7.7 Data Type Range 3-31
3.7.8 BACnet Diagnostic Tools 3-32
SECTION 3.8 VIEWING VERSION INFORMATION 3-35
SECTION 3.9 VIEWING LOG INFORMATION 3-35
SECTION 3.10 UPGRADING THE SOFTWARE 3-36
SECTION 3.11 SAVING CHANGES 3-36
SECTION 3.12 RELOADING SYSTEM CONFIGURATION 3-36

CHAPTER 4 IMPLEMENTING THE BACNET PROTOCOL **4-1**

SECTION 4.1 BACNET OBJECT TYPES 4-1
SECTION 4.2 PARAMETER OFFSET FOR MODULE BLOCK 4-5
SECTION 4.3 USER INTERFACE OPTIONS 4-5
SECTION 4.4 TECHNICAL ASSISTANCE 4-6

APPENDIX A CONFIGURATION INFORMATION **A-1**

INDEX **I-1**

Preface

Read this preface to familiarize yourself with the rest of the manual. This preface covers the following topics:

- Who should use this manual
- How to use this manual
- Rockwell Automation technical support
- Documentation
- Conventions used in this manual

Who Should Use This Manual

Use this manual if you are responsible for designing, installing, programming, or troubleshooting control systems that use Allen-Bradley I/O and/or compatible controllers, such as CompactLogix and ControlLogix.

How to Use This Manual

As much as possible, we organized this manual to explain, in a task-by-task manner, how to install, configure, program, operate, and troubleshoot a control system using the Micro800™ BACnet Communication Module.

Rockwell Automation Technical Support

For technical support, please contact your local Rockwell Automation TechConnect Office for all Spectrum products. Contact numbers are as follows:

• USA	1-440-646-6900 (US/global, English only
• United Kingdom	+44 0 1908 635 230 (EU phone, UK local)
• Australia, China, India, and other East Asia locations:	1-800-722-778 or +61 39757 1502
• Mexico	001-888-365-8677
• Brazil	55-11-5189-9500 (general support)
• Europe	+49-211-41553-630 (Germany/general support)

or send an email to support@spectrumcontrols.com

Documentation

If you would like a .PDF version of a manual, you can download a free electronic version at www.spectrumcontrols.com

Conventions Used in This Manual

The following conventions are used throughout this manual:

- Bulleted lists (like this one) provide information not procedural steps.
- Numbered lists provide sequential steps or hierarchical information.
- *Italic* type is used for emphasis.
- **Bold** type identifies headings and sub-headings:

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

ATTENTION ⚠	Actions ou situations risquant d'entraîner des blessures pouvant être mortelles, des dégâts matériels ou des pertes financières. Les messages « Attention » vous aident à identifier un danger, à éviter ce danger et en discerner les conséquences.

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

Chapter 1
Module Overview

Section 1.1 General Description

The 2080sc-BAC Communication Module is a two-channel communication, plug-in module for use with Rockwell Automation Micro800™ systems as a slave node on a BACnet network.

The plug-in module supports two channels of data communications: one channel is configured for RS-485, half duplex serial communications, and the other channel is configured for 10/100M Ethernet full duplex serial communications. After installation, the module is configured via the Ethernet port. By default, the Ethernet port is the port that the module uses to communicate with external devices such as other BACnet modules or personal computers.

The module plugs into any spare plug-in slot on the PLC. The module interfaces with the controller via Asynchronous Parallel Interface (API) and communicates with other BACnet modules using the BACnet protocol. The module stores the data internally. During module setup, you map Micro800 PLC tags to BACnet tags so that the Micro800 system can receive, and respond to, BACnet messages.

The BACnet configuration software resides on the BACnet module. You use a web browser to access this software to configure the parameters for the module.

The module configuration includes setting module slave ID address, IP address, serial baud rate, and static or dynamic IP addressing etc.

The structured text is part of a ladder diagram running on the PLC. Ladder development for the system is done using Connected Component Workbench (CCW) software.

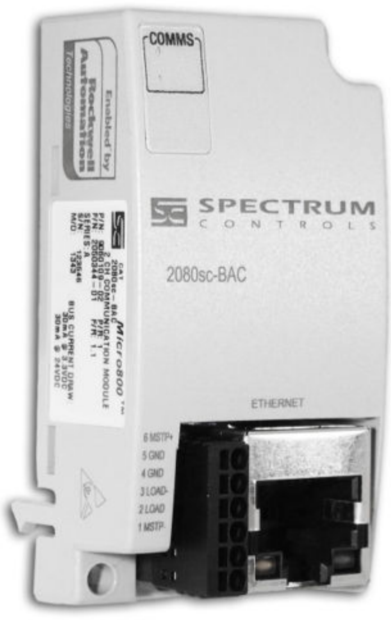
The amount of data that can be exchanged between the PLC and the module is determined by memory size on the BACnet module and the PLC.

Sequence of operation is as follows:

- BACnet module configuration: Configure IP address, configure slave id, configure password for downloading, configure tags name, type and index, configure module time using web browser.
- BACnet to PLC communication: The configuration software generates a structured text file that is copied to the CCW block that serves as part of the CCW programming code. The code moves data between the PLC and the module's backplane. Based on the PLC tag data type, the structured text assigns each PLC tag to a backplane memory offset of the module. For example, a USINT type tag is allocated 2 bytes of memory; a FLOAT type tag is allocated 4 bytes of memory; a BOOL type tag receives 1 bit of memory, and so on. Depending on the attribute of the tag as READ or READ/WRITE, the structured text will determine the direction the data moves between PLC and module backplane. If the tag has a READ attribute, the module only moves data from the PLC to the

module. If the tag is READ/WRITE, the module moves data from, and to, the PLC.

The module is designed to operate continuously. If an error occurs in the firmware, or the firmware stops running, the module is able to recover without user intervention in most situations. You may also use the configuration software to reset the module. You can always retrieve the last known good tag setting on the module, even if you reset the module back to the default state.



Power for the module is provided across the backplane. MS/TP signals from the field side are connected to the module via the 6-pin connector (RS-485). The Ethernet port handles BACnet/IP traffic. You may choose to use one of these; both cannot be used simultaneously.

Section 1.2 Environment and Enclosure

WARNING ⚠	This equipment is intended for use in a Pollution Degree 2 industrial environment, in overvoltage Category II applications (as defined in IEC publication 60664-1), at altitudes up to 2000 meters (6562 feet) without derating. This equipment is considered Group 1, Class A industrial equipment according to IEC/CISPR Publication 11. Without appropriate precautions, there may be potential difficulties ensuring electromagnetic compatibility in other environments due to conducted as well as radiated disturbance. This equipment is supplied as open-type equipment. It must be mounted within an enclosure that is suitably designed for those specific environmental conditions that will be present and appropriately designed to prevent personal injury resulting from accessibility to live parts. The enclosure must have suitable flame-retardant properties to prevent or minimize the spread of flame, complying with a flame spread rating of 5 VA, V2, V1, V0 (or equivalent) if non-metallic. The interior of the enclosure must be accessible only by the use of a tool. Subsequent sections of this publication may contain additional information regarding specific enclosure type ratings that are required to comply with certain product safety certifications. In addition to this publication, see: • Industrial Automation Wiring and Grounding Guidelines, Allen-Bradley publication 1770-4.1, for additional installation requirements. NEMA Standards publication 250 and IEC publication 60529, as applicable, for explanations of the degrees of protection provided by different types of enclosure.

WARNING ⚠	Cet équipement est prévu pour fonctionner en environnement industriel avec une pollution de niveau 2, dans des applications de surtension de catégorie II (telles que définies dans la publication 60664-1 de la CEI) et à une altitude maximum de 2000 m sans déclassement. Cet équipement est considéré comme étant un équipement industriel du Groupe 1, classe A selon CEI/CISPR 11. En l'absence de précautions appropriées, des problèmes de compatibilité électromagnétique peuvent survenir dans des environnements résidentiels et dans d'autres environnement en raison de perturbations conduites et rayonnées. Cet équipement est fourni en tant qu'équipement de type « ouvert ». Il doit être installé à l'intérieur d'une armoire fournissant une protection adaptée aux conditions d'utilisation ambiantes et suffisante pour éviter toute blessure pouvant résulter d'un contact direct avec des composants sous tension. L'armoire doit posséder des propriétés ignifuges capables d'empêcher ou de limiter la propagation des flammes, correspondant à un indice de propagation de 5VA, V2, V1, V0 (ou équivalent) dans le cas d'une armoire non métallique. L'accès à l'intérieur de l'armoire ne doit être possible qu'à l'aide d'un outil. Cette armoire doit permettre des connexions d'alimentation par un système de câblage de Classe I, Division 2, conformément au code électrique national (NEC). Certaines sections de la présente publication peuvent comporter des recommandations supplémentaires portant sur les indices de protection spécifiques à respecter pour maintenir la conformité à certaines normes de sécurité. En plus de cette publication, consultez: • La publication Rockwell Automation 1770-4.1, « Industrial Automation Wiring and Grounding Guidelines », pour d'autres critères d'installation. • La publication 250 de la norme NEMA ou la publication 60529 de la CEI, selon le cas, pour obtenir une description des indices de protection que fournissent les différents types d'armoires.

WARNING ⚠	Electrostatic discharge can damage integrated circuits or semiconductors if you touch bus connector pins. Follow these guidelines when you handle the module: • Touch a grounded object to discharge static potential. • Wear an approved wrist-strap grounding device. • Do not touch connectors or pins on component boards. • Do not touch circuit components inside the module. • If available, use a static-safe workstation. When not in use, keep the module in its static-shield box.

Section 1.3 Prevent Electrostatic Discharge

WARNING ⚠	Cet équipement est sensible aux décharges électrostatiques, lesquelles peuvent entraîner des dommages internes et nuire à son bon onctionnement. Conformez-vous aux directives suivantes lorsque vous manipulez cet équipement: • Touchez un objet mis à la terre pour vous décharger de toute électricité statique éventuelle. • Portez au poignet un bracelet antistatique agréé. • Ne touchez pas les connecteurs ni les broches figurant sur les cartes des composants. • Ne touchez pas les circuits internes de l’équipement. • Utilisez si possible un poste de travail antistatique. Lorsque vous n’utilisez pas l’équipement, stockez-le dans un emballage antistatique.

WARNING ⚠	To comply with the CE Low Voltage Directive (LVD), all connected I/O must be powered from a source compliant with the following: Safety Extra Low Voltage (SELV) or Protected Extra Low Voltage (PELV).

WARNING ⚠	Pour se conformer à la Directive basse tension CE, cet équipement doit être alimenté à partir d’une source ayant les caractéristiques suivantes: très basse tension de sécurité (TBTS) ou très basse tension de protection (TBTP).

Section 1.4 Parts List

Your package contains one Micro800 BACnet Communication Module, installation screws, and one Quick Start Guide.

You can choose to wire the plug-in before inserting it into the controller or wire it once the module is secured in place.

WARNING ⚠	• This equipment is considered Group 1, Class A industrial equipment according to IEC/CISPR 11. Without appropriate precautions, there may be difficulties with electromagnetic compatibility in residential and other environments due to conducted and radiated disturbance. • Be careful when stripping wires. Wire fragments that fall into the controller could cause damage. Once wiring is complete, make sure the controller is free of all metal fragments before removing the protective debris strip. • Do not wire more than 2 conductors on any single terminal. • If you insert or remove the plug-in module while power is on, 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. • Do not insert or remove the plug-in module while power is applied; otherwise, permanent damage to equipment may occur.

WARNING ⚠	• Cet équipement est considéré comme étant un équipement industriel du Groupe 1, classe A selon CEI/CISPR 11. En l'absence de precautions appropriées, des problèmes de compatibilité électromagnétique peuvent survenir dans des environnements résidentiels et dans d'autres environnements en raison de perturbations conduites et rayonnées. • Soyez vigilant en dénudant les fils. Tout fragment de fil tombé dans l'automate risquerait de le détériorer. Une fois le câblage terminé, veillez à ce que l'automate ne présente aucun copeau de métal avant de retirer la bande de protection. • Ne câblez pas plus de 2 conducteurs sur une même borne. • L'insertion ou le retrait du module enfichable sous tension peut provoquer un arc électrique, susceptible de provoquer une explosion dans un environnement dangereux. Assurez-vous que l'alimentation est coupée ou que l'environnement est classé non dangereux avant de poursuivre. • N'insérez pas et ne retirez pas le module enfichable quand l'équipement est sous tension, au risque de provoquer des dommages irrémédiables à l'équipement.

Section 1.5 Hardware Features

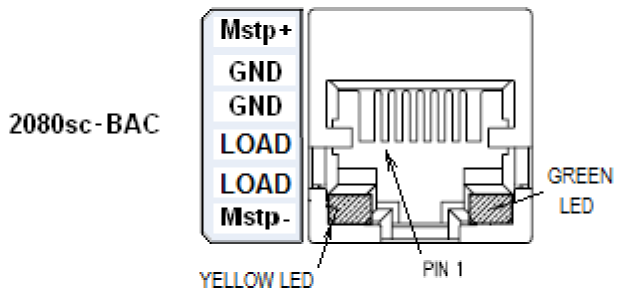
The module plugs into, and communicates with, a controller in the Micro800 family. BACnet communication I/O signals are connected to the module through a 6-pin terminal block or an RJ-45 connector:

1.5.1 I/O MS/TP Serial I/O Connector

NOTE	Pins in following table are listed from 6 to 1 to match connector on front panel of module.

The six-pin Connector pinouts are as follows:

Pin	Signal	Description
6	MSTP+	Positive signal on twisted pair.
5	GND	Twisted pair shield (if available).
4	GND	See Pin 5 above.
3	LOAD	Provides 120 Ohm terminal load (no polarity) when shorted with Pin 2. Short using a jumper.
2	LOAD	See Pin 3 above.
1	MSTP-	Negative signal on twisted pair.



1.5.2 Ethernet Connector

The Ethernet connector has a default IP address that may be changed during setup. The Ethernet connector may be used as an external communication port to a personal computer or to another BACnet module. The Ethernet connector is also used to configure the module.

The default IP address for the module is **169.254.3.3**. See Chapter 3, Configuring the Module Using Software.

Pinouts for the connector (crossover cable) are:

2080sc-BACnet Module	Personal Computer
1 TX+	**8** Not connected
2 TX-	**7** Not connected
3 RX+	**6** TX-
4 Not connected	**5** Not connected
5 Not connected	**4** Not connected
6 RX-	**3** TX+
7 Not connected	**2** RX-
8 Not connected	**1** RX+

Section 1.6 LED Indicator

A single LED indicator is provided with the module. The LED blinks for 5 seconds after a reset. The LED is steady green for ON.
Software Upgrade

The module software can be upgraded in the field.

Section 1.7 Module DC Power Specifications

The controller provides two Power Supplies to the module:

- 3.3 Volts (3.0 V Min, 3.6 V Max), Current Rating: 40 mA
- 24 Volts (20.4 V Min, 26.4 V Max), Current Rating: 50 mA

You may not use an external power source to power the module. Refer to the specifications in the Appendix for further information.

Section 1.8 Module Chassis Earth Ground

The Micro800 controller does not have a chassis (earth) ground. The 2080sc-BACnet module connects to an isolated ground, ISO-GND, which is exclusive to the external communication interfaces. The purpose of the isolated ground is to prevent possible interference on the I/O channels from permanently damaging the module itself.

Chapter 2
Installation and Wiring

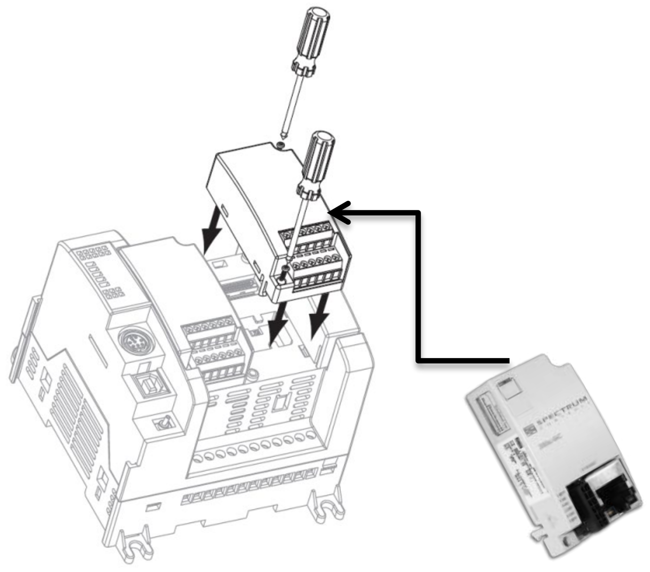
Section 2.1 Insert Module into Controller

Follow the instructions to insert and secure the plug-in module to the controller.

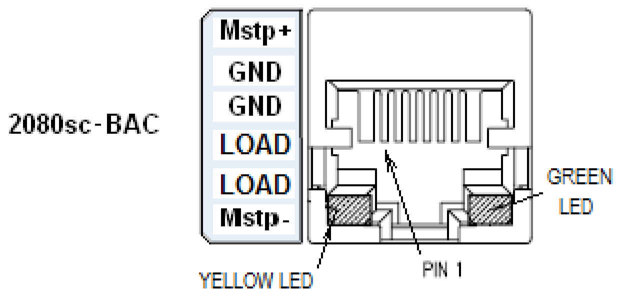
WARNING ⚠	Electrostatic discharge can damage integrated circuits or semiconductors if you touch bus connector pins. Follow these guidelines when you handle the module: • Touch a grounded object to discharge static potential. • Wear an approved wrist-strap grounding device. • Do not touch connectors or pins on component boards. • Do not touch circuit components inside the module. • If available, use a static-safe workstation. • When not in use, keep the module in its static-shield box.

WARNING ⚠	Cet équipement est sensible aux décharges électrostatiques, lesquelles peuvent entraîner des dommages internes et nuire à son bon onctionnement. Conformez-vous aux directives suivantes lorsque vous manipulez cet équipement: • Touchez un objet mis à la terre pour vous décharger de toute électricité statique éventuelle. • Portez au poignet un bracelet antistatique agréé. • Ne touchez pas les connecteurs ni les broches figurant sur les cartes des composants. • Ne touchez pas les circuits internes de l'équipement. • Utilisez si possible un poste de travail antistatique. • Lorsque vous n'utilisez pas l'équipement, stockez-le dans un emballage antistatique.

4. Position the plug-in module with the terminal block facing the front of the controller as shown. The 2080sc-BACnet module has a different front panel setup, but the installation in the controller is the same:



5. Snap the module into the module bay.
6. Using a screwdriver, tighten the supplied, self-tapping screw.
7. Wire the module using the 6-pin connector as shown:



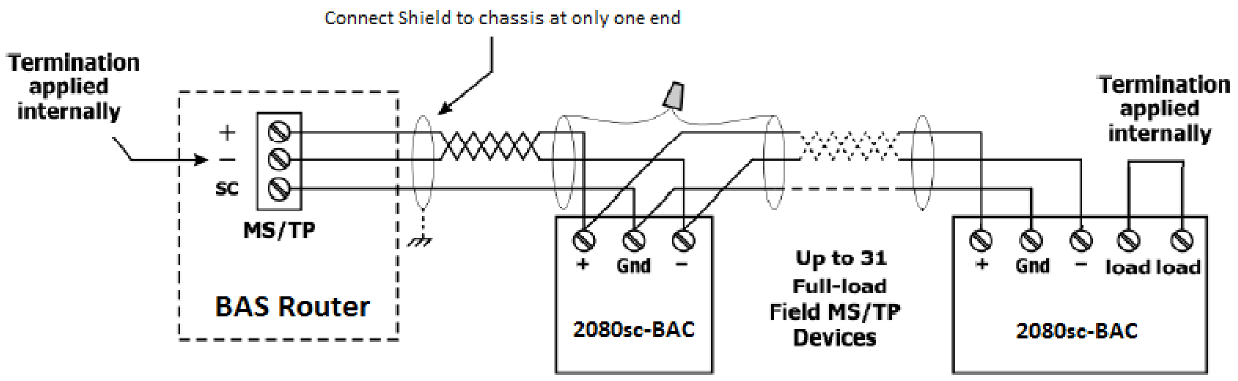
OR
Connect an RJ-45 Ethernet cable and connector between a personal computer and the connector on the front of the module.
Set up the tag mapping using the software described in Chapter 3.

Section 2.2 MS/TP Wiring Best Practices

Use the following best wiring practices:

- Ensure the MS/TP segment is wired in a daisy chain configuration (NOT Star, Bus, “T”, or other configurations).
- Ensure all MAC addresses are unique for that MS/TP segment. If you have 2 addresses that are the same, that means 2 devices are talking at the same time.
- Ensure module device instance is unique for the BACnet network.

- Ensure all devices for that MS/TP segment are running at the same baud rate.
- Ensure there are only 2 terminations for that MS/TP segment, at the ends.
- Having more than 30 MS/TP devices on one segment is not recommended. Break up that segment into separate ones with an MS/TP BACnet router.
- Having an MS/TP segment that goes over 1050 feet is not recommended. Break up that segment.
- If you have other BACnet (non Micro800) devices on the network, remove them while troubleshooting. If they are causing issues, then isolate the Micro800 devices on a separate MS/TP segment
- CAT5, CAT5E, or CAT6 network cables have excellent specifications for BACnet MS/TP.
- Route the BACnet network away from high voltage lines, VFDs, fluorescent lights, etc.


Connect Shield to chassis at only one end
Termination applied internally
+
−
SC
MS/TP
BAS Router
+
Gnd
−
2080sc-BAC
Up to 31 Full-load Field MS/TP Devices
Termination applied internally
+
Gnd
−
load load
2080sc-BAC

3-wire MS/TP Bus

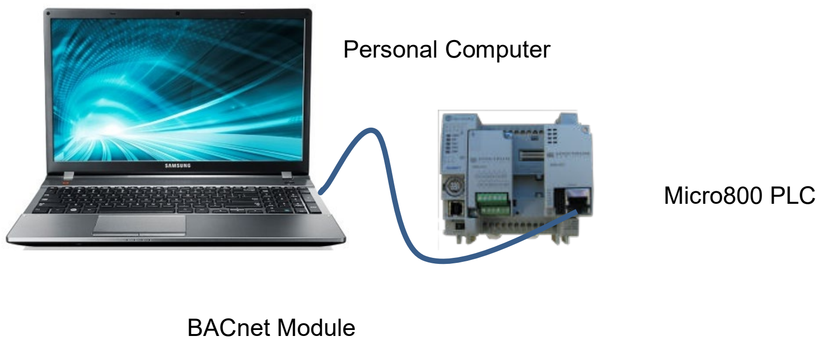
Chapter 3
Configuring the Module using Software

Before configuring the module with the BACnet software:

1. Install your BACnet module in the Rockwell Micro800 controller.
2. Connect an Ethernet cable between the Ethernet port on the personal computer and the Ethernet port on the BACnet module.

An unconfigured module has a default address of 169.254.3.3.

A configured module may have a different static IP address.



Once the PLC is powered on and the module is successfully connected to the PC:

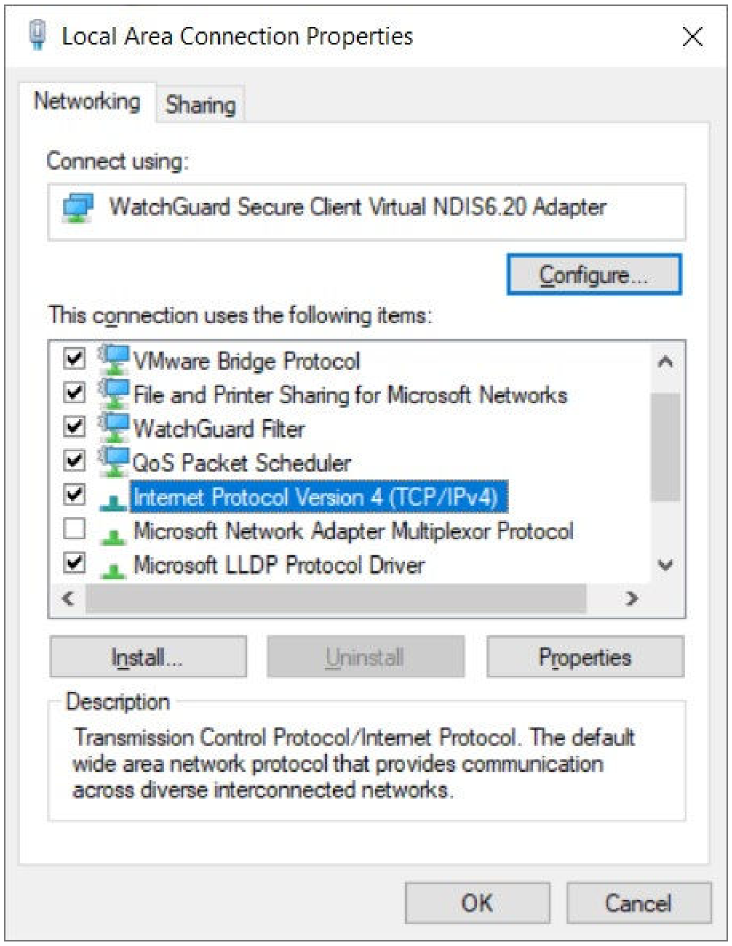
1. Change the personal computer IP address to a static IP address in the same subnet as the default address of the module. If you need additional assistance with changing your personal computer's IP address, refer to the Windows Help documentation or use the Access the Software on your Module information provided next.
2. Use the web browser to access the module and to map tags between the Micro800 PLC and the BACnet module.
3. Generate the XML mapping file between the PLC and the module.
4. Download the XML map file to the module.
5. Generate the structured text to be used in the PLC program.

After you have configured your system, if the BACnet module is configured for BACnet IP, the BACnet module will listen on Ethernet port **0×BAC0** for data from the network and will handle the data according to the BACnet protocol. Detailed instructions for each step of this process are provided below.

NOTE	The software automatically logs you out after 30 minutes of inactivity.

To access the software on your module:

1. Access your PC local area connection properties dialog as shown below: **Local Area Connection Settings:Properties**.
2. From the Local Area Connection Properties dialog, select the **Internet Protocol Version 4 (TCP/IPv4)** option:


Local Area Connection Properties

Networking | Sharing

Connect using:

WatchGuard Secure Client Virtual NDIS6.20 Adapter

Configure...

This connection uses the following items:

- [✓] VMware Bridge Protocol
- [✓] File and Printer Sharing for Microsoft Networks
- [✓] WatchGuard Filter
- [✓] QoS Packet Scheduler
- [✓] Internet Protocol Version 4 (TCP/IPv4)
- [ ] Microsoft Network Adapter Multiplexor Protocol
- [✓] Microsoft LLDP Protocol Driver

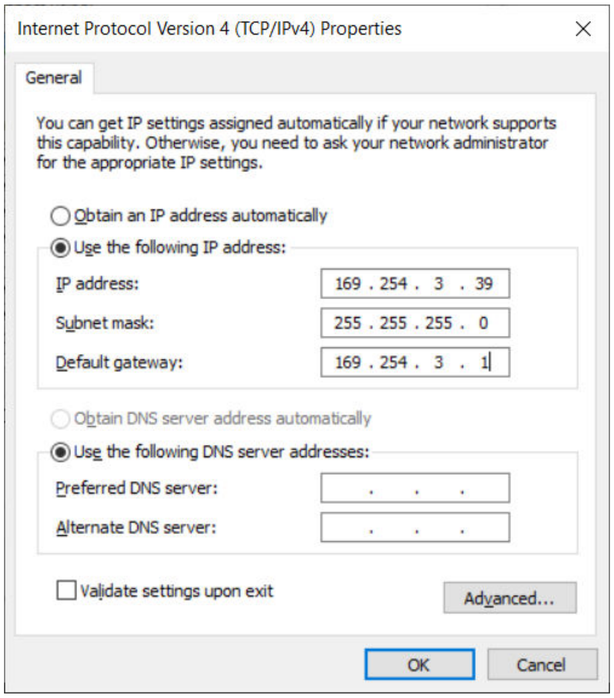
Install... | Uninstall | Properties

Description

Transmission Control Protocol/Internet Protocol. The default wide area network protocol that provides communication across diverse interconnected networks.

OK | Cancel

3. Change the settings to the equivalent settings for your personal computer as shown next:



Internet Protocol Version 4 (TCP/IPv4) Properties

General

You can get IP settings assigned automatically if your network supports this capability. Otherwise, you need to ask your network administrator for the appropriate IP settings.

( ) Obtain an IP address automatically
(●) Use the following IP address:

IP address:	169 . 254 . 3 . 39
Subnet mask:	255 . 255 . 255 . 0
Default gateway:	169 . 254 . 3 . 1

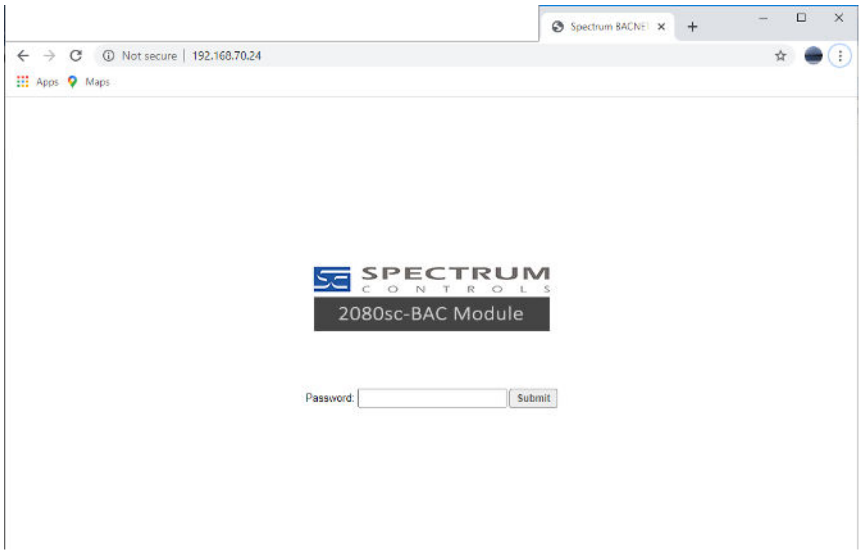
( ) Obtain DNS server address automatically
(●) Use the following DNS server addresses:

Preferred DNS server:	. . .
Alternate DNS server:	. . .

☐ Validate settings upon exit

4. Open a web browser and type in the IP default address of your BACnet module, which is **169.254.3.3**. The Gateway address is 169.254.3.1.

The following dialog appears:



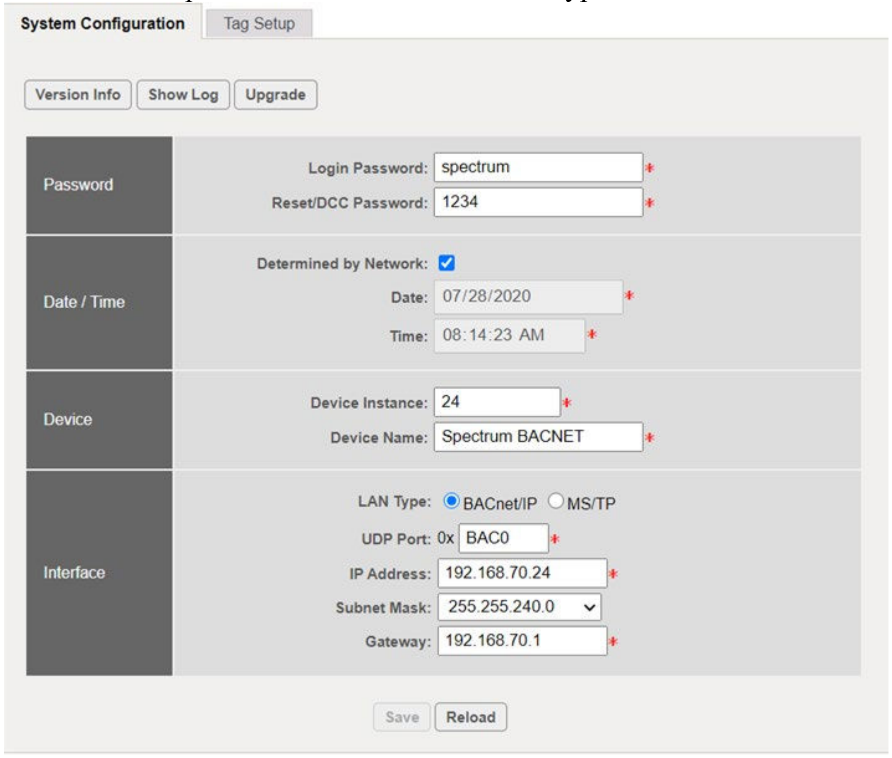
Not secure | 192.168.70.24

5. Type the password **spectrum** into the **Password** field and click **Submit**.

The BACnet configuration dialog appears.

The example shown below uses the MS/TP LAN type:

This next example shows the BACnet/IP LAN type:



Password	Login Password:	spectrum *
	Reset/DCC Password:	1234 *
Date / Time	Determined by Network:	✓
	Date:	07/28/2020 *
	Time:	08:14:23 AM *
Device	Device Instance:	24 *
	Device Name:	Spectrum BACNET *
Interface	LAN Type:	[✓ BACnet/IP] MS/TP
	UDP Port:	0x BAC0 *
	IP Address:	192.168.70.24 *
	Subnet Mask:	255.255.240.0
	Gateway:	192.168.70.1 *

6. View or modify the following options:
   - **System Configuration**. Use to set up password, date and time, device, and LAN or MS/TP settings.
   - **Tag Setup**. Use to set up tag mapping between your Micro800 controller and the BACnet Communications module.

Section 3.1 Configuring the System

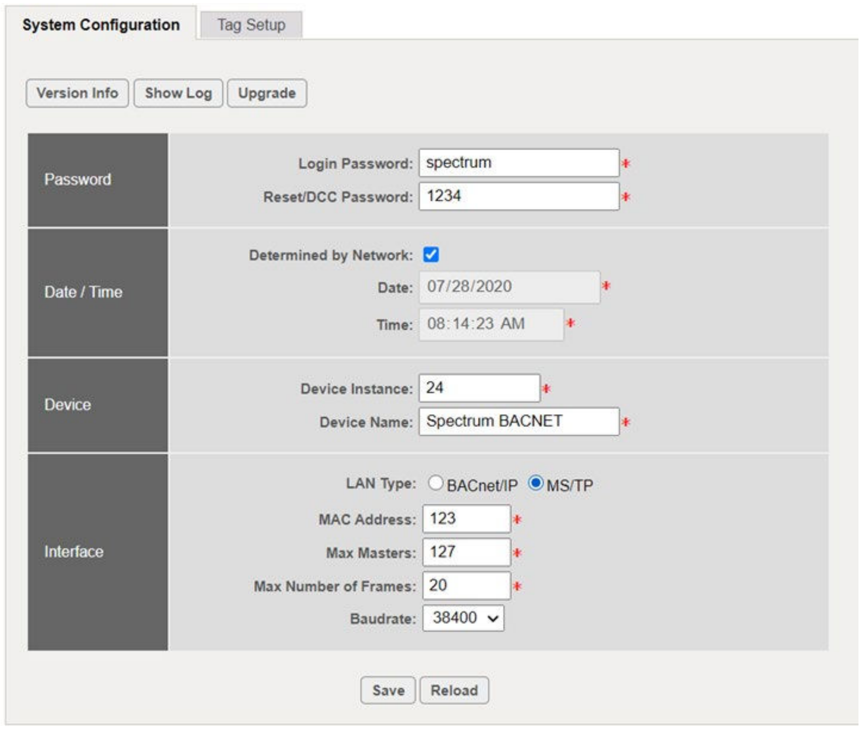
System configuration includes setting up or changing your system password, setting the date and time, choosing your device, and setting up the LAN.

System Configuration Summary

Item	PLC Code	Module Configuration	Notes
Module IP Address	NA	169.254.3.3	This is the default; revise when Network IP address is known.
Module MS/TP MAC Address	NA	127	This is the default; revise when MS/TP Network MAC address is known.
Module MS/TP Baud	NA	38400	This is the default; revise when MS/TP Network Baud rate is known.
Module Slot Number	`BACNET_MODULE_SLOT := 1;`	1	This is the default for the example code; revise if module not in slot 1

To set up or change system configuration settings:

1. Access the System Configuration tab:



Login Password: spectrum *
Reset/DCC Password: 1234 *
Determined by Network: ✓
Date: 07/28/2020 *
Time: 08:14:23 AM *
Device Instance: 24 *
Device Name: Spectrum BACNET *
LAN Type: BACnet/IP [✓ MS/TP]
MAC Address: 123 *
Max Masters: 127 *
Max Number of Frames: 20 *
Baudrate: 38400

2. View or specify the following options:

   - **Password**. The software ships with a default password, **spectrum**. You log onto the module software User Interface with this password when you first log onto the module. However, Spectrum Controls, Inc. highly recommends that you immediately change this password to one of your own choosing. There are two passwords:

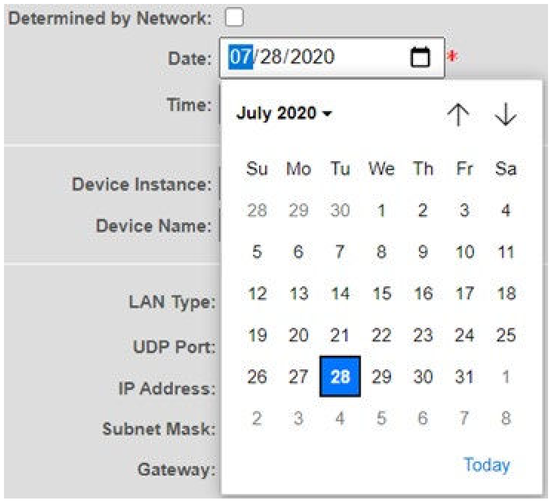
     - <u>Login Password</u>. This is the password you use to log onto software which ships with the BACnet module. The default user password is **spectrum**. If you reset your module to factory default settings, this is the password that you use to log in.

- Reset/DCC Password. BACnet has a remote reset command. When the module receives the remote reset command, it reboots itself. You use this password as the BACnet reset message password to reboot the module. The default value is 1234.

- **Date/Time**. At startup, the module retrieves the NIST time, if network is set to retrieve it. If connected to the Internet at that point, the module's time is set from the server.

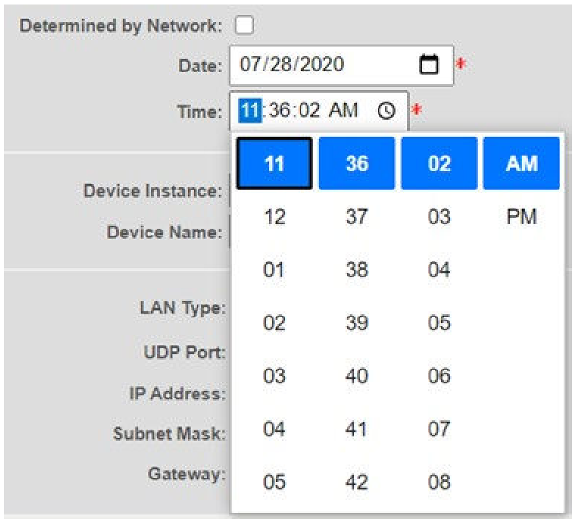
  The date and time are held for 2 to 3 days if power is lost. If needed, you may manually enter the date and time:

  - Determined by Network. Checkbox. When selected, **Date** and **Time** controls are disabled.
  - Date. Available when **Determined by Network** checkbox is deselected. Enter the date as *MM/DD/YYYY* where *MM* is month, *DD* is day, and *YYYY* is year. If necessary, select the date from the calendar provided.



  - Time. Available when **Determined by Network** checkbox is deselected. 24-hour time format.

- Select the time in hours, minutes, and seconds.



- **Device**. Enter the device instance and name. The device instance is the unique ID of a device used by BACnet. Default Device ID is **#200121**. All messages directed to the module are addressed to this ID. The allowable range is from 1 to 4,194,303:
  - Device Instance. Specify the device instance entry.
  - Device Name. Enter the device name.

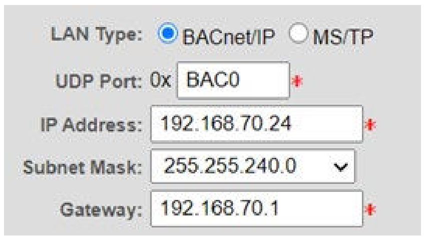
NOTE	• In a large network, the BACnet administrator/designer would designate the device instances and names. Also, required in an MS/TP network are the MAC addresses, baud rate, and allowable max frames, and in a BACnet IP network, the UDP Port, address, and gateway. • Review the BACnet Diagnostic Tools section for information on the commands that allow a BACnet master to discover devices on the network (WhoIs), read

- **Interface.** The interface connection may be either via LAN over the Ethernet connection or MS/TP, which provides a serial connection:
  - LAN Type. Specifies LAN type and communication parameters for the LAN type selected. After saving your values, you must cycle power on the module to allow the new settings to take effect.

    The module and personal computer's IP addresses must also be on the same network to allow the two devices to communicate:

NOTE	If you change the PC subnet mask, you must cycle power to the module so that the module reloads the user interface with the new value.

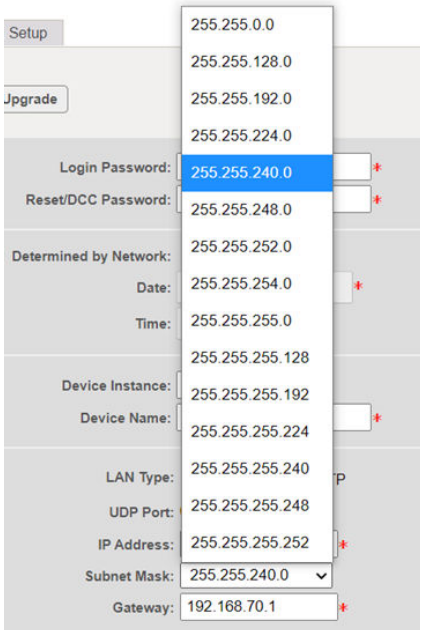
BACnet/IP: When you set up your module to use an Ethernet physical interface, you must specify a UDP port number for IP address, Subnet Mask, and Gateway. The UDP port communicates between the PC and the module. It is used to retrieve the module's IP address and password authentication:



LAN Type: [✓ BACnet/IP] MS/TP
UDP Port: 0x BAC0 *
IP Address: 192.168.70.24 *
Subnet Mask: 255.255.240.0
Gateway: 192.168.70.1 *

View or specify the following options:

- *UDP Port:* Hex number only. *0x XXXN*. Default address: **0×BAC0**
- *IP Address*. Enter IP address. Example: **192.168.70.24**
- *Subnet Mask*. From drop down list, select subnet mask. Default mask value is **255.255.240.0**:

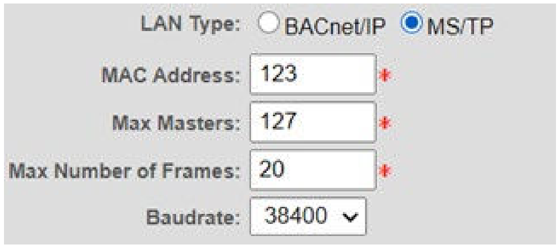


255.255.0.0
255.255.128.0
255.255.192.0
255.255.224.0
255.255.240.0
255.255.248.0
255.255.252.0
255.255.254.0
255.255.255.0
255.255.255.128
255.255.255.192
255.255.255.224
255.255.255.240
255.255.255.248
255.255.255.252

Subnet Mask: 255.255.240.0
Gateway: 192.168.70.1

- *Gateway*. Enter gateway address. Example: **192.168.70.1**

- MS/TP. (Master-Slave/Token Ring Passing.)



LAN Type: BACnet/IP [✓ MS/TP]

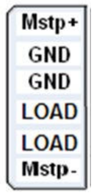
MAC Address: 123 *

Max Masters: 127 *

Max Number of Frames: 20 *

Baudrate: 38400

This connection is set up using the 6-pin connector on the front panel of the BACnet module. **Mstp+** is **RS-485+**; **Mtsp-** is **RS-485-**:

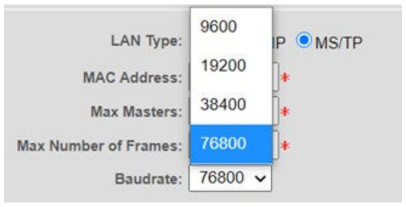


When you set up your module to use an MS/TP physical interface, which is RS-485, half-duplex communication, you specify the following fields:

- *MAC Address*. Enter the module MS/TP MAC address. Default suggested value is any value between 1 and 127. If there are more devices on your MS/TP network, you will need to determine what MAC address to use that makes this device unique on your network. You will need to exercise care to avoid entering duplicate MAC addresses in your device network.
- *Max Masters*. In a token ring network, each node is responsible for searching for the next node and passing the token to it. **Max Masters** is the maximum MAC address this module searches for. For example, if this module's MAC address is **5** and the **Max Master** value is **20**, this module will poll MAC address from 6 to 20 for the next node. Enter the **Max Masters** address. Default suggested value is **127** (range is 1 to 127).
- *Max Number of Frames*. In a token ring network, when a node receives a token, it needs to pass the token to the next node in a timely manner. Setting **Max Number of Frames** to 20 means when this module receives the token, it can send out a maximum of 20 messages before it has to pass the token to the next node. A good starting value to enter is **20**. If you have a very busy network, you may need to reduce this number to allow your network to function

optimally. Range is 1 to 100.

- *Baudrate*. Select an MS/TP communication baud rate (9600, 19200, 38400, or 76800):



LAN Type: ◉ MS/TP
MAC Address:
Max Masters:
Max Number of Frames:
Baudrate: 76800

Section 3.2 System Configuration Save and Reload Buttons

When you are finished setting up your module to use an MS/TP or BACnet physical interface, you need to save the configuration to the module.

- Clicking **Save** while on the system configuration tab saves the configuration to the module.
- If you change the configuration, but want to abandon the change, clicking **Reload** restores the configuration from the module memory.

Section 3.3 Setting Up Tags

You use the Tag Setup dialog to map PLC tags to BACnet objects. The 2080sc-BAC Communication Module supports BACnet AnalogInput (AI), AnalogOutput (AO), BinaryInput (BI), and BinaryOutput (BO) objects. When you map a BACnet object to a PLC tag, you should match the same or similar data types (covered later in this chapter). For example, map PLC data type BOOL to BACnet binary input or output objects. Other PLC data types can be mapped to BACnet analog input or analog output objects.

- The data flow for the AI and BI BACnet object types is FROM the Micro800 PLC (PLUGIN_WRITE) TO the BACnet master (a BACnet READ). These are inputs to the BACnet master.
- The data flow for the AO and BO BACnet object types is FROM the BACnet master (a BACnet WRITE) TO the Micro800 PLC (PLUGIN_READ). These are outputs from the BACnet master.

Tag Setup

Tag Setup						
PLC Tagname	**PLC Data Type**	**PLC Attribute**	**BACnet Object Name**	**BACnet Object Type**	**BACnet Object ID**	**Notes**
Temp_Chiller	REAL	Read	Temp_Chiller1	AnalogInput	0	First of 3 Cooling towers, first BACnet AI object. PLC writes to this.

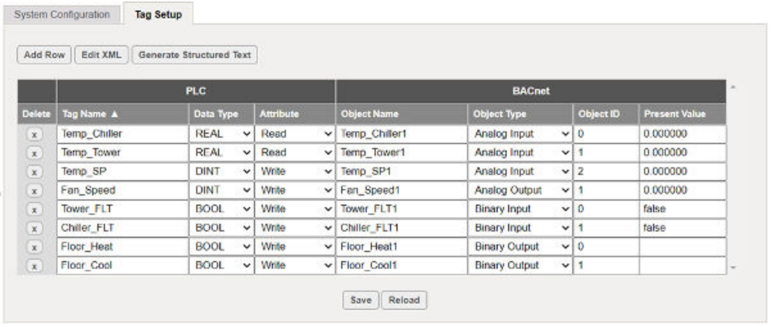
Tag Setup						
PLC Tagname	**PLC Data Type**	**PLC Attribute**	**BACnet Object Name**	**BACnet Object Type**	**BACnet Object ID**	**Notes**
Temp_Tower	REAL	Read	Temp_Tower1	AnalogInput	1	First of 3 Cooling towers, second BACnet AI object. PLC writes to this.
Temp_SP	DINT	Write	Temp_SP1	AnalogOutput	0	First of 3 Cooling towers, first BACnet AO object. PLC reads this.
Fan_Speed	DINT	Write	Fan_Speed1	AnalogOutput	1	First of 3 Cooling towers, second BACnet AO object. PLC reads this.
Pump_Speed	DINT	Write	Pump_Speed1	AnalogOutput	16	First of 3 Cooling towers, second BACnet AO object. PLC reads this.
Tower_FLT	BOOL	Read	Tower_FLT1	BinaryInput	0	First of 3 Cooling towers, first BACnet BI object. PLC writes to this.
Chiller_FLT	BOOL	Read	Chiller_FLT1	BinaryInput	1	First of 3 Cooling towers, second BACnet BI object. PLC writes to this.
Floor_Enable	Bool	Read	Floor_Enable1	BinaryOutput	2	First of 3 Cooling towers, second BACnet BI object. PLC writes to this.
Floor_Heat	BOOL	Write	Floor_Heat1	BinaryOutput	0	First of 3 Cooling towers, first BACnet BO object. PLC reads this.

Tag Setup						
PLC Tagname	**PLC Data Type**	**PLC Attribute**	**BACnet Object Name**	**BACnet Object Type**	**BACnet Object ID**	**Notes**
Floor_Cool	BOOL	Write	Floor_Cool1	BinaryOutput	1	First of 3 Cooling towers, second BACnet BO object. PLC reads this.

To set up tag mapping between the Micro800 PLC and BACnet objects:

1. Select the Tag Setup tab.

   The Tag Setup dialog appears:



Delete	PLC			BACnet			
	Tag Name ▲	Data Type	Attribute	Object Name	Object Type	Object ID	Present Value
x	Temp_Chiller	REAL	Read	Temp_Chiller1	Analog Input	0	0.000000
x	Temp_Tower	REAL	Read	Temp_Tower1	Analog Input	1	0.000000
x	Temp_SP	DINT	Write	Temp_SP1	Analog Input	2	0.000000
x	Fan_Speed	DINT	Write	Fan_Speed1	Analog Output	1	0.000000
x	Tower_FLT	BOOL	Write	Tower_FLT1	Binary Input	0	false
x	Chiller_FLT	BOOL	Write	Chiller_FLT1	Binary Input	1	false
x	Floor_Heat	BOOL	Write	Floor_Heat1	Binary Output	0	
x	Floor_Cool	BOOL	Write	Floor_Cool1	Binary Output	1	

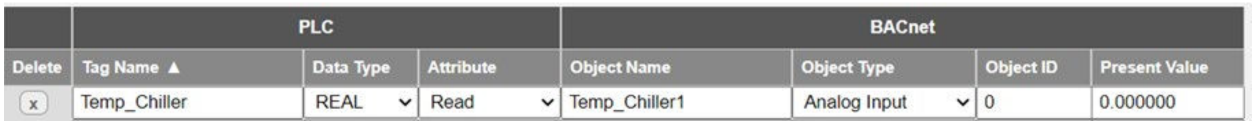
2. View or specify the following options:

   - **Add Row**. To add a tag, click the **Add Row** button:

     A new row is added to the tag list. 64 rows per object type is maximum allowed.

     Enter the tag data for each of the editable fields in the PLC and BACnet mapping fields as follows:

     - Tag Name. Enter the PLC variable tag name. Click on the field and type in the name:

- 

Delete	PLC			BACnet			
	Tag Name ▲	Data Type	Attribute	Object Name	Object Type	Object ID	Present Value
x	Temp_Chiller	REAL	Read	Temp_Chiller1	Analog Input	0	0.000000

     - Data Type. Select the PLC variable data type. Click on the field and select the correct data type associated with the tag:

     - Attribute. Click on the field and select the PLC variable attribute. The best practice for this module is to set the Attribute to READ for `AnalogInput` and `BinaryInput` types, and to WRITE for `AnalogOutput` and `BinaryOutput` types. That explicitly states the data direction.

     - Object Name. Enter the BACnet object name. Use a meaningful description, such as **Damper Angle**.

- Object Type. Click on the field and select the relevant object type for the PLC tag:
- Object ID. This is the BACnet object instance, and the valid range is from 0 to 63. Each object is uniquely defined by its combination of object instance and type. The BACnet administrator/designer needs to assign a unique value to each object instance. The module allows 64 object instances of each object type for a total of 256 BACnet objects (64 AI + 64 BI + 64 AO + 64 BO = 256). Click on the field and type in the ID value.
- Present Value. Not editable (from user interface). When the BACnet module receives a request from the system to report a specific value, the module returns the present value in the specific index. Example: When the module receives a BACnet command to report the present value of Analog Output 1, the module sends the present value of Analog Output 1 to the BACnet master. This value is then displayed in the **Present Value** column.

- **Save** . Saves tag changes to the module's memory. You must save changes before you may generate structured text from the tag information.
- **Reload**. Erases the current tag configuration, loads tag configuration from the modules memory and updates the present values. Useful for verifying data flow to and from the PLC.
- **Edit XML**. Allows you to manually edit tag information in XML format. See Editing XML Tag Information next.
- **Generate Structured Text**. Converts tag information to CCW structured text. See Generating Structured Text later in this chapter.

Section 3.4 Editing XML Tag Information

You normally modify tags through the software's user interface. However, the module uses an XML file format to save the tag mapping information. To save time, you may also modify the XML file directly.

NOTE	When modifying XML directly, be sure to make a backup of your original file first. Otherwise you may find errors or accidental deletions of entries cause problems with your file. Once you save the file, the changes are permanent.

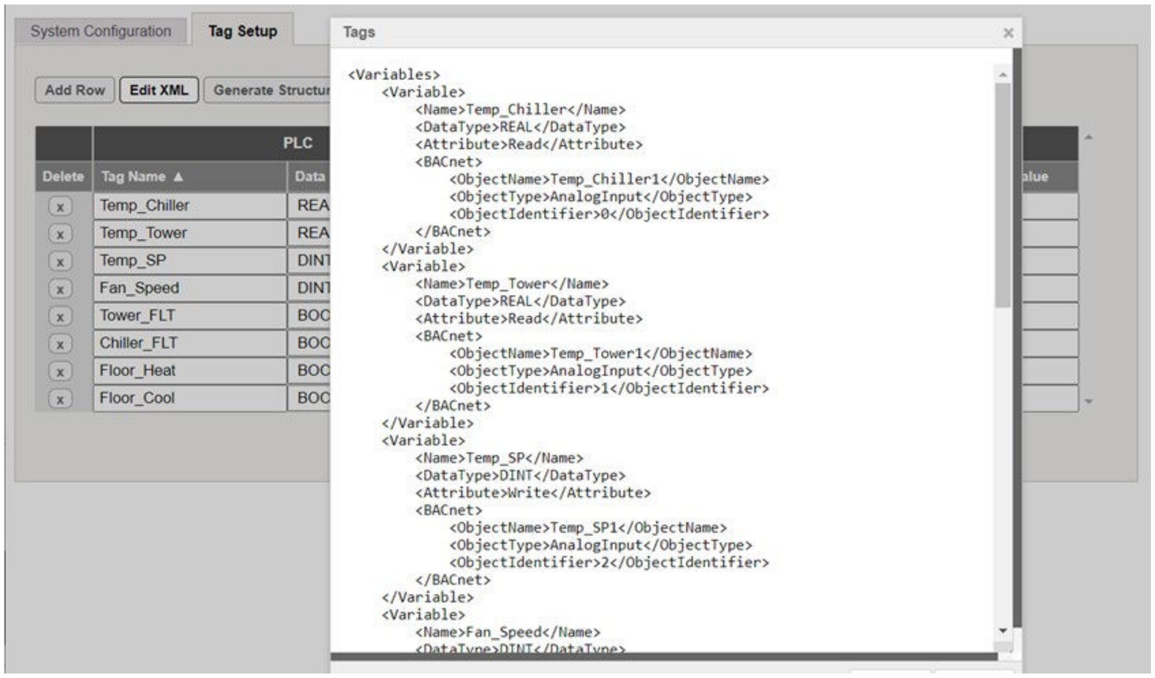
NOTE	To expedite the tag mapping process, you may copy and paste PLC tags from CCW directly into the XML window. Use this feature when you are editing a large number of tags.

This module uses XML file format to save the tag mapping.

To view all the variables in the XML file:

1. Click **Edit XML**:

The Tags dialog appears with all the tags in the list in XML format:



	PLC	
Delete	Tag Name ▲	Data
x	Temp_Chiller	REA
x	Temp_Tower	REA
x	Temp_SP	DINT
x	Fan_Speed	DINT
x	Tower_FLT	BOO
x	Chiller_FLT	BOO
x	Floor_Heat	BOO
x	Floor_Cool	BOO

```
<Variables>
    <Variable>
        <Name>Temp_Chiller</Name>
        <DataType>REAL</DataType>
        <Attribute>Read</Attribute>
        <BACnet>
            <ObjectName>Temp_Chiller1</ObjectName>
            <ObjectType>AnalogInput</ObjectType>
            <ObjectIdentifier>0</ObjectIdentifier>
        </BACnet>
    </Variable>
    <Variable>
        <Name>Temp_Tower</Name>
        <DataType>REAL</DataType>
        <Attribute>Read</Attribute>
        <BACnet>
            <ObjectName>Temp_Tower1</ObjectName>
            <ObjectType>AnalogInput</ObjectType>
            <ObjectIdentifier>1</ObjectIdentifier>
        </BACnet>
    </Variable>
    <Variable>
        <Name>Temp_SP</Name>
        <DataType>DINT</DataType>
        <Attribute>Write</Attribute>
        <BACnet>
            <ObjectName>Temp_SP1</ObjectName>
            <ObjectType>AnalogInput</ObjectType>
            <ObjectIdentifier>2</ObjectIdentifier>
        </BACnet>
    </Variable>
    <Variable>
        <Name>Fan_Speed</Name>
        <DataType>DINT</DataType>
```

2. If needed, manually modify individual entries in the XML file.

   Example. In the above Tags list, you may manually change:

   <Name>__IO_EM_DO_00</Name>

   TO

   <Name>__IO_EM_DO_123</Name>.

3. To implement the changes, click **Submit**. The name changes to **__IO_EM_DO_123**.

Section 3.5 Saving Tag Setup and Reloading Using XML

On the Tag Setup tab, there is an Edit XML button, which allows you to manually edit tag information in XML format. It also allows you to copy the finished XML code, paste it into a text file, and save it to your PC. You then have a backup or a master copy of your tag setup.

To initialize a new 2080sc-BAC module with an identical copy of your master tag setup:

- In a blank tag setup screen, click the **Edit XML** button.
- Paste the text from the master copy of your tag setup into the blank screen.
- Click **Submit**.

To initialize a new 2080sc-BAC module with a slightly different version of your master tag setup:

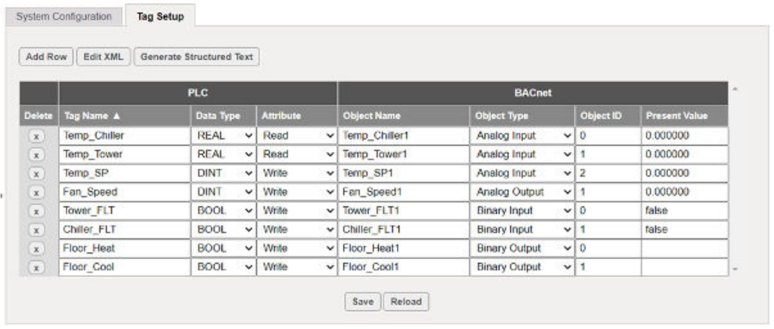
- Modify the text from the master copy of your tag setup in XML format.
- In a blank tag setup screen, click the **Edit XML** button
- Paste the modified text of your tag setup into the blank screen.
- Click **Submit**.

Section 3.6 Generating Structured Text

Rockwell Connected Component Workshop (CCW) software is used to program the Micro800 PLC with the BACnet module installed in a plugin slot. The tags to be mapped to the BACnet module need to be installed in your program in the Global Variables list. Then, those same tag names and data types need to be installed in the BACnet module Tag Setup tab to exactly match what is in the program. When you finish mapping tags, you convert the tag information to structured text code, and paste that code into your CCW program.

The generated structured text code allows the PLC program to access data written by the BACnet master (AO and BO objects) and to provide data the BACnet master wants to read (AI and BI objects).

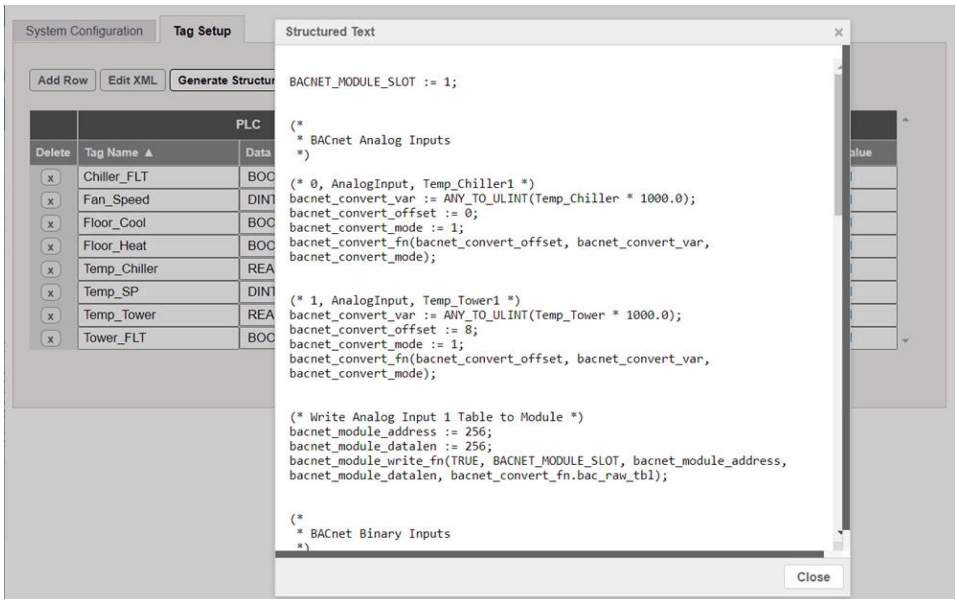
1. To convert tag information to structured text code, add your tags using the Tags dialog:



	PLC			BACnet			
Delete	Tag Name ▲	Data Type	Attribute	Object Name	Object Type	Object ID	Present Value
x	Temp_Chiller	REAL	Read	Temp_Chiller1	Analog Input	0	0.000000
x	Temp_Tower	REAL	Read	Temp_Tower1	Analog Input	1	0.000000
x	Temp_SP	DINT	Write	Temp_SP1	Analog Input	2	0.000000
x	Fan_Speed	DINT	Write	Fan_Speed1	Analog Output	1	0.000000
x	Tower_FLT	BOOL	Write	Tower_FLT1	Binary Input	0	false
x	Chiller_FLT	BOOL	Write	Chiller_FLT1	Binary Input	1	false
x	Floor_Heat	BOOL	Write	Floor_Heat1	Binary Output	0	
x	Floor_Cool	BOOL	Write	Floor_Cool1	Binary Output	1	

2. From the Tags dialog, click **Generate Structured Text**:

The software generates the CCW structured text and shows the text in the Structured Text dialog:



PLC		
Delete	Tag Name ▲	Data
x	Chiller_FLT	BOO
x	Fan_Speed	DINT
x	Floor_Cool	BOO
x	Floor_Heat	BOO
x	Temp_Chiller	REA
x	Temp_SP	DINT
x	Temp_Tower	REA
x	Tower_FLT	BOO

```
BACNET_MODULE_SLOT := 1;


(*
 * BACnet Analog Inputs
 *)

(* 0, AnalogInput, Temp_Chiller1 *)
bacnet_convert_var := ANY_TO_ULINT(Temp_Chiller * 1000.0);
bacnet_convert_offset := 0;
bacnet_convert_mode := 1;
bacnet_convert_fn(bacnet_convert_offset, bacnet_convert_var,
bacnet_convert_mode);


(* 1, AnalogInput, Temp_Tower1 *)
bacnet_convert_var := ANY_TO_ULINT(Temp_Tower * 1000.0);
bacnet_convert_offset := 8;
bacnet_convert_mode := 1;
bacnet_convert_fn(bacnet_convert_offset, bacnet_convert_var,
bacnet_convert_mode);


(* Write Analog Input 1 Table to Module *)
bacnet_module_address := 256;
bacnet_module_datalen := 256;
bacnet_module_write_fn(TRUE, BACNET_MODULE_SLOT, bacnet_module_address,
bacnet_module_datalen, bacnet_convert_fn.bac_raw_tbl);


(*
 * BACnet Binary Inputs
 *)
```

3. Copy the structured text and paste it into the Main POU of your CCW project.

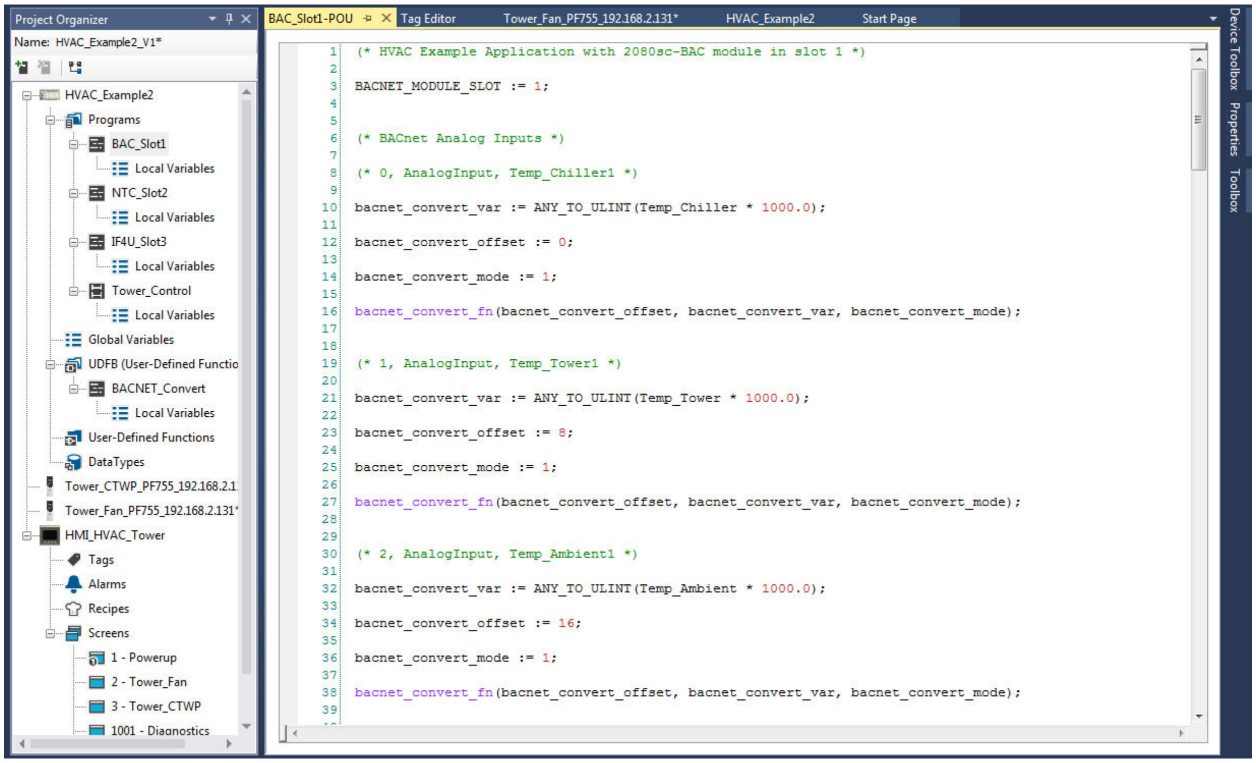
Section 3.7 Using the CCW Structured Text Example

Spectrum Controls, Inc. provides a sample CCW project with example programs), and a Function block named BACNET_Convert in a zipped project file available upon request. If you have the same PLC as the sample project, you may directly use this project.

If you are using a different Micro800 PLC, the global variables, the programs and the BACNET_Convert UDF (User Defined Function) have been exported so you can import them directly into your favorite flavor of Micro800 PLC. Import all and build the project to verify there are no errors.

The CCW sample project consists of example BAC_Slot1, NTC_Slot2, IF4U_Slot3, Tower_Control programs and the BACNET_Convert user defined function block (UDF). The example programs consist of the structured text generated by the Module when you click the **Generate Structured Text** button on the Tag Setup tab. The function block **BACNET_Convert** is provided in the sample project. The Function block **BACNET_Convert** does not need to change when you make a tag mapping change.

This is a view of the CCW program showing on the left, the project organizer with a view of the Programs and **BACNET_Convert** UDF and on the right, the first few lines of the structured text that make up the first of the provided examples, under Programs, the BAC_Slot1 program:



```
(* HVAC Example Application with 2080sc-BAC module in slot 1 *)

BACNET_MODULE_SLOT := 1;


(* BACnet Analog Inputs *)

(* 0, AnalogInput, Temp_Chiller1 *)

bacnet_convert_var := ANY_TO_ULINT(Temp_Chiller * 1000.0);

bacnet_convert_offset := 0;

bacnet_convert_mode := 1;

bacnet_convert_fn(bacnet_convert_offset, bacnet_convert_var, bacnet_convert_mode);


(* 1, AnalogInput, Temp_Tower1 *)

bacnet_convert_var := ANY_TO_ULINT(Temp_Tower * 1000.0);

bacnet_convert_offset := 8;

bacnet_convert_mode := 1;

bacnet_convert_fn(bacnet_convert_offset, bacnet_convert_var, bacnet_convert_mode);


(* 2, AnalogInput, Temp_Ambient1 *)

bacnet_convert_var := ANY_TO_ULINT(Temp_Ambient * 1000.0);

bacnet_convert_offset := 16;

bacnet_convert_mode := 1;

bacnet_convert_fn(bacnet_convert_offset, bacnet_convert_var, bacnet_convert_mode);

```

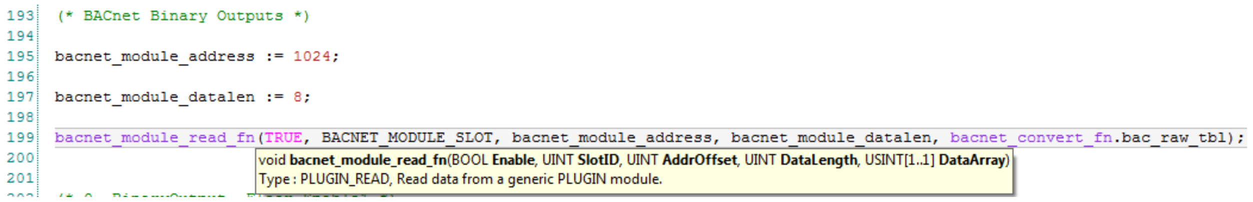
NOTE	There is no defined behavior for the module when the controller enters Program Mode. When placed in Program Mode, the module will still respond to BACnet requests with the same functionality as exists in normal operation, with the exception that PLC will not receive updated Output values from the module, nor will it update Input values to the module. When the controller enters Run mode, the module writes the value.

3.7.1 Program BAC_Slot1

The example BAC_Slot1 program uses the generated structured text from the above tag setup.

There are 3 main functions types used in the programs to transfer data between the PLC and the BAC module. Examples are provided below:

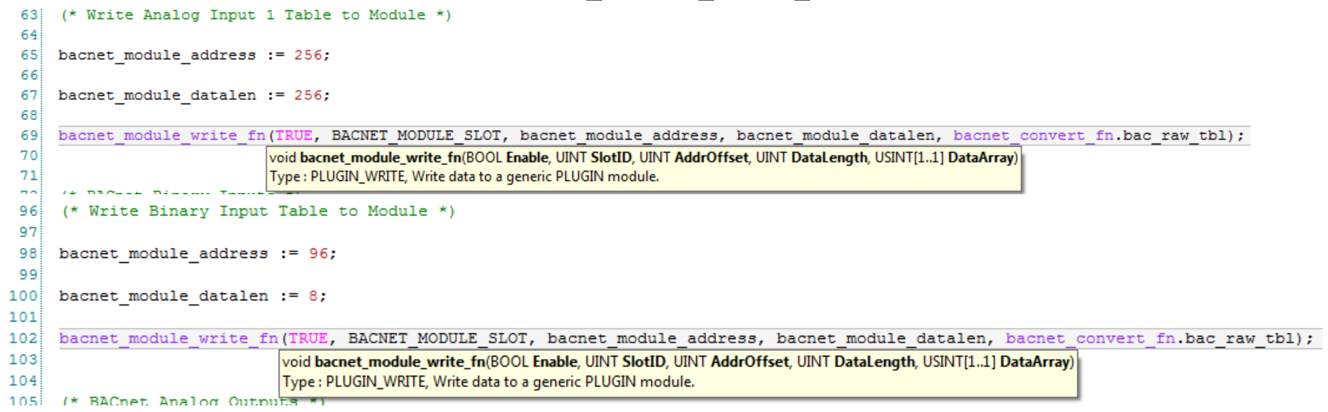
Read functions (`bacnet_module_read_fn`):



```
(* BACnet Binary Outputs *)

bacnet_module_address := 1024;

bacnet_module_datalen := 8;

bacnet_module_read_fn(TRUE, BACNET_MODULE_SLOT, bacnet_module_address, bacnet_module_datalen, bacnet_convert_fn.bac_raw_tbl);
```

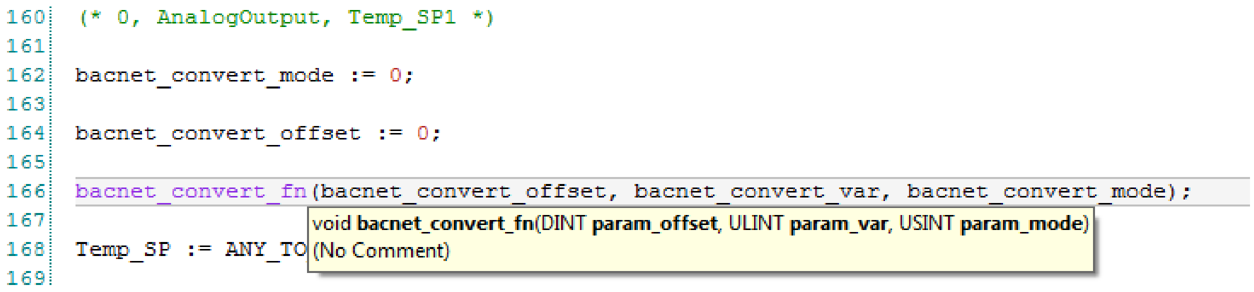
void **bacnet_module_read_fn**(BOOL **Enable**, UINT **SlotID**, UINT **AddrOffset**, UINT **DataLength**, USINT[1..1] **DataArray**)
Type : PLUGIN_READ, Read data from a generic PLUGIN module.

Write functions (`bacnet_module_write_fn`):



```
(* Write Analog Input 1 Table to Module *)

bacnet_module_address := 256;

bacnet_module_datalen := 256;

bacnet_module_write_fn(TRUE, BACNET_MODULE_SLOT, bacnet_module_address, bacnet_module_datalen, bacnet_convert_fn.bac_raw_tbl);
                void bacnet_module_write_fn(BOOL Enable, UINT SlotID, UINT AddrOffset, UINT DataLength, USINT[1..1] DataArray)
                Type : PLUGIN_WRITE, Write data to a generic PLUGIN module.

(* Write Binary Input Table to Module *)

bacnet_module_address := 96;

bacnet_module_datalen := 8;

bacnet_module_write_fn(TRUE, BACNET_MODULE_SLOT, bacnet_module_address, bacnet_module_datalen, bacnet_convert_fn.bac_raw_tbl);
                void bacnet_module_write_fn(BOOL Enable, UINT SlotID, UINT AddrOffset, UINT DataLength, USINT[1..1] DataArray)
                Type : PLUGIN_WRITE, Write data to a generic PLUGIN module.
```

Convert functions (`bacnet_convert_fn`):

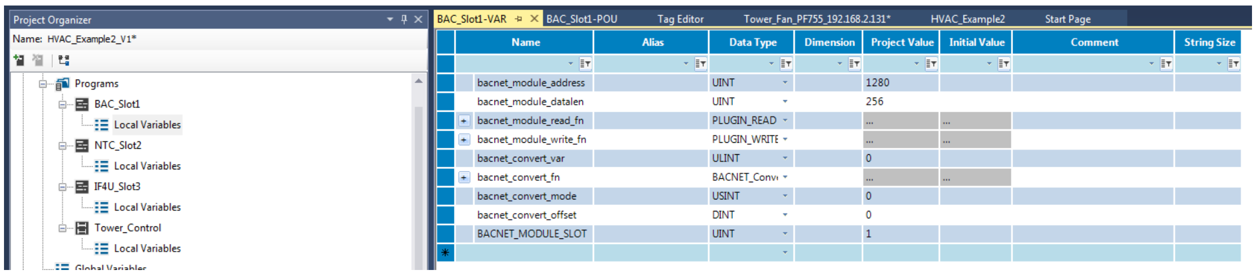


```
(* 0, AnalogOutput, Temp_SP1 *)

bacnet_convert_mode := 0;

bacnet_convert_offset := 0;

bacnet_convert_fn(bacnet_convert_offset, bacnet_convert_var, bacnet_convert_mode);
                void bacnet_convert_fn(DINT param_offset, ULINT param_var, USINT param_mode)
Temp_SP := ANY_TO(No Comment)
```

There are a variety of variables for providing the read, write and convert functions with the proper parameters. Examples are:

- Address of the module (`bacnet_module_address`)
- Length of the data (`bacnet_module_datalen`)
- Variable parameter (`bacnet_convert_var`)
- Convert mode (`bacnet_convert_mode`)
- Array offset (`bacnet_convert_offset`)
- Slot that the module is in (`BACNET_MODULE_SLOT`)

The first step is to inspect the local variables for the example BAC_Slot1.



Name	Alias	Data Type	Dimension	Project Value	Initial Value	Comment	String Size
bacnet_module_address		UINT		1280			
bacnet_module_datalen		UINT		256			
bacnet_module_read_fn		PLUGIN_READ		...	...		
bacnet_module_write_fn		PLUGIN_WRITE		...	...		
bacnet_convert_var		ULINT		0			
bacnet_convert_fn		BACNET_Conv		...	...		
bacnet_convert_mode		USINT		0			
bacnet_convert_offset		DINT		0			
BACNET_MODULE_SLOT		UINT		1			

The second step is to inspect the example BAC_Slot1. First examined will be the first line of this example program.

The first line of code in the program tells the BACNET_Convert, user-defined function block in which plugin slot the module is located. In this case, the module is in slot 1.

BACNET_MODULE_SLOT := 1;

Analog Input

The next section is the analog input portion of the BAC_Slot1 example program.

The first variable (called Temp_Chiller1 to the BACnet system and `Temp_Chiller` to the PLC) is converted to ULINT and placed into variable `bacnet_convert_var` at offset 0 of the byte array table because it is writing AnalogInput 0. Variable `bacnet_convert_mode` is 1, which means this is an operation to convert AnalogInput objects in preparation to write them to the 2080sc-BAC module.

If it is AnalogInput 1, the offset will be 8 as each BACNET object instance takes 8 bytes.

The next variable (called `Temp_Tower1` to the BACnet system and `Temp_Tower` to the PLC) is converted to ULINT and placed into variable `bacnet_convert_var` at offset 8 of the byte array table because it is writing AnalogInput 1.

Each BACNET object instance takes 8 bytes, so the AnalogInput with ID 2 is at offset 16, and the AnalogInput with ID 3 is at offset 24.

`Temp_Chiller1` is informing the BACnet system of the temperature of the water coming from the condenser of the chiller to be pumped to the cooling tower.

`Temp_Tower1` is informing the BACnet system of the temperature of the water returning to the condenser of the chiller from the cooling tower.

`Temp_Ambient1` is informing the BACnet system of the temperature of the air at the cooling tower.

`Humidity_Ambient1` is informing the BACnet system of the humidity of the air at the cooling tower.

`Temp_Floor1` is informing the BACnet system of the temperature of the floor at the cooling tower.

`Temp_Humidity1` is informing the BACnet system of the relative humidity of the air at the cooling tower.

```
(* BACnet Analog Inputs *)

(* 0, AnalogInput, Temp_Chiller1 *)
bacnet_convert_var := ANY_TO_ULINT(Temp_Chiller * 1000.0);
bacnet_convert_offset := 0;
bacnet_convert_mode := 1;
bacnet_convert_fn(bacnet_convert_offset,
bacnet_convert_var, bacnet_convert_mode);

(* 1, AnalogInput, Temp_Tower1 *)
bacnet_convert_var := ANY_TO_ULINT(Temp_Tower * 1000.0);
bacnet_convert_offset := 8;
bacnet_convert_mode := 1;
bacnet_convert_fn(bacnet_convert_offset,
bacnet_convert_var, bacnet_convert_mode);

 (* 2, AnalogInput, Temp_Ambient1 *)
bacnet_convert_var := ANY_TO_ULINT(Temp_Ambient * 1000.0);
bacnet_convert_offset := 16;
bacnet_convert_mode := 1;
bacnet_convert_fn(bacnet_convert_offset,
bacnet_convert_var, bacnet_convert_mode);

(* 1, AnalogInput, Temp_Floor1 *)
bacnet_convert_var := ANY_TO_ULINT(Temp_Floor * 1000.0);
bacnet_convert_offset := 24;
bacnet_convert_mode := 1;
bacnet_convert_fn(bacnet_convert_offset,
bacnet_convert_var, bacnet_convert_mode);

 (* 3, AnalogInput, Humidity_Ambient1 *)
bacnet_convert_var := ANY_TO_ULINT(Humidity_Ambient *
1000.0);
bacnet_convert_offset := 32;
bacnet_convert_mode := 1;
bacnet_convert_fn(bacnet_convert_offset,
bacnet_convert_var, bacnet_convert_mode);

(* Write Analog Input 1 Table to Module *)
bacnet_module_address := 256;
bacnet_module_datalen := 256;
bacnet_module_write_fn(TRUE, BACNET_MODULE_SLOT,
bacnet_module_address, bacnet_module_datalen,
bacnet_convert_fn.bac_raw_tbl);
```

Binary Input

The next section is the binary input portion of the BAC_Slot1 example program.

The binary inputs are handled a little differently than the analog inputs.

First, the bits are converted to ULINT and placed into variable `bacnet_convert_`var at offset 0, 1, 2, 3, 4, and 5 using convert mode 3.

Variable `bacnet_convert_mode` is 3, which means this is an operation to convert BinaryInput objects in preparation to write them to the 2080sc-BAC module.

`Tower_FLT1` is informing the BACnet system of the status of the cooling tower system.

`Chiller_FLT1` is informing the BACnet system of the status of the chiller condenser system.

`Occupied1` is informing the BACnet system of the status of personnel in the vicinity of the cooling tower and chiller condenser area.

`Tower_AUTO1` is informing the BACnet system of the state of the inputs local to the PLC.

`Tower_HAND1` is informing the BACnet system of the state of the inputs local to the PLC.

`IntoBackWash` is informing the BACnet system when a tower goes into backwash (self-cleaning) operation. This means the tower is at less than maximum capacity when in operation.

```
(* BACnet Binary Inputs *)

(* 0, BinaryInput, Tower_FLT1 *)
bacnet_convert_var := ANY_TO_ULINT(Tower_FLT);
bacnet_convert_offset := 0;
bacnet_convert_mode := 3;
bacnet_convert_fn(bacnet_convert_offset, bacnet_convert_var,
bacnet_convert_mode);

(* 1, BinaryInput, Chiller_FLT1 *)
bacnet_convert_var := ANY_TO_ULINT(Chiller_FLT);
bacnet_convert_offset := 1;
bacnet_convert_mode := 3;
bacnet_convert_fn(bacnet_convert_offset, bacnet_convert_var,
bacnet_convert_mode);


(* 2, BinaryInput, Occupied_1 *)
bacnet_convert_var := ANY_TO_ULINT(Occupied);
bacnet_convert_offset := 2;
bacnet_convert_mode := 3;
bacnet_convert_fn(bacnet_convert_offset, bacnet_convert_var,
bacnet_convert_mode);

(* 3, BinaryInput, Tower_AUTO1 *)
bacnet_convert_var := ANY_TO_ULINT(Tower_AUTO);
bacnet_convert_offset := 3;
bacnet_convert_mode := 3;
bacnet_convert_fn(bacnet_convert_offset, bacnet_convert_var,
bacnet_convert_mode);


(* 4, BinaryInput, Tower_HAND1 *)
bacnet_convert_var := ANY_TO_ULINT(Tower_HAND);
bacnet_convert_offset := 4;
bacnet_convert_mode := 3;
bacnet_convert_fn(bacnet_convert_offset, bacnet_convert_var,
bacnet_convert_mode);


(* 5, BinaryInput, IntoBackWash *)
bacnet_convert_var := ANY_TO_ULINT(INTO_BW);
bacnet_convert_offset := 5;
```

```
bacnet_convert_mode := 3;
bacnet_convert_fn(bacnet_convert_offset, bacnet_convert_var,
bacnet_convert_mode);

(* Write Binary Input Table to Module *)
bacnet_module_address := 96;
bacnet_module_datalen := 8;
bacnet_module_write_fn(TRUE, BACNET_MODULE_SLOT,
bacnet_module_address, bacnet_module_datalen,
bacnet_convert_fn.bac_raw_tbl);
```

Analog Output

For Analog Output, a `bacnet_module_read_fn` is performed to read the whole analog output table to `bac_raw_tbl`, defined in function block `bacnet_convert_fn`.

Next, the AnalogOutput objects are written to the PLC variables after conversion to the proper PLC tag data type (Real or DINT). This write occurs every PLC scan.

`Temp_SP1` is the BACnet system command value of the temperature setpoint for the water returning to the condenser of the chiller from the cooling tower.

`Fan_Speed1` is the BACnet system command value of the speed of the cooling tower fan.

`Pump_Speed1` is the BACnet system command value of the speed of the pump sending water to cooling tower.

```
 (* BACnet Analog Outputs *)

(* Read Analog Output 1 Table from Module *)
bacnet_module_address := 1280;
bacnet_module_datalen := 256;
bacnet_module_read_fn(TRUE, BACNET_MODULE_SLOT,
bacnet_module_address, bacnet_module_datalen,
bacnet_convert_fn.bac_raw_tbl);

(* 0, AnalogOutput, Temp_SP1 *)
bacnet_convert_mode := 0;
bacnet_convert_offset := 0;
bacnet_convert_fn(bacnet_convert_offset, bacnet_convert_var,
bacnet_convert_mode);
Temp_SP := ANY_TO_REAL(bacnet_convert_fn.tmp_raw_var) / 1000.0;

(* 1, AnalogOutput, Fan_Speed1 *)
bacnet_convert_mode := 0;
bacnet_convert_offset := 8;
bacnet_convert_fn(bacnet_convert_offset, bacnet_convert_var,
bacnet_convert_mode);
Fan_Speed := ANY_TO_DINT(bacnet_convert_fn.tmp_raw_var);

(* 2, AnalogOutput, Pump_Speed1 *)
bacnet_convert_mode := 0;
bacnet_convert_offset := 16;
bacnet_convert_fn(bacnet_convert_offset, bacnet_convert_var,
bacnet_convert_mode);
Pump_Speed := ANY_TO_DINT(bacnet_convert_fn.tmp_raw_var);
```

Binary Output

For Binary Output, a `bacnet_module_read_fn` is performed to read the whole binary output table to `bac_raw_tbl`, defined in function block `bacnet_convert_fn`.

Next, the BinaryOutput objects are written to the PLC variables after conversion to the Bool PLC tag data type. This write occurs every PLC scan.

Variable `bacnet_convert_mode` is 2, which means this is an operation to convert BinaryOutput objects after reading them from the 2080sc-BAC module.

`Floor_Enable1` is the BACnet system command to allow changing the temperature of the floor by using water going to or coming from the cooling tower.

`Floor_Heat1` is the BACnet system command to raise the temperature of the floor by changing valves to use warm water from the condenser of the chiller before sending it the cooling tower.

`Floor_Cool1` is the BACnet system command to lower the temperature of the floor by changing valves to use cool water returning to the condenser of the chiller from the cooling tower.

```
(* BACnet Binary Outputs *)

bacnet_module_address := 1024;
bacnet_module_datalen := 8;
bacnet_module_read_fn(TRUE, BACNET_MODULE_SLOT,
bacnet_module_address, bacnet_module_datalen,
bacnet_convert_fn.bac_raw_tbl);

(* 0, BinaryOutput, Floor_Enable1 *)
bacnet_convert_mode := 2;
bacnet_convert_offset := 0;
bacnet_convert_fn(bacnet_convert_offset, bacnet_convert_var,
bacnet_convert_mode);
Floor_Enable := ANY_TO_BOOL(bacnet_convert_fn.tmp_raw_var);

(* 1, BinaryOutput, Floor_Heat1 *)
bacnet_convert_mode := 2;
bacnet_convert_offset := 1;
bacnet_convert_fn(bacnet_convert_offset, bacnet_convert_var,
bacnet_convert_mode);
Floor_Heat := ANY_TO_BOOL(bacnet_convert_fn.tmp_raw_var);

(* 2, BinaryOutput, Floor_Cool1 *)
bacnet_convert_mode := 2;
bacnet_convert_offset := 2;
bacnet_convert_fn(bacnet_convert_offset, bacnet_convert_var,
bacnet_convert_mode);
Floor_Cool := ANY_TO_BOOL(bacnet_convert_fn.tmp_raw_var);
```

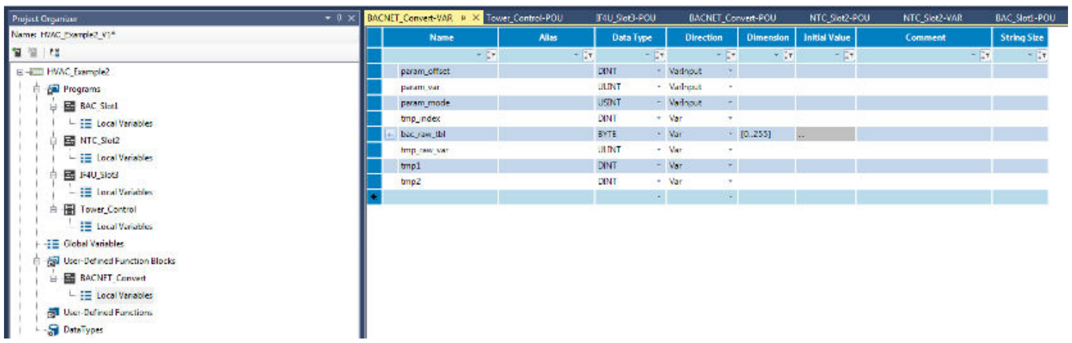
3.7.2 BACnet_Convert Function Block

Spectrum Controls, Inc. also provides a sample CCW project with a function block named BACNET_Convert in the zipped project file, available upon request. If you have the same PLC as the sample project, you may directly use this project.

If you are using a different Micro800 PLC, the BACNET_Convert UDF (User

Defined Function) has been exported so you can import it directly into your favorite flavor of Micro800 PLC.

This is a view of the CCW program showing on the left, the project organizer with a view of the BACNET_Convert UDF and on the right, the local variables of the UDF block.



Name	Alias	Data Type	Direction	Dimension	Initial Value	Comment	String Size
param_offset		DINT	VarInput				
param_var		ULINT	VarInput				
param_mode		USINT	VarInput				
tmp_index		DINT	Var				
bac_raw_tbl		BYTE	Var	[0..255]	...		
tmp_raw_var		ULINT	Var				
tmp1		DINT	Var				
tmp2		DINT	Var				

Here is the code listing in structured text style of the BACNET_Convert UDF:

```
(*
 if param_mode = 0
  convert byte array bac_raw_tbl to ULINT variable tmp_raw_var
  This is for Analog Output Table
*)

IF param_mode = 0 THEN
              tmp_index := param_offset;
              tmp_raw_var := 0;
              tmp_raw_var := tmp_raw_var +
ANY_TO_ULINT(bac_raw_tbl[tmp_index]) * 1;
              tmp_index := tmp_index + 1;
              tmp_raw_var := tmp_raw_var +
ANY_TO_ULINT(bac_raw_tbl[tmp_index]) * 16#100;
              tmp_index := tmp_index + 1;
              tmp_raw_var := tmp_raw_var +
ANY_TO_ULINT(bac_raw_tbl[tmp_index]) * 16#10000;
              tmp_index := tmp_index + 1;
              tmp_raw_var := tmp_raw_var +
ANY_TO_ULINT(bac_raw_tbl[tmp_index]) * 16#1000000;
              tmp_index := tmp_index + 1;
              tmp_raw_var := tmp_raw_var +
ANY_TO_ULINT(bac_raw_tbl[tmp_index]) * 16#100000000;
              tmp_index := tmp_index + 1;
              tmp_raw_var := tmp_raw_var +
ANY_TO_ULINT(bac_raw_tbl[tmp_index]) * 16#10000000000;
              tmp_index := tmp_index + 1;
              tmp_raw_var := tmp_raw_var +
ANY_TO_ULINT(bac_raw_tbl[tmp_index]) * 16#1000000000000;
              tmp_index := tmp_index + 1;
              tmp_raw_var := tmp_raw_var +
ANY_TO_ULINT(bac_raw_tbl[tmp_index]) * 16#100000000000000;

 (*
 if param_mode == 1
  convert ULINT variable tmp_raw_var to byte array bac_raw_tbl
  This is for Analog Input Table
*)

ELSIF param_mode = 1 THEN
       tmp_index := param_offset;
       tmp_raw_var := param_var;

              bac_raw_tbl[tmp_index] := ANY_TO_BYTE(tmp_raw_var);
              tmp_raw_var := tmp_raw_var / 256;
              tmp_index := tmp_index + 1;
              bac_raw_tbl[tmp_index] := ANY_TO_BYTE(tmp_raw_var);
              tmp_raw_var := tmp_raw_var / 256;
              tmp_index := tmp_index + 1;
              bac_raw_tbl[tmp_index] := ANY_TO_BYTE(tmp_raw_var);
              tmp_raw_var := tmp_raw_var / 256;
              tmp_index := tmp_index + 1;
              bac_raw_tbl[tmp_index] := ANY_TO_BYTE(tmp_raw_var);
              tmp_raw_var := tmp_raw_var / 256;
              tmp_index := tmp_index + 1;
              bac_raw_tbl[tmp_index] := ANY_TO_BYTE(tmp_raw_var);
              tmp_raw_var := tmp_raw_var / 256;
              tmp_index := tmp_index + 1;
              bac_raw_tbl[tmp_index] := ANY_TO_BYTE(tmp_raw_var);
              tmp_raw_var := tmp_raw_var / 256;
              tmp_index := tmp_index + 1;
              bac_raw_tbl[tmp_index] := ANY_TO_BYTE(tmp_raw_var);
              tmp_raw_var := tmp_raw_var / 256;
              tmp_index := tmp_index + 1;
              bac_raw_tbl[tmp_index] := ANY_TO_BYTE(tmp_raw_var);
              tmp_index := tmp_index + 1;

(*
 if param_mode == 2
  This is for Binary Output Table
*)

ELSIF param_mode = 2 THEN
       tmp_index := param_offset / 8;
       tmp1 := MOD(param_offset, 8);

       tmp2 := ANY_TO_DINT(bac_raw_tbl[tmp_index]);
       tmp2 := SHR(tmp2, tmp1);
       tmp2 := AND_MASK(tmp2, 1);
       IF tmp2 = 1 THEN
              tmp_raw_var := 1;
       ELSE
              tmp_raw_var := 0;
       END_IF;
```

```
(*
 if param_mode == 3
  This is for Binary Input Table
*)

ELSIF param_mode = 3 THEN
       tmp_index := param_offset / 8;
       tmp1 := MOD(param_offset, 8);
       tmp2 := 1;
       tmp2 := SHL(tmp2, tmp1);

       tmp1 := ANY_TO_DINT(bac_raw_tbl[tmp_index]);
       IF param_var = 0 THEN      (* set 0 *)
              tmp1 := NOT_MASK(tmp1);
              tmp1 := OR_MASK(tmp1, tmp2);
              tmp1 := NOT_MASK(tmp1);
       ELSE   (* set 1 *)
              tmp1 := OR_MASK(tmp1, tmp2);
       END_IF;
       bac_raw_tbl[tmp_index] := ANY_TO_BYTE(tmp1);
ELSE
       (* do nothing *)
       (* tmp_index := param_offset / 8; *)
END_IF;
```

In the BAC_Slot1 program, the function `bacnet_convert_fn` type as `BACNET_Convert` is called to convert data types between whatever the PLC variable is and the requirements of the BAC module.

Since CCW has many different data types, this function converts the different PLC data types to a byte array. When the program reads from the module, the module fills this byte array first. Then the program reads the array and converts the data to the data type of the PLC variable. When the program writes to the module, it converts the variable to type BYTE, puts it in the proper place in the array, and writes the array to the module.

The following examples shows how analog output variables are read from the module, and how an analog input variable is written to the module using the structured text generated by the module.

For this Analog Input example, the variable called `Humidity_Ambient` is converted to ULINT variable `bacnet_convert_var` and then placed at offset 23 of the byte array table because you are writing AnalogInput 4 (4 X 8). Variable `bacnet_convert_mode` is 1 which means this is an operation to write AnalogInput objects to the 2080sc-BAC module. The `bacnet_convert_fn` is then called to insert the data into the correct place in the byte array.

After all the AI have been assembled into the array, the `bacnet_module_write_fn` is then called to move the data from the array into the module so the BACnet system can read it.

```
(* 3, AnalogInput, Humidity_Ambient1 *)
bacnet_convert_var := ANY_TO_ULINT(Humidity_Ambient * 1000.0);
bacnet_convert_offset := 32;
bacnet_convert_mode := 1;
```

```
bacnet_convert_fn(bacnet_convert_offset, bacnet_convert_var,
bacnet_convert_mode);

(* Write Analog Input 1 Table to Module *)
bacnet_module_address := 256;
bacnet_module_datalen := 256;
bacnet_module_write_fn(TRUE, BACNET_MODULE_SLOT,
bacnet_module_address, bacnet_module_datalen,
bacnet_convert_fn.bac_raw_tbl);
```

For this Analog Output example, a read operation moves the analog output table from the module to bac_raw_tbl defined in function block `bacnet_convert_fn`. Then individual AnalogOutput objects are pulled out of the table and placed into the variables after conversion from ULINT to the proper data type for the tag.

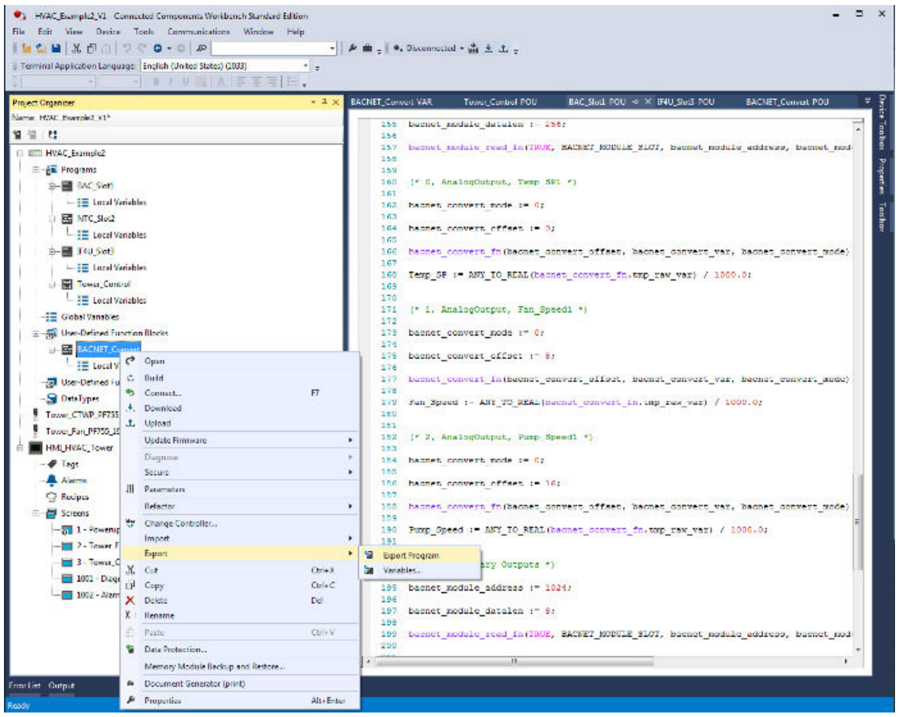
The variable `Temp_SP` is pulled from the table at offset 0, and the variable `Fan_Speed` is pulled from the table at offset 8.

Variable `bacnet_convert_mode` is 0 which means this is an operation to convert AnalogOutput objects after reading them from the 2080sc-BAC module.

```
(* Read Analog Output 1 Table from Module *)
bacnet_module_address := 1280;
bacnet_module_datalen := 256;
bacnet_module_read_fn(TRUE, BACNET_MODULE_SLOT,
bacnet_module_address, bacnet_module_datalen,
bacnet_convert_fn.bac_raw_tbl);

(* 0, AnalogOutput, Temp_SP1 *)
bacnet_convert_mode := 0;
bacnet_convert_offset := 0;
bacnet_convert_fn(bacnet_convert_offset, bacnet_convert_var,
bacnet_convert_mode);
Temp_SP := ANY_TO_REAL(bacnet_convert_fn.tmp_raw_var) / 1000.0;

(* 1, AnalogOutput, Fan_Speed1 *)
bacnet_convert_mode := 0;
bacnet_convert_offset := 8;
bacnet_convert_fn(bacnet_convert_offset, bacnet_convert_var,
bacnet_convert_mode);
Fan_Speed := ANY_TO_DINT(bacnet_convert_fn.tmp_raw_var);
```

3.7.3 CCW Export and Import of Function Blocks

CCW provides a way to export function blocks as shown below. You may also import the example BACNET_Convert UDF to your new CCW project.



```
bacnet_module_datalen := 256;

bacnet_module_read_fn(TRUE, BACNET_MODULE_SLOT, bacnet_module_address, bacnet_mod


(* 0, AnalogOutput, Temp_SP1 *)

bacnet_convert_mode := 0;

bacnet_convert_offset := 0;

bacnet_convert_fn(bacnet_convert_offset, bacnet_convert_var, bacnet_convert_mode)

Temp_SP := ANY_TO_REAL(bacnet_convert_fn.tmp_raw_var) / 1000.0;


(* 1, AnalogOutput, Fan_Speed1 *)

bacnet_convert_mode := 0;

bacnet_convert_offset := 8;

bacnet_convert_fn(bacnet_convert_offset, bacnet_convert_var, bacnet_convert_mode)

Fan_Speed := ANY_TO_REAL(bacnet_convert_fn.tmp_raw_var) / 1000.0;


(* 2, AnalogOutput, Pump_Speed1 *)

bacnet_convert_mode := 0;

bacnet_convert_offset := 16;

bacnet_convert_fn(bacnet_convert_offset, bacnet_convert_var, bacnet_convert_mode)

Pump_Speed := ANY_TO_REAL(bacnet_convert_fn.tmp_raw_var) / 1000.0;


ary Outputs *)

bacnet_module_address := 1024;

bacnet_module_datalen := 8;

bacnet_module_read_fn(TRUE, BACNET_MODULE_SLOT, bacnet_module_address, bacnet_mod
```

3.7.4 IP Address and User Interface Password Recovery

During startup, the IP address is written to the analog output object memory for you to read. This is helpful if you forget the IP address of the module. The password for the user interface is written to the same location in case you need to recover the password.

IP addresses and the user interface password are in the Analog Output 2 data area as shown in the following table:

User Interface Password	0×6D0- 0×6EF	Analog Output Tag 61
IP Address	0×6FC-0×6FF	Analog Output Tag 63

To read the values using structured text in CCW, add the following code from the example project provided by Spectrum Controls, Inc. to the Main program.

The code snippet reads FPGA address 0×600 into the CCW global variable's `tag_id`, which is defined as 256 bytes of BYTE.

NOTE	You will need to add `tag_id` as a global variable.

The code reads the red-colored, FPGA address, 1536 (this address is 0×600) for 256 bytes into the `tag_id` array.

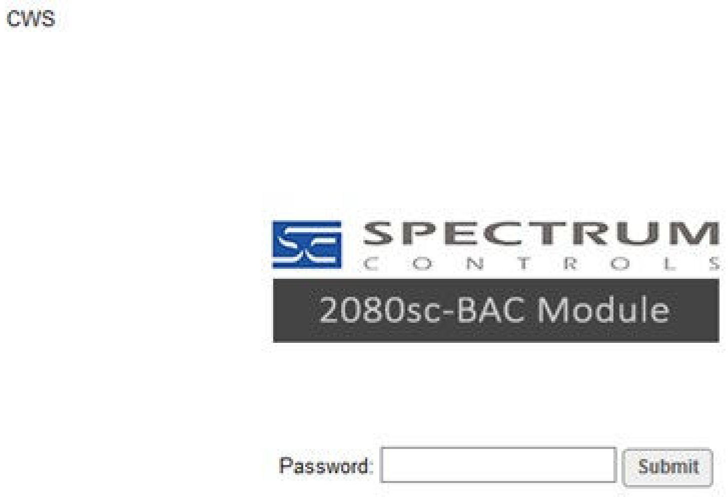
```
    (* Read Table from Module *)

    bacnet_module_address := 1536;

    bacnet_module_datalen := 256;

    bacnet_module_read_fn(TRUE, BACNET_MODULE_SLOT,
    bacnet_module_address, bacnet_module_datalen, tag_id);
```

Download the new program to the PLC and run the program.

You can read the IP address and the user interface password by monitoring the `tag_id` global variable.

3.7.5 Upgrading the Software from Version 1.*n* to Version 2.*n*

Software versions released before Version 2.*n* used Adobe Flash. From Version 2.*n* onwards, the user interface uses HTML5. To upgrade to Version 2.*n* from any Version 1.*n* release through the user interface, click the upgrade button and select an updated .UPT file. The upgrade starts automatically. Once the upgrade is complete, the module restarts. The upgraded user interface will show the text 'CWS' on the upper-left corner of the interface. This does not affect the functioning of the interface or the module:



To remove the lettering, upgrade your software using a later version of the 2.*n* software that you can download from the Spectrum Controls, Inc. website and select the **Upgrade** button on the user interface.

3.7.6 Memory Mapping

This section discusses memory mapping between a BACnet module and the PLC. The BACnet module has internal memory that is mapped to the PLC memory range. On the PLC, each slot has 2 Kbytes of memory. The internal RAM in the Module is arranged as below.

This section is for reference only as all the mappings are automatically generated in the structured text.

BACnet Binary Input Block (0×60 – 0×67)

Register Name	Addr.	Comments	Default	R/W From PLC	R/W From Module
BAC_BI	0×60 – 0×67	BACnet Binary Input data (tags 0 – 63)	0	W	R

Each bit of the BACnet Binary Input represents a true/false value for one BACnet Binary Input object. This block supports up to 64 such objects.

BACnet Binary Output Block (0×400 – 0×407)

Register Name	Addr.	Comments	Default	R/W From PLC	R/W From Module
BAC_BO	0×400 – 0×407	BACnet Binary Output data (tags 0 – 63)	0	W	R

Each bit of the BACnet Binary Value represents a true/false value for one BACnet Binary Value object. This block supports up to 64 such objects.

BACnet Analog Input 1 Block (0×100 – 0×1FF)

Register Name	Addr.	Comments	Default	R/W From PLC	R/W From Module
BAC_AI_0	0×100 – 0×107	BACnet Analog Input tag 0	0	W	R
BAC_AI_1	0×108 – 0×10F	BACnet Analog Input tag 1	0	W	R
…	…	…	…	…	…
BAC_AI_31	0×1F8 – 0×1FF	BACnet Analog Input tag 31	0	W	R

This block is for the first 32 BACnet Analog Input tag values.

BACnet Analog Input objects are read from a user-specified PLC variable, written by the PLC to the module, and then made available via the BACnet stack as a read-only value.

BACnet Analog Input 2 Block (0×200 – 0×2FF)

Register Name	Addr.	Comments	Default	R/W From PLC	R/W From Module
BAC_AI_32	0×200 – 0×207	BACnet Analog Input tag 32	0	W	R
…	…	…	…	…	…
BAC_AI_63	0×2F8 – 0×2FF	BACnet Analog Input tag 63	0	W	R

This block is for the last 32 BACnet Analog Input tag values.

BACnet Analog Output 1 Block (0×500 – 0×5FF)

Register Name	Addr.	Comments	Default	R/W From PLC	R/W From Module
BAC_AO_0	0×500 – 0×507	BACnet Analog Output tag 0	0	R	W
BAC_AO_1	0×508 – 0×50F	BACnet Analog Output tag 1	0	R	W
…	…	…	…	…	…
BAC_AO_31	0×5F8 – 0×5FF	BACnet Analog Output tag 31	0	R	W

This block is for the first 32 BACnet Analog Output tag values.

BACnet Analog Output objects are made available via the BACnet stack as a writable value that is then written by the module to the PLC. The data are then copied into a user-specified PLC variable.

BACnet Analog Output 2 Block (0×600 – 0×6FF)

Register Name	Addr.	Comments	Default	R/W From PLC	R/W From Module
BAC_AO_32	0×600 – 0×607	BACnet Analog Output tag 32	0	R	W
…	…	…	…	…	…
BAC_AO_63	0×6F8 – 0×6FF	BACnet Analog Output tag 63	0	R	W

This block is for the last 32 BACnet Analog Output tag values.

Analog Output Block address 0×600 to 0×6FF values are initialized to 0. However, BAC_AO_61 to BAC_AO_63 initial values are not 0 because they contain the module IP address and user interface password. The module will set the initial value to AO_61 to AO_63 for servicing purpose. Those initial values will only be written once after module is powered up, then AO_61 to AO_63 are available for you to use if you set up the module to use AO_61 to AO_63.

Module Block (0×00 – 0×1F)

Register Name	Addr.	Comments	Default	R/W From PLC	R/W From Module
MOD_ID_LO	0×00	Module ID	196	R	W
MOD_ID_HI	0×01		0	R	W
VENDOR_ID_LO	0×02	Vendor ID	58	R	W
VENDOR_ID_HI	0×03		0	R	W
PRODUCT_TYPE_LO	0×04		12	R	W
PRODUCT_TYPE_HI	0×05		0	R	W

3.7.7 Data Type Range

The maximum and minimum value of the tags is decided by the PLC data type and it is also limited by data storage and conversion.

BOOL	TRUE or FALSE
SINT	-128 to 127
USINT	0 to 255
BYTE	0 to 255
INT	-32768 to 32767
UINT	0 to 65535
WORD	0 to 65535
DINT	-2147483648 to 2147483647
UDINT	0 to 429467295
DWORD	0 to 429467295
LINT	-1e18 to 1e18[1]
ULINT	0 to 1e18
LWORD	0 to 1e18
REAL	-1e18 to 1e18[2]
LREAL	-1e18 to 1e18

3.7.8 BACnet Diagnostic Tools

Spectrum Controls, Inc. provides multiple Windows command-line executables you may use to communicate with the module. These utilities are compiled from an open source BACnet stack. The tools require an Ethernet-to-BACnet MS/TP router if you are using the RS-485 multidrop physical layer. This gives you the ability to send BACnet messages across the BACnet MS/TP network for easier troubleshooting. You can find the source code on sourceforge at http://sourceforge.net/projects/bacnet/files/bacnet-stack/

Setup

To access the BACnet/IP traffic, set the PC IP address using a network address that is on the same network as the BACnet router IP address.

Using the 2080sc-BAC module User Interface, set the baud rate and other parameters, then connect the MS/TP side of the BACnet router to the MS/TP connection on the 2080sc-BAC module.

The router converts BACnet/IP packets to MS/TP packets that are then routed to the module.

BACnet WhoIs Service

The executable bacwi.exe, looks for who is on the BACnet network. When a device is discovered on the network, the master is then able to talk to the device.

Examples:

- bacwi 123 sends a BACnet WhoIs service request to Device 123.

[1] LINT and ULINT cannot have the same upper range.
[2] LREAL has twice as many bits to work with than REAL

- bacwi 1000 9000 sends a BACnet WhoIs service request to Devices from 1000 to 9000.
- bacwi -1 sends a BACnet WhoIs service request to all devices.

When the BACnet Master wishes to discover if there is a device at a specific address, it sends out a 'BACnet WhoIs service *n*' message, where *n* is the Device ID. The device responds with an 'I-Am *n*', and provides the master with either an IP address if the device is an Ethernet type, or a MAC address if the device is an MS/TP type.

The BACnet Master can also send a 'BACnet WhoIs service *m* to *n*' message, where *m* is the lower limit, and *n* is the upper limit. Each device in that range will respond with an 'I-Am *n*'.

It may also send a 'BACnet WhoIs service null', which is a broadcast to all reachable devices, and all devices on the network will respond with an 'I-Am *n*'.

After discovery of all devices on the network, the BACnet Master will bind all responding devices' addresses to their device ID.

ReadProperty

The executable bacrp.exe (**BACnet ReadProperty service**) will read a property of the object-type from the device-instance (the assigned node) on the network.

Usage: `bacrp device-instance object-type object-instance property [index]`

Examples:

To read the present value of AI1 at device 200121, do this:

```
bacrp 200121 0 1 85
```

- `{param 1} – 200121. This is a module instance number.`
- `{param 2} – 0. This is the object type. Object type is defined as AI=0, AO=1, BI=3, BO=4.`
- `{param 3} – 1. This is the object instance.`
- `{param 4} – 85. This is the property. (85=present value).`

To read the present value of BI4 at device 123, do this:

```
bacwp 123 3 4 85
```

- `{param 1} – 123. This is a module instance number.`
- `{param 2} – 1. This is the object type. Object type is defined as AI=0, AO=1, BI=3, BO=4.`
- `{param 3} – 4. This is the object instance.`
- `{param 4} – 85. This is the property. (85=present value).`

The BACnet object-instance (an integer) is the object instance number of the object that you are reading from. For example, if you were reading from Analog Output 2, the object-instance would be 2.

For more information on bacrp, review the help by executing the command bacrp --help.

WriteProperty

The executable bacwp.exe (**BACnet WriteProperty service**) will write a property to the object-type at the device-instance (the assigned node) on the network.

Usage: `bacwp device-instance object-type object-instance property priority index tag value [tag value...]`

Examples:

To write a present value of 100 to AO7 at device 123 at priority 16, do this:

```
bacwp 123 1 7 85 16 -1 4 100
```

- `{param 1} – 123. This is the module instance number.`
- `{param 2} – 1. This is the object type. Object type is defined as AI=0, AO=1, BI=3, BO=4.`
- `{param 3} – 7. This is the object instance.`
- `{param 4} – 85. This is the property (85=present value).`
- `{param 5} – 16. This is the priority.`
- `{param 6} – -1. This is the index. Use -1 for a single property write.`
- `{param 7} – 4. Value type. Use 4 for AO.`
- `{param 8} – 100. Value. This is the actual value to be written to the tag.`

To write a present value of 1 to BO3 at device 200121 at priority 15, do this:

```
bacwp 200121 4 3 85 15 -1 9 1
```

- `{param 1} – 200121. This is the module instance number.`
- `{param 2} – 4. This is the object type. Object type is defined as AI=0, AO=1, BI=3, BO=4.`
- `{param 3} – 3. This is the object instance.`
- `{param 4} – 85. This is the property (85=present value).`
- `{param 5} – 15. This is the priority.`
- `{param 6} – -1. This is the index. Use -1 for a single property write.`
- `{param 7} – 9. Value type. Use 9 for BO.`
- `{param 8} – 1. Value. This is the actual value to be written to the tag.`

The BACnet priority levels range from 1 to 16, where 16 is the lowest priority. High priority (lower number) overrides lower priority.

The BACnet index (an integer) parameter is the index number of an array. If the property is an array, individual elements can be written to if supported. If this parameter is -1, the index is ignored.

The BACnet object-instance (an integer) is the object instance number of the object that you are writing to. For example, if you were writing to Analog Output 2, the object-instance would be 2.

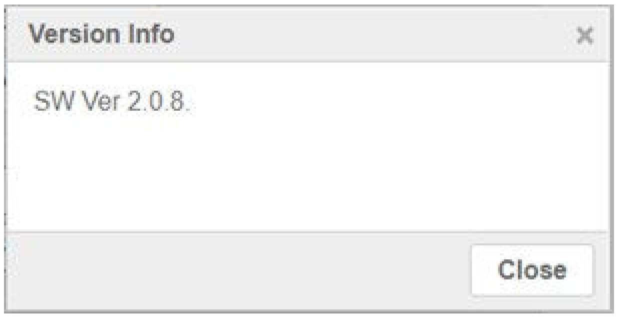
For more information on bacwp, review the help by executing the command bacwp --help.

Section 3.8 Viewing Version Information

To view the current version of your software:

1. From the main software dialog, click **Version Info**:

   The Version Info dialog appears:

   SW Ver 2.0.8.

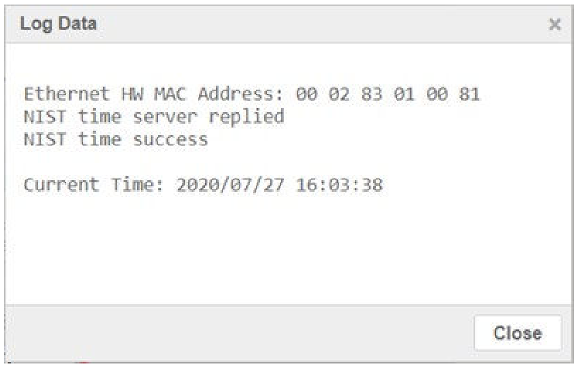
   The current software version is listed.
2. To exit, click **Close**.

Section 3.9 Viewing Log Information

To view current Log Information:

1. From the main software dialog, click **Show Log**:
2.

   The Log Data dialog appears:

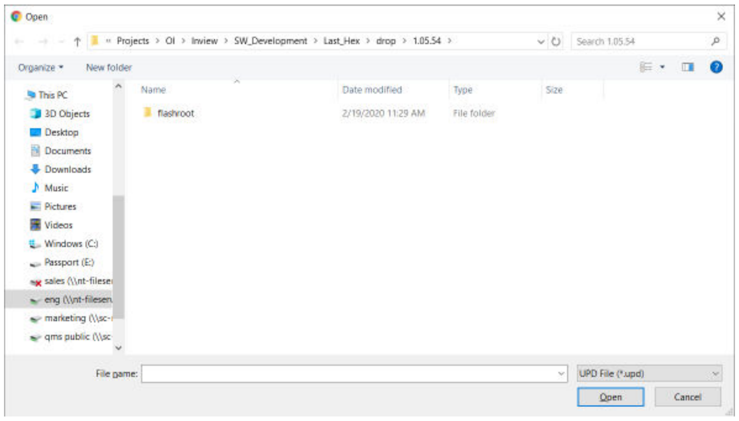
```
Ethernet HW MAC Address: 00 02 83 01 00 81
NIST time server replied
NIST time success

Current Time: 2020/07/27 16:03:38
```

   The Ethernet hardware MAC address is listed, along with server response success and current date and time information.
3. To exit, click **Close**.

Section 3.10 Upgrading the Software

To upgrade to a new version of software:

1. Navigate to the 2080sc-BAC location on the Spectrum Controls, Inc. web site (www.spectrumcontrols.com), and download the firmware to your personal computer. This file contains both the application and the user interface software. The file will have a .UPT file extension.
2. From the main software dialog, click **Upgrade**:
3. Upgrade
4. Use the dialog that opens to navigate to where you downloaded the upgrade file and click on the file.



The software upgrades and the module reboots once the upgrade is complete.

Section 3.11 Saving Changes

To save changes made to the software fields:

1. From the main software dialog, click **Save**:

   Save

2. The software saves the system configuration, or system configuration changes to the module.

Section 3.12 Reloading System Configuration

To reload the system configuration from the module:

1. From the main software dialog, click **Reload**:

   Reload

2. The software reloads the system configuration from the module into the software.

Chapter 4 Implementing the BACnet Protocol

This chapter describes the BACnet protocol and its implementation in the BACnet Communications Module:

- BACnet Protocol Requirements.
- Tag Mapping.
- API Address Mapping and other considerations.

Section 4.1 BACnet Object Types

The following data shows the supported BACnet object types and the properties supported for each object type. Each table also includes the property data type, the identifier number, and whether the property is read or read/write.

Analog Input-BACnetObjectType = 0

Properties Supported	Property Data	Type Identifier	Read/Write
Object_Identifier	BACnetObjectIdentifier	75	R
Object_Name	CharacterString	77	R
Object_Type	BACnetObjectType	79	R
Present_Value	REAL	85	R/W
Status_Flags	BACnetStatusFlags	111	R
Event_State	BACnetEventState	36	R
Out_Of_Service	BOOLEAN	81	R
Units	BACnetEngineeringUnits	117	R

Analog Output-BACnetObjectType = 1

Properties Supported	Property Data	Type Identifier	Read/Write
Object_Identifier	BACnetObjectIdentifier	75	R
Object_Name	CharacterString	77	R
Object_Type	BACnetObjectType	79	R
Present_Value	REAL	85	R/W
Status_Flags	BACnetStatusFlags	111	R
Event_State	BACnetEventState	36	R
Units	BACnetEngineeringUnits	117	R

Properties Supported	Property Data	Type Identifier	Read/Write
Priority Array	BACnetPriorityArray	87	R
Relinquish Default	REAL	104	R

Binary Input-BACnetObjectType = 3

Properties Supported	Property Data	Type Identifier	Read/Write
Object_Identifier	BACnetObjectIdentifier	75	R
Object_Name	CharacterString	77	R
Object_Type	BACnetObjectType	79	R
Present_Value	BACnetBinaryPV	85	R/W
Status_Flags	BACnetStatusFlags	111	R
Event_State	BACnetEventState	36	R
Out_Of_Service	BOOLEAN	81	R
Polarity	BACnetPolarity	84	R

Binary Output-BACnetObjectType = 4

Properties Supported	Property Data	Type Identifier	Read/Write
Object_Identifier	BACnetObjectIdentifier	75	R
Object_Name	CharacterString	77	R
Object_Type	BACnetObjectType	79	R
Present_Value	BACnetBinaryPV	85	R/W
Status_Flags	BACnetStatusFlags	111	R
Event_State	BACnetEventState	36	R
Out_Of_Service	BOOLEAN	81	R
Polarity	BACnetPolarity	84	R
Priority Array	BACnetPriorityArray	87	R
Relinquish Default	REAL	104	R

Device-BACnetObjectType = 8

Properties Supported	**Property Data**	**Type Identifier**	**Read/Write**
Object_Identifier	BACnetObjectIdentifier	75	R
Object_Name	CharacterString	77	R
Object_Type	BACnetObjectType	79	R
System_Status	BACnetDeviceStatus	112	R
Vendor_Name	CharacterString	121	R
Vendor_Identifier	Unsigned16	120	R
Model_Name	CharacterString	70	R
Firmware_Revision	CharacterString	44	R
Application_Software_Version	Version CharacterString	12	R
Protocol_Version	Version Unsigned	98	R
Protocol_Revision	Revision Unsigned	139	R
Protocol_Services_Supported	BACnetServicesSupported	97	R
Protocol_Object_Type_Supported	BACnetObjectTypesSupported	96	R
Object_List	Sequence of BACnetObjectIdentifier	76	R
Max_APDU_Length_Supported	Unsigned	62	R
Segmentation_Supported	BACnetSegmentation	107	R
Local_Time	Time	57	R
Local_Date	Date	56	R
UTC_Offset	Integer	119	R/W
APDU_Timeout	Unsigned	11	R
Number_Of_ADPU_Retries	Unsigned	73	R
Max_Master	Unsigned	64	R/W
Max_Info_Frames	Unsigned	63	R/W
Device_Address_Binding	Sequence of BACnetAddressBinding	30	R
Database_Revision	Unsigned	155	R

The following table lists BACnet object types and properties supported by the BACnet Communications Module.

Supported Object Types Properties

Property	Device	Binary Input	Binary Output	Analog Input	Analog Output
Object Identifier	√	√	√	√	√
Object Name	√	√	√	√	√
Object Type	√	√	√	√	√
System Status	√				
Vendor Name	√				
Vendor Identifier	√				
Model Name	√				
Firmware Revision	√				
Appl Software revision	√				
Protocol Version	√				
Protocol Revision	√				
Services Supported	√				
Object Types Supported	√				
Object List	√				
Max APDU Length	√				
Segmentation Support	√				
Local Time	√				
Local Date	√				
UTC Offset	√				
APDU Timeout	√				
Number APDU Retries	√				
Max Master	√				
Max Info Frames	√				
Device Address Binding	√				
Database Revision	√				
Present Value		√	√	√	√
Status Flags		√	√	√	√
Event State		√	√	√	√
Out-of-Service		√	√	√	√
Units				√	√

Property	Device	Binary Input	Binary Output	Analog Input	Analog Output
Priority Array			√		√
Relinquish Default			√		√
Polarity		√	√		

Section 4.2 Parameter Offset for Module Block

Register	Address	Comments	Default
MOD_ID_LO	0×00	Module ID	196
MOD_ID_HI	0×01		0
VENDOR_ID_LO	0×02	Vendor ID	58 (0×3a)
VENDOR_ID_HI	0×03		0
PRODUCT_TYPE_LO	0×04		12 (0×C)
PRODUCT_TYPE_HI	0×05		0
PRODUCT_CODE_LO	0×06		80 (0× 50)
PRODUCT_CODE_HI	0×07		0
MOD_REV_LO	0×08	Minor revision, 1-255	3
MOD_REV_HI	0×09	Major revision, 1-127	1

Section 4.3 User Interface Options

The configuration software uses a web browser running on a personal computer. You access the configuration software via the browser to set up the system. See Chapter 3, Configuring the Module using Software.

Section 4.4 Technical Assistance

Note that your module contains electrostatic components that are susceptible to damage from electrostatic discharge (ESD). An electrostatic charge can accumulate on the surface of ordinary wrapping or cushioning material. **In the unlikely event that the module should need to be returned to Spectrum Controls, Inc., please ensure that the unit is enclosed in approved ESD packaging (such as static-shielding/metallized bag or black conductive container).** Spectrum Controls, Inc. reserves the right to void the warranty on any unit that is improperly packaged for shipment.

For further information or assistance, please contact your local distributor, or call the technical support number provided under the Technical Support section in the Preface.

Appendix A Configuration Information

This appendix contains configuration information as follows:

Environmental Specifications

Environmental Tests	Industry Standards	Test Level Limits
Temperature (Operating) (Performance Criteria A)	IEC60068-2-1: (Test Ad, Operating Cold), IEC60068-2-2: (Test Bd, Operating Dry Heat), IEC60068-2-14: (Test Nb, Operating Thermal Shock)	-20 °C to 65 °C (-4 °F to 149 °F)
Temperature (Non-operating) (Performance Criteria B)	IEC60068-2-1: (Test Ab, Unpackaged Non-operating Cold), IEC60068-2-2: (Test Bb, Unpackaged Non-operating Dry Heat), IEC60068-2-14: (Test Na, Unpackaged Non-operating Thermal Shock)	-40 °C to 85 °C (-40 °F to 185 °F)
Humidity (Operating) (Performance Criteria A)	IEC60068-2-30: (Test Db, Unpackaged Damp Heat):	5 to 95% non-condensing
Vibration (Operating) (Performance Criteria A)	IEC60068-2-6: (Test Fc, Operating)	5 G at 10 to 500 Hz, 0.030 in. max. peak-to-peak
Shock (Operating) (Performance Criteria A)	IEC60068-2-27: (Test Ea, Unpackaged Shock)	30 g, 11 ms half-sine (3 mutually perpendicular axes)
Shock (Non-operating) (Performance Criteria B)	IEC60068-2-27: (Test Ea, Unpackaged Shock)	50 g, 11 ms half-sine (3 mutually perpendicular axes)
Radiated Emissions	CSIPR 11; Group 1, Class A	(Enclosure) Class A, 30 MHz–1 GHz
Conducted Emissions	IEC 61000-6-4:2007	Group 1, Class A (AC Mains), 15 0 kHz–30 MHz

Environmental Tests	Industry Standards	Test Level Limits
ESD immunity (Performance Criteria B)	IEC 61000-4-2	6 kV Indirect (Coupling Plate) 6 kV Contact Discharge (to points of initial contact) 8 kV Air Discharge (to points of initial contact)
Radiated RF immunity (Performance Criteria A)	IEC 61000-4-3: Level 3	10 V/M with 1 kHz sine-wave 80% AM from 80…2000 MHz 10 V/M with 200 Hz sine-wave 50% Pulse 100% AM at 900 MHz 10 V/M with 200 Hz sine-wave 50% Pulse 100% AM at 1890 MHz 1 V/M with 1 kHz sine-wave 80% AM from 2000…2700 MHz (3 V/M goal)
EFT/B immunity (Performance Criteria B)	IEC 61000-4-4*	Signal Ports: ±3 kV at 5 kHz for 5 minutes, Criteria B (Marine) ±2 kV at 5 kHz for 5 minutes, Criteria A (Marine) ±2 kV at 5 kHz for 5 minutes, Criteria B (standard) Power Ports: ±2 kV at 5 kHz for 5 minutes, Criteria A (Marine) ±2 kV at 5 kHz for 5 minutes, Criteria B (standard)
Surge transient immunity (Performance Criteria B)	IEC 61000-4-5	Signal Ports: ±2 kV line-earth {CM}at 2Ω on shielded ports Power Ports ±2 kV CM at 12Ω ±1 kV DM at 2Ω
Conducted RF immunity (Performance Criteria A)	IEC 61000-4-6	10 V rms with 1 kHz sine wave 80% AM from 150 kHz…80 MHz on signal and power ports
Magnetic Field (Performance Criteria A)	IEC 61000-4-8	30 Arms/m

Environmental Tests	Industry Standards	Test Level Limits
AC Mains Voltage Dips, Interruptions and Variations	IEC 61000-4-11	Follow the 61000-4-11.

Safety Tests and Test Limits

Safety Tests	Industry Standards
cUL	UL 508 Industrial Control Equipment Seventeenth Edition Dated January 28 1999, with revisions through March 15, 2013 (ANSI/UL 508-2005) (NRAQ, NRAQ7) cUL CSA C22.2 No. 142 -M1987 Process Control Equipment May 1987 ULH ANSI/ISA–12.12.01–2007 Non-incendive Electrical Equipment for Use in Class I, Division 2 Hazardous (Classified) Locations CULH CSA C22.2 No. 213-M1987 - Non-incendive Electrical Equipment for use in Class I Division 2 Hazardous Locations - March 1987
CE EMC Directive	EN 61131-2 Programmable Controllers: Third Edition 2007-02, Clause 8, Zones A&B EN 61000-6-2: Generic Industrial Immunity EN 61000-6-4: Generic Industrial Emissions
UKCA EMC Directive	Electromagnetic Compatibility Regulations 2016 BS EN 61131-2, BS EN 61000-6-4, BS EN 61000-6-2
CMIM	Arrêté ministériel n° 6404-15 du 29 ramadan 1436 (16 juillet 2015) NM EN 61131-2, NM EN 61000-6-4, NM EN 61000-6-2, NM EN 61010-2-201

Performance Requirements

Input Specifications	
Inputs per module	2 serial channels, non-concurrent operation
Interface, channel 1	RS-485, configurable during setup
Hardware flow control	None
Baud rates	1200, 2400, 4800, 9600, 19.2 k, 38.4 k, 76.8 k
Interface, channel 2	10/100M Ethernet, auto sensing
Crosstalk	-40 dB, minimum
Input protection	24 VDC continuous.
Power source	3.3 VDC and 24 VDC from backplane, 40 mA from 3.3 VDC and 50 mA maximum from 24 VDC
Power, RTC Backup	72 hours, minimum
Accuracy, RTC Backup	+1.0, - 3.0 minutes/month.

Input Specifications	
Power consumption	<40 mA at 3.3 V, <50 mA at 24 V, <1.5 W Total
Inrush current	<500 mA at 3.3 V, <500 mA at 24 V
Fusing	2.7 Ω 1/10 W resistor, 24 VDC input 0.47 Ω 1/10 W resistor, 3.3 VDC input
Input to backplane isolation	707 VDC for 1 minute
Channel-to-channel isolation	None
Fault detection	None
Wire size	#22 to #30 AWG (for the mating connector)
Operating temperature	-20 ºC to 65 ºC
Storage temperature	-40 ºC to 85 ºC
Operating humidity	5% to 95% (non-condensing)
Manufacturing	RoHS & REACH compliant
Dimensions	58.4 mm × 29.3 mm × 25 mm

Reliability

The Mean Time between Failure (MTBF) target for the 2080sc-BAC is 250,000 hours.

Index

2080sc-BACnet Communication Module
  config uring 3-1
  DC power specifications 1-8
  enclosure requirements 1-3
  environment requirements 1-3
  hardware features 1-7
  installing 2-1
  Parts List 1-6
Add Tags
  icon 3-12
Analog Input 3-19
Analog Input-BACnetObjectType 4-1
Analog Output 3-22
Analog Output-BACnetObjectType 4-1
Attribute
  field 3-12
BAC_Slot1 function 3-17
BACnet
  implementing protocol 4-1
  Object Types 4-1
BACnet Analog Input 1 Block 3-30
BACnet Analog Input 2 Block 3-30
BACnet Analog Output 1 Block 3-31
BACnet Analog Output 2 Block 3-31
BACnet Binary Input Block 3-30
BACnet Binary Output Block 3-30
BACnet Diagnostic Tools 3-32
BACnet WhoIs Service 3-32
BACnet/IP
  field 3-8
BACnet_Convert Function Block 3-23
BACnetObjectType = 3 4-2
Baudrate
  field 3-9
Binary Input 3-20
Binary Output 3-23
Binary Output-BACnetObjectType 4-2
CCW Export and Import of Function Blocks 3-28
Changes
  changes 3-saving 3-36
Configuring
  system 3-4
Connector
  Ethernet 1-7
Conventions
  used in the manual, vi
Data Type
  field 3-12
Data Type Range 3-31
Date
  field 3-6
Date/Time
  section 3-6
DC power specifications
  2080sc-BACnet Communication Module 1-8
Determined by Network
  field 3-6
Device
  section 3-7
Device Instance
  field 3-7
Device Name
  field 3-7
Device-BACnetObjectType 4-3
Edit XML
  button 3-14
  option 3-13
Electrostatic discharge
  prevention guidelines 1-5
Ethernet
  Connector 1-7
  LED indicator 1-8
  pinouts 1-7
Gateway
  field 3-8
General description
  1-1
Generate Structured Text
  option 3-13
Ground
  earth ground not present 1-8
  isolated 1-8
Hardware Features
  2080sc-BACnet Communication Module 1-7
How to use this manual, v
Important Notes
  About the guide, ii
Indicator
  Ethernet 1-8
Installing
  2080sc-BACnet, 1
Instance
  device 3-7
Interface

field 3-7
IP address
169.254.3.3, 3-1
IP Address
field 3-8
IP Address and User Interface Password Recovery 3-28
LAN Type
section 3-7
LED
indicator 1-8
Limited Warranty, ii
Log
viewing information 3-35
MAC Address
field 3-9
Max Masters
field 3-9
Max Number of Frames
field 3-9
Memory Mapping 3-29
MS/TP
field 3-8
MS/TP Wiring Best Practices 1-2
Notice
user requirement, ii
Object ID
field 3-13
Object Name
field 3-12
Object Type
field 3-13
Parameter
module block offset 4-5
Parts List
2080sc-BACnet Communication Module 1-6
Password
entering 3-3
login 3-5
Reset/DCC 3-6
section 3-5
spectrum 3-3
Pinouts
Connector 1-7
Preface, v
Present Value
field 3-13
Properties
supported object types 4-4
Protocol
implementing BACnet 4-1
ReadProperty 3-33
Reload
button 3-10
Reload Tags
option 3-13
Save
button 3-10
Save Tags
option 3-13
Saving
changes 3-36
Serial I/O RJ-45 Connector 1-7
Setting up
tags 3-10
Setup 3-32
Software
upgrading 3-36
Software Upgrade 1-8
Structured text
example 3-16
generating 3-15
Subnet Mask
field 3-8
System
configuring 3-4
System Configuration
reloading 3-36
tab 3-4
Tag
ediiting XML information 3-13
Tag Name
field 3-12
Tag Setup
button 3-14
tab 3-4
Tags
setting up 3-10
Technical Assistance 4-6
Technical support
contact information, v
Text
generating structured 3-15
Time
field 3-6
UDP Port
field 3-8
Upgrading
software 3-36
User Interface Options 4-5
Version
viewing 3-35
Viewing
version 3-35
Who should use this manual, v
WriteProperty 3-33

SILVER
Technology Partner
A ROCKWELL AUTOMATION PARTNER

Corporate Headquarters
Spectrum Controls Inc.
P.O. Box 6489
Bellevue, WA 98006 USA
Fax: 425-641-9473
Tel: 425-746-9481

Web Site: www.spectrumcontrols.com
E-mail: spectrum@spectrumcontrols.com

SPECTRUM
CONTROLS
An Allient Company