User Guide

Babel Buster 3 BACnet-Modbus Network Gateway with Proprietary Protocol Support

Model BB3-7101-SP
Model MX-71-SP
Rev. 1.0 – June 2021

BB3-7101-SP/MX-71-SP User Guide Contents

1. Introduction

1.1 How to Use This Guide

This user guide provides background information on how the gateway works, and an overview of the configuration process. There are several sections for groups of tabs found in the web interface in the gateway which is accessed by opening a web browser and browsing to the IP address of the device.

You should at least read Sections 2 and 3, and other sections specific to your intended use. There is a "Quick Help" section at the bottom of each web page in the gateway which is generally sufficient for quick reference in setting up the gateway.

1.2 Important Safety Notice

Proper system design is required for reliable and safe operation of distributed control systems incorporating any Control Solutions product. It is extremely important for the user and system designer to consider the effects of loss of power, loss of communications, and failure of components in the design of any monitoring or control application. This is especially important where the potential for property damage, personal injury, or loss of life may exist. By using ANY Control Solutions, Inc., product, the user has agreed to assume all risk and responsibility for proper system design as well as any consequence for improper system design.

1.3 Warranty

This documentation is provided “as is,” without warranty of any kind, either expressed or implied, including, but not limited to, the implied warranties of fitness or merchantability for a particular purpose. Control Solutions may make improvements and/or changes in this documentation at any time. This documentation could include technical inaccuracies, typographical errors, and the like. Changes are periodically made to the information herein; these changes may be made without notice.

Product Warranty: All Control Solutions products are warranted against defects in materials and workmanship for a period of time from date of shipment from factory as follows: Two years on non-mechanical parts, one year on mechanical parts (e.g. relays). Defective units will be repaired or replaced, at manufacturer’s discretion, at no cost to user except when negligence or improper use has resulted in damage. The express warranty stated herein is in lieu of all other warranties, express or implied, including without limitation any warranties of merchantability or fitness for a particular purpose and all other warranties are hereby disclaimed and excluded by Control Solutions, Inc.

Configuration errors made by customer are not covered under warranty. Damage caused by incorrect electrical connection is not covered under warranty. Removing circuit boards from their enclosures will void the warranty - the complete product with all of its original circuit boards and components must be returned for warranty consideration.

2. Connecting Gateway for the First Time

2.1 Where to Start

The Babel Buster BB3-7101-SP or MX-71-SP is used to share data between devices with proprietary serial or IP protocols, BACnet IP devices and Modbus devices. If you are looking for a connection to some other protocol, then start by looking for a different model number.

Start by getting familiar with this User Guide and the sections that pertain to your application. Be sure to review the remainder of this section. Online videos are also available to demonstrate key operations in setting up the BB3-7101-SP or MX-71-SP.

You may need to obtain information such as a "register map" from the vendor of the Modbus device you wish to connect. The register map may go by any of several names in the manufacturer's documentation. You will need to know what registers to read in order to interface a Modbus device to the BACnet network.

If you are going the other direction - interfacing a BACnet device to the Modbus network - then you will need to obtain an object list for the BACnet device you are going to be polling. The BB3-7101 does not support BACnet auto-discovery, so you will need to obtain an object list from the manufacturer of the device.

If you get stuck, you can open a support ticket at https://ticket.csimn.com where response time is generally 24 hours or less, and often as little as 2 hours, and at no cost.

NOTE: Screen shots throughout this User Guide illustrate BB3-7101; however, the screens in the MX-71 are identical with the only exception being model number indicated at the top of the page.

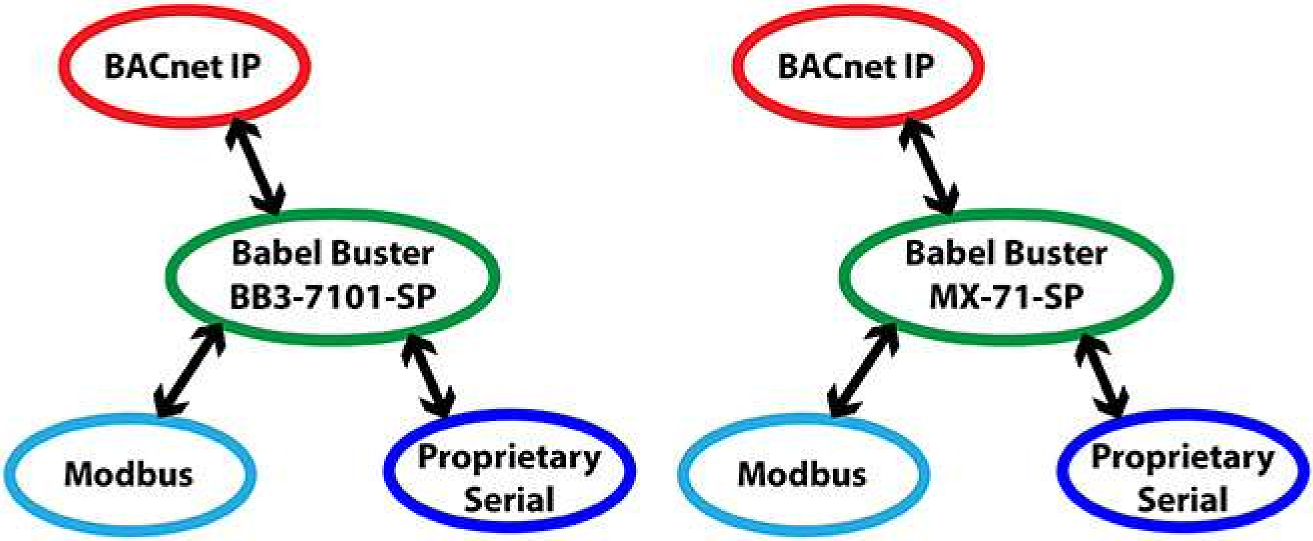
2.2 Overview of Model BB3-7101/MX-71

2.2.1 Application of the BB3-7101-SP/MX-71-SP

The Babel Buster BB3-7101/MX-71 is a BACnet gateway used primarily to interface between BACnet devices and Modbus devices. The gateway may be used as BACnet IP client and server, Modbus TCP client and server, and Modbus RTU master or slave. The -SP variant of this gateway adds the ability to interface devices having properietary serial protocols (in lieu of Modbus RTU) or proprietary protocols transported via TCP or UDP.

The BB3-7101/MX-71 may be thought of as a data server with multiple network ports that have access to the internal database. The BB3-7101/MX-71 is not a router - BACnet devices cannot interact directly with Modbus devices or vice versa. Both BACnet and Modbus have access to the internal data objects which are updated according to rules you define. Proprietary protocols are implemented with Script Basic programming that you provide, and your Basic program also has access to these internal data objects.

From a strictly BACnet perspective, the BB3-7101/MX-71 should be thought of as a collection of sensors and actuators. The means by which the sensors and actuators interface to hardware is what BACnet protocol would refer to as a "local matter". In the case of the BB3-7101/MX-71, that hardware consists of one or more Modbus devices or proprietary protocol devices which the BB3-7101/MX-71 is set up to automatically interact with, without any constraints on the BACnet side.



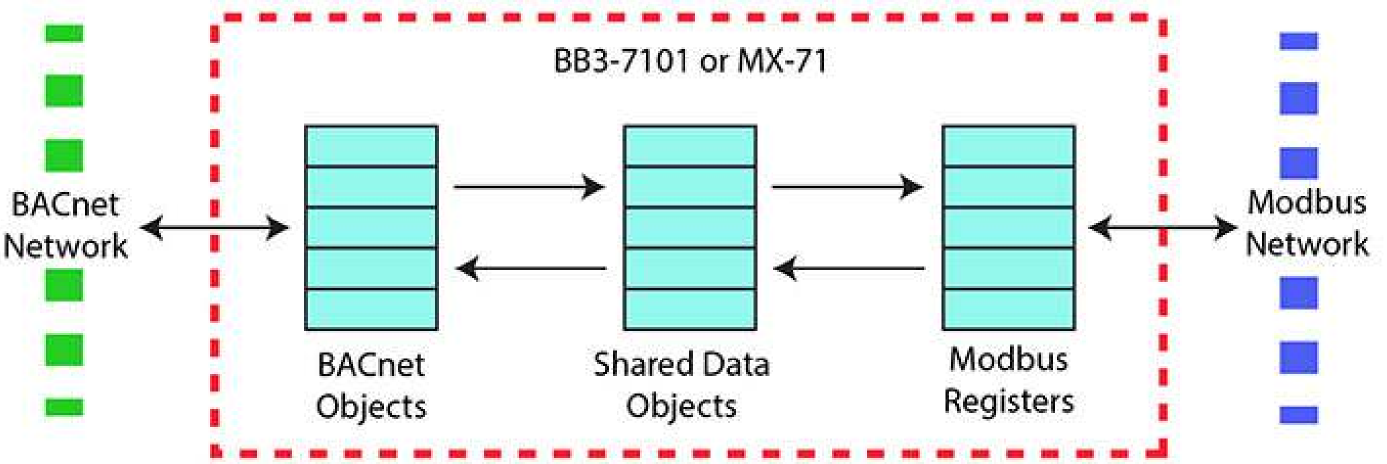
The most common application for the BB3-7101-SP/MX-71-SP is interfacing a serial communication device to a BACnet IP network. Your Script Basic program will automatically interact with the serial device, and store the data received in BACnet objects you assign. The BACnet system may then use standard BACnet services such as Read Property to access the content of the data objects. The BB3-7101 will also accept COV subscriptions such that other devices will receive a COV notification when the content of a local object changes.

The BB3-7101 can be configured as a BACnet IP client. This means the BB3-7101 will be reading and writing properties in other BACnet devices, storing copies of their object's Present Value in the BB3-7101. The stored values may later be accessed by Modbus or by your Script Basic program for sharing data with your proprietary device.

2.2.2 Object Server Model for a Gateway

Control Solutions gateways are not simple protocol translators. It is not possible to do an effective job of simply converting one protocol directly to another. Any attempt to do so would likely have negative effects on the networks on both sides of the gateway. An effective solution requires an intelligent device that can properly and efficiently act as a native device on each network. Control Solutions gateways function as two native devices, one on each network, with a shared data base in between them. They function as clients and/or servers on each network.

The central data element in every Control Solutions gateway is an “object”. Each object has rules for accessing that object which are specific to the protocol of the network. Each object has at least two sets of rules, one set for each of the two (or more) networks that may access the object. The object model is often optimized to cater to a specific protocol, and will most often favor the more complex protocol.



Control Solutions gateways will function as servers, providing a copy of the most recent data found in its data base when a client requests that data. In master/slave terms, the server is a slave while the client is a master. Some applications will treat the gateway as a server from both (all) networks connected. But most applications will want the gateway to be a server on one side, and a client on the other side. The most frequent application of the standard BB3-7101/MX-71 will have it functioning as a Modbus master (client), but the application of the -SP variant is really user defined.

Client functionality of a Control Solutions gateway is autonomous. In other words, when acting as a Modbus master (client), the gateway will continuously poll the Modbus slave device(s) on its own, and keep a copy of the most recent data obtained from (or sent to) the Modbus slave device(s). Most often, the gateway is configured to read slave devices periodically, and write to the slave devices when new data is received from a client.

Client functionality of the BB3-7101/MX-71 can also pertain to the BACnet side. When configured as a BACnet client, the BB3-7101/MX-71 will automatically poll other BACnet devices and retain data for subsequent access by Modbus.

2.3 What is New in Model BB3-7101/MX-71

The BB3-7101 is a significant enhancement over its predecessor, the BB2-7010. The MX-71 is the upgraded version of the SPX-B. The hardware includes a faster processor and hardware encryption engine for efficient rendering of secure web pages. The software includes numerous enhancements.

- Faster processor
- Secure (HTTPS) web server

- Higher point count, up to 5,000 BACnet objects
- User defined register map for Modbus slave
- CSV import of register maps
- Menu options to clear part or all of configuration

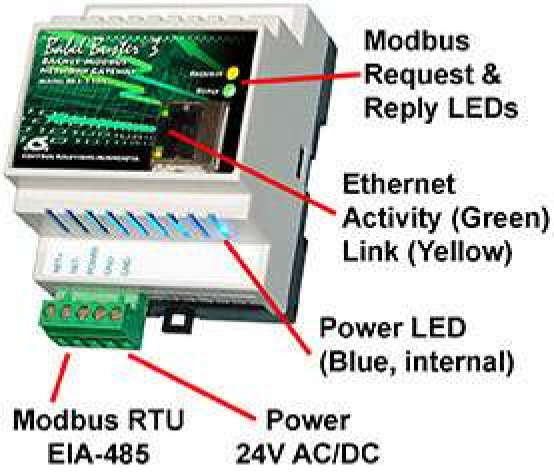
Control Solutions has benchmark tested a configuration in which the BB3-7101 was reading 5,000 registers from Modbus TCP and mappign them to 5,000 BACnet Analog Input objets. The scan rate was 7 seconds for all 5,000 points.

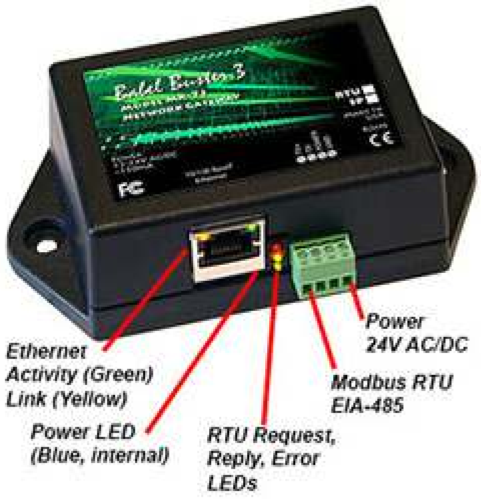
IMPORTANT: Configuration files from older gateways are not directly usable in the BB3-7101, but the Babel Buster Configuration Builder program can be used to convert a BB2-7010 configuration into a BB3-7101 configuration file. See Appendix H.

2.4 Connectors and Indicators

Follow these steps to make the initial connection to the BB3-7101 or MX-71.

(a) Connect power. Apply +12VDC to +24VDC or 24VAC to the terminal marked "POWER", and common or ground to one of the terminals marked "GND".


Modbus Request & Reply LEDs
Ethernet Activity (Green) Link (Yellow)
Power LED (Blue, internal)
Modbus RTU EIA-485
Power 24V AC/DC


Ethernet Activity (Green) Link (Yellow)
Power LED (Blue, internal)
RTU Request, Reply, Error LEDs
Modbus RTU EIA-485
Power 24V AC/DC

(b) Connect a CAT5 cable between the RJ-45 jack on the gateway, and your network switch or hub. You cannot connect directly to your PC unless you use a "crossover" cable (or your PC supports auto-MDX, which many newer laptops do).

(c) Apply power.

A blue LED inside the case should light indicating power is present.

If the link LED on the RJ45 jack is not on, check your Ethernet cable connections. Both link and activity LEDs on the RJ45 jack will be on solid for a short time during boot-up. The entire bootup process will take 1-2 minutes, during which time you will not be able to connect with a browser.

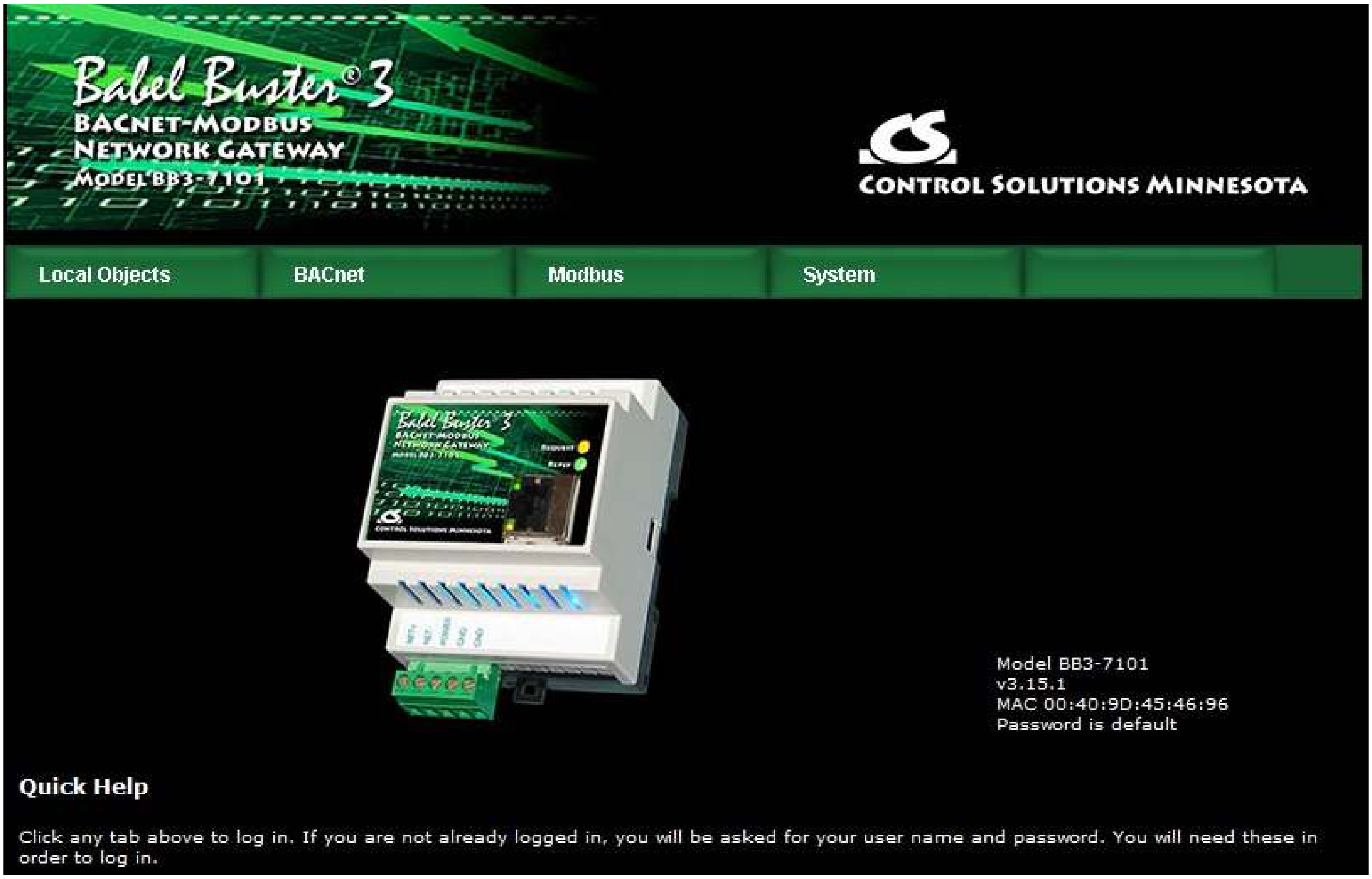
Ethernet link LED is the yellow LED integrated into the CAT5 connector. Ethernet activity LED is the green LED integrated into the CAT5 connector.

Refer to Appendix A for additional detail pertaining to connections and indicators as well as optional internal jumper settings.

2.5 Opening the Web User Interface

The default IP address as shipped is 10.0.0.101. Open your browser, and enter “http://10.0.0.101/” in the address window. You should see a page with the “Babel Buster 3” header shown below (BB3-7101 illustrated, MX-71 will be similar). From this point, you will find help on each page in the web site contained within the product.

If your PC is not already on the 10.0.0.0 domain, and you are unable to connect, you may need to temporarily change your computer’s IP address to a static IP address that starts with 10.0.0. and ends with anything but 101.



Model BB3-7101
v3.15.1
MAC 00:40:9D:45:46:96
Password is default

Quick Help

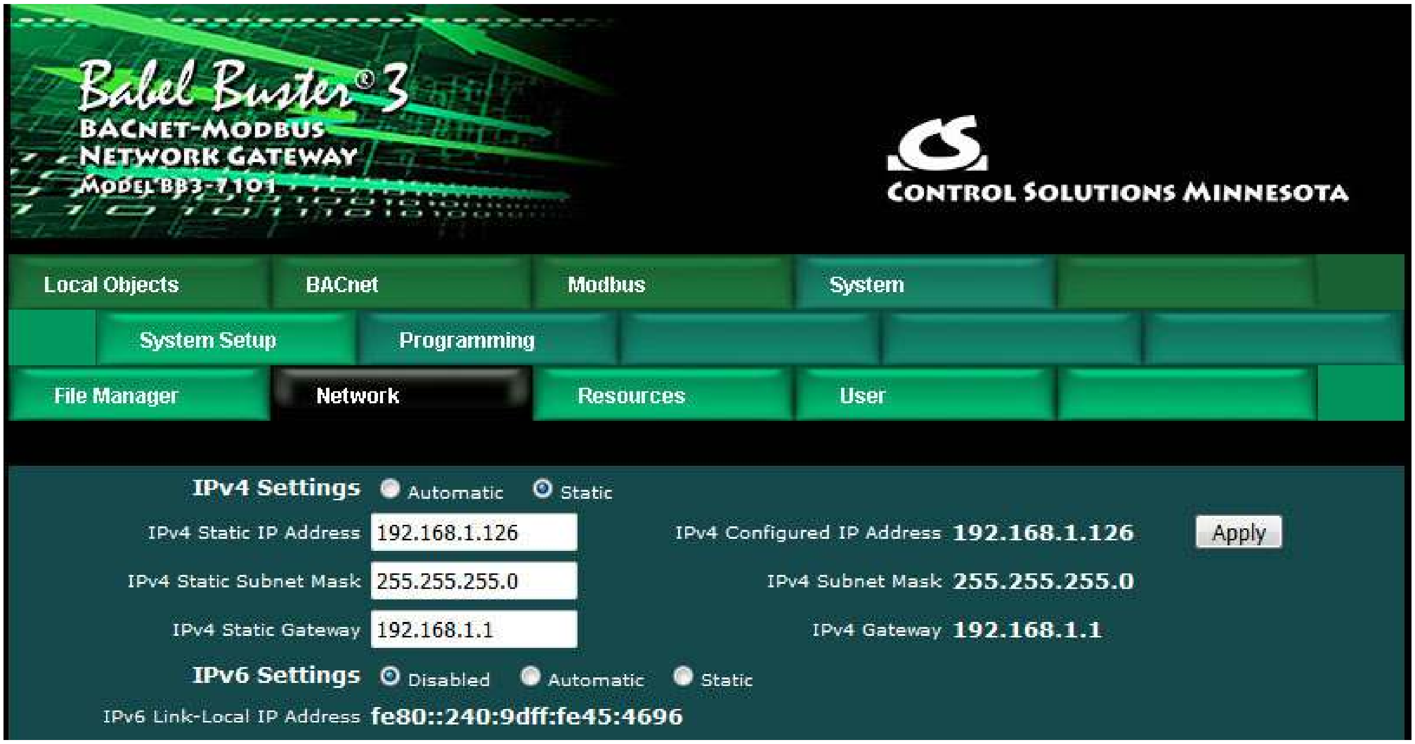
Click any tab above to log in. If you are not already logged in, you will be asked for your user name and password. You will need these in order to log in.

When you click on any of the page tabs such as System, you will be asked for a user name and password. The only login as shipped is user name “root” with a unique password generated specifically for your Babel Buster. Your password should be included on a document included with the gateway, or on a label attached to the gateway.

If the unique automatically generated password is currently in effect for user "root", it will be indicated by "Password is default" as shown in the above screen shot. If you have changed the root password to something of your own making, then this line is absent.

There is no way to get the BB3-7101/MX-71 to show you what the default root password is. If you have lost track of it, make a note of the MAC address, and open a support ticket at https://ticket.csimn.com to request the default root password (you will need to provide the MAC address in order to obtain the password).

To change the IP address of the gateway, go to the Network page under System :: System Setup. The following page should appear (only top portion illustrated here). Change the IP address, and subnet mask and gateway if applicable. Click Change IP to save the changes. The process of programming this into Flash takes around half a minute. The new IP address only takes effect following the next system restart or power cycle.



IPv4 Settings ○ Automatic ● Static

IPv4 Static IP Address 192.168.1.126 IPv4 Configured IP Address 192.168.1.126

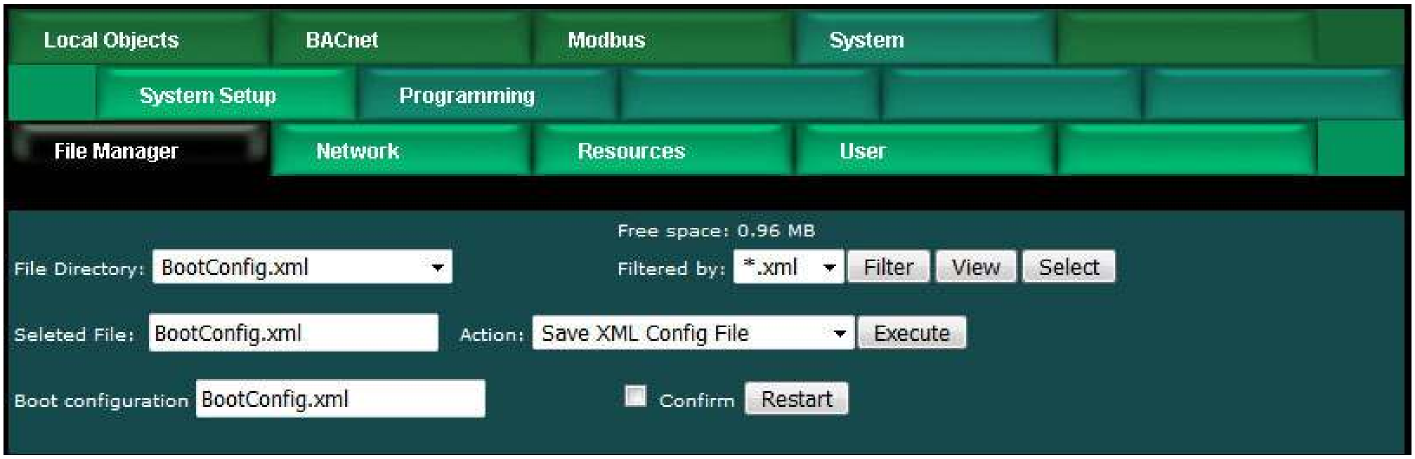
IPv4 Static Subnet Mask 255.255.255.0 IPv4 Subnet Mask 255.255.255.0

IPv4 Static Gateway 192.168.1.1 IPv4 Gateway 192.168.1.1

IPv6 Settings ● Disabled ○ Automatic ○ Static

IPv6 Link-Local IP Address fe80::240:9dff:fe45:4696

Most changes are stored in an XML configuration file in the device's Flash file system. Only a few are stored differently, and the IP address is one of those. Normally, clicking Update on any configuration page only stores that configuration information to a temporary RAM copy of the configuration file. To make your changes other than IP address permanent, you must select your file, select the Save XML Config File action, and then click Execute on the File Manager page. Refer to Section 11 for more about the File Manager.



Free space: 0.96 MB

File Directory: BootConfig.xml Filtered by: *.xml

Seleted File: BootConfig.xml Action: Save XML Config File

Boot configuration BootConfig.xml ☐ Confirm

Babel Buster® 3
MODEL BB3-7101-SP
MODEL MX-71-SP
NETWORK GATEWAY

CONTROL SOLUTIONS MINNESOTA

3. Configuring Local Objects

Babel Buster gateways do not come with a predefined set of BACnet objects. The gateway will initially have a handful of objects, but it is up to the user to allocate the number needed, up to the maximum permitted by available resources.

3.1 Behavior of Input vs Output Objects

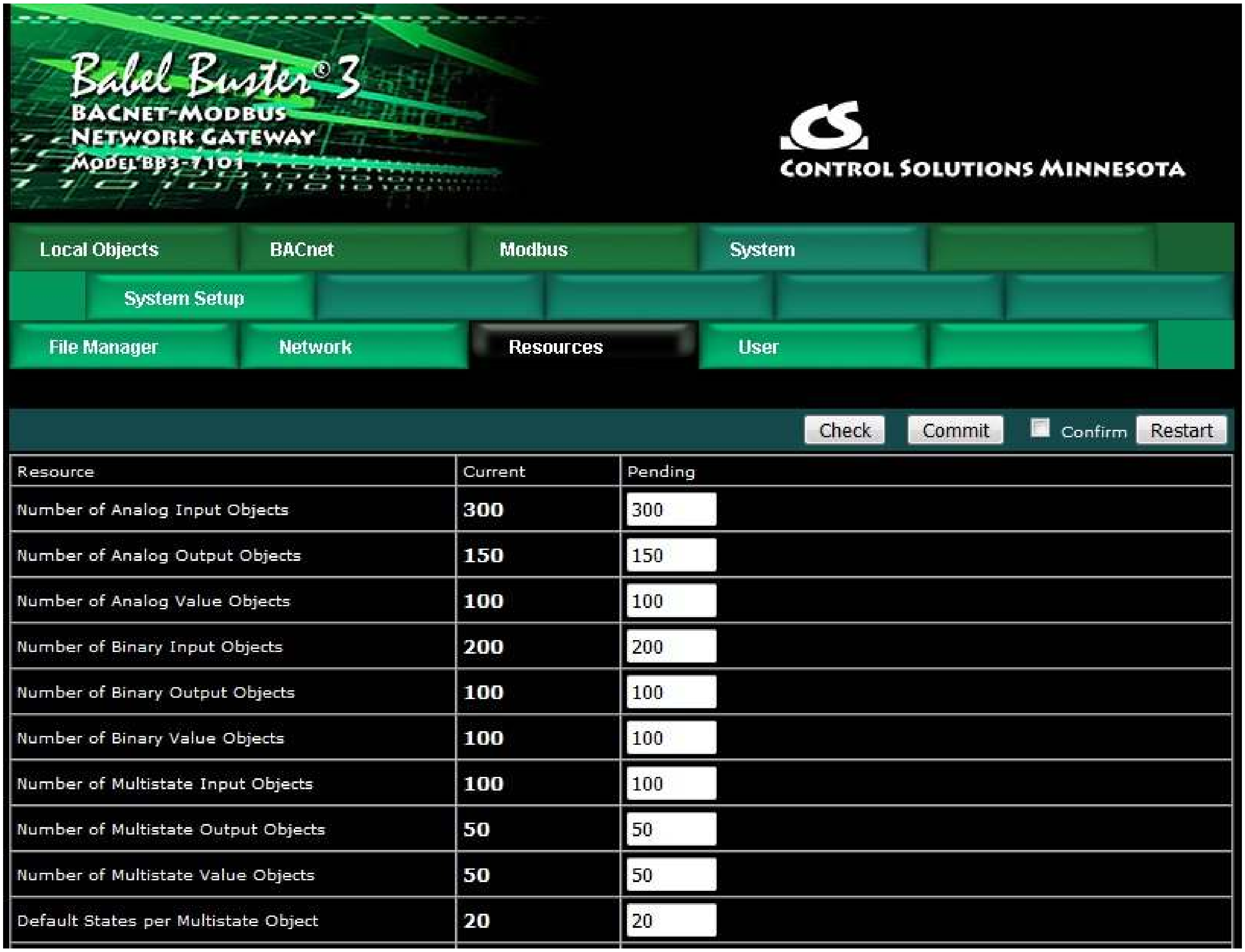
The easiest way to keep track of input versus output is to think about a BACnet device's role in the system. The system will receive input from the BACnet device, and provide output to the BACnet device. Inside the BACnet device, hardware will physically associate BACnet Input Objects with sensor inputs such as temperature or pressure sensors, etc. The system then receives the sensor input information via BACnet Input Objects. When the system wants to control an actuator, it will send setpoints to the actuator via BACnet Output Objects. Hardware inside the BACnet device will physically associate the Output Object with a physical actuator such as valve position servo or motor speed controller.

Keeping track of input versus output in a gateway can be a bit trickier; however, the choice of input versus output does not change from the BACnet perspective. Only the nature of the physical sensor and actuator hardware changes. In the case of the BB3-7101/MX-71, sensors and actuators both consist of Modbus devices (from BACnet's perspective). Therefore, use a BACnet Output Object to send data to a Modbus device, and use a BACnet Input Object to receive data from a Modbus device.

We have not mentioned BACnet Value Objects yet just to avoid confusing the discussion. A Value Object can be input or output, or both at the same time. If you are familiar with Modbus, the BACnet Value Object is most synonymous with the holding register that you can both read and write. When using a Value Object, it is best to think about its role as input or output when deciding how to apply maps or rules in the gateway.

3.2 Allocating Local Objects

The resource allocation page is where you set the number of each type of available BACnet object that you will use. It is a good idea to determine ahead of time how many objects you will need, then allocate that number, possibly including a spare object or two. It is not a good idea to allocate a large number of objects that will remain unused since this simply clutters the screen when a front end system auto-discovers all objects in the device.



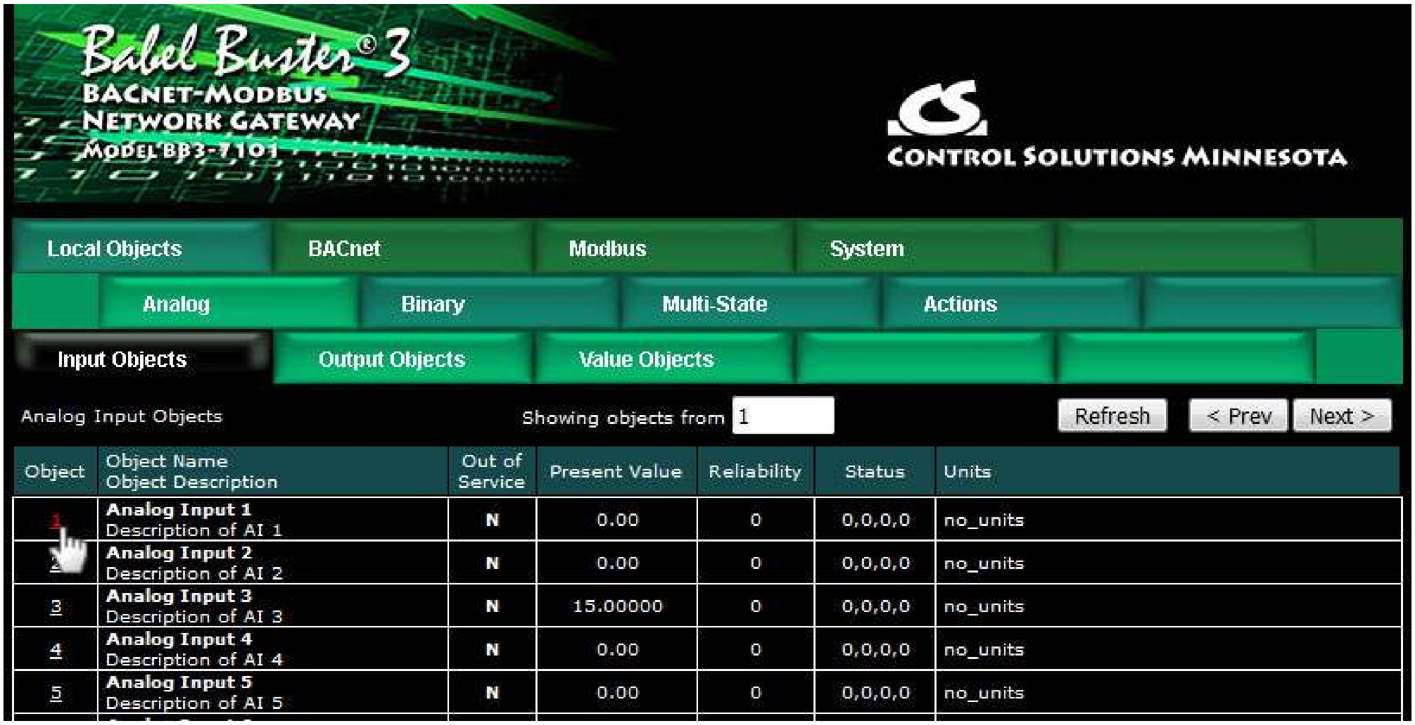
Resource	Current	Pending
Number of Analog Input Objects	300	300
Number of Analog Output Objects	150	150
Number of Analog Value Objects	100	100
Number of Binary Input Objects	200	200
Number of Binary Output Objects	100	100
Number of Binary Value Objects	100	100
Number of Multistate Input Objects	100	100
Number of Multistate Output Objects	50	50
Number of Multistate Value Objects	50	50
Default States per Multistate Object	20	20

The portion of the Resources page dedicated to BACnet object counts is shown above. For a complete discussion of the Resources page, including how to change the counts, refer to Section 11.4 in this User Guide.

3.3 Configuring Local Objects

There is a different web page for each BACnet object type in the device. Objects are listed in tabular form with name and description, present value, reliability code and status. Additional information as applicable to the object type may also be listed.

Click on the object number in the first column to open the expanded view of that object and gain access to its configuration.



Analog Input Objects

Showing objects from 1

Object	Object Name Object Description	Out of Service	Present Value	Reliability	Status	Units
1	**Analog Input 1** Description of AI 1	N	0.00	0	0,0,0,0	no_units
2	**Analog Input 2** Description of AI 2	N	0.00	0	0,0,0,0	no_units
3	**Analog Input 3** Description of AI 3	N	15.00000	0	0,0,0,0	no_units
4	**Analog Input 4** Description of AI 4	N	0.00	0	0,0,0,0	no_units
5	**Analog Input 5** Description of AI 5	N	0.00	0	0,0,0,0	no_units

Reliability codes may be any of the following:

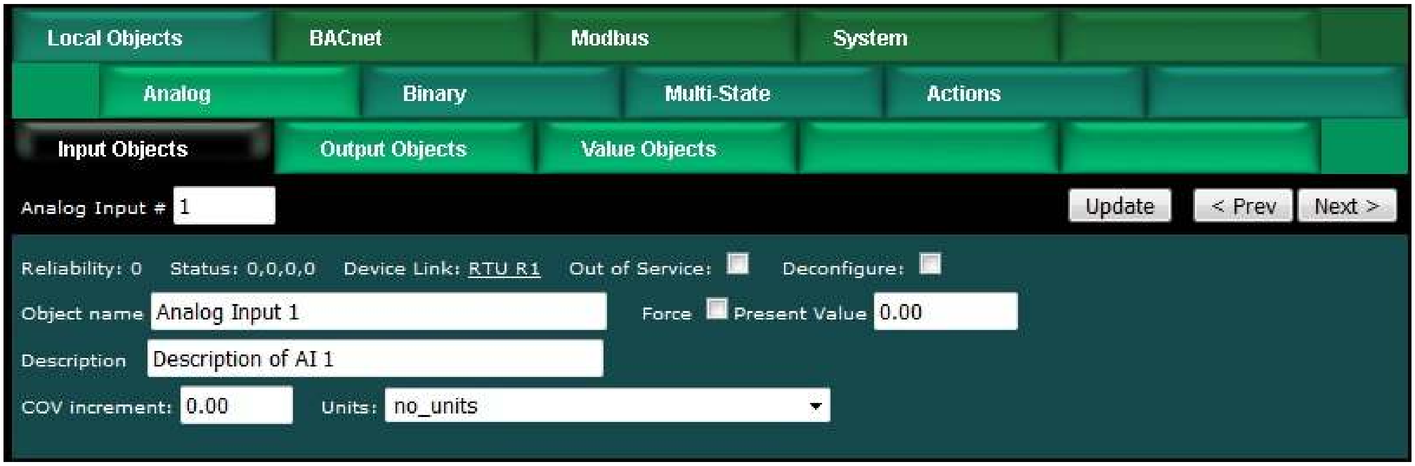
Modbus client/master, no response from slave (64)
Modbus client/master, crc error (65)
Modbus exception, illegal function code (66)
Modbus exception, illegal data address (67)
Modbus exception, illegal data value (68)
Modbus exception, code+65, rarely used (69..79)
Local device, configuration property fault (80)
Faulty Modbus packet(81)
BACnet IP client, device timeout (82)
BACnet IP client, error returned by server (83)

Status flags A,B,C,D indicate the following, 0 meaning not true, 1 meaning true:

A = in alarm
B = fault
C = overridden
D = out of service

3.3.1 Analog Input Objects

The source of data for an Analog Input object will typically be reading from some other BACnet or Modbus device.



Analog Input # 1

Reliability: 0 Status: 0,0,0,0 Device Link: RTU R1 Out of Service: □ Deconfigure: □

Object name Analog Input 1 Force □ Present Value 0.00

Description Description of AI 1

COV increment: 0.00 Units: no_units

The object name, units, value, and status are shown for the object number entered at the top of the page. Click Prev/Next to scroll through the list. Click Refresh to update the page, or Update to accept changes.

The object name and description may be changed here. BACnet units may be selected. Initial COV increment may be entered. When any of these are changed, be sure to save the updated configuration by executing "Save XML Config File" on the File Manager page.

The object may be set Out of Service by checking that box and clicking Update. The present value may be changed by entering a value, checking Force, and clicking Update.

The source of data for an Analog Input object will typically be reading from some other BACnet or Modbus device via the map indicated by the Device Link. The mapped device will be polled at the rate specified by the Read Map.

Out of Service means polling of the mapped remote device will stop. While out of service, the present value may be written by the BACnet client. Data may be forced via this web page at any time, but will be overwritten by the next mapped client update unless the object is out of service.

Reliability codes indicate errors as itemized on the tabular object list.

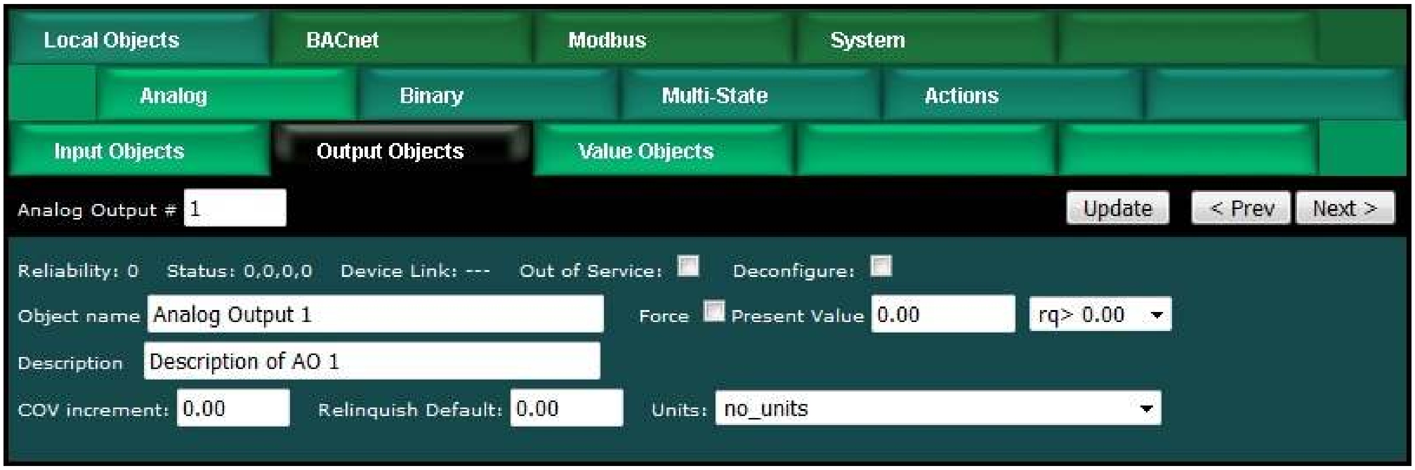
Status flags A,B,C,D indicate the following, 0 meaning not true, 1 meaning true:
A = in alarm
B = fault
C = overridden
D = out of service

Device link will indicate BIP, RTU, or TCP, followed by R for read or W for write, and a number which is the map number in the table of read or write maps for mapping to this BACnet object. The designation R means read from a remote device, and W means write to a remote device.

Check 'Deconfigure' and click Update to erase configuration for this object.

3.3.2 Analog Output Objects

The destination of data for an Analog Output object will typically be some other BACnet or Modbus device.



Analog Output # 1

Reliability: 0 Status: 0,0,0,0 Device Link: --- Out of Service: □ Deconfigure: □

Object name Analog Output 1 Force □ Present Value 0.00 rq> 0.00

Description Description of AO 1

COV increment: 0.00 Relinquish Default: 0.00 Units: no_units

The object name, units, value, and status are shown for the object number entered at the top of the page. Click Prev/Next to scroll through the list. Click Refresh to update the page, or Update to accept changes.

The object name and description may be changed here. BACnet units may be selected. Initial COV increment may be entered. When any of these are changed, be sure to save the updated configuration by executing "Save XML Config File" on the File Manager page.

The destination of data for an Analog Output object will be writing the remote BACnet or Modbus device via the map indicated by the Device Link. The remote device will be updated upon change of source data and/or periodically as defined by the Write Map.

The Analog Output object is commandable, meaning the BACnet client must write both a value and a priority level for that value. The highest level value will be the one written to the remote device (if one is mapped). If all values are relinquished, the relinquish default value will be written to the remote device.

To set an output object manually from this page, check the Force box, enter a value in the Present Value window, and select a priority level to assign to your forced value. Then click Update. To return a given priority level to NULL, simply type the word NULL in the Present Value window, check Force, and click Update.

Out of service means the mapped remote device will not be written to. Values written by the BACnet client will be retained, but only applied when this object is placed back in service. At that time, the highest priority value will be written to the remote device.

Reliability codes indicate errors as itemized on the tabular object list.

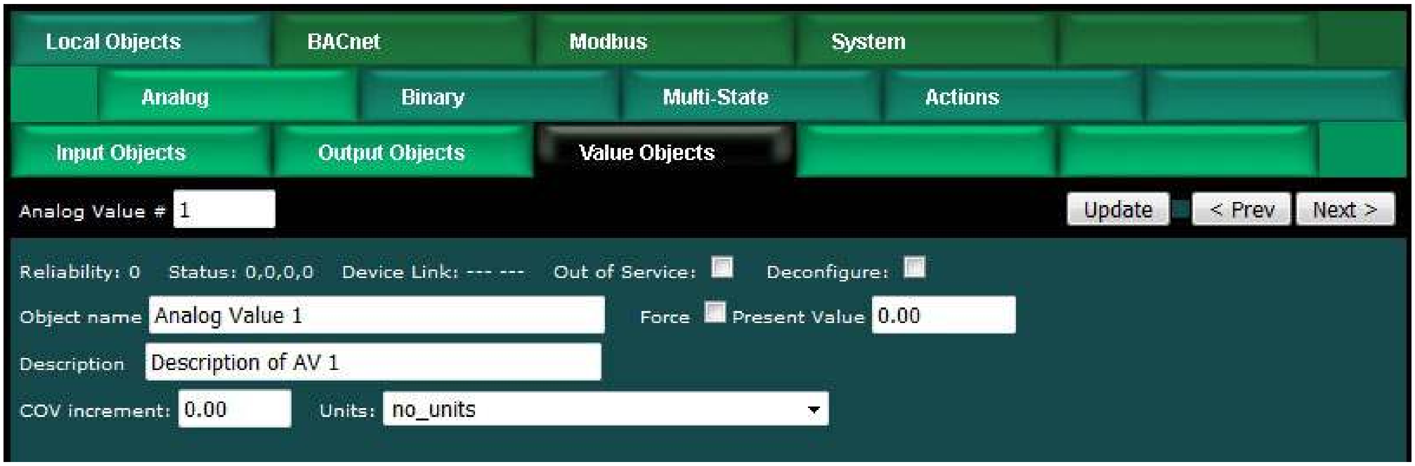
Status flags A,B,C,D indicate the following, 0 meaning not true, 1 meaning true:
A = in alarm
B = fault
C = overridden
D = out of service

Device link will indicate BIP, RTU, or TCP, followed by R for read or W for write, and a number which is the map number in the table of read or write maps for mapping to this BACnet object. The designation R means read from a remote device, and W means write to a remote device.

Check 'Deconfigure' and click Update to erase configuration for this object.

3.3.3 Analog Value Objects

Analog Value objects may be both a source and destination for some other BACnet or Modbus device.



Analog Value # 1

Reliability: 0 Status: 0,0,0,0 Device Link: --- --- Out of Service: □ Deconfigure: □

Object name Analog Value 1 Force □ Present Value 0.00

Description Description of AV 1

COV increment: 0.00 Units: no_units

The object name, units, value, and status are shown for the object number entered at the top of the page. Click Prev/Next to scroll through the list. Click Refresh to update the page, or Update to accept changes.

The object name and description may be changed here. BACnet units may be selected. Initial COV increment may be entered. When any of these are changed, be sure to save the updated configuration by executing "Save XML Config File" on the File Manager page.

The object may be set Out of Service by checking that box and clicking Update. The present value may be changed by entering a value, checking Force, and clicking Update.

Analog Value objects may be both a source and destination for some other BACnet or Modbus device. The source of data for the Value object will be reading from a remote device when associated with a Read Map. The destination of data for the Value object will be writing to a remote device when associated with a Write Map. If a remote device is mapped, the device links are displayed above. You may click on either link to view the respective mapping.

The Value object may be simultaneously associated with both Read and Write maps pointing to the same remote device object. When this Value object receives new data (from any source), this data will be written to the mapped remote device before any subsequent read from the same device. Thus the Value data is not discarded by the read operation before the new data can be written.

Out of Service means polling of the remote device will stop. While out of service, the present value may be written by an external BACnet client but it will not be written to any mapped remote device. Data may be forced via this web page at any time, but will be overwritten by the next read from a remote device unless the object is out of service.

Reliability codes indicate errors as itemized on the tabular object list.

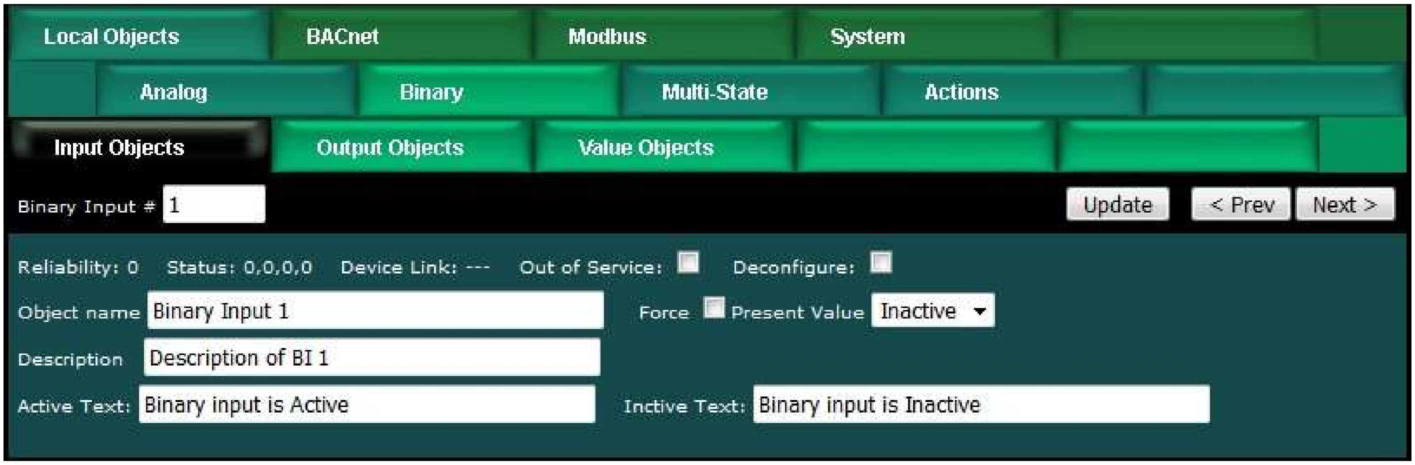
Status flags A,B,C,D indicate the following, 0 meaning not true, 1 meaning true:
A = in alarm
B = fault
C = overridden
D = out of service

Device link will indicate BIP, RTU, or TCP, followed by R for read or W for write, and a number which is the map number in the table of read or write maps for mapping to this BACnet object. The designation R means read from a remote device, and W means write to a remote device.

Check 'Deconfigure' and click Update to erase configuration for this object.

3.3.4 Binary Input Objects

The source of data for a Binary Input object will typically be reading from some other BACnet or Modbus device.



Binary Input # 1

Reliability: 0 Status: 0,0,0,0 Device Link: --- Out of Service: □ Deconfigure: □

Object name Binary Input 1 Force □ Present Value Inactive

Description Description of BI 1

Active Text: Binary input is Active Inctive Text: Binary input is Inactive

The object name, units, value, and status are shown for the object number entered at the top of the page. Click Prev/Next to scroll through the list. Click Refresh to update the page, or Update to accept changes.

The object name and description may be changed here. State text may be entered. When any of these are changed, be sure to save the updated configuration by executing "Save XML Config File" on the File Manager page.

The object may be set Out of Service by checking that box and clicking Update. The present value may be changed by entering a value, checking Force, and clicking Update.

The source of data for an Binary Input object will typically be reading from some other BACnet or Modbus device via the map indicated by the Device Link. The mapped device will be polled at the rate specified by the Read Map.

Out of Service means polling of the mapped remote device will stop. While out of service, the present value may be written by the BACnet client. Data may be forced via this web page at any time, but will be overwritten by the next mapped client update unless the object is out of service.

Reliability codes indicate errors as itemized on the tabular object list.

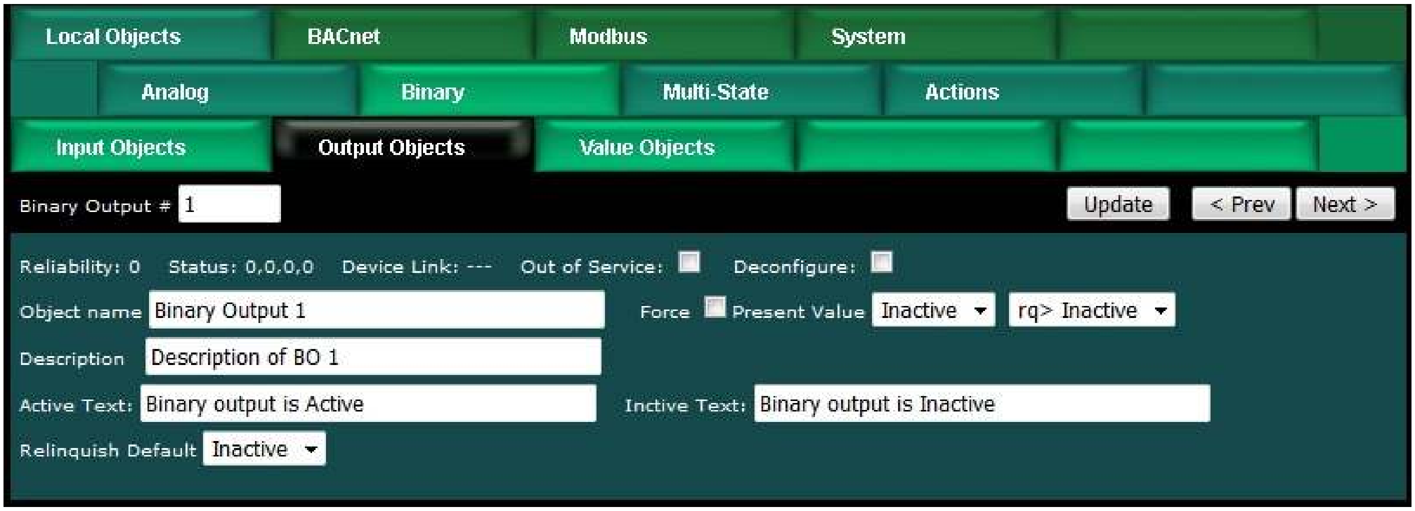
Status flags A,B,C,D indicate the following, 0 meaning not true, 1 meaning true:
A = in alarm
B = fault
C = overridden
D = out of service

Device link will indicate BIP, RTU, or TCP, followed by R for read or W for write, and a number which is the map number in the table of read or write maps for mapping to this BACnet object. The designation R means read from a remote device, and W means write to a remote device.

Check 'Deconfigure' and click Update to erase configuration for this object.

3.3.5 Binary Output Objects

The destination of data for a Binary Output object will typically be some other BACnet or Modbus device.



Binary Output # 1

Reliability: 0 Status: 0,0,0,0 Device Link: --- Out of Service: □ Deconfigure: □

Object name Binary Output 1 Force □ Present Value Inactive rq> Inactive

Description Description of BO 1

Active Text: Binary output is Active Inctive Text: Binary output is Inactive

Relinquish Default Inactive

The object name, units, value, and status are shown for the object number entered at the top of the page. Click Prev/Next to scroll through the list. Click Refresh to update the page, or Update to accept changes.

The object name and description may be changed here. State text may be entered. When any of these are changed, be sure to save the updated configuration by executing "Save XML Config File" on the File Manager page..

The destination of data for a Binary Output object will be writing the remote BACnet or Modbus device via the map indicated by the Device Link. The remote device will be updated upon change of source data and/or periodically as defined by the Write Map.

The Binary Output object is commandable, meaning the BACnet client must write both a value and a priority level for that value. The highest level value will be the one written to the remote device (if one is mapped). If all values are relinquished, the relinquish default value will be written to the remote device.

To set an output object manually from this page, check the Force box, enter a value in the Present Value window, and select a priority level to assign to your forced value. Then click Update. To return a given priority level to NULL, simply type the word NULL in the Present Value window, check Force, and click Update.

Out of service means the mapped remote device will not be written to. Values written by the BACnet client will be retained, but only applied when this object is placed back in service. At that time, the highest priority value will be written to the remote device.

Reliability codes indicate errors as itemized on the tabular object list.

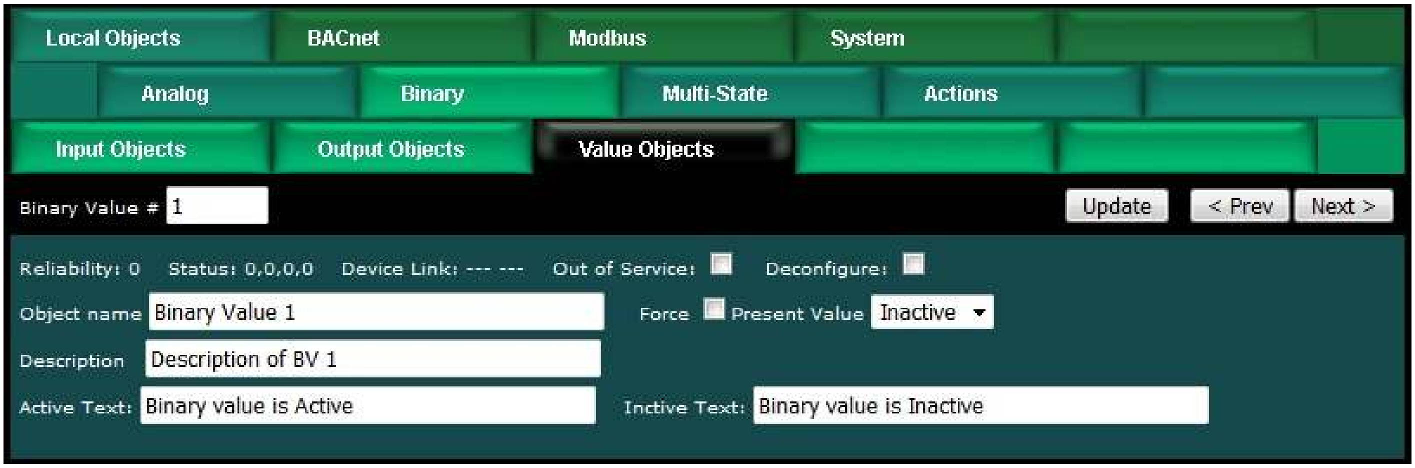
Status flags A,B,C,D indicate the following, 0 meaning not true, 1 meaning true:
A = in alarm
B = fault
C = overridden
D = out of service

Device link will indicate BIP, RTU, or TCP, followed by R for read or W for write, and a number which is the map number in the table of read or write maps for mapping to this BACnet object. The designation R means read from a remote device, and W means write to a remote device.

Check 'Deconfigure' and click Update to erase configuration for this object.

3.3.6 Binary Value Objects

Binary Value objects may be both a source and destination for some other BACnet or Modbus device.



Binary Value # 1

Reliability: 0 Status: 0,0,0,0 Device Link: --- --- Out of Service: □ Deconfigure: □

Object name Binary Value 1 Force □ Present Value Inactive

Description Description of BV 1

Active Text: Binary value is Active Inctive Text: Binary value is Inactive

The object name, units, value, and status are shown for the object number entered at the top of the page. Click Prev/Next to scroll through the list. Click Refresh to update the page, or Update to accept changes.

The object name and description may be changed here. State text may be entered. When any of these are changed, be sure to save the updated configuration by executing "Save XML Config File" on the File Manager page.

The object may be set Out of Service by checking that box and clicking Update. The present value may be changed by entering a value, checking Force, and clicking Update.

Binary Value objects may be both a source and destination for some other BACnet or Modbus device. The source of data for the Value object will be reading from a remote device when associated with a Read Map. The destination of data for the Value object will be writing to a remote device when associated with a Write Map. If a remote device is mapped, the device links are displayed above. You may click on either link to view the respective mapping.

The Value object may be simultaneously associated with both Read and Write maps pointing to the same remote device object. When this Value object receives new data (from any source), this data will be written to the mapped remote device before any subsequent read from the same device. Thus the Value data is not discarded by the read operation before the new data can be written.

Out of Service means polling of the remote device will stop. While out of service, the present value may be written by an external BACnet client but it will not be written to any mapped remote device. Data may be forced via this web page at any time, but will be overwritten by the next read from a remote device unless the object is out of service.

Reliability codes indicate errors as itemized on the tabular object list.

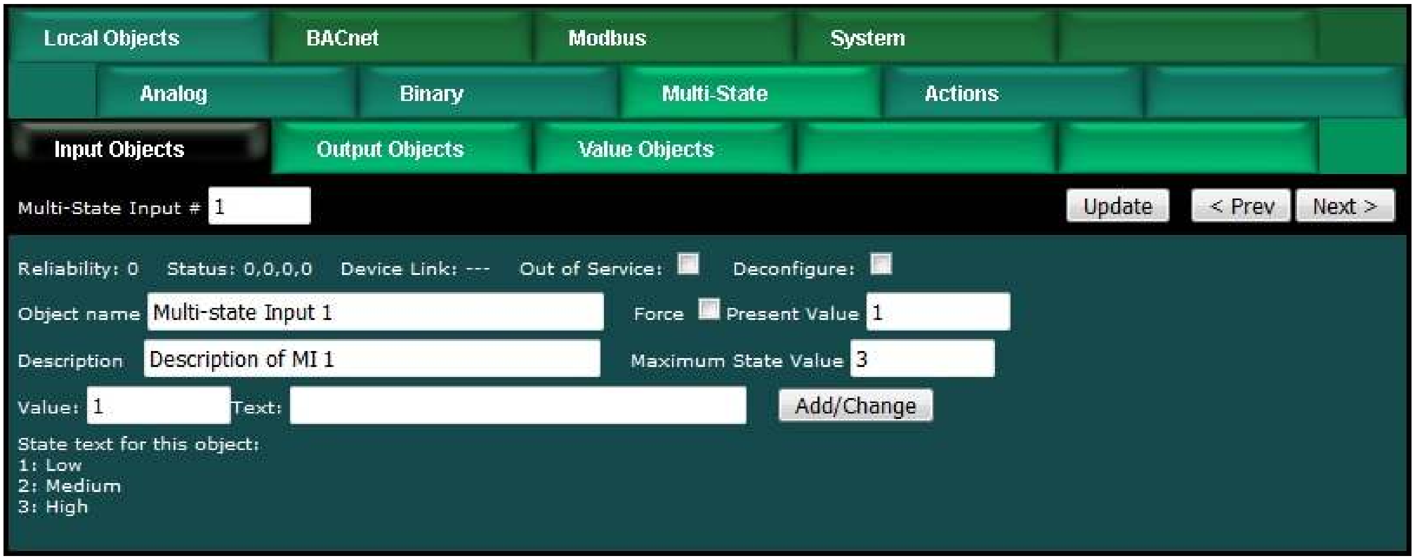
Status flags A,B,C,D indicate the following, 0 meaning not true, 1 meaning true:
A = in alarm
B = fault
C = overridden
D = out of service

Device link will indicate BIP, RTU, or TCP, followed by R for read or W for write, and a number which is the map number in the table of read or write maps for mapping to this BACnet object. The designation R means read from a remote device, and W means write to a remote device.

Check 'Deconfigure' and click Update to erase configuration for this object.

3.3.7 Multistate Input Objects

The source of data for a Multistate Input object will typically be reading from some other BACnet or Modbus device.



Multi-State Input # 1

Reliability: 0 Status: 0,0,0,0 Device Link: --- Out of Service: □ Deconfigure: □

Object name Multi-state Input 1 Force □ Present Value 1

Description Description of MI 1 Maximum State Value 3

Value: 1 Text:

State text for this object:
1: Low
2: Medium
3: High

The object name, value, and status are shown for the object number entered at the top of the page. Click Prev/Next to scroll through the list. Click Refresh to update the page, or Update to accept changes.

The object name and description may be changed here. When changed, be sure to save the updated configuration by executing "Save XML Config File" on the File Manager page.

State text may be added. Before adding text, set the maximum state value for this object. Then add text strings corresponding to each of the number of states allocated by entering the value, corresponding text, and clicking Add/Change. When changed, be sure to save the updated configuration by executing "Save XML Config File" on the File Manager page.

The object may be set Out of Service by checking that box and clicking Update. The present value may be changed by entering a value, checking Force, and clicking Update.

The source of data for a Multistate Input object will typically be reading from some other BACnet or Modbus device via the map indicated by the Device Link. The mapped device will be polled at the rate specified by the Read Map.

Out of Service means polling of the mapped remote device will stop. While out of service, the present value may be written by the BACnet client. Data may be forced via this web page at any time, but will be overwritten by the next mapped client update unless the object is out of service.

Reliability codes indicate errors as itemized on the tabular object list.

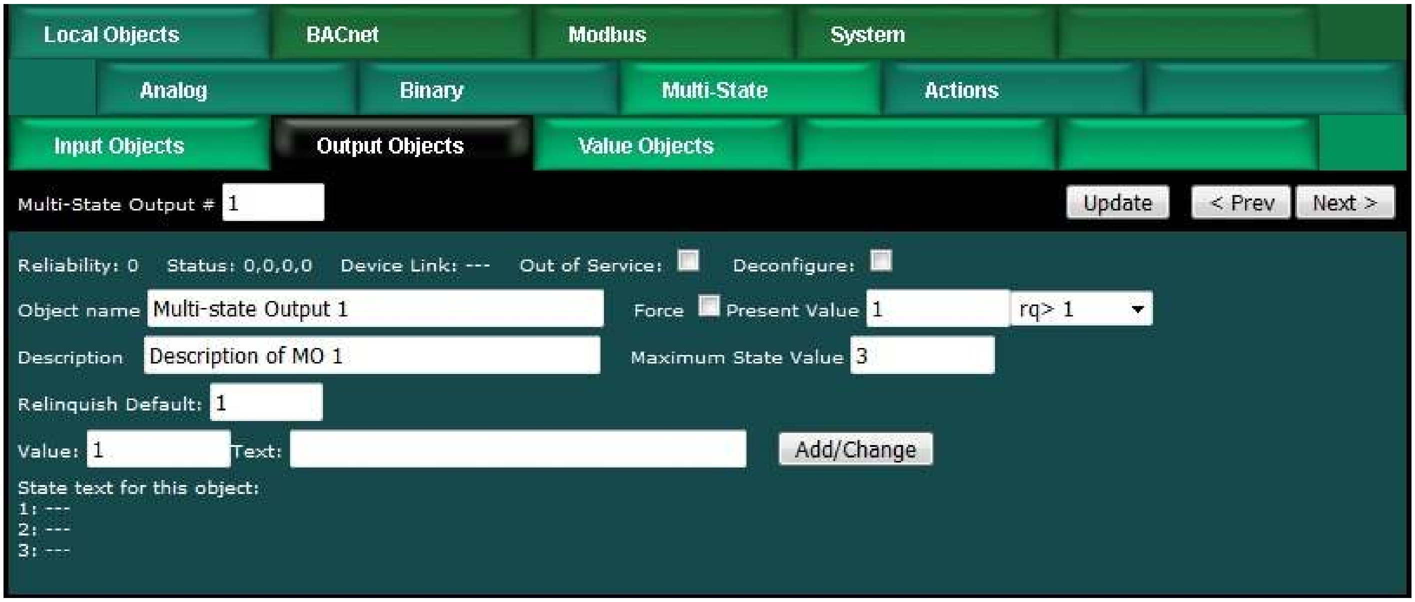
Status flags A,B,C,D indicate the following, 0 meaning not true, 1 meaning true:
A = in alarm
B = fault
C = overridden
D = out of service

Device link will indicate BIP, RTU, or TCP, followed by R for read or W for write, and a number which is the map number in the table of read or write maps for mapping to this BACnet object. The designation R means read from a remote device, and W means write to a remote device.

Check 'Deconfigure' and click Update to erase configuration for this object.

3.3.8 Multistate Output Objects

The destination of data for a Multistate Output object will typically be some other BACnet or Modbus device.



Multi-State Output # 1

Reliability: 0 Status: 0,0,0,0 Device Link: --- Out of Service: □ Deconfigure: □

Object name Multi-state Output 1 Force □ Present Value 1 rq> 1

Description Description of MO 1 Maximum State Value 3

Relinquish Default: 1

Value: 1 Text:

State text for this object:
1: ---
2: ---
3: ---

The object name, value, and status are shown for the object number entered at the top of the page. Click Prev/Next to scroll through the list. Click Refresh to update the page, or Update to accept changes.

The object name and description may be changed here. When changed, be sure to save the updated configuration by executing "Save XML Config File" on the File Manager page.

State text may be added. Before adding text, set the maximum state value for this object. Then add text strings corresponding to each of the number of states allocated by entering the value, corresponding text, and clicking Add/Change. When changed, be sure to save the updated configuration by executing "Save XML Config File" on the File Manager page.

The object may be set Out of Service by checking that box and clicking Update. The present value may be changed by entering a value, checking Force, and clicking Update.

The Multistate Output object is commandable, meaning the BACnet client must write both a value and a priority level for that value. The highest level value will be the one written to the remote device (if one is mapped). If all values are relinquished, the relinquish default value will be written to the remote device.

To set an output object manually from this page, check the Force box, enter a value in the Present Value window, and select a priority level to assign to your forced value. Then click Update. To return a given priority level to NULL, simply type the word NULL in the Present Value window, check Force, and click Update.

Out of service means the mapped remote device will not be written to. Values written by the BACnet client will be retained, but only applied when this object is placed back in service. At that time, the highest priority value will be written to the remote device.

Reliability codes indicate errors as itemized on the tabular object list.

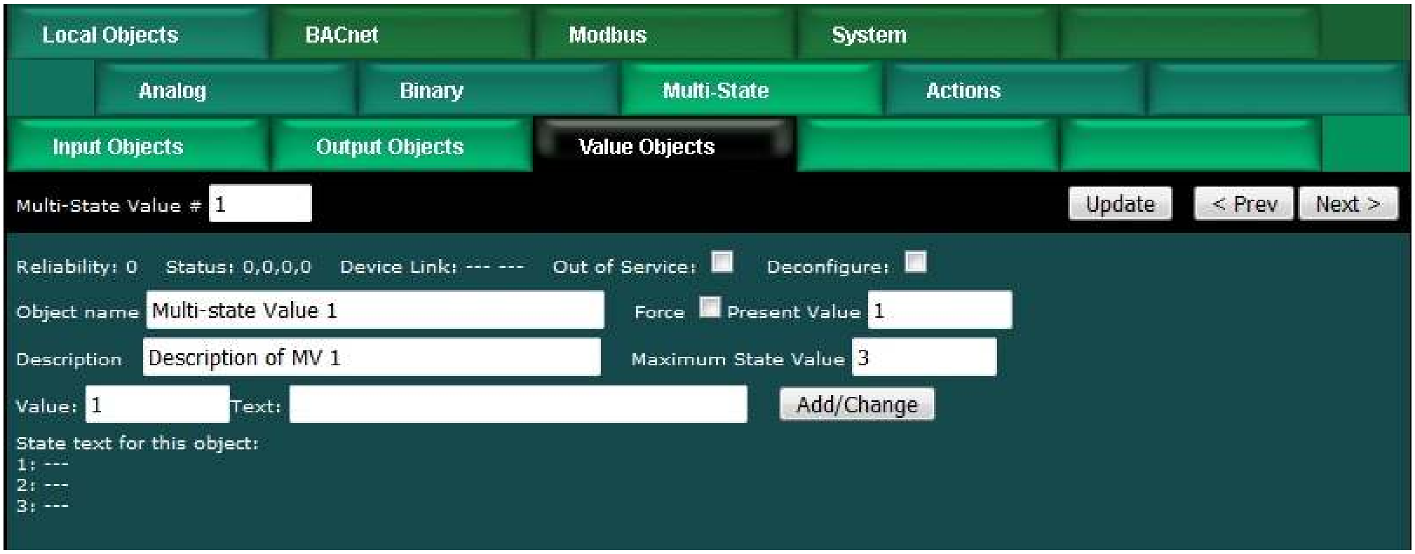
Status flags A,B,C,D indicate the following, 0 meaning not true, 1 meaning true:
A = in alarm
B = fault
C = overridden
D = out of service

Device link will indicate BIP, RTU, or TCP, followed by R for read or W for write, and a number which is the map number in the table of read or write maps for mapping to this BACnet object. The designation R means read from a remote device, and W means write to a remote device.

Check 'Deconfigure' and click Update to erase configuration for this object.

3.3.9 Multistate Value Objects

Multistate Value objects may be both a source and destination for some other BACnet or Modbus device.



Multi-State Value # 1

Reliability: 0 Status: 0,0,0,0 Device Link: --- --- Out of Service: □ Deconfigure: □

Object name Multi-state Value 1 Force □ Present Value 1

Description Description of MV 1 Maximum State Value 3

Value: 1 Text:

State text for this object:
1: ---
2: ---
3: ---

The object name, value, and status are shown for the object number entered at the top of the page. Click Prev/Next to scroll through the list. Click Refresh to update the page, or Update to accept changes.

The object name and description may be changed here.When changed, be sure to save the updated configuration by executing "Save XML Config File" on the File

Manager page.

State text may be added. Before adding text, set the maximum state value for this object. Then add text strings corresponding to each of the number of states allocated by entering the value, corresponding text, and clicking Add/Change. When changed, be sure to save the updated configuration by executing "Save XML Config File" on the File Manager page.

The object may be set Out of Service by checking that box and clicking Update. The present value may be changed by entering a value, checking Force, and clicking Update.

Multistate Value objects may be both a source and destination for some other BACnet or Modbus device. The source of data for the Value object will be reading from a remote device when associated with a Read Map. The destination of data for the Value object will be writing to a remote device when associated with a Write Map. If a remote device is mapped, the device links are displayed above. You may click on either link to view the respective mapping.

The Value object may be simultaneously associated with both Read and Write maps pointing to the same remote device object. When this Value object receives new data (from any source), this data will be written to the mapped remote device before any subsequent read from the same device. Thus the Value data is not discarded by the read operation before the new data can be written.

Out of Service means polling of the remote device will stop. While out of service, the present value may be written by an external BACnet client but it will not be written to any mapped remote device. Data may be forced via this web page at any time, but will be overwritten by the next read from a remote device unless the object is out of service.

Reliability codes indicate errors as itemized on the tabular object list.

Status flags A,B,C,D indicate the following, 0 meaning not true, 1 meaning true:
A = in alarm
B = fault
C = overridden
D = out of service

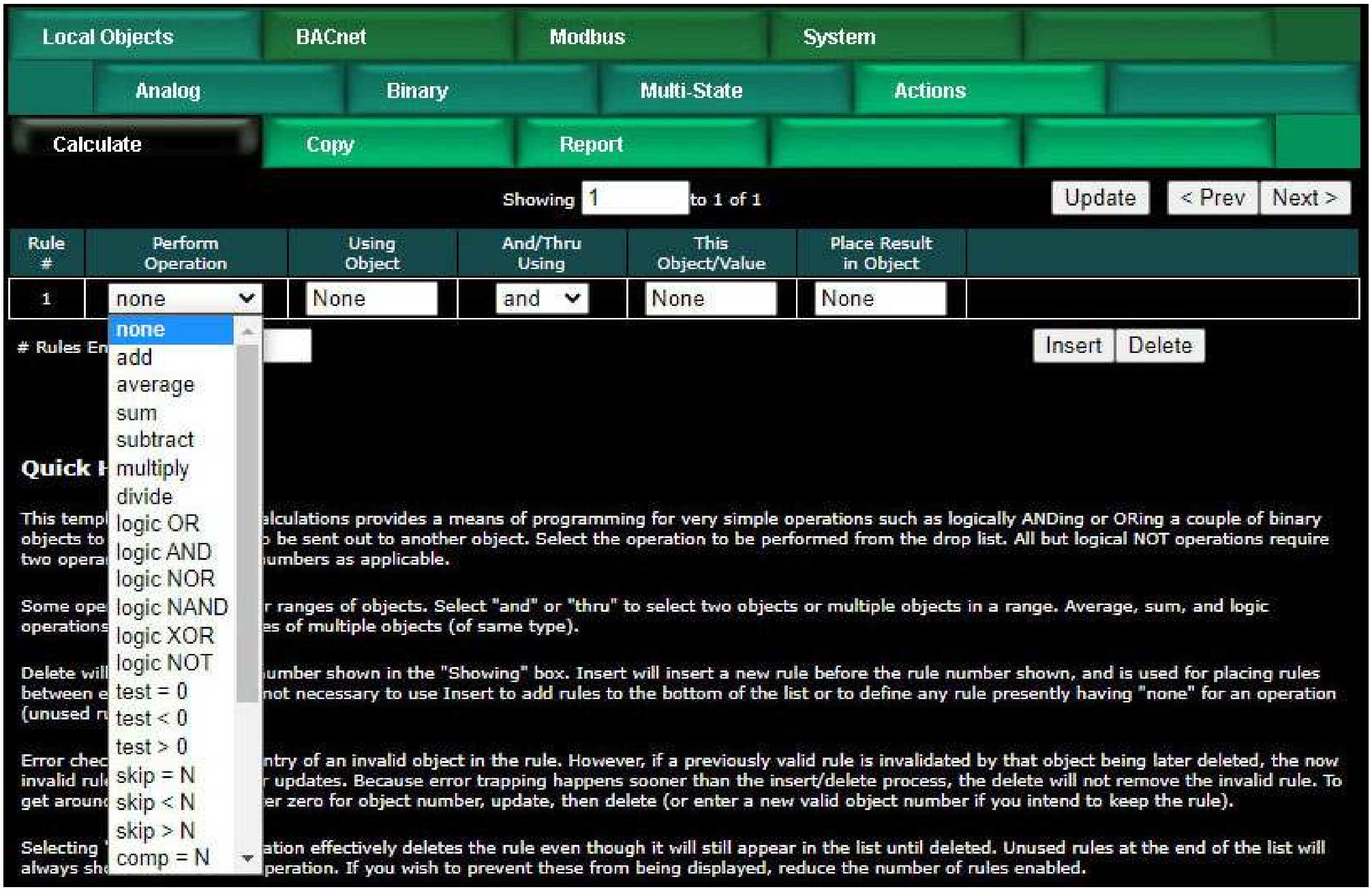
Device link will indicate BIP, RTU, or TCP, followed by R for read or W for write, and a number which is the map number in the table of read or write maps for mapping to this BACnet object. The designation R means read from a remote device, and W means write to a remote device.

Check 'Deconfigure' and click Update to erase configuration for this object.

3.4 Local Object Calculate Rules

The Babel Buster BB3-7101/MX-71 includes the ability to do simple calculations based on simple template rules. Select the operation, one or two operands as applicable, and a object to place the result in. Operations like "multiply" will use objects A "and" B.

Operations like "sum" can add up the contents of a series of objects by selecting "thru" instead of "and". These template rules can be useful for doing minor data manipulation or testing for purposes of enabling rules, or for generating derived values.



Showing 1 to 1 of 1

Rule #	Perform Operation	Using Object	And/Thru Using	This Object/Value	Place Result in Object
1	none	None	and	None	None

Rules En

Quick H

This templ alculations provides a means of programming for very simple operations such as logically ANDing or ORing a couple of binary
objects to be sent out to another object. Select the operation to be performed from the drop list. All but logical NOT operations require
two opera umbers as applicable.

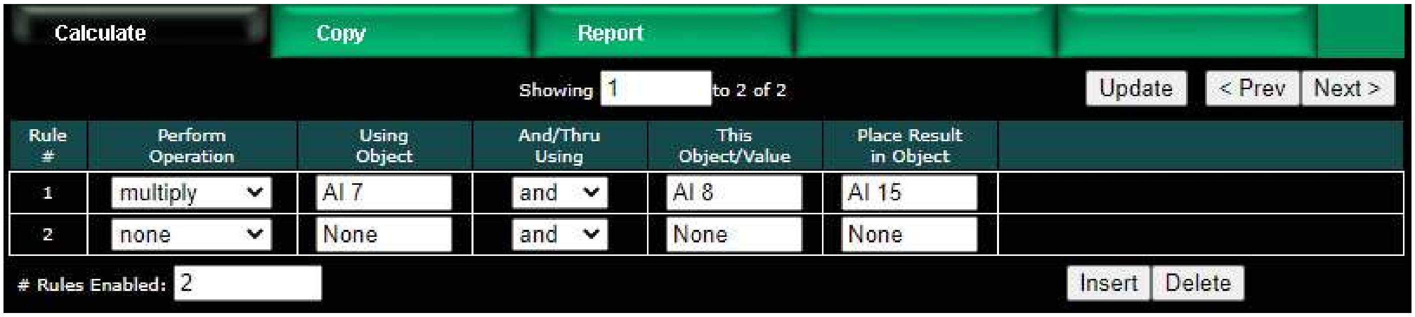
Some ope r ranges of objects. Select "and" or "thru" to select two objects or multiple objects in a range. Average, sum, and logic
operations es of multiple objects (of same type).

Delete will umber shown in the "Showing" box. Insert will insert a new rule before the rule number shown, and is used for placing rules
between e not necessary to use Insert to add rules to the bottom of the list or to define any rule presently having "none" for an operation
(unused r

Error chec ntry of an invalid object in the rule. However, if a previously valid rule is invalidated by that object being later deleted, the now
invalid rul r updates. Because error trapping happens sooner than the insert/delete process, the delete will not remove the invalid rule. To
get aroun er zero for object number, update, then delete (or enter a new valid object number if you intend to keep the rule).

Selecting ' ation effectively deletes the rule even though it will still appear in the list until deleted. Unused rules at the end of the list will
always sh peration. If you wish to prevent these from being displayed, reduce the number of rules enabled.

Here is an example of a template rule that multiplies the value of Analog Input 7 by value of Analog Input 8 and places the result in Analog Input 15. An example of application would be to multiply a voltage reading input by a current reading input to derive a power value presented as an input.



Showing 1 to 2 of 2

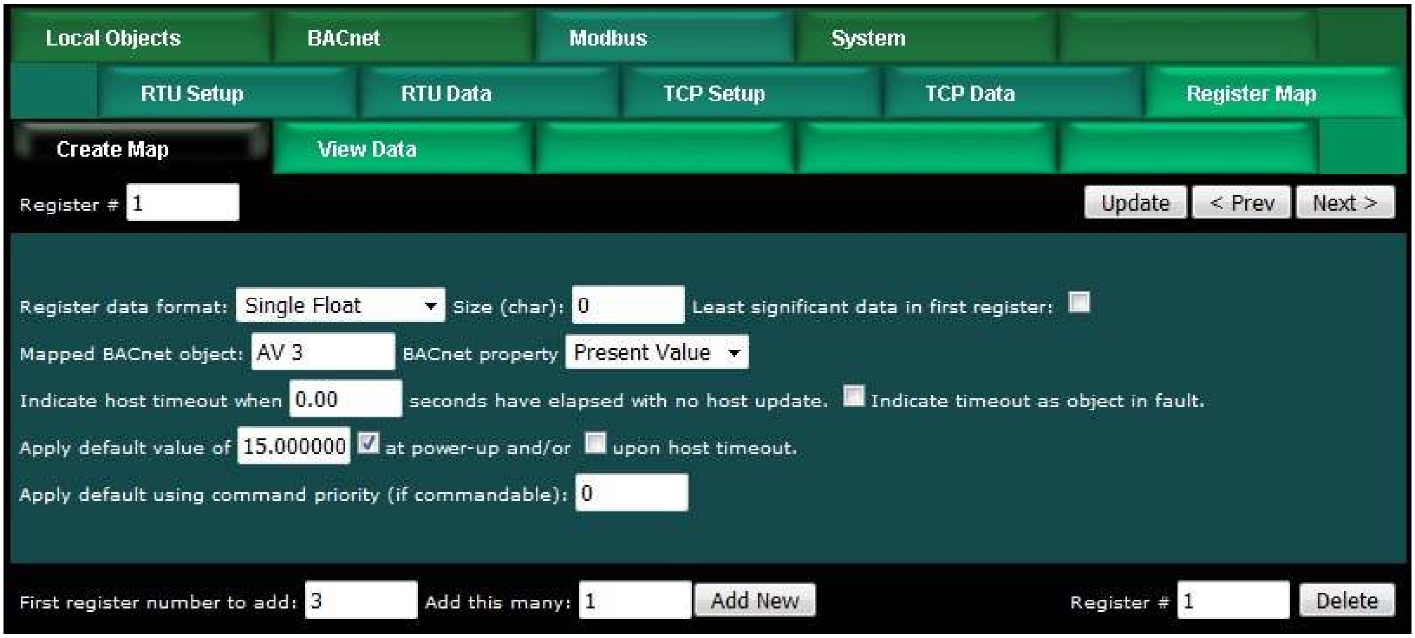
Rule #	Perform Operation	Using Object	And/Thru Using	This Object/Value	Place Result in Object
1	multiply	AI 7	and	AI 8	AI 15
2	none	None	and	None	None

Rules Enabled: 2

There are times when you may want to make a calculation based on a constant value. There are "tricks" you can apply to use an object for a constant value.

Constants may be introduced into the calculation by reserving a commandable object to hold that constant, and then configuring the relinquish default to be that value. Then reference that object in your calculate rule.

The Modbus Register Map allows applying a default value at power-up (also applies when reloading the configuration file). The screen shot below illustrates using Analog Value #3 as a holder for the constant value 15. This constant may now be used in math calculation sequences.



Register # 1

Register data format: Single Float Size (char): 0 Least significant data in first register: □

Mapped BACnet object: AV 3 BACnet property Present Value

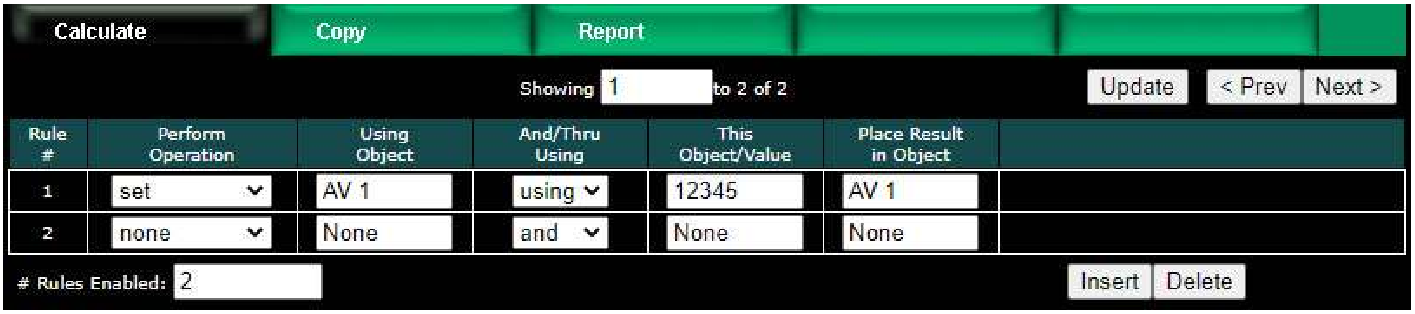
Indicate host timeout when 0.00 seconds have elapsed with no host update. □ Indicate timeout as object in fault.

Apply default value of 15.000000 ☑ at power-up and/or □ upon host timeout.

Apply default using command priority (if commandable): 0

First register number to add: 3 Add this many: 1 Register # 1

The "set" operation can be used to directly place an unsigned integer value into an object. The screen shot below illustrates setting Analog Value 1 to a value of 12345. The value in a set operation can only be unsigned integer as the value was originally intended for use in bit mask operations.



Showing 1 to 2 of 2

Rule #	Perform Operation	Using Object	And/Thru Using	This Object/Value	Place Result in Object
1	set	AV 1	using	12345	AV 1
2	none	None	and	None	None

Rules Enabled: 2

Operations available on two or more objects using 'and' or 'thru':

add	Add two objects
average	Average two or more objects
sum	Sum two or more objects
subtract	Subtract second object from first
multiply	Multiply two objects
divide	Divide first object by second
logic OR	Logically OR two or more objects
logic AND	Logically AND two or more objects
logic NOR	Logically NOR two or more objects
logic NAND	Logically NAND two or more objects
logic XOR	Logically Exclusive-OR two objects

Operations available on one object:

logic NOT	Generate bit-wise negation of object
test = 0	Set result to 'true' if object is zero
test < 0	Set result to 'true' if object is less than zero
test > 0	Set result to 'true' if object is greater than zero
relinquish	Relinquish command priority previosly written to a commandable object

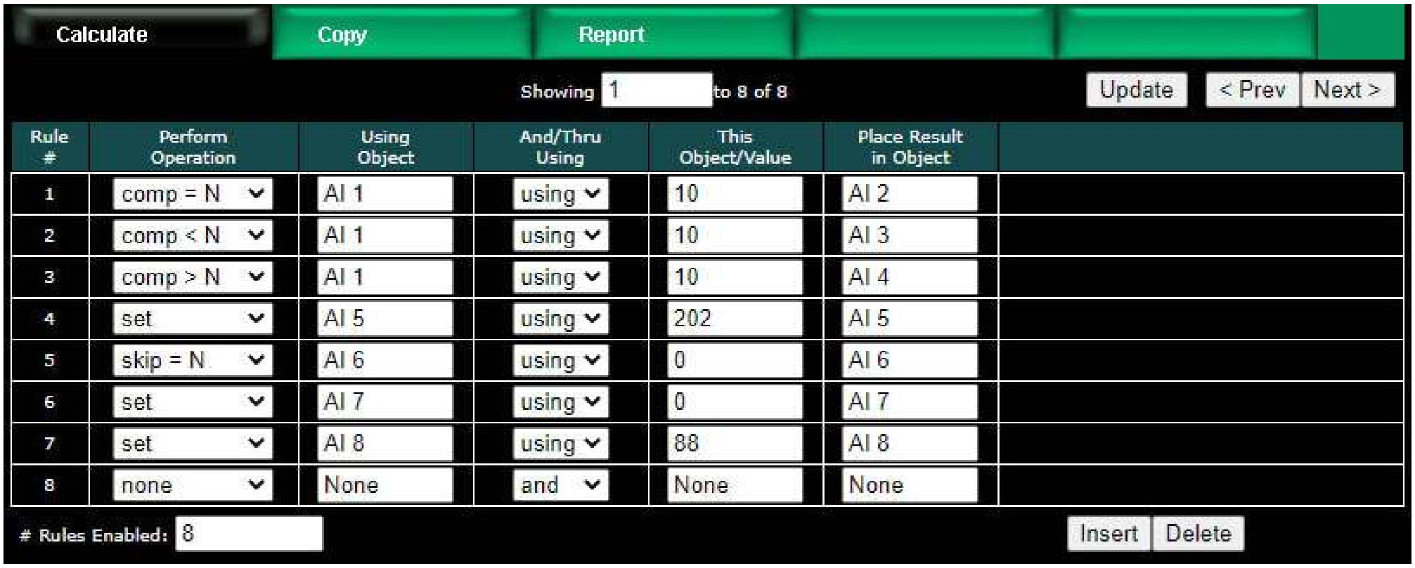
Operations available on one object 'using' a given value:

set	Set object to given value (unsigned 32-bit integer)
skip = N	Skip next operation if object is equal to given value
skip < N	Skip next operation if object is less than given value
skip > N	Skip next operation if object is greater than given value
comp = N	Compare, set result 'true' if object is equal to given value
comp < N	Compare, set result 'true' if object is less than given value
comp > N	Compare, set result 'true' if object is greater than given value
pack	Perform Pack operation (see text)
fill	Perform Fill operation (see text)
unpack	Perform Unpack operation (see text)
priority	Sets command priority that will be used in any subsequent write to a commandable object

Operations "using" a given value will have an unsigned integer value in the "This Object/Value" column rather than an object number. These values will be displayed as integer for most operations, but will be displayed in hexadecimal for pack, fill, and unpack operations since these operate primarily on bit mask values.

The result of a test or compare will be zero if false, or one if true when the result object is a Analog or Binary object. The result of a test or compare when the result object is Multi-State will be 1 if false and 2 if true (since Multi-State cannot use zero).

The next two screen shots illustrate compare, set, and skip operations. Rule 5 says that rule 6 will not be executed if AI 6 contains a zero. If AI 6 is not equal to zero, then rule 6 will be executed. (The numbers rule 6 and AI 6 are not related in any other way, this is just conincidence in the example.)

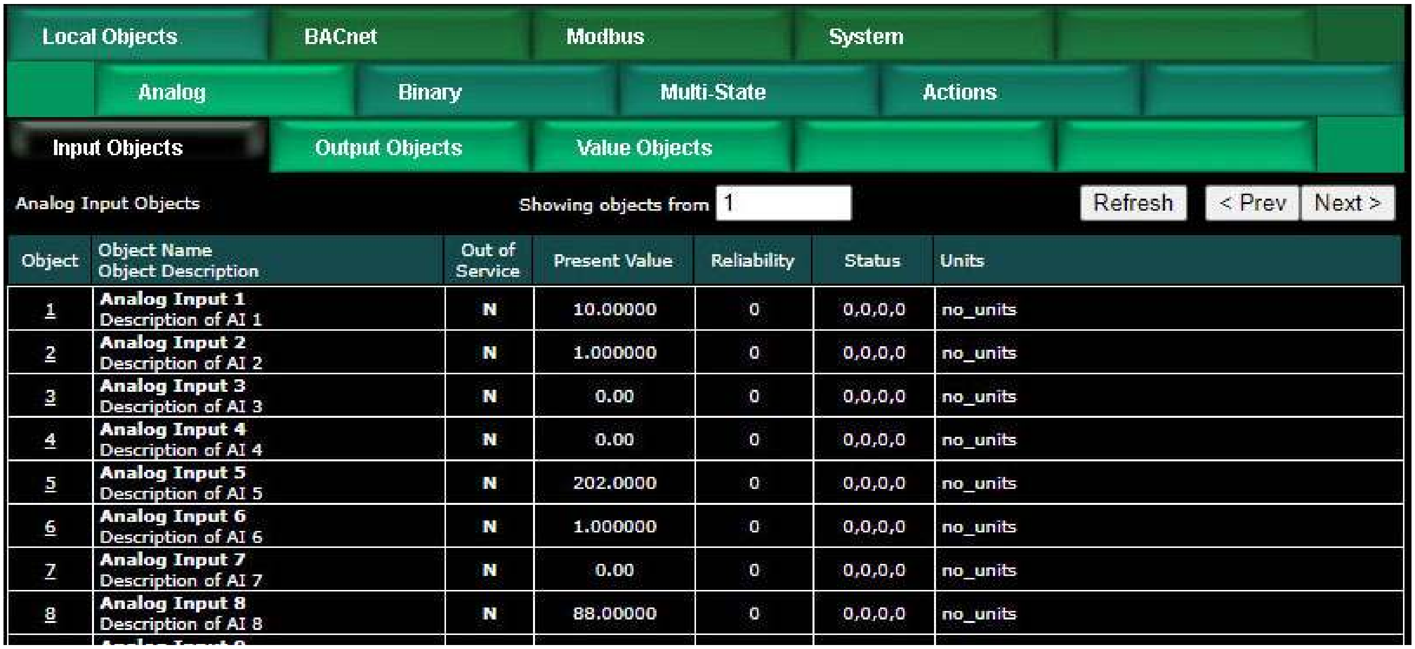


Showing 1 to 8 of 8

Rule #	Perform Operation	Using Object	And/Thru Using	This Object/Value	Place Result in Object
1	comp = N	AI 1	using	10	AI 2
2	comp < N	AI 1	using	10	AI 3
3	comp > N	AI 1	using	10	AI 4
4	set	AI 5	using	202	AI 5
5	skip = N	AI 6	using	0	AI 6
6	set	AI 7	using	0	AI 7
7	set	AI 8	using	88	AI 8
8	none	None	and	None	None

Rules Enabled: 8

Object values for examples using the above operations are illustrated below.

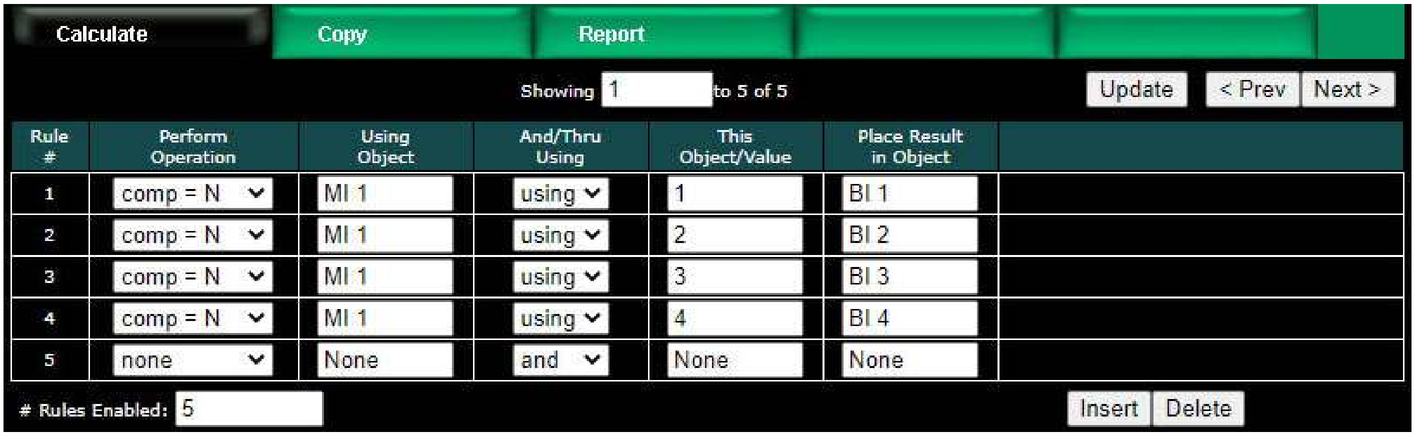


Analog Input Objects

Showing objects from 1

Object	Object Name Object Description	Out of Service	Present Value	Reliability	Status	Units
1	**Analog Input 1** Description of AI 1	N	10.00000	0	0,0,0,0	no_units
2	**Analog Input 2** Description of AI 2	N	1.000000	0	0,0,0,0	no_units
3	**Analog Input 3** Description of AI 3	N	0.00	0	0,0,0,0	no_units
4	**Analog Input 4** Description of AI 4	N	0.00	0	0,0,0,0	no_units
5	**Analog Input 5** Description of AI 5	N	202.0000	0	0,0,0,0	no_units
6	**Analog Input 6** Description of AI 6	N	1.000000	0	0,0,0,0	no_units
7	**Analog Input 7** Description of AI 7	N	0.00	0	0,0,0,0	no_units
8	**Analog Input 8** Description of AI 8	N	88.00000	0	0,0,0,0	no_units

The following screen shot illustrates the use of calculate rules to set the states of multiple Binary Input objects based on the value of a single Multi-State Input object. In this example, BI 1 will be active if the MI 1 state is 1, BI 2 will be active for state 2, and so on.



Showing 1 to 5 of 5

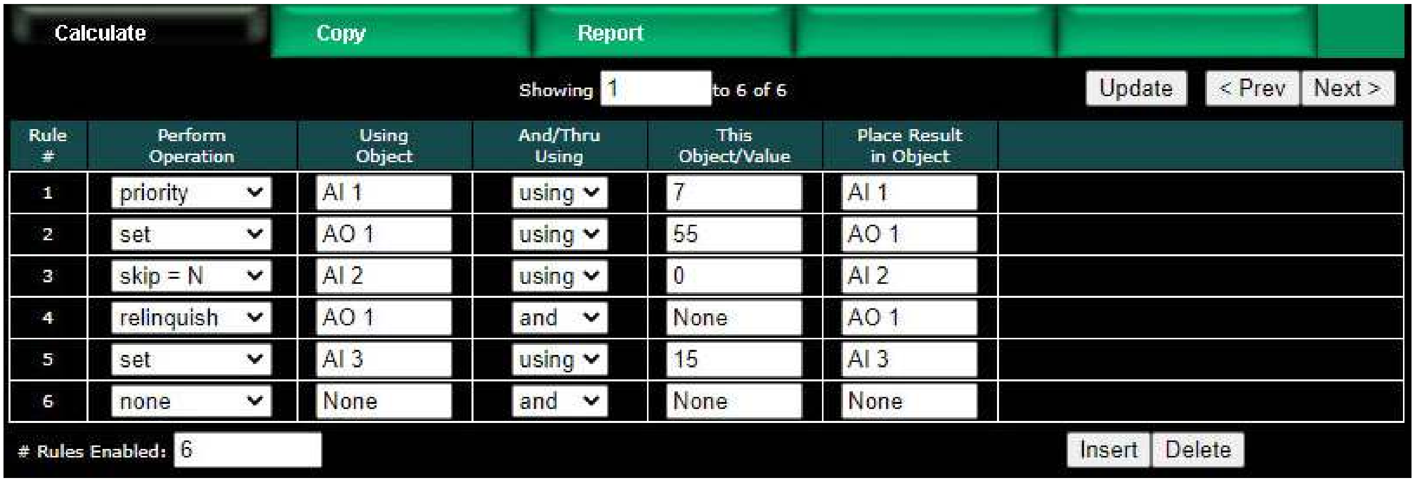
Rule #	Perform Operation	Using Object	And/Thru Using	This Object/Value	Place Result in Object
1	comp = N	MI 1	using	1	BI 1
2	comp = N	MI 1	using	2	BI 2
3	comp = N	MI 1	using	3	BI 3
4	comp = N	MI 1	using	4	BI 4
5	none	None	and	None	None

Rules Enabled: 5

The calculate rules have access to command priority and relinquish when the result regiter is a commandable object.

The command priority is set using the priority operation as illustrated below. In this case, the "using object" and "place result in object" are only place holders to keep the rule validator happy. The only thing actually used in this operation is the "this value". In the example below, the command priority is being set to 7. This command priority will be used for any subsequent operations that place a result in a commandable object, and will remain in effect until another priority operation is used. If no priority operation is ever included, then the default local command priority on the BACnet settings page will be used.

Once a commandable object has been set by a calculate rule, it can be relinquished by using the relinquish operation as illustrated below. The command priority currently in effect as the result of the most recent priority operation will be relinquished. The calculate rules themselves do not have any ability to remember command priorities - it is up to you to keep track of command priority using the priority operation.



Showing 1 to 6 of 6

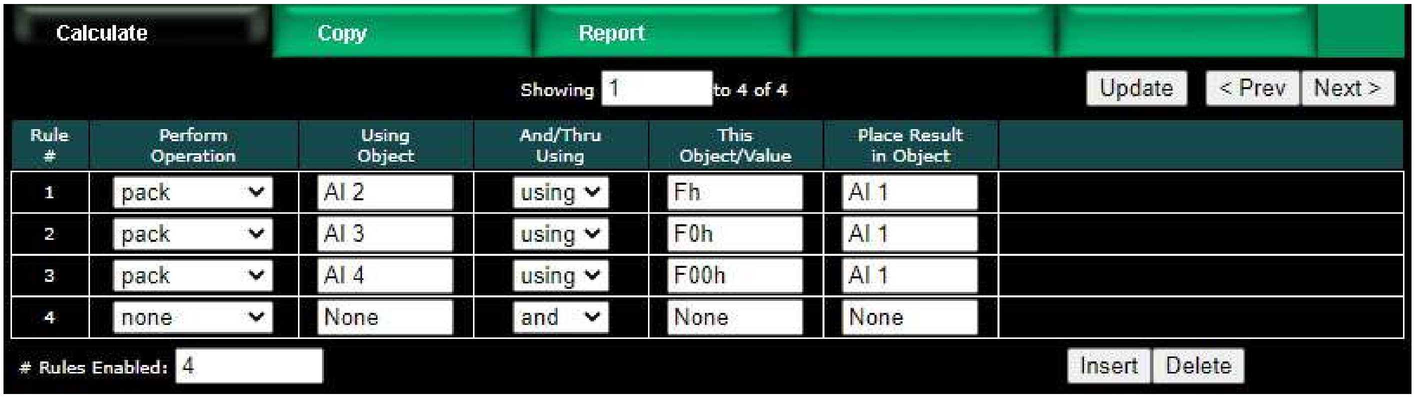
Rule #	Perform Operation	Using Object	And/Thru Using	This Object/Value	Place Result in Object
1	priority	AI 1	using	7	AI 1
2	set	AO 1	using	55	AO 1
3	skip = N	AI 2	using	0	AI 2
4	relinquish	AO 1	and	None	AO 1
5	set	AI 3	using	15	AI 3
6	none	None	and	None	None

Rules Enabled: 6

Pack and fill are used for packing multiple local objects into a single object for purposes of emulating existing equipment when the Babel Buster is functioning as a server (slave). When pack and fill are used, "using" should be selected, and the second entry is a hexadecimal mask or fill value. The hexadecimal value should include "h" at the end to signify hexadecimal (otherwise the value will be parsed as decimal).

The pack mask is both a bit mask and position indicator. To calculate the contribution of a given calculate rule, the mask is right shifted until the least significant bit is nonzero, then this shifted mask is logically AND-ed with the local object content. The resulting masked value is then left shifted back to the original mask position. This final shifted result is then logically OR-ed into the result object (after first clearing the bits in the affected position of the result object).

Fill is simple - it simply logically OR's the bit mask into the result object.

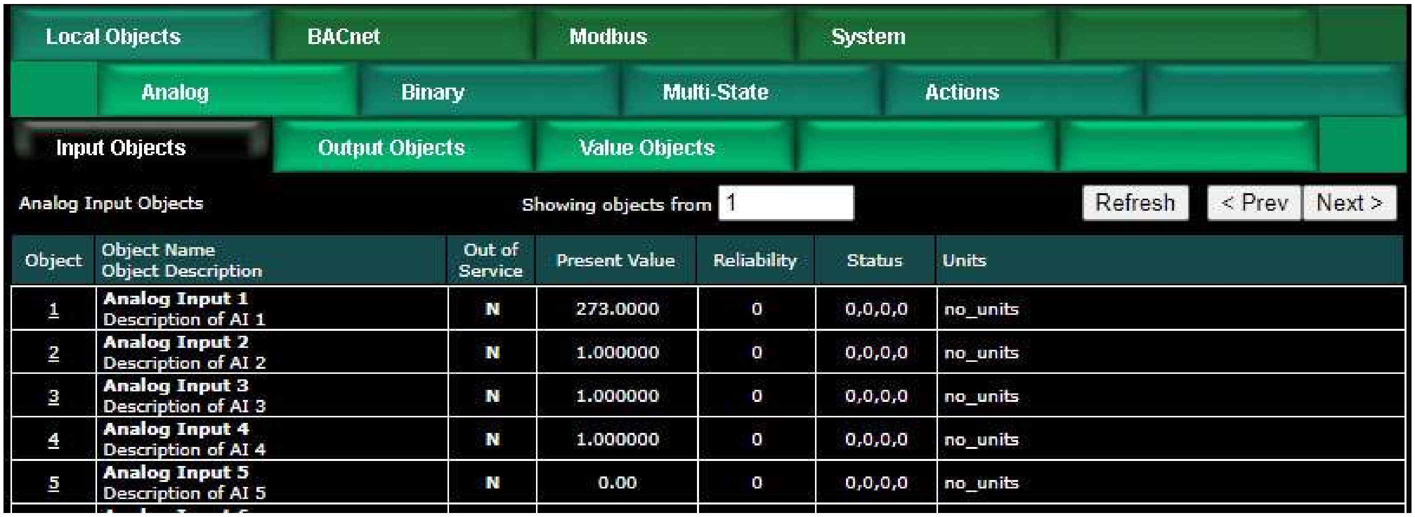


Showing 1 to 4 of 4

Rule #	Perform Operation	Using Object	And/Thru Using	This Object/Value	Place Result in Object
1	pack	AI 2	using	Fh	AI 1
2	pack	AI 3	using	F0h	AI 1
3	pack	AI 4	using	F00h	AI 1
4	none	None	and	None	None

Rules Enabled: 4

Using the above rules, an example of resulting data would be as follows.

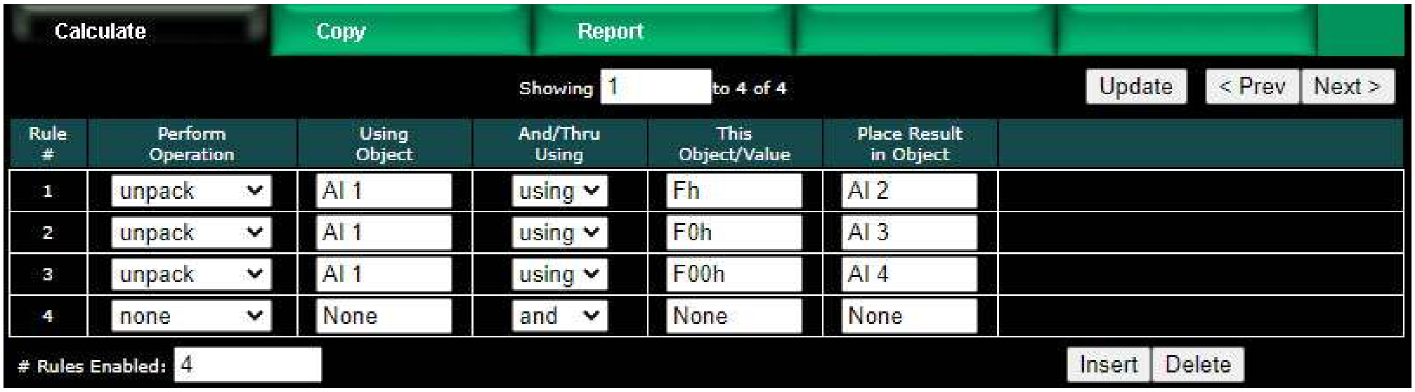


Analog Input Objects

Showing objects from 1

Object	Object Name Object Description	Out of Service	Present Value	Reliability	Status	Units
1	**Analog Input 1** Description of AI 1	N	273.0000	0	0,0,0,0	no_units
2	**Analog Input 2** Description of AI 2	N	1.000000	0	0,0,0,0	no_units
3	**Analog Input 3** Description of AI 3	N	1.000000	0	0,0,0,0	no_units
4	**Analog Input 4** Description of AI 4	N	1.000000	0	0,0,0,0	no_units
5	**Analog Input 5** Description of AI 5	N	0.00	0	0,0,0,0	no_units

A set of calculate rules that would exactly reverse the above operation would be as follows.



Showing 1 to 4 of 4

Rule #	Perform Operation	Using Object	And/Thru Using	This Object/Value	Place Result in Object
1	unpack	AI 1	using	Fh	AI 2
2	unpack	AI 1	using	F0h	AI 3
3	unpack	AI 1	using	F00h	AI 4
4	none	None	and	None	None

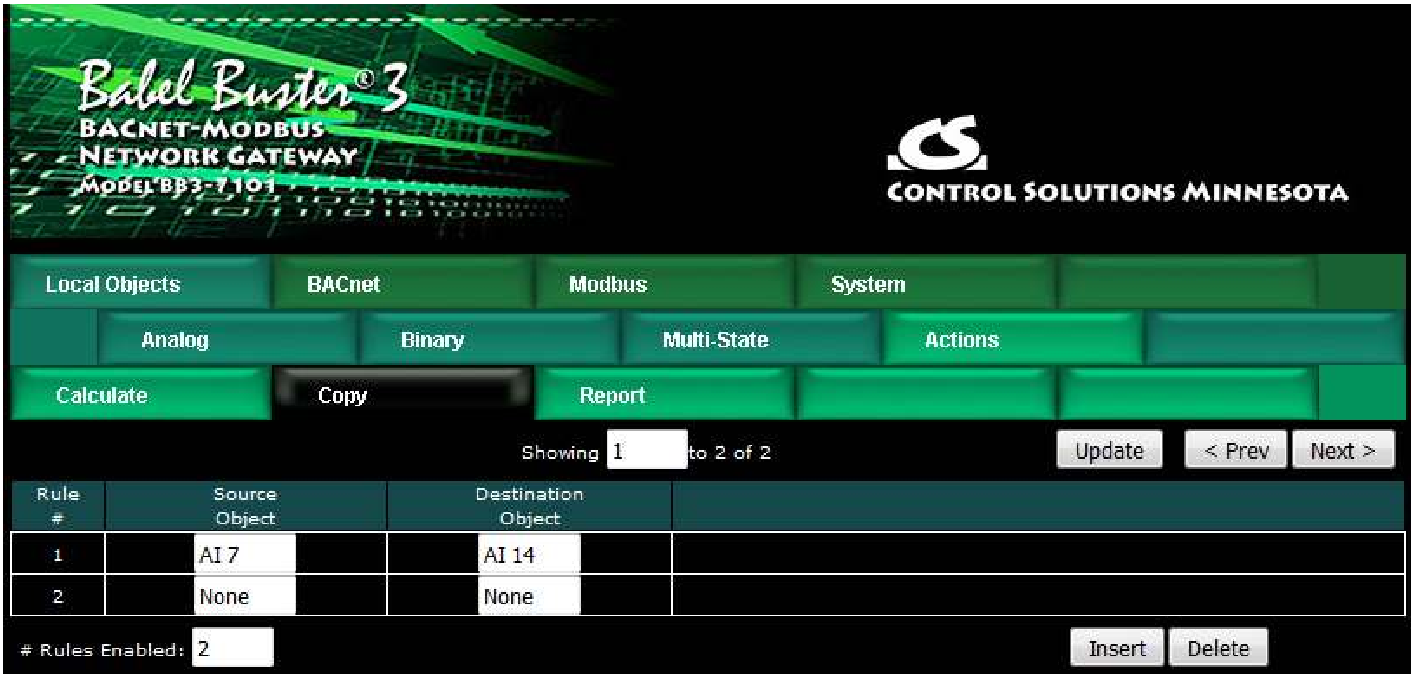
Rules Enabled: 4

The pack, fill, and unpack instructions are primarily targeting Modbus applications.

They are less useful when dealing with BACnet objects, but are retained in the calculate rule set for consistency across the gateway family.

3.5 Local Object Copy Rules

The copy rules provide a means of simply copying the content of one object to another.



Showing 1 to 2 of 2

Rule #	Source Object	Destination Object
1	AI 7	AI 14
2	None	None

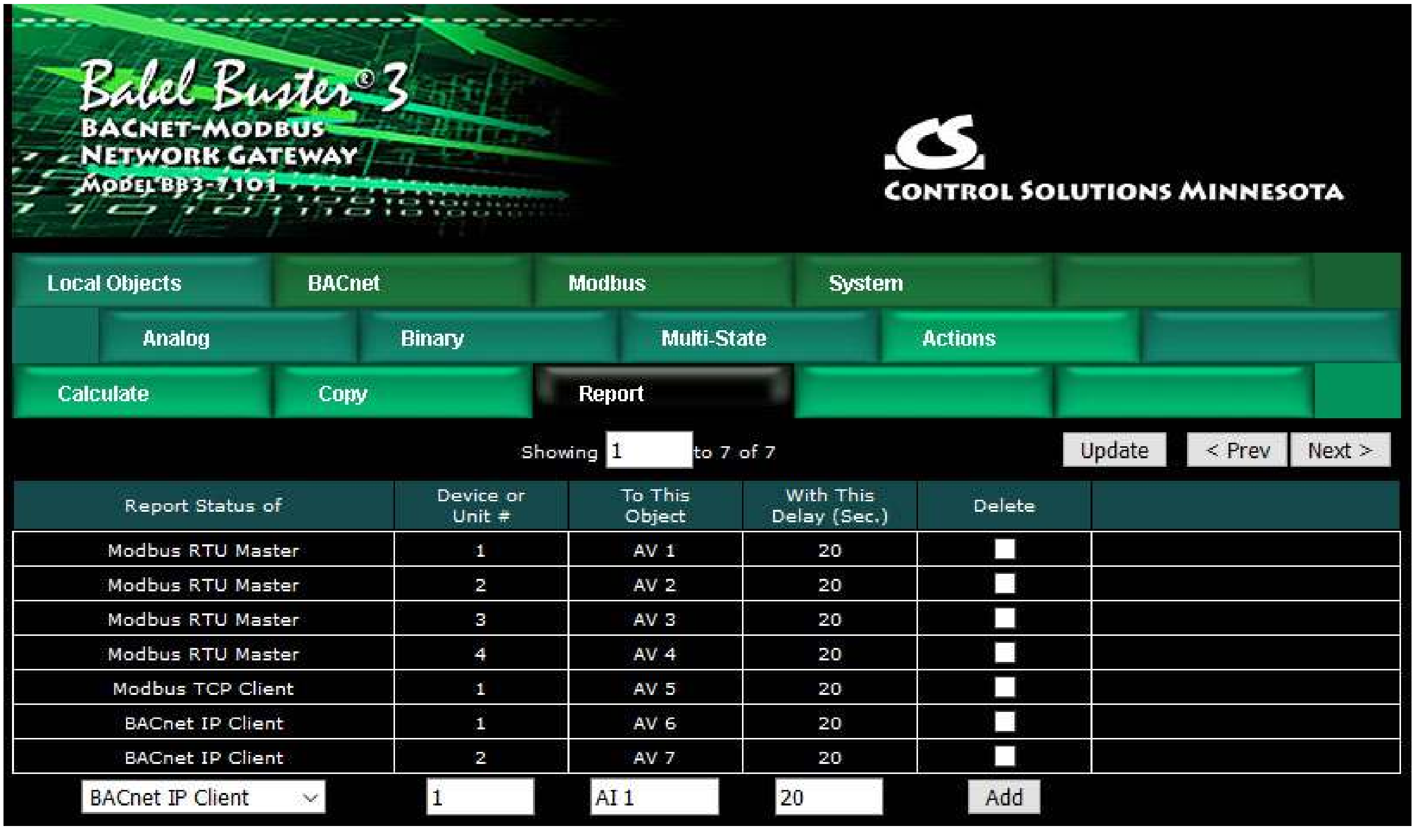
Rules Enabled: 2

The above rule would cause the value of AI 7 to be placed in AI 14. If a floating point (Analog) value is copied to a Binary object, the Binary object will be set Active if the value was nonzero, or cleared to Inactive if zero. Analog values copied to a Multistate object will be not only truncated, but bounded to the maximum number of states (not a recommended use of Copy).

3.6 Device Status Reporting

The Babel Buster BB3-7101/MX-71 read maps include the ability to set a default value upon 'n' read fails, meaning that if the Babel Buster gets an error 'n' times attempting to read that point, it will automatically set the corresponding local object to the given default value to indicate the problem. This indication applies on a point by point basis, but of course any one point can be used as an indication that the entire device may be offline.

The BB3-7101/MX-71 also includes the ability to report device errors to an assigned status object rather than rely on default values. This reporting is configured on the Report page.



Showing 1 to 7 of 7

Report Status of	Device or Unit #	To This Object	With This Delay (Sec.)	Delete
Modbus RTU Master	1	AV 1	20	□
Modbus RTU Master	2	AV 2	20	□
Modbus RTU Master	3	AV 3	20	□
Modbus RTU Master	4	AV 4	20	□
Modbus TCP Client	1	AV 5	20	□
BACnet IP Client	1	AV 6	20	□
BACnet IP Client	2	AV 7	20	□
BACnet IP Client	1	AI 1	20	Add

This optional list allows reporting device errors as object values to make it easier to monitor communication failures. The length of the list is variable. To add to the list, select the type of device to report on, select the device instance or unit number to report on, and select an object in which to put the status indication. Enter a delay if desired, and then click Add.

The delay is optional. If zero, there is no delay. If some number of seconds is entered, then the error condition will not be reported until this time period expires. If the error clears before the time is up, then the error is never reported. This is useful for spurious errors that would result in nuisance indications.

To remove a report from the list, check the box in the Delete column and then click Update. Click Prev or Next to scroll through the list.

Error codes placed into the reporting object will be as follows:

0 = No error
1 = Timeout, no response from remote device
2 = Error message received from remote device (e.g. Modbus exception)
3 = Line fault (e.g. CRC error, socket connection error, etc)

Once a Timeout error indication has been set (following delay if applicable), it will automatically return to zero upon the next successful communication with that device.

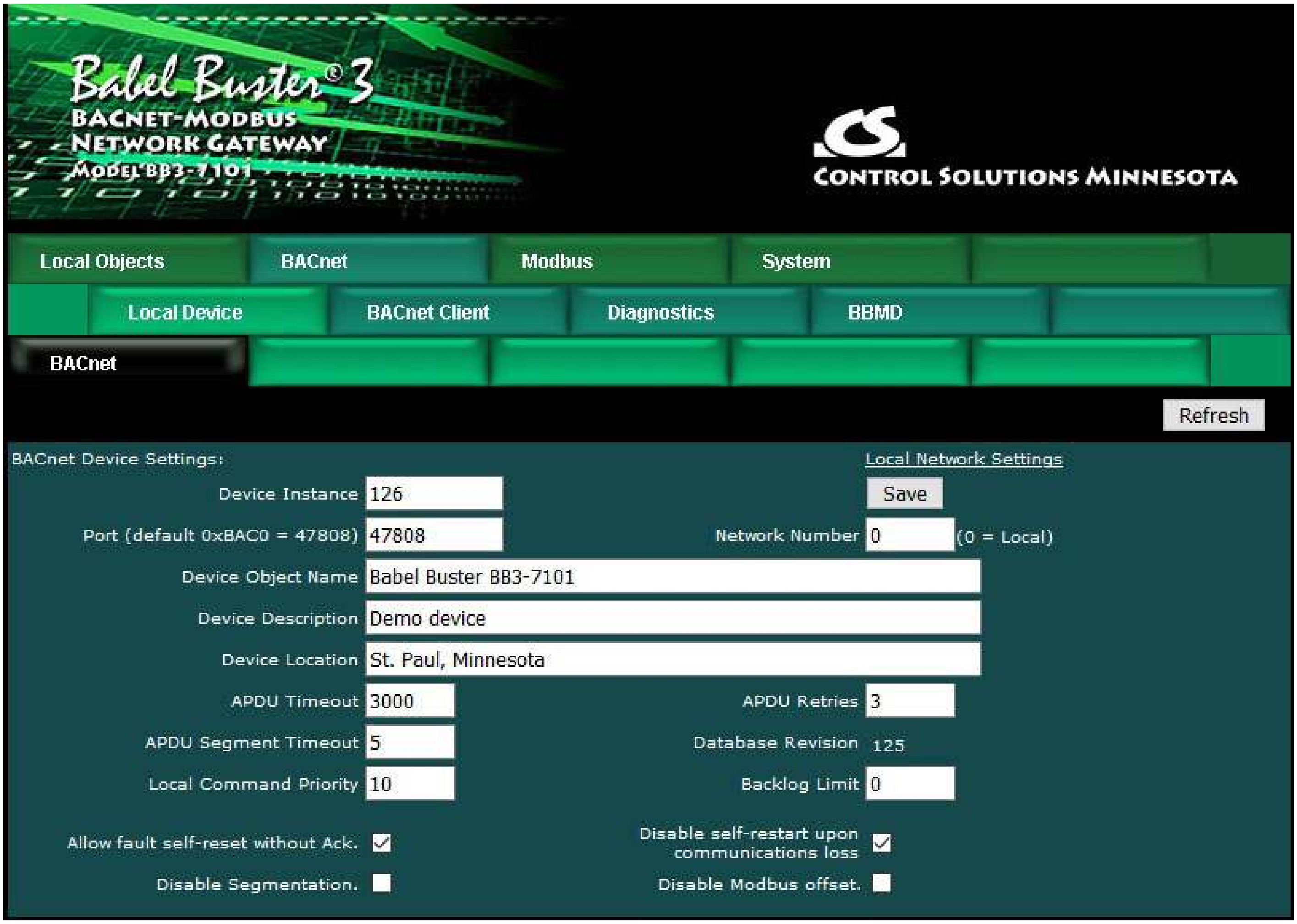
Once either the error message or line fault indication has been set, following delay if applicable, communication must continue free of this same error condition for at least the same delay period before the indication will be reset to zero. If an error message (e.g. Modbus exception) is reported for one data point, but multiple others are error free, then the one error would be hidden without this delay before reset.

Ideally, this delay period should be at least as great as the poll period for the slowest point mapped.

4. Configuring Gateway as a BACnet Device

4.1 Device Object Parameters

The identity of the gateway as a BACnet device is entered on this page, along with other device object parameters.



BACnet Device Settings:

Device Instance	126		
Port (default 0xBAC0 = 47808)	47808	Network Number	0 (0 = Local)
Device Object Name	Babel Buster BB3-7101		
Device Description	Demo device		
Device Location	St. Paul, Minnesota		
APDU Timeout	3000	APDU Retries	3
APDU Segment Timeout	5	Database Revision	125
Local Command Priority	10	Backlog Limit	0
Allow fault self-reset without Ack.	☑	Disable self-restart upon communications loss	☑
Disable Segmentation.	□	Disable Modbus offset.	□

Enter a device instance from 1 to 4,194,303. Enter a port number (note that 47808 is the standard port expected by most BIP devices).

The device object name, description, and location are entered here. The device object name is expected to be unique to the entire BACnet network. Standard BACnet timeout and retry values are also entered on this page. These values are stored in a special area of non-volatile memory rather than the XML configuration file.

Network Number is optional, and normally left set to zero here. Normally, BACnet routers will take care of any network number translation needed. If you leave this network number entry set to zero, then this BB3-7101/MX-71 WILL respond to any local broadcast message. If you enter a network number here, then this device WILL NOT respond to local broadcast messages, and will only respond to remote broadcast messages or global broadcast messages. The default setting is zero, and BTL certification was done with the default setting of zero - no other number is needed here for normal typical use.

Local command priority is used when the result of a Calculate or Copy rule is written to a commandable object. It is also used if the result of a client read map is saved to a local commandable object, although this would not be recommended. Output objects are commandable. Client read maps should store results in input or value objects, while client write maps take their data from value or output objects. In other words, output objects should not be used for input.

Check the "allow" check box to allow faults to self-reset. These faults are those conditions indicated by a non-zero reliability code in any of the data objects (see list on data objects pages). Normally an external client needs to read the realiability code to acknowledge the fault before it will automatically reset. By checking the "allow" check box, faults will automatically self-reset without acknowledgement. This is required any time the client does not periodically read reliability codes but does check fault status - a behavior known to be common to BMS front ends.

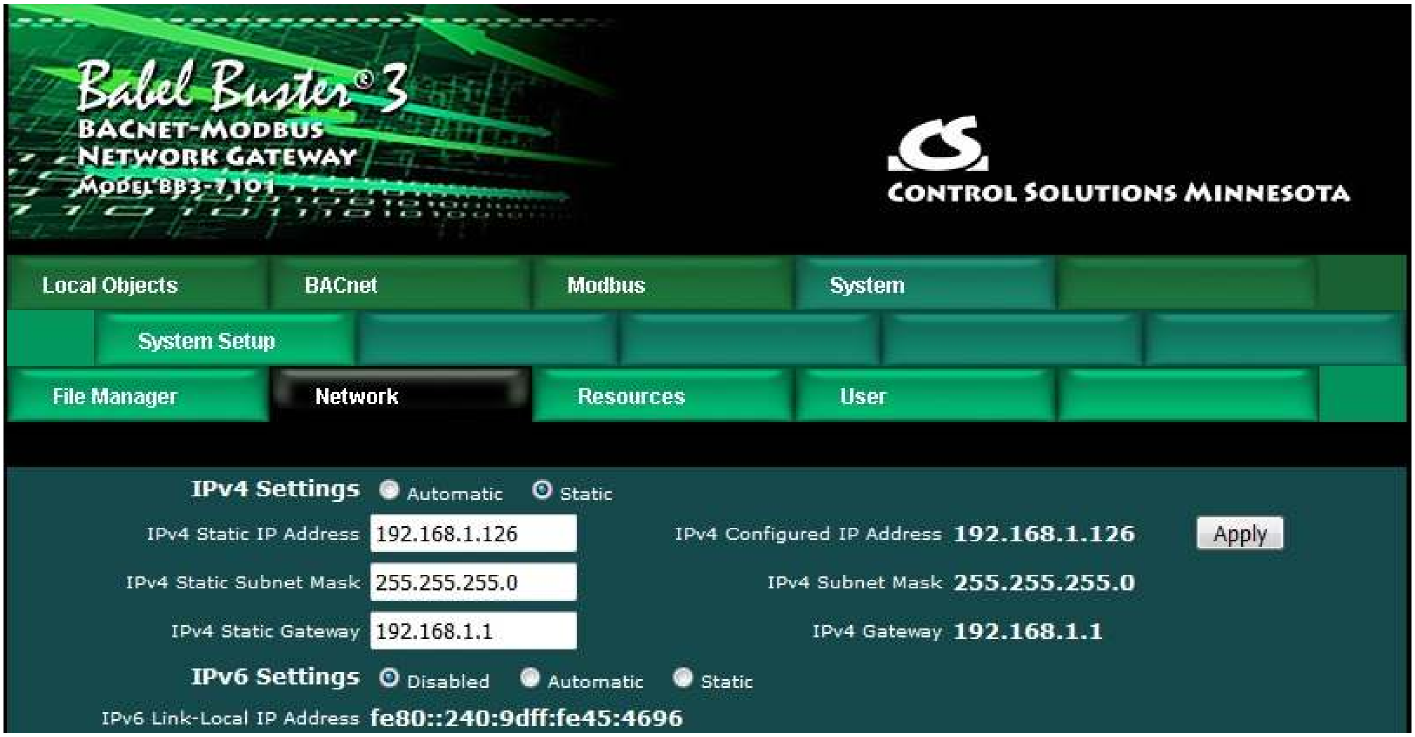
Check the "disable self-restart" box to disable self restart upon communication loss. If this box is not checked, this gateway will restart itself in an attempt to auto-recover if communications with devices has started and then stopped.

By default, when Modbus reads values that are mapped to a Multi-State object, the values are offset by one. Multi-State objects are not allowed to contain a value of zero per BACnet protocol standard. Since Modbus doesn't often deal very well with zero being prohibited, values are offset by one. Therefore, where Modbus sees zero, the BACnet Multi-State object will see one. This offset only applies to Multi-State objects. You can disable this offset by checking the "Disable Modbus offset" box and saving the setting. Just be aware that with the offset disabled, Modbus will not know the difference between zero and one because any attempt to write zero to the Multi-State object will be forced to one. You can safely disable the offset when only reading Multi-State objects, but use extra caution when writing to Multi-State objects from Modbus with the offset disabled.

Click Save to store. This store process will take a little while as these parameters are being saved to non-volatile memory. A change in port number will not take effect until the next system restart.

4.2 Network Settings

The two most important things that must be unique on the BACnet IP network are device instance, and IP address. The IP address is set on the Network page.



IPv4 Settings ○ Automatic ◉ Static

IPv4 Static IP Address 192.168.1.126 IPv4 Configured IP Address 192.168.1.126

IPv4 Static Subnet Mask 255.255.255.0 IPv4 Subnet Mask 255.255.255.0

IPv4 Static Gateway 192.168.1.1 IPv4 Gateway 192.168.1.1

IPv6 Settings ◉ Disabled ○ Automatic ○ Static

IPv6 Link-Local IP Address fe80::240:9dff:fe45:4696

Select either Static or Automatic for IPv4. To change the Static IP address of this device, enter the address, subnet mask, and gateway, then click Apply.
Select Automatic to specify that DHCP should be used to obtain an IP address upon power-up. IP address change will take effect upon next power cycle.

The above screen shot is only a portion of the Network setup page, and is the only part of the Network page that is required for BACnet IP. The remainder of the Network page is discussed in Section 11.3.

The Web User Interface is accessible via IPv6; however, the BACnet IP Client does not yet support IPv6. The only demonstrated version of BACnet IP over IPv6 does not use actual IPv6 addresses - it uses Virtual MAC addresses (VMAC) and address translation tables. The VMAC approach allows IPv6 to coexist with original IPv4 devices. As of the BTL testing of this device, test specifications for BACnet IP over IPv6 were not available and thus IPv6 support is not included in the BTL tested device.

Babel Buster® 3
MODEL BB3-7101-SP
MODEL MX-71-SP
NETWORK GATEWAY

CONTROL SOLUTIONS MINNESOTA

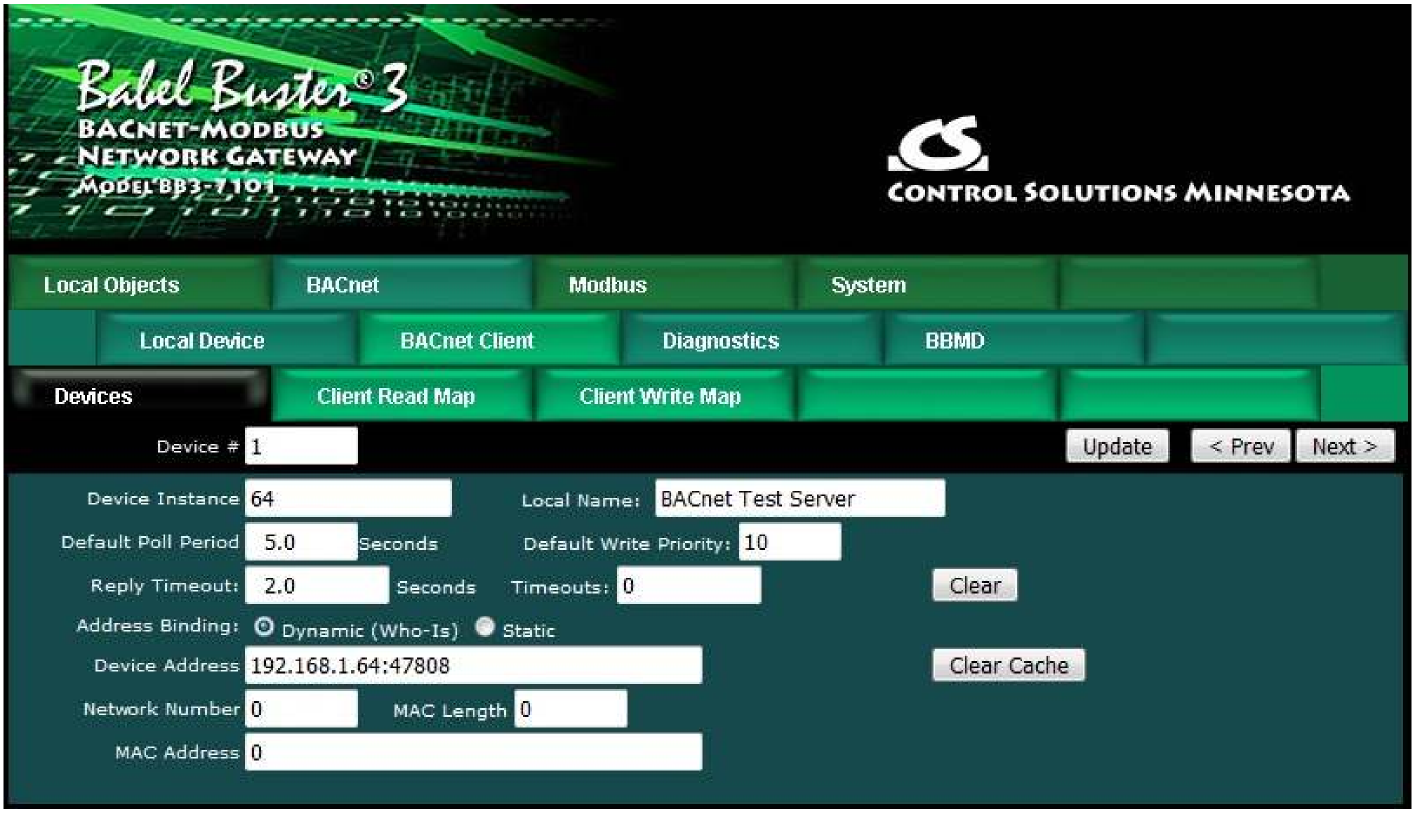
5. Configuring Gateway as a BACnet Client

The BACnet client is used to query other BACnet devices, obtain their Present Value data (or other property value), and store a copy of that data in the BB3-7101/MX-71's own local objects. From there, the data may be accessed by Modbus devices (or other BACnet devices if you have some reason to want to remap objects).

This data exchange with other BACnet devices requires that you define those devices in the list on the BACnet Client Devices page, and then create some number of Read and Write Maps. The maps may be created via the web pages talked about below. But you also have the option of using a standard spread sheet program to create a list that you save as a CSV file, and then import that via the File Manager in the BB3-7101/MX-71. Refer to Section 11.1 for more about importing CSV files, and also Appendix B for CSV file format information.

5.1 BACnet Device List

Setting up the BACnet client consists of identifying one or more BACnet devices, then listing the objects that should be queried (whether read or written). The client configuration pages are illustrated below.



Device # 1

Device Instance 64

Local Name: BACnet Test Server

Default Poll Period 5.0 Seconds

Default Write Priority: 10

Reply Timeout: 2.0 Seconds

Timeouts: 0

Address Binding: ● Dynamic (Who-Is) ○ Static

Device Address 192.168.1.64:47808

Network Number 0

MAC Length 0

MAC Address 0

Device number simply shows you where you are on the internal local device list. Click "next" and "prev" to scroll through the list.

Remote BACnet IP devices to be accessed by this device are specified here. Enter the Device Instance of the remote device, a name to reference in other pages, and a poll rate. Then click "Update".

Select dynamic or static address binding. Dynamic binding is used most often, and simply means the gateway will send out a "Who-Is" request asking for the device instance to respond, at which time the gateway learns its IP address automatically (or MS/TP address via an IP router).

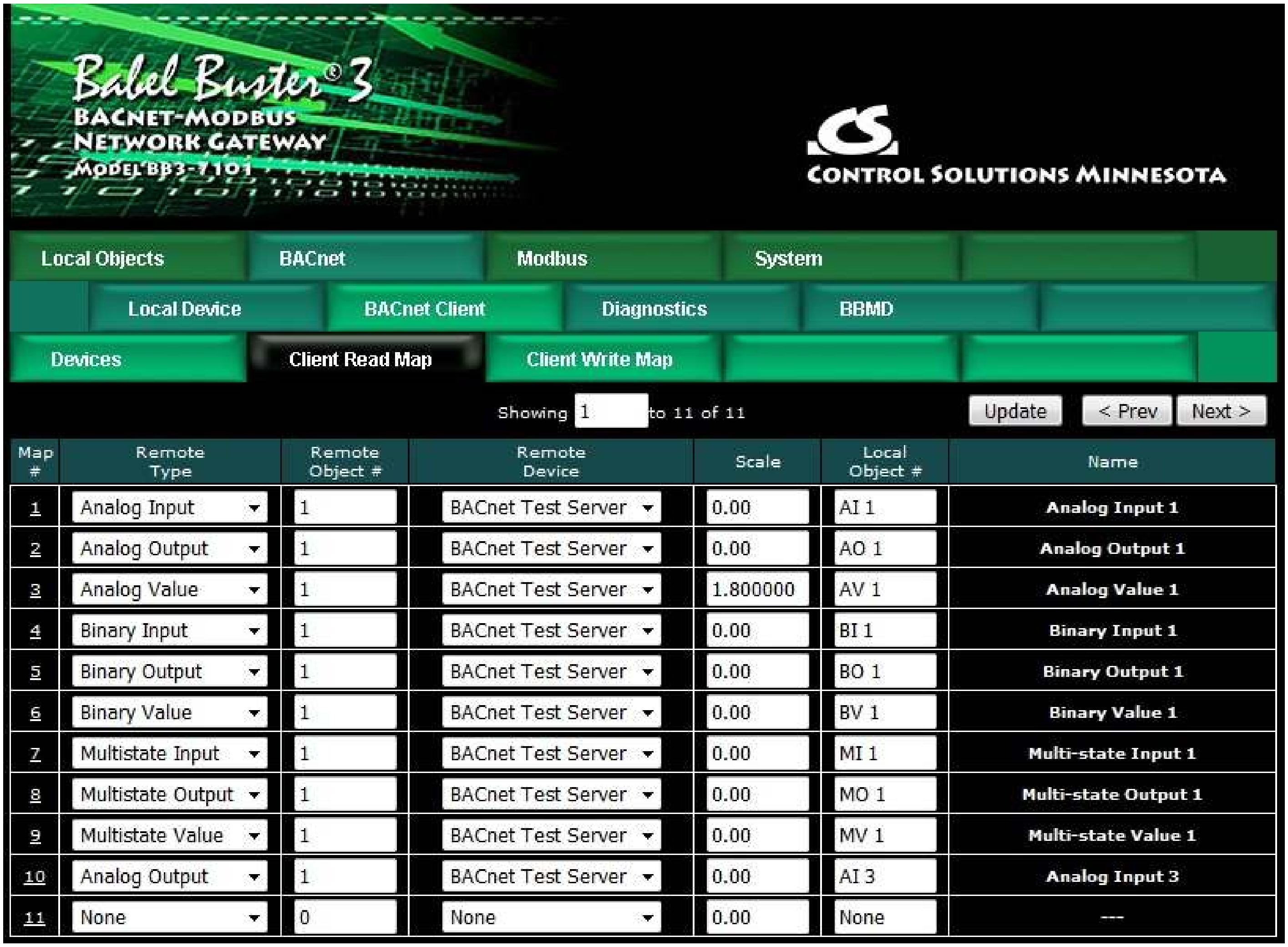
If static binding must be used, enter the fixed IP address you know the device instance to be found at. If no port is given, it will default to 0xBAC0 (47808). Enter IP as a.b.c.d or IP with port as a.b.c.d:p, for example 192.168.1.99:47808.

Include network number, mac length, and mac address ONLY if static binding to a device on the other side of a BACnet router.

When dynamic address binding is used (default), the gateway broadcasts a "Who-Is" looking for this device instance when a read or write map wants to use this device. When (if) it responds, its IP address is listed here simply as a diagnostic. Timeouts resulting from inability to reach this device are tabulated on this page as well, and may be cleared by clicking the Clear button. To cause the who-is process to be repeated, click Clear Cache. When dynamic binding is used, the IP address is read-only and any changes entered will be ignored.

5.2 BACnet Client Read Maps

Getting the gateway to read objects from another BACnet device requires setting up a "Read Map" as shown here.



Showing 1 to 11 of 11

Map #	Remote Type	Remote Object #	Remote Device	Scale	Local Object #	Name
1	Analog Input	1	BACnet Test Server	0.00	AI 1	Analog Input 1
2	Analog Output	1	BACnet Test Server	0.00	AO 1	Analog Output 1
3	Analog Value	1	BACnet Test Server	1.800000	AV 1	Analog Value 1
4	Binary Input	1	BACnet Test Server	0.00	BI 1	Binary Input 1
5	Binary Output	1	BACnet Test Server	0.00	BO 1	Binary Output 1
6	Binary Value	1	BACnet Test Server	0.00	BV 1	Binary Value 1
7	Multistate Input	1	BACnet Test Server	0.00	MI 1	Multi-state Input 1
8	Multistate Output	1	BACnet Test Server	0.00	MO 1	Multi-state Output 1
9	Multistate Value	1	BACnet Test Server	0.00	MV 1	Multi-state Value 1
10	Analog Output	1	BACnet Test Server	0.00	AI 3	Analog Input 3
11	None	0	None	0.00	None	---

Map number simply tells you where you're at on the list of object maps. Click "next" and "prev" to scroll through the list. To advance directly to a specific map, enter the desired number in the "Showing" box, then click Update.

Maps entered on this page only read data from remote devices. Go to the Client Write Map to write data to those devices. The full parameter set is different for read versus write.

An abbreviated version of a list of maps is shown on this page. Any of the parameters shown may be changed here and registered by clicking the Update button. To view and/or modify the complete set of parameters, click on the map number in the left most column.

For each remote object to be read, select the object instance and type, and remote device. The names in the remote device list are defined in the Devices page. The property read will default to Present Value. If you wish to read a different property, click on the Map # in the first column for the expanded view of the map and enter the property number.

When the remote object is read, data may be manipulated before being written to the local object. The value will be multiplied by the scale factor. The final result is written to the local object number given. The name is optional and used only for display purposes.

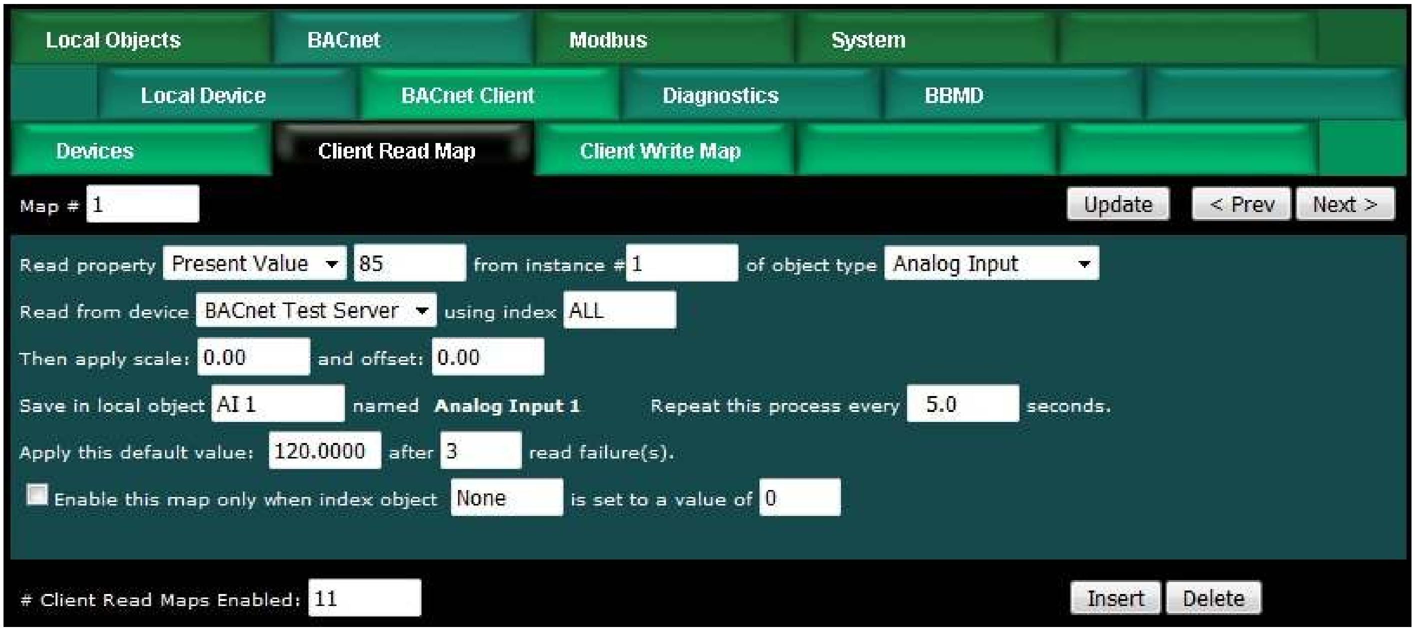
Selecting "none" for remote type effectively deletes the map even though it will still appear in the list until deleted. Unused maps at the end of the list will always show none as the type.

Local Object is internally a coded number consisting of BACnet object type multiplied by 10000, then added to the object number starting from #1. These numbers will appear as "register numbers" in XML configuration files. These are translated into abbreviations that are easy to interpret on the web page as follows:

AI n = Analog Input #n
AO n = Analog Output #n
AV n = Analog Value #n
BI n = Binary Input #n
BO n = Binary Output #n
BV n = Binary Value #n
MI n = Multi-state Input #n
MO n = Multi-state Output #n
MV n = Multi-state Value #n

Local object numbers start at #1. The maximum available number varies by object type, and these limits are set on the Resources page (under System).

Click on a Map # in the first column of maps to get the expanded view of that map as follows:



Map # 1

Read property Present Value 85 from instance # 1 of object type Analog Input

Read from device BACnet Test Server using index ALL

Then apply scale: 0.00 and offset: 0.00

Save in local object AI 1 named **Analog Input 1** Repeat this process every 5.0 seconds.

Apply this default value: 120.0000 after 3 read failure(s).

☐ Enable this map only when index object None is set to a value of 0

Client Read Maps Enabled: 11

Map number simply tells you where you're at on the list of object maps. Click "next" and "prev" to scroll through the list. To advance directly to a specific map, enter the desired number in the "Map #" box, then click Update.

For each remote object to be read, enter the property number, object instance and type, and select a remote device. The names in the device list are defined in the Devices page. Use index value of "ALL" if no index (use this by default if you do not have a known index number).

The most commonly read property will be Present Value, which is property number 85. For other property numbers, refer to Appendix D, BACnet Codes.

When the remote object is read, data may be manipulated before being written to the local object. The value will be multiplied by the scale factor, then the offset is added. The final result is written to the local object number given. The name is optional and used only for display purposes.

The periodic poll time ("Repeat this process") determines how often the remote object will be read. This number, if nonzero, will override the default poll time given in the Devices page for the remote device being read.

The default value will be stored into the local object after the given number of read failures if the fail count is non-zero. Setting the count to zero will disable the default, and the object will retain the most recent value obtained.

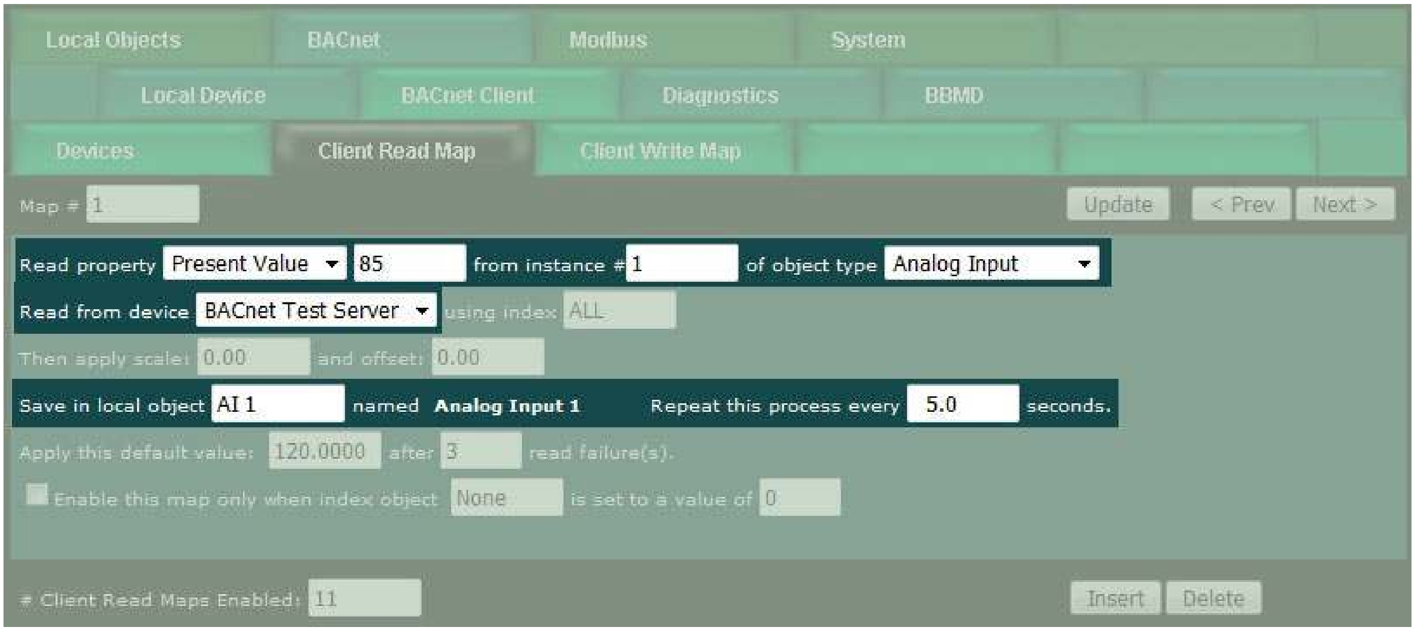
You have the option of enabling this map only when a selected object contains a given value. Any local object may be used as the index object. As the name implies, you could have the same local object contain different values based on different maps as indexed by the index object.

Delete will remove the map number shown in the "Map #" box. Insert will insert a new map before the map number shown, and is used for placing maps between existing maps. It is not necessary to use Insert to add maps to the bottom of the list or to define any map presently having zero for a source object or "none" for remote type.

Selecting "none" for remote type effectively deletes the map even though it will still appear in the list until deleted. Unused maps at the end of the list will always show none as the type. If you wish to prevent these from being displayed, reduce the number of maps enabled.

The number of maps enabled simply limits the scope of map review so that you do not have to review a lot of unused maps. If the displayed maps are used up and you need more, increase the enabled number.

The expanded view of the Client Read Map may look daunting at first. Here is the same screen shot with the parts you are most likely to not use grayed out for illustration purposes. If you are only concerned with reading Present Value and you have set the default poll time on the Device page, then you really never need to look at the expanded view at all. Your configuration can be entered entirely on the tabular list of maps. The same applies to Write Maps below.



Map # 1

Read property Present Value 85 from instance # 1 of object type Analog Input

Read from device BACnet Test Server using index ALL

Then apply scale: 0.00 and offset: 0.00

Save in local object AI 1 named **Analog Input 1** Repeat this process every 5.0 seconds.

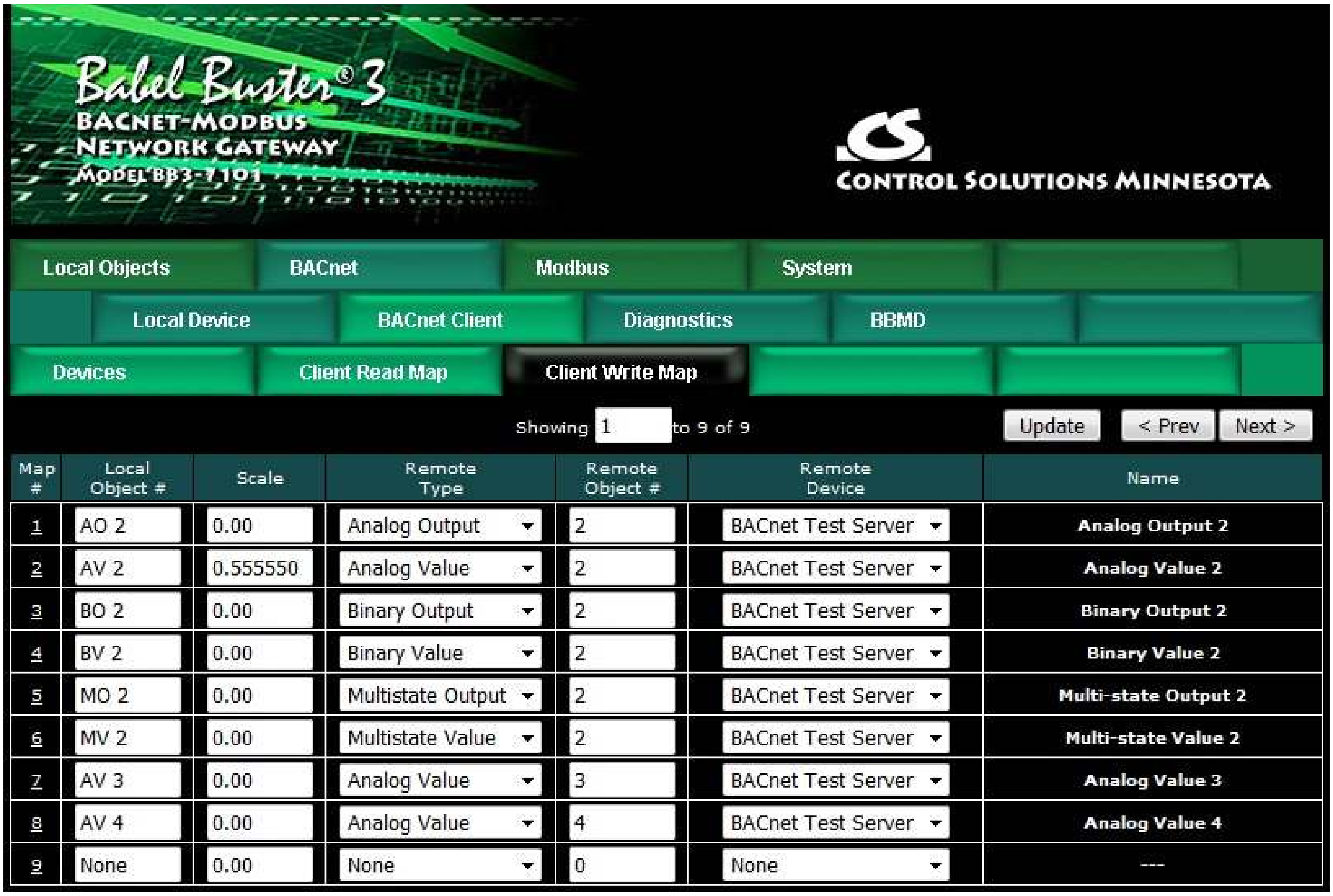
Apply this default value: 120.0000 after 3 read failure(s).

□ Enable this map only when index object None is set to a value of 0

Client Read Maps Enabled: 11

5.3 BACnet Client Write Maps

Getting the gateway to write objects to another BACnet device requires setting up a "Write Map" as shown here. Much of the Write Map is configured the same as a Read Map.



Showing 1 to 9 of 9

Map #	Local Object #	Scale	Remote Type	Remote Object #	Remote Device	Name
1	AO 2	0.00	Analog Output	2	BACnet Test Server	**Analog Output 2**
2	AV 2	0.555550	Analog Value	2	BACnet Test Server	**Analog Value 2**
3	BO 2	0.00	Binary Output	2	BACnet Test Server	**Binary Output 2**
4	BV 2	0.00	Binary Value	2	BACnet Test Server	**Binary Value 2**
5	MO 2	0.00	Multistate Output	2	BACnet Test Server	**Multi-state Output 2**
6	MV 2	0.00	Multistate Value	2	BACnet Test Server	**Multi-state Value 2**
7	AV 3	0.00	Analog Value	3	BACnet Test Server	**Analog Value 3**
8	AV 4	0.00	Analog Value	4	BACnet Test Server	**Analog Value 4**
9	None	0.00	None	0	None	---

Map number simply tells you where you're at on the list of object maps. Click "next" and "prev" to scroll through the list. To advance directly to a specific map, enter the desired number in the "Showing" box, then click Update.

Maps entered on this page only write data to remote devices. Go to the Client Read Map to read data from those devices. The full parameter set is different for read versus write.

An abbreviated version of a list of maps is shown on this page. Any of the parameters shown may be changed here and registered by clicking the Update button. To view and/or modify the complete set of parameters, click on the map number in the left most column.

For each remote object to be written, select the object instance and type, and remote device. The names in the device list are defined in the Devices page. The property written will default to Present Value. If you wish to write a different property, click on the Map # in the first column for the expanded view of the map and enter the property number.

Data from the local object given will be multiplied by the scale factor before being written.

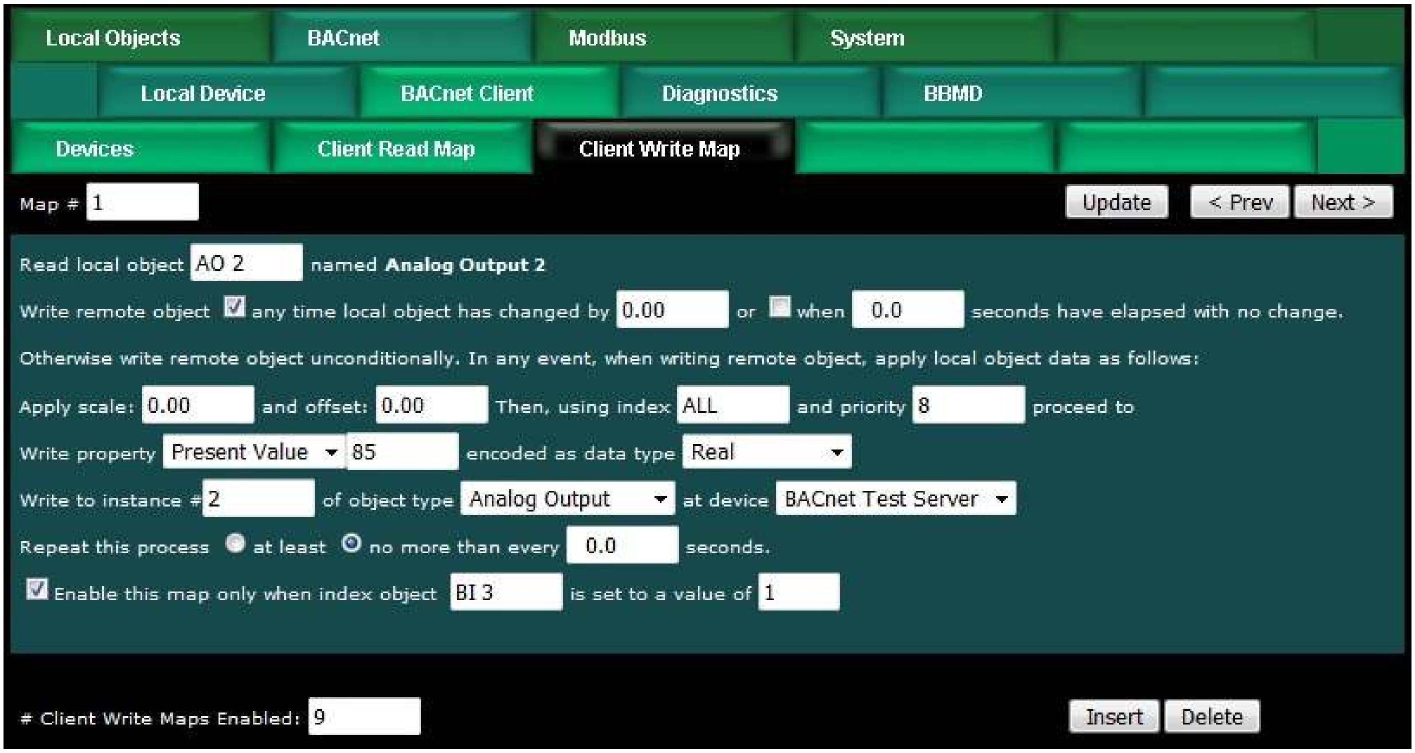
Selecting "none" for remote type effectively deletes the map even though it will still appear in the list until deleted. Unused maps at the end of the list will always show none as the type.

Local Object is internally a coded number consisting of BACnet object type multiplied by 10000, then added to the object number starting from #1. These numbers will appear as "register numbers" in XML configuration files. These are translated into abbreviations that are easy to interpret on the web page as follows:

AI n = Analog Input #n
AO n = Analog Output #n
AV n = Analog Value #n
BI n = Binary Input #n
BO n = Binary Output #n
BV n = Binary Value #n
MI n = Multi-state Input #n
MO n = Multi-state Output #n
MV n = Multi-state Value #n

Object numbers start at #1. The maximum available number varies by object type, and these limits are set on the Resources page (under System).

Click on a Map # in the first column of maps to get the expanded view of that map as follows:



Map # 1

Read local object AO 2 named **Analog Output 2**

Write remote object ☑ any time local object has changed by 0.00 or □ when 0.0 seconds have elapsed with no change.

Otherwise write remote object unconditionally. In any event, when writing remote object, apply local object data as follows:

Apply scale: 0.00 and offset: 0.00 Then, using index ALL and priority 8 proceed to

Write property Present Value 85 encoded as data type Real

Write to instance # 2 of object type Analog Output at device BACnet Test Server

Repeat this process ○ at least ● no more than every 0.0 seconds.

☑ Enable this map only when index object BI 3 is set to a value of 1

Client Write Maps Enabled: 9

Map number simply tells you where you're at on the list of object maps. Click "next" and "prev" to scroll through the list. To advance directly to a specific map, enter the desired number in the "Map #" box, then click Update.

The local object data may be written periodically, or when it changes, or both. To send upon change (send on delta), check the first box and enter the amount by which the local object must change before being written to the remote device. To guarantee that the remote object will be written at least occasionally even if the data does not change, check the second box and enter some amount of time. This time period will be referred to as the "maximum quiet time".

Data from the local object may be manipulated before being written to the remote object. The local data is first multiplied by the scale factor. The offset is then added to it.

For the remote object to be written, enter the property number, object instance and type, index if applicable (enter "ALL" if no index), and priority to use of the object being written is commandable. Select a remote device to write to. The names in the device list are defined in the Devices page.

The most commonly written property will be Present Value, which is property number 85. For other property numbers, refer to Appendix D, BACnet Codes.

The repeat time may determine how often the remote object will be written. If send on delta and maximum quiet time are not checked above, clicking the "at least" button will establish a periodic update time. If send on delta is used and you wish to limit the network traffic in the event changes are frequent, click the "no more than" button and enter the minumum time that should elapse before another write to the remote device.

You have the option of enabling this map only when a selected object contains a given value. Any local object may be used as the index object. As the name implies, you can write different values to the remote object based on different maps as indexed by the index object.

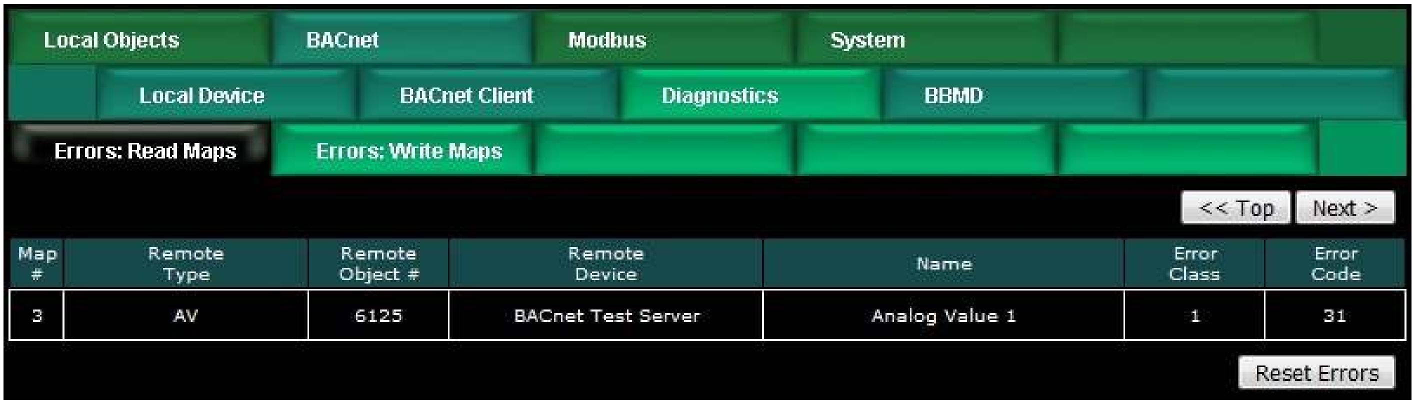
Delete will remove the map number shown in the "Map #" box. Insert will insert a new map before the map number shown, and is used for placing maps between existing maps. It is not necessary to use Insert to add maps to the bottom of the list or to define any map presently having zero for a source object or "none" for remote type.

Selecting "none" for remote type effectively deletes the map even though it will still appear in the list until deleted. Unused maps at the end of the list will always show none as the type. If you wish to prevent these from being displayed, reduce the number of maps enabled.

The number of maps enabled simply limits the scope of map review so that you do not have to review a lot of unused maps. If the displayed maps are used up and you need more, increase the enabled number.

5.4 BACnet Client Diagnostics

If errors are detected in the course of reading or writing other BACnet objects via the client's maps, they will be indicated on the errors pages.



Map #	Remote Type	Remote Object #	Remote Device	Name	Error Class	Error Code
3	AV	6125	BACnet Test Server	Analog Value 1	1	31

Errors for BACnet client read/write maps are shown on these pages. Only those maps with errors to report are listed. Refer to the code and class lists below for interpretation. In the illustration above, error class 1 says the error refers to "object" and the code says "unknown object". In other words, AV 6125 does not exist in the device shown.

Proprietary class 82, code 0, is generated locally indicating a timeout, no response received from remote server. All other codes listed below are returned by the remote server.

0 = ERROR_CLASS_DEVICE
1 = ERROR_CLASS_OBJECT
2 = ERROR_CLASS_PROPERTY
3 = ERROR_CLASS_RESOURCES
4 = ERROR_CLASS_SECURITY

```
5 = ERROR_CLASS_SERVICES

/* valid for all classes */
0 = ERROR_CODE_OTHER

/* Error Class - Device */
2 = ERROR_CODE_CONFIGURATION_IN_PROGRESS
3 = ERROR_CODE_DEVICE_BUSY
25 = ERROR_CODE_OPERATIONAL_PROBLEM

/* Error Class - Object */
4 = ERROR_CODE_DYNAMIC_CREATION_NOT_SUPPORTED
17 = ERROR_CODE_NO_OBJECTS_OF_SPECIFIED_TYPE
23 = ERROR_CODE_OBJECT_DELETION_NOT_PERMITTED
24 = ERROR_CODE_OBJECT_IDENTIFIER_ALREADY_EXISTS
27 = ERROR_CODE_READ_ACCESS_DENIED
31 = ERROR_CODE_UNKNOWN_OBJECT
36 = ERROR_CODE_UNSUPPORTED_OBJECT_TYPE

/* Error Class - Property */
8 = ERROR_CODE_INCONSISTENT_SELECTION_CRITERION
9 = ERROR_CODE_INVALID_DATA_TYPE
32 = ERROR_CODE_UNKNOWN_PROPERTY
37 = ERROR_CODE_VALUE_OUT_OF_RANGE
40 = ERROR_CODE_WRITE_ACCESS_DENIED
41 = ERROR_CODE_CHARACTER_SET_NOT_SUPPORTED
42 = ERROR_CODE_INVALID_ARRAY_INDEX
44 = ERROR_CODE_NOT_COV_PROPERTY
45 = ERROR_CODE_OPTIONAL_FUNCTIONALITY_NOT_SUPPORTED
47 = ERROR_CODE_DATATYPE_NOT_SUPPORTED
50 = ERROR_CODE_PROPERTY_IS_NOT_AN_ARRAY

/* Error Class - Resources */
18 = ERROR_CODE_NO_SPACE_FOR_OBJECT
19 = ERROR_CODE_NO_SPACE_TO_ADD_LIST_ELEMENT
20 = ERROR_CODE_NO_SPACE_TO_WRITE_PROPERTY

/* Error Class - Security */
1 = ERROR_CODE_AUTHENTICATION_FAILED
6 = ERROR_CODE_INCOMPATIBLE_SECURITY_LEVELS
12 = ERROR_CODE_INVALID_OPERATOR_NAME
15 = ERROR_CODE_KEY_GENERATION_ERROR
26 = ERROR_CODE_PASSWORD_FAILURE
28 = ERROR_CODE_SECURITY_NOT_SUPPORTED
30 = ERROR_CODE_TIMEOUT

/* Error Class - Services */
5 = ERROR_CODE_FILE_ACCESS_DENIED
7 = ERROR_CODE_INCONSISTENT_PARAMETERS
```

10 = ERROR_CODE_INVALID_FILE_ACCESS_METHOD
11 = ERROR_CODE_ERROR_CODE_INVALID_FILE_START_POSITION
13 = ERROR_CODE_INVALID_PARAMETER_DATA_TYPE
14 = ERROR_CODE_INVALID_TIME_STAMP
16 = ERROR_CODE_MISSING_REQUIRED_PARAMETER
22 = ERROR_CODE_PROPERTY_IS_NOT_A_LIST
29 = ERROR_CODE_SERVICE_REQUEST_DENIED
43 = ERROR_CODE_COV_SUBSCRIPTION_FAILED
46 = ERROR_CODE_INVALID_CONFIGURATION_DATA
48 = ERROR_CODE_DUPLICATE_NAME
49 = ERROR_CODE_DUPLICATE_OBJECT_ID

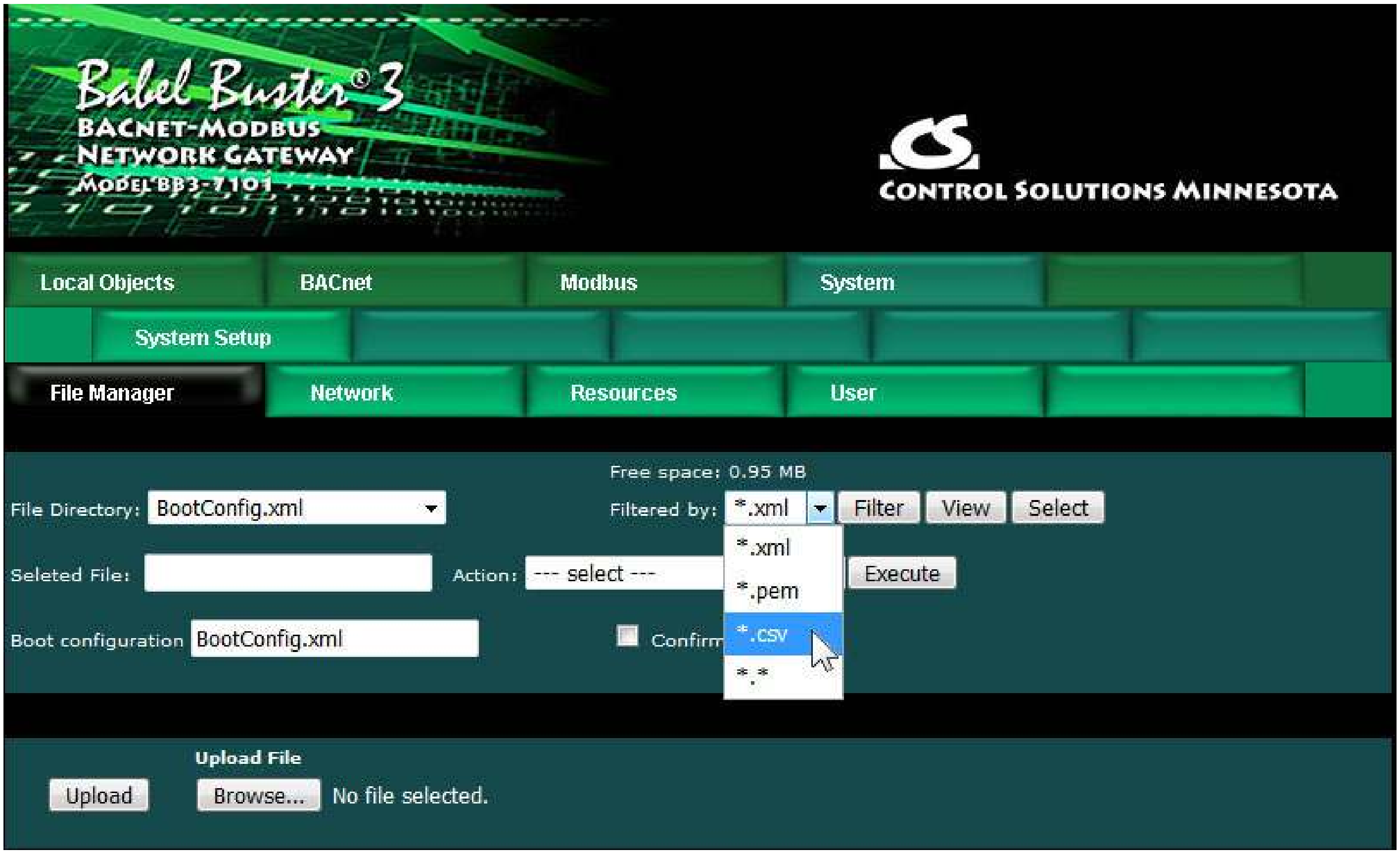
5.5 Importing BACnet Client Maps from CSV File

The built-in web user interface is user friendly, but can get tedious if you have a lot of maps to enter. You may already have a list of BACnet objects available in spread sheet form. With a bit of editing, you can turn this into a CSV file that can be directly imported into the BB3-7101/MX-71 to quickly configure a lot of read and write maps. If you are proficient with spread sheets, you can probably create a rather large set of maps quickly and speed up the process of configuring the BB3-7101/MX-71.

There is more discussion about the File Manager in Section 11, but a summary of what you need to do to import maps from a CSV file is given here.

Start by uploading your CSV file. Use the Browse button to locate the file on your PC, then click Upload.

Select *.csv as the file filter. This will result in showing the list of CSV files currently stored in the BB3-7101/MX-71.



Free space: 0.95 MB

File Directory: BootConfig.xml

Filtered by: *.xml

Seleted File:

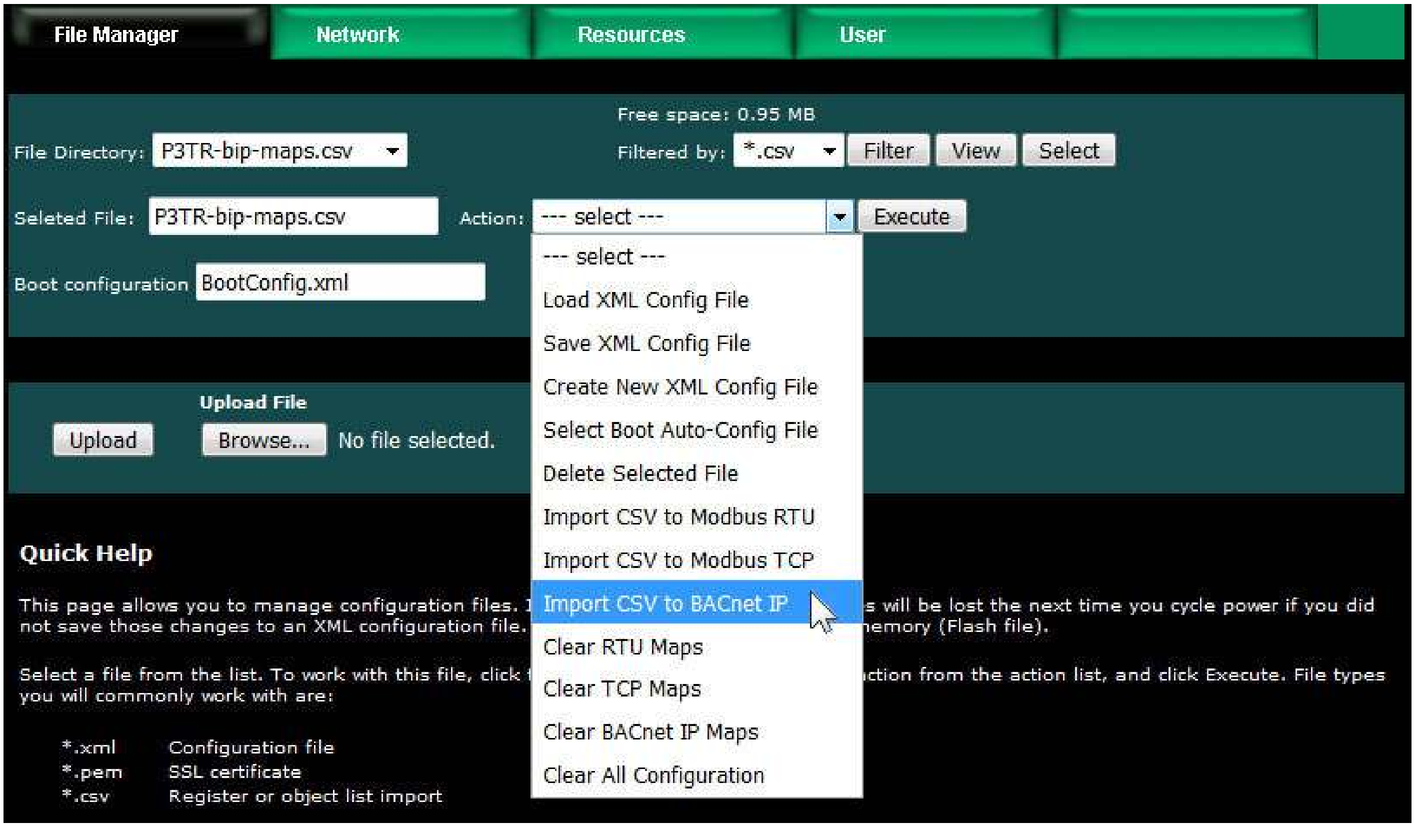
Action: --- select ---

Boot configuration BootConfig.xml

□ Confirm

No file selected.

Select your file from the File Directory drop-down list, then click the Select button on the right. Select "Import CSV to BACnet IP" from the Action list, and click Execute.



Free space: 0.95 MB

File Directory: P3TR-bip-maps.csv

Filtered by: *.csv

Seleted File: P3TR-bip-maps.csv

Action: --- select ---

Boot configuration BootConfig.xml

No file selected.

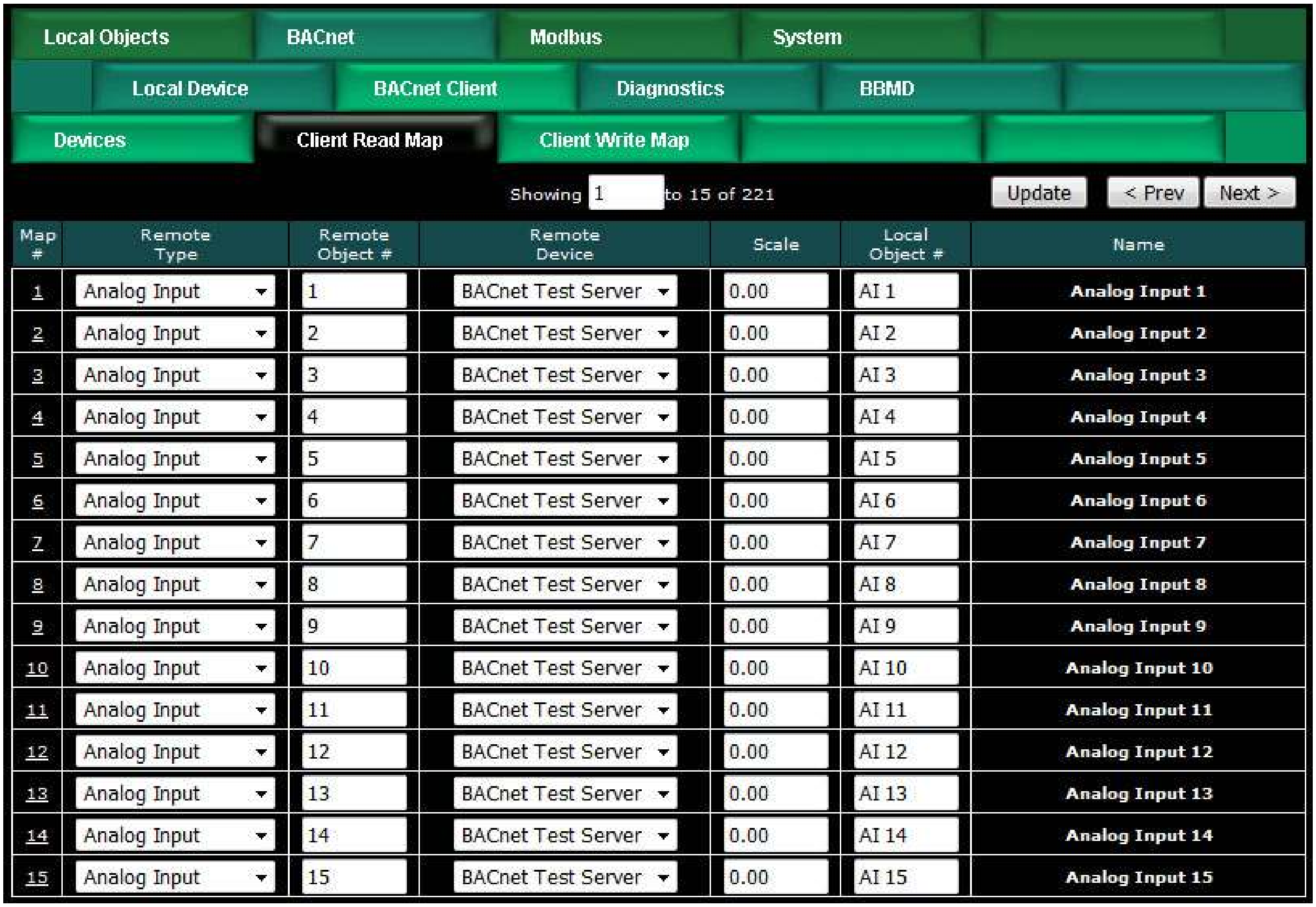
Quick Help

This page allows you to manage configuration files. I s will be lost the next time you cycle power if you did not save those changes to an XML configuration file. emory (Flash file).

Select a file from the list. To work with this file, click t ction from the action list, and click Execute. File types you will commonly work with are:

*.xml	Configuration file
*.pem	SSL certificate
*.csv	Register or object list import

We imported 220 read maps in our example test case.

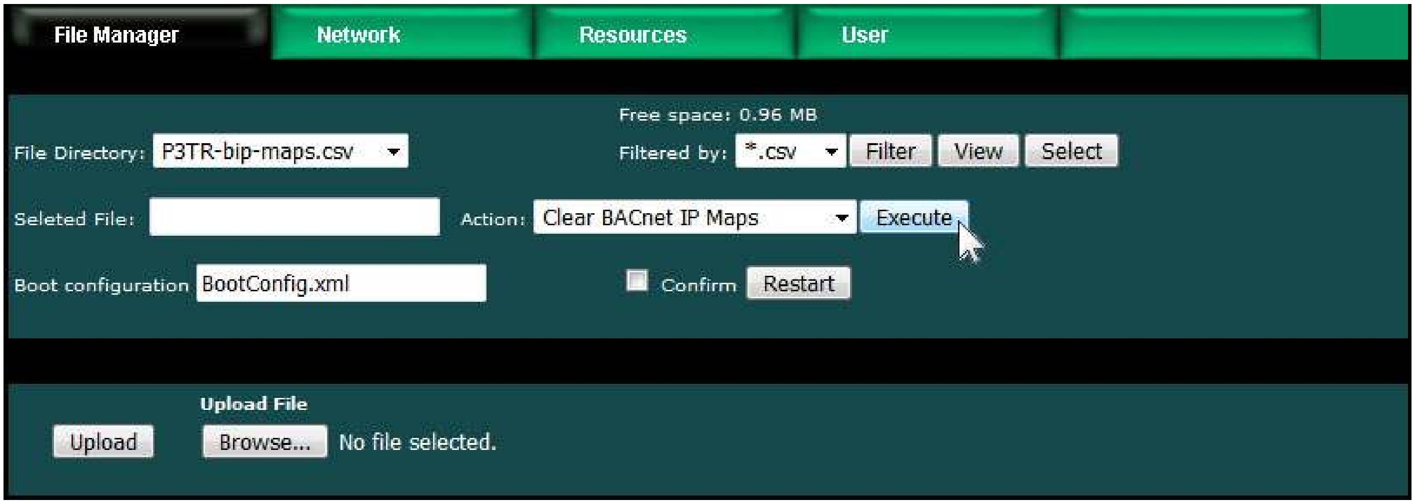


Showing 1 to 15 of 221

Map #	Remote Type	Remote Object #	Remote Device	Scale	Local Object #	Name
1	Analog Input	1	BACnet Test Server	0.00	AI 1	Analog Input 1
2	Analog Input	2	BACnet Test Server	0.00	AI 2	Analog Input 2
3	Analog Input	3	BACnet Test Server	0.00	AI 3	Analog Input 3
4	Analog Input	4	BACnet Test Server	0.00	AI 4	Analog Input 4
5	Analog Input	5	BACnet Test Server	0.00	AI 5	Analog Input 5
6	Analog Input	6	BACnet Test Server	0.00	AI 6	Analog Input 6
7	Analog Input	7	BACnet Test Server	0.00	AI 7	Analog Input 7
8	Analog Input	8	BACnet Test Server	0.00	AI 8	Analog Input 8
9	Analog Input	9	BACnet Test Server	0.00	AI 9	Analog Input 9
10	Analog Input	10	BACnet Test Server	0.00	AI 10	Analog Input 10
11	Analog Input	11	BACnet Test Server	0.00	AI 11	Analog Input 11
12	Analog Input	12	BACnet Test Server	0.00	AI 12	Analog Input 12
13	Analog Input	13	BACnet Test Server	0.00	AI 13	Analog Input 13
14	Analog Input	14	BACnet Test Server	0.00	AI 14	Analog Input 14
15	Analog Input	15	BACnet Test Server	0.00	AI 15	Analog Input 15

5.6 Clearing Configuration

Read and write maps imported from a CSV file will be added to the list of maps already in place. If you wish to reload the list, you must first clear it. Clear the BACnet client maps by going to the File Manager page, then selecting "Clear BACnet IP Maps" from the action list and clicking Execute.



Free space: 0.96 MB

File Directory: P3TR-bip-maps.csv

Filtered by: *.csv

Seleted File:

Action: Clear BACnet IP Maps

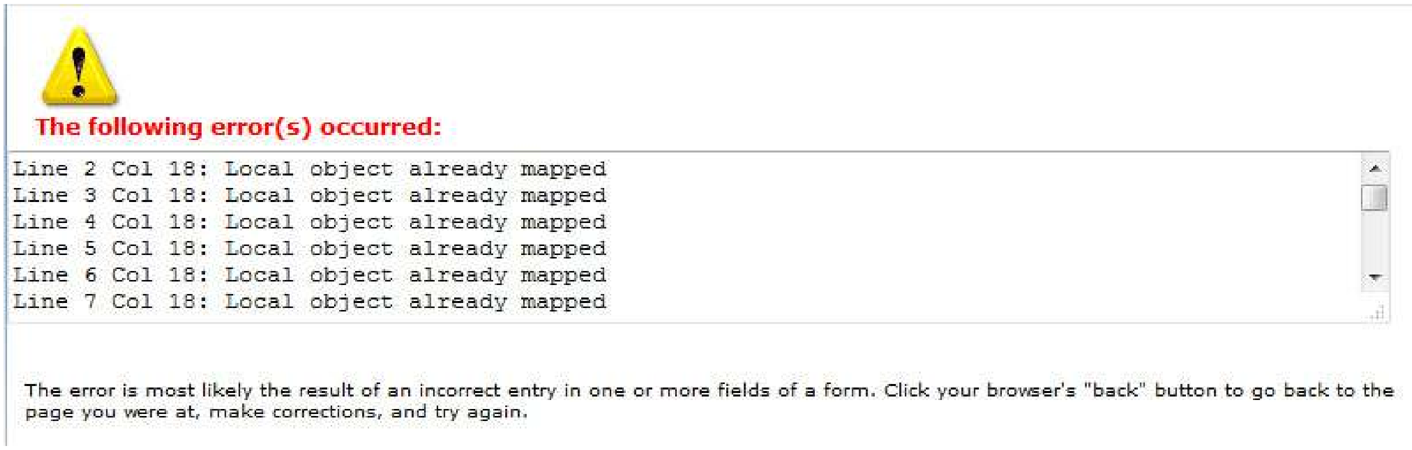
Boot configuration BootConfig.xml

□ Confirm

Upload File

No file selected.

If you forget to clear the maps before re-importing them, you will get an error notice something like this:



The following error(s) occurred:

```
Line 2 Col 18: Local object already mapped
Line 3 Col 18: Local object already mapped
Line 4 Col 18: Local object already mapped
Line 5 Col 18: Local object already mapped
Line 6 Col 18: Local object already mapped
Line 7 Col 18: Local object already mapped
```

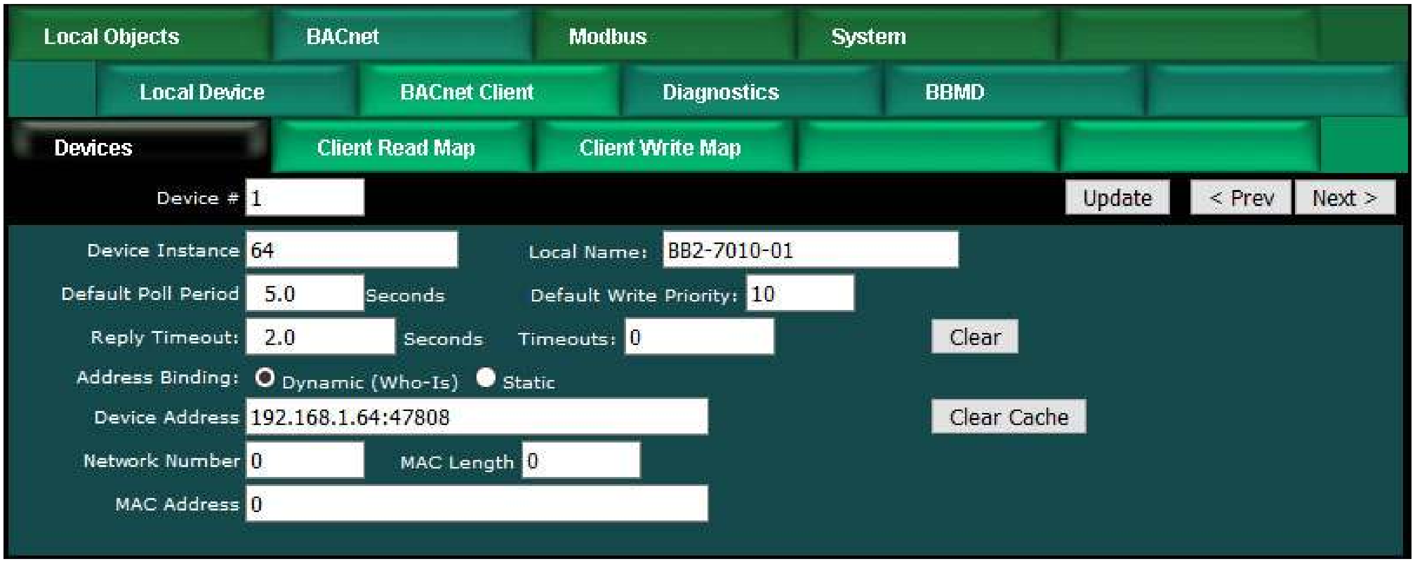
The error is most likely the result of an incorrect entry in one or more fields of a form. Click your browser's "back" button to go back to the page you were at, make corrections, and try again.

5.7 Understanding BACnet Client Timeout Settings

The Babel Buster gateway includes a BACnet client that can be configured to read and write objects in other BACnet devices. Each point to be read or written is defined by a client read map or client write map. These maps reference a device, and each device has an entry on the Devices page. For each device, the BACnet Device Instance, a name, default poll period, and timeout are provided by the user.

The Reply Timeout is the amount of time the client will wait for a response before calling it a timeout if no response is received. The client will then move on to the next read or write map. The client will eventually come back around to the same point and try again. If the client times out a second time, then the mapped object’s reliability code will be set to the nonzero value indicating timeout, no response.

If repeated timeouts are observed, one should confirm that the device in question is operating. If so, then set a longer timeout period as needed.



Device # 1

Device Instance 64 Local Name: BB2-7010-01

Default Poll Period 5.0 Seconds Default Write Priority: 10

Reply Timeout: 2.0 Seconds Timeouts: 0

Address Binding: ○ Dynamic (Who-Is) ● Static

Device Address 192.168.1.64:47808

Network Number 0 MAC Length 0

MAC Address 0

Much of the time, especially with BACnet IP, the client device timeout is the only timeout one needs to be pay much attention to. However, it is important to understand what is going on inside BACnet behind the scenes, especially if the client is making requests to an MS/TP device on the other side of a router.

The BACnet Device has settings that apply to all requests made by this device, and these are found on the BACnet settings page (illustrated below). Of particular concern with respect to timeouts are the APDU Timeout and APDU Retries.

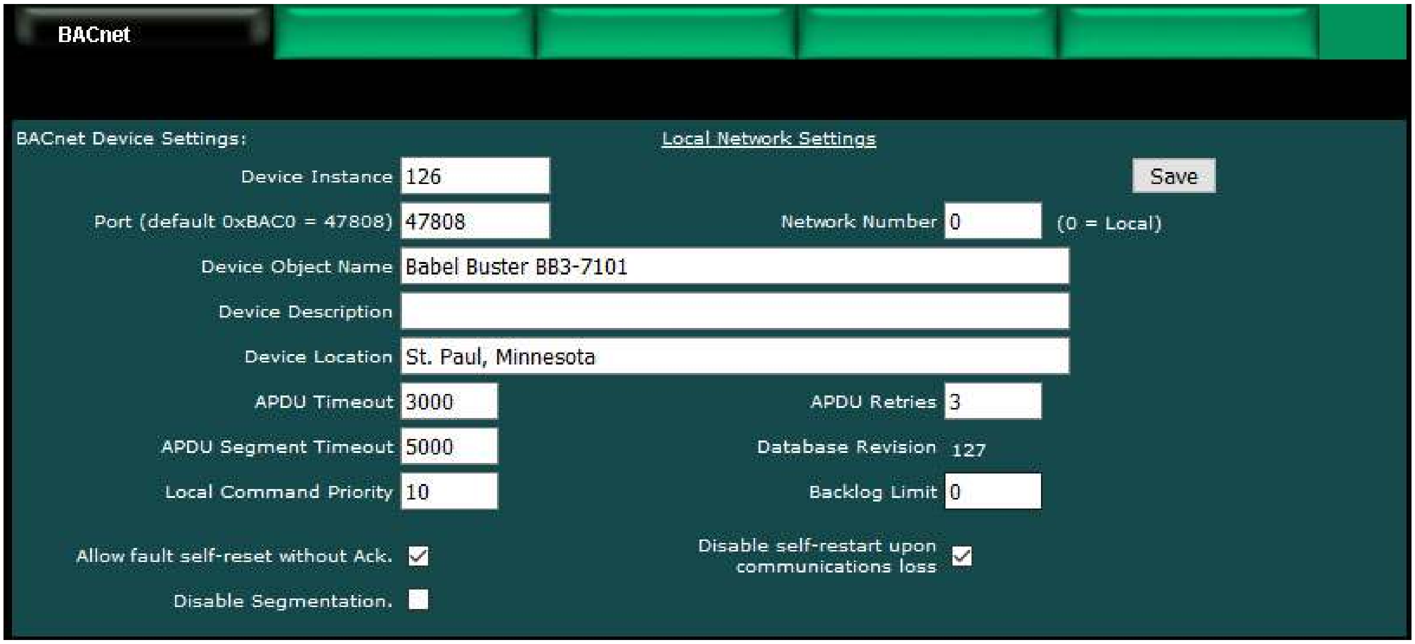
Any time a request is made by this BACnet device, the request initiates a Transaction State Machine (TSM). The Invoke ID you will see if you use Wireshark to look at network traffic identifies this TSM. This invoke ID is used to associate a reply with a request. If the TSM does not see a reply within the APDU Timeout (given in milliseconds), then the TSM will automatically retransmit the request and wait another APDU Timeout period. This retransmission will be repeated up to the retry count. If the retry count is 3 as illustrated below (with a timeout of 3000 milliseconds), and no reply is ever received, this means the request will have been transmitted a total of 4 times (over 12 seconds).

The APDU Timeout will default to 3000 (milliseconds) and APDU Retries will default to 3 as recommended by BACnet protocol. However, whether these numbers make sense for your application is left for you to determine.

It makes sense to have the BACnet client device timeout set to 2 seconds as illustrated above so that a timeout doesn't hang up the client for too long. However, if the default values for the BACnet Device are left as illustrated below, then here is what is going to happen when the target device does not reply: The client will send its initial request. Then 2 seconds later, it decides there is no response and moves on to the next point and sends the next request. Meanwhile the TSM has waited 3 seconds, then retransmitted the original request. Over a period of 12 seconds, the TSM will be sending the same original first request 4 times. As you can see, the client has not really waited for the final result in this instance. Furthermore, the client is kicking off more TSM's faster than they can complete their retry sequence. If the target device is a slow MS/TP device on the other side of a router, it is highly likely that you will flood the router with more requests than it can handle, and you will start to get "reject - router busy" replies from the router (which will be indicated simply as timeout on the client end).

This snowball effect and request log jam will often clear itself when the BACnet client is only polling BACnet IP devices. But the snowball effect can have very adverse effects on a slow MS/TP network on the other side of a router.

When choosing a timeout value for the devices listed on the Devices page in the BACnet client, be sure to also examine the APDU Timeout and Retries on the BACnet settings page. If the default values illustrated below are left as is, then the most suitable timeout value for the client device above would be 12 seconds, not 2 seconds.



BACnet Device Settings: Local Network Settings

Device Instance 126

Port (default 0xBAC0 = 47808) 47808

Network Number 0 (0 = Local)

Device Object Name Babel Buster BB3-7101

Device Description

Device Location St. Paul, Minnesota

APDU Timeout 3000

APDU Retries 3

APDU Segment Timeout 5000

Database Revision 127

Local Command Priority 10

Backlog Limit 0

Allow fault self-reset without Ack. ☑

Disable self-restart upon communications loss ☑

Disable Segmentation. □

The other setting one should pay attention to especially when talking to MS/TP devices on the other side of a router is the "Backlog Limit". The BACnet client will not necessarily wait for the reply from device A before sending a request to device B. If the client is polling 20 different devices, it is quite plausible that the client will send 20 requests faster than the first reply can come back. Thus it is quite easy for a BACnet IP client to overrun an MS/TP router by pumping out requests faster than the router can forward them to MS/TP. Therefore, one should use the Backlog Limit to throttle the client. If the limit is set to 4 (a reasonable number for MS/TP), this means the client will send no more than 4 requests before pausing and waiting for replies to those requests.

The other aspect of how Backlog Limit may affect required timeout setting is that when there is a large backlog of replies to process, the remote device may have responded promptly and within the client timeout setting, but by the time the client gets through the backlog of replies, a short client timeout may have expired. Therefore, timeout is not just a matter of how fast the other device responds, but also a matter of how busy you are keeping the client.

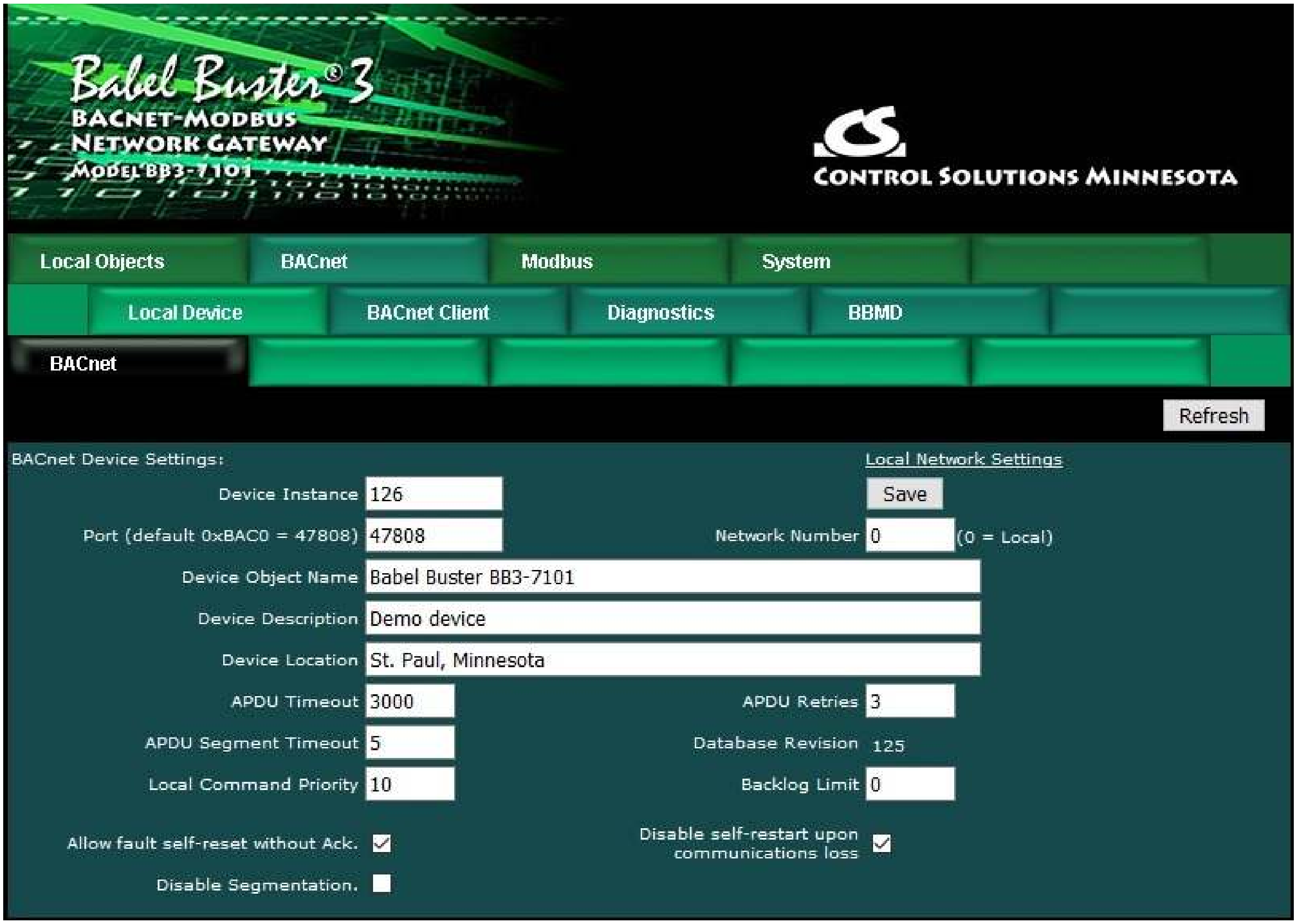
Babel Buster® 3
MODEL BB3-7101-SP
MODEL MX-71-SP
NETWORK GATEWAY
CONTROL SOLUTIONS MINNESOTA

6. Configuring Gateway as a BACnet Server

6.1 Server Configuration

The BB3-7101/MX-71 contains a set of BACnet objects whose only purpose is to store copies of data obtained from other devices. This copy of data may then be queried by different devices.

The only configuration needed to use the BB3-7101/MX-71 as a BACnet server is to set the Device instance on the BACnet page. The device should also be given an object name that will be unique on the entire network. Configuring the gateway as a BACnet Device is described in more detail in Section 4.



BACnet Device Settings:

Device Instance 126
Port (default 0xBAC0 = 47808) 47808
Network Number 0 (0 = Local)
Device Object Name Babel Buster BB3-7101
Device Description Demo device
Device Location St. Paul, Minnesota
APDU Timeout 3000
APDU Retries 3
APDU Segment Timeout 5
Database Revision 125
Local Command Priority 10
Backlog Limit 0
Allow fault self-reset without Ack. ☑
Disable self-restart upon communications loss ☑
Disable Segmentation. □

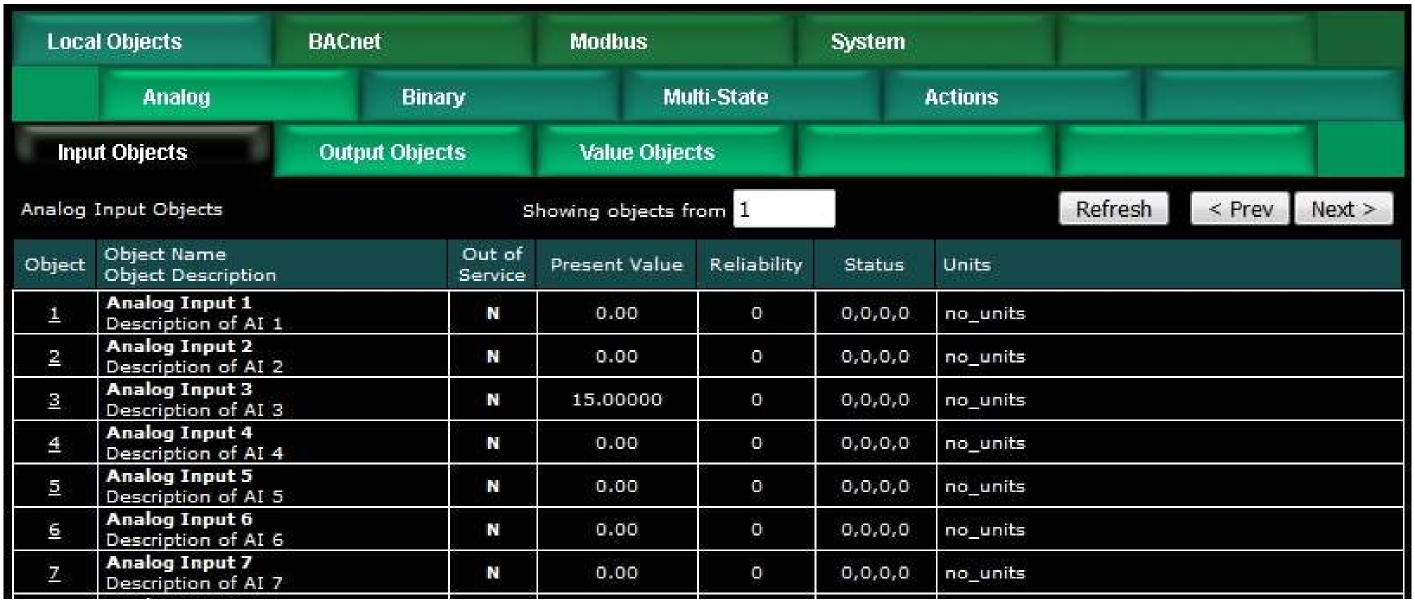
6.2 Accessing Local Objects

The collection of local objects includes Analog, Binary, and Multi-State types of objects, and includes Input, (commandable) Output, and (writeable) Value types of each of those objects. The BB3-7101/MX-71 also contains a Device object which is configured in the above screen.

Data may be placed in the local objects by other devices writing to the BB3-7101/MX-71, or by the BB3-7101/MX-71 querying other devices. When the BB3-7101/MX-71 is configured to query other devices, these operations are defined by “read maps” and “write maps” associated with the respective client function (BACnet client or Modbus client/master).

The following pages illustrate the Analog Input object pages and the Binary Output object pages. The remaining object pages found in the BB3-7101/MX-71 are virtually identical, and are not replicated here. (See also Configuring Local Objects, Section 3.)

Each object page initially comes up as a table of object data. Click on the object number in the left-hand column to expand the view of that object and access the windows that let you locally force values, assign units or names, etc.



Analog Input Objects

Showing objects from 1

Object	Object Name Object Description	Out of Service	Present Value	Reliability	Status	Units
1	**Analog Input 1** Description of AI 1	N	0.00	0	0,0,0,0	no_units
2	**Analog Input 2** Description of AI 2	N	0.00	0	0,0,0,0	no_units
3	**Analog Input 3** Description of AI 3	N	15.00000	0	0,0,0,0	no_units
4	**Analog Input 4** Description of AI 4	N	0.00	0	0,0,0,0	no_units
5	**Analog Input 5** Description of AI 5	N	0.00	0	0,0,0,0	no_units
6	**Analog Input 6** Description of AI 6	N	0.00	0	0,0,0,0	no_units
7	**Analog Input 7** Description of AI 7	N	0.00	0	0,0,0,0	no_units

The object name, units, value, and status are shown for a list of objects starting with the number entered at the top of the page. Click Prev/Next to scroll through the list. Click on the object number in the first column to change name, units, COV, and out-of-service state.

The source of data for an Analog Input object will typically be reading from some other BACnet or Modbus device. Click on the object number in the first column for more detail including the link to any client map providing data to this object.

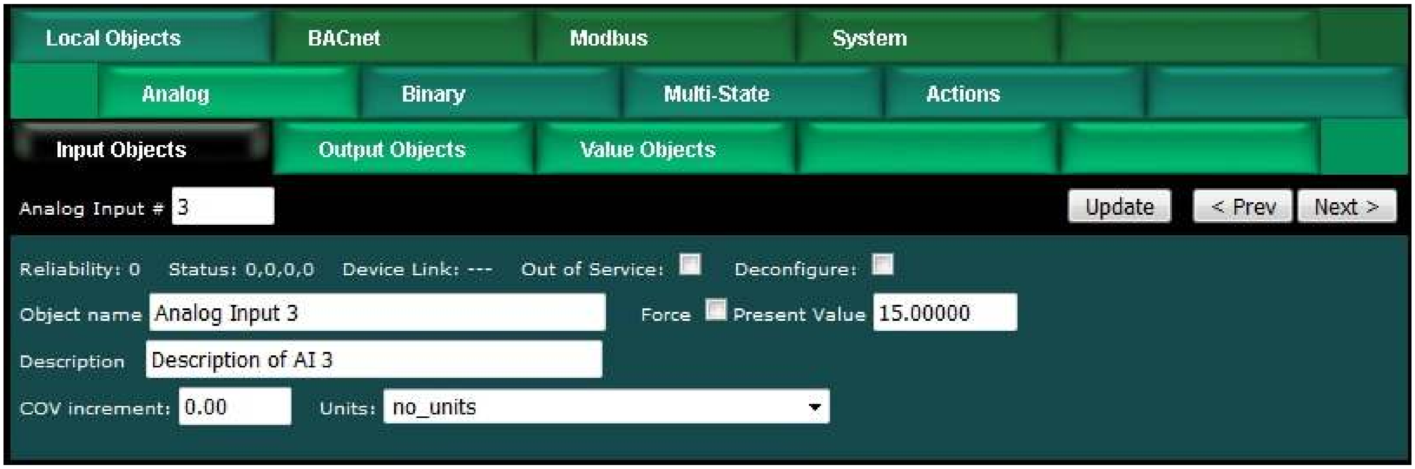
Out of Service means polling for data will stop. While out of service, the present value may be written by the BACnet client. Data may be forced via this web page at any time, but will be overwritten by the next mapped client update unless the object is out of service.

Reliability codes may be any of the following:

Modbus client/master, no response from slave (64)
Modbus client/master, crc error (65)
Modbus exception, illegal function code (66)
Modbus exception, illegal data address (67)
Modbus exception, illegal data value (68)
Modbus exception, code+65, rarely used (69..79)
Local device, configuration property fault (80)
Faulty Modbus packet(81)
BACnet IP client, device timeout (82)
BACnet IP client, error returned by server (83)

Status flags A,B,C,D indicate the following, 0 meaning not true, 1 meaning true:
A = in alarm
B = fault
C = overridden
D = out of service

Click on an Object number in the first column of maps to get the expanded view of that object as follows:



Analog Input # 3

Reliability: 0 Status: 0,0,0,0 Device Link: --- Out of Service: □ Deconfigure: □

Object name Analog Input 3 Force □ Present Value 15.00000

Description Description of AI 3

COV increment: 0.00 Units: no_units

The object name, units, value, and status are shown for the object number entered at the top of the page. Click Prev/Next to scroll through the list. Click Refresh to update the page, or Update to accept changes.

The object name and description may be changed here. BACnet units may be selected. Initial COV increment may be entered. When any of these are changed, be sure to save the updated configuration by clicking Save on the Config File page under System Setup.

The object may be set Out of Service by checking that box and clicking Update. The present value may be changed by entering a value, checking Force, and clicking Update.

The source of data for an Analog Input object will typically be reading from some other BACnet or Modbus device via the map indicated by the Device Link. The mapped device will be polled at the rate specified by the Read Map.

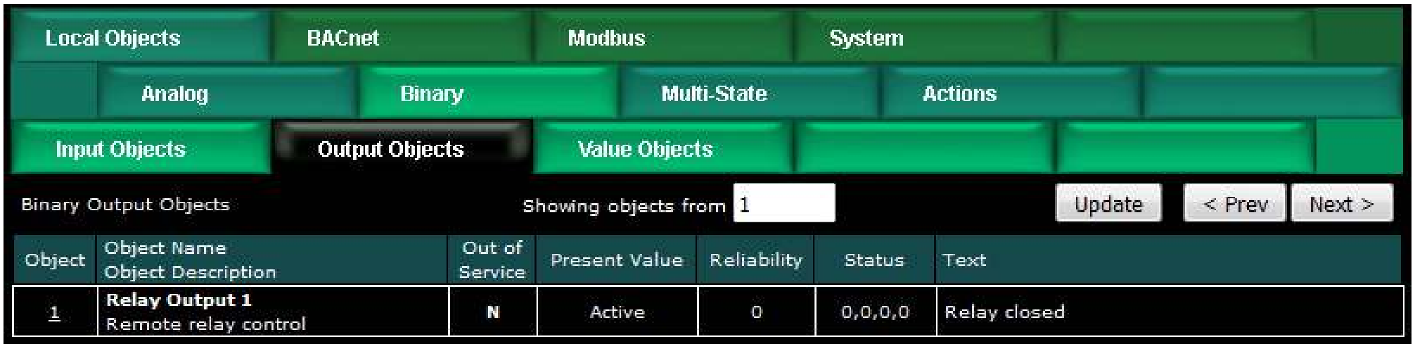
Out of Service means polling of the mapped remote device will stop. While out of service, the present value may be written by the BACnet client. Data may be forced via this web page at any time, but will be overwritten by the next mapped client update unless the object is out of service.

Reliability codes indicate errors as itemized on the tabular object list.

Status flags A,B,C,D indicate the following, 0 meaning not true, 1 meaning true:
A = in alarm
B = fault
C = overridden
D = out of service

Device link will indicate BIP, RTU, or TCP, followed by R for read or W for write, and a number which is the map number in the table of read or write maps for mapping to this BACnet object. The designation R means read from a remote device, and W means write to a remote device.

Check 'Deconfigure' and click Update to erase configuration for this object.



Binary Output Objects

Showing objects from 1

Object	Object Name Object Description	Out of Service	Present Value	Reliability	Status	Text
1	**Relay Output 1** Remote relay control	N	Active	0	0,0,0,0	Relay closed

The object name, value, and status are shown for a list of objects starting with the number entered at the top of the page. Click Prev/Next to scroll through the list. Click on the object number in the first column to change name or out-of-service state.

The destination of data for a Binary Output object will typically be some other BACnet or Modbus device. Click on the object number in the first column for more detail including the link to any client map receiving data from this object.

The Binary Output object is commandable, meaning the BACnet client must write both a value and a priority level for that value. The highest level value will be the one written to the remote device (if mapped). If all values are relinquished, the relinquish default value will be written to the remote device.

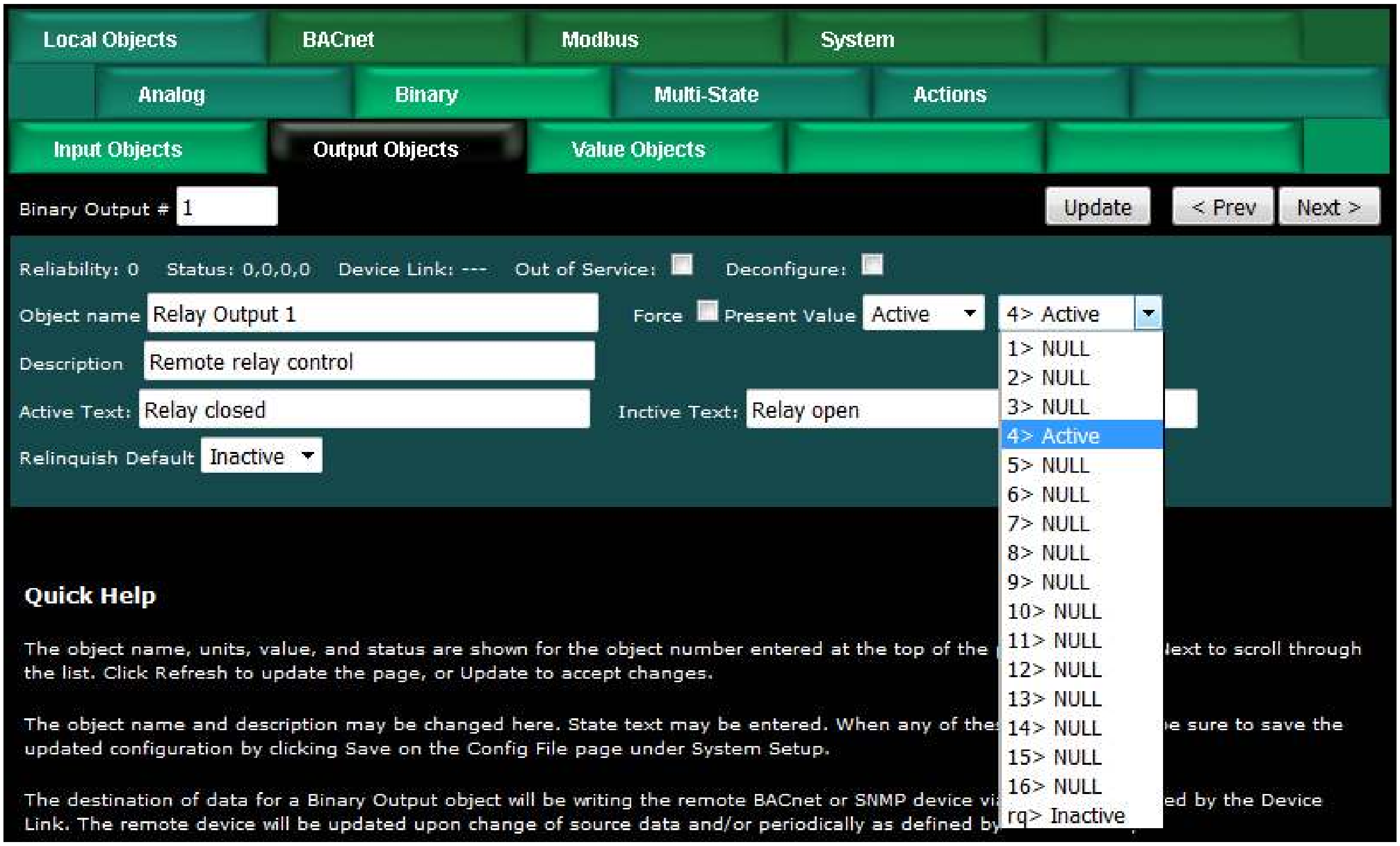
Out of service means the mapped remote device will not be written to. Values written by the BACnet client will be retained, but only applied when this object is placed back in service. At that time, the highest priority value will be written to the remote device.

Reliability codes may be any of the following:
Modbus client/master, no response from slave (64)
Modbus client/master, crc error (65)
Modbus exception, illegal function code (66)
Modbus exception, illegal data address (67)

Modbus exception, illegal data value (68)
Modbus exception, code+65, rarely used (69..79)
Local device, configuration property fault (80)
Faulty Modbus packet(81)
BACnet IP client, device timeout (82)
BACnet IP client, error returned by server (83)

Status flags A,B,C,D indicate the following, 0 meaning not true, 1 meaning true:
A = in alarm
B = fault
C = overridden
D = out of service

Click on an Object number in the first column of maps to get the expanded view of that object as follows:



Binary Output # 1

Reliability: 0 Status: 0,0,0,0 Device Link: --- Out of Service: □ Deconfigure: □

Object name Relay Output 1 Force □ Present Value Active 4> Active

Description Remote relay control

Active Text: Relay closed Inctive Text: Relay open

Relinquish Default Inactive

Quick Help

The object name, units, value, and status are shown for the object number entered at the top of the lext to scroll through the list. Click Refresh to update the page, or Update to accept changes.

The object name and description may be changed here. State text may be entered. When any of the e sure to save the updated configuration by clicking Save on the Config File page under System Setup.

The destination of data for a Binary Output object will be writing the remote BACnet or SNMP device vi ed by the Device Link. The remote device will be updated upon change of source data and/or periodically as defined by

The object name, units, value, and status are shown for the object number entered at the top of the page. Click Prev/Next to scroll through the list. Click Refresh to update the page, or Update to accept changes.

The object name and description may be changed here. State text may be entered. When any of these are changed, be sure to save the updated configuration by clicking Save on the Config File page under System Setup.

The destination of data for a Binary Output object will be writing the remote BACnet or Modbus device via the map indicated by the Device Link. The remote device will be updated upon change of source data and/or periodically as defined by the Write Map.

The Binary Output object is commandable, meaning the BACnet client must write both a value and a priority level for that value. The highest level value will be the one written to the remote device (if one is mapped). If all values are relinquished, the relinquish default value will be written to the remote device.

To set an output object manually from this page, check the Force box, enter a value in the Present Value window, and select a priority level to assign to your forced value. Then click Update. To return a given priority level to NULL, simply type the word NULL in the Present Value window, check Force, and click Update.

Out of service means the mapped remote device will not be written to. Values written by the BACnet client will be retained, but only applied when this object is placed back in service. At that time, the highest priority value will be written to the remote device.

Reliability codes indicate errors as itemized on the tabular object list.

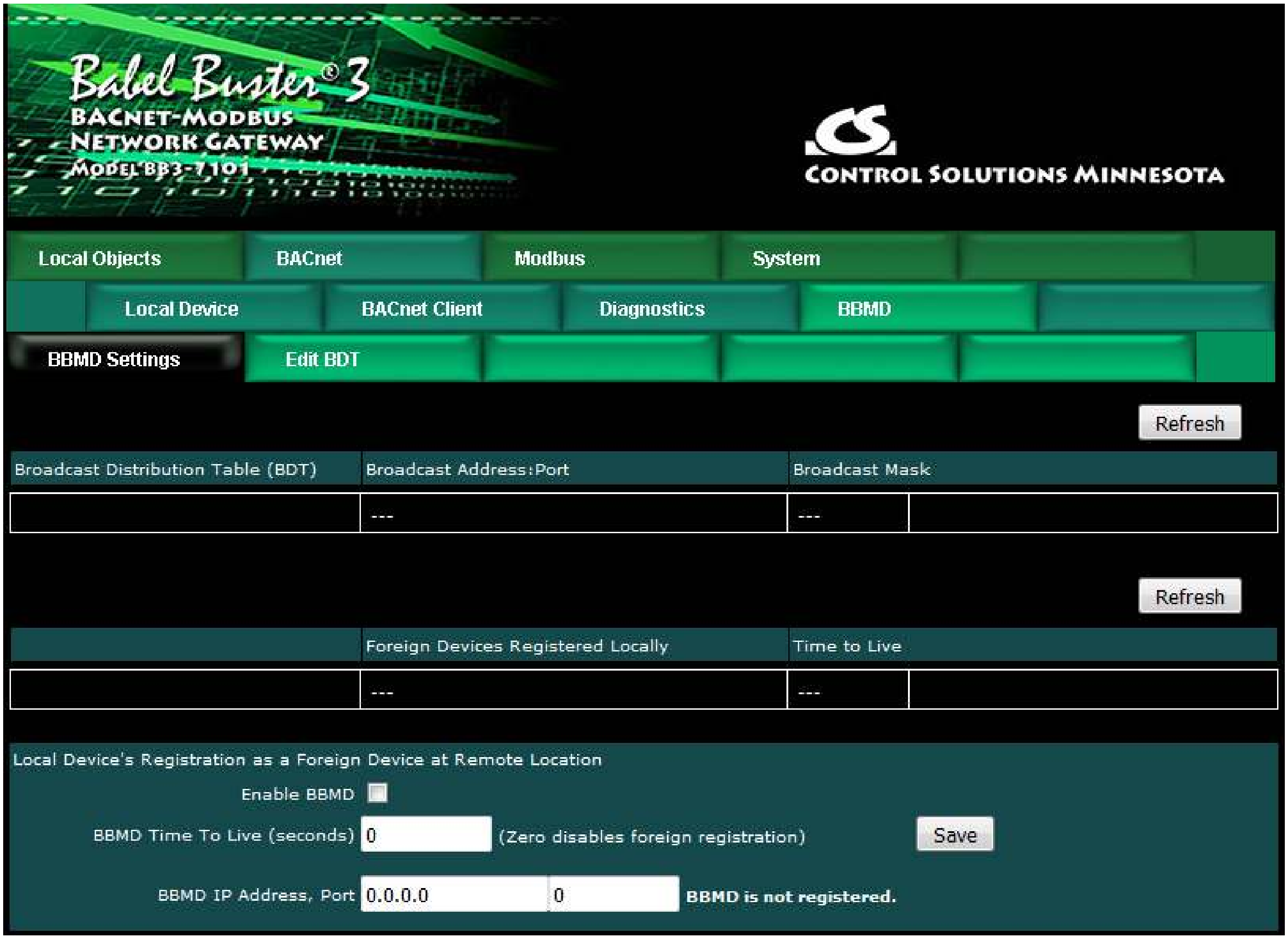
Status flags A,B,C,D indicate the following, 0 meaning not true, 1 meaning true:
A = in alarm
B = fault
C = overridden
D = out of service

Device link will indicate BIP, RTU, or TCP, followed by R for read or W for write, and a number which is the map number in the table of read or write maps for mapping to this BACnet object. The designation R means read from a remote device, and W means write to a remote device.

Check 'Deconfigure' and click Update to erase configuration for this object.

7. Configuring BBMD

BBMD stands for BACnet Broadcast Management Device. Messages such as "Who-Is" and "I-Am" are broadcast. Most routers, however, to not pass broadcast messages along. The BBMD solves this problem by explicitly directing broadcast messages to a specific IP address.



Broadcast Distribution Table (BDT)	Broadcast Address:Port	Broadcast Mask	
	---	---	

	Foreign Devices Registered Locally	Time to Live	
	---	---	

Local Device's Registration as a Foreign Device at Remote Location

Enable BBMD □

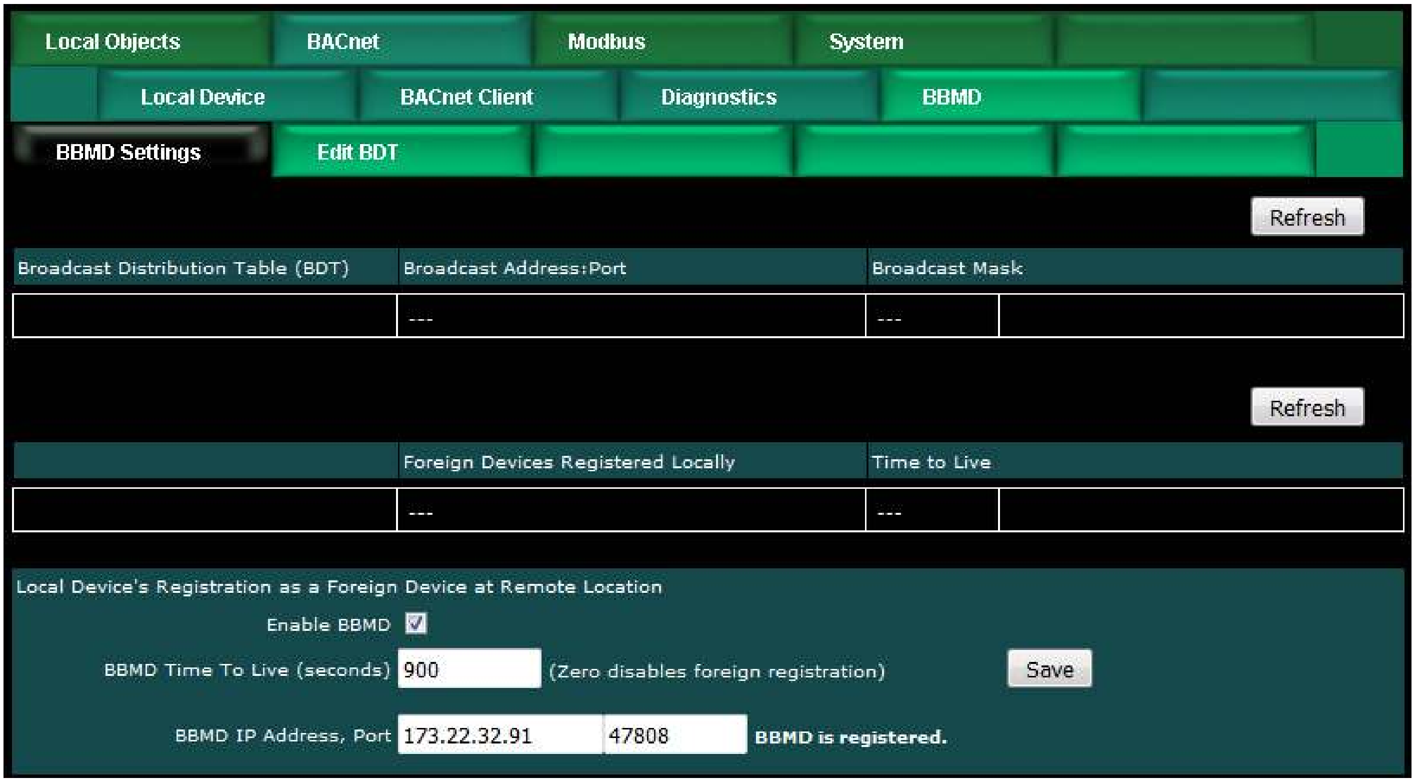
BBMD Time To Live (seconds) 0 (Zero disables foreign registration)

BBMD IP Address, Port 0.0.0.0 0 **BBMD is not registered.**

The BBMD Settings page appears as shown above when no part of BBMD support is enabled, as is the case when shipped. Do not enable BBMD if you are not aware of needing it and/or do not understand how BBMD works. The three elements of BBMD support are discussed in the following sections, and their use is often mutually exclusive, meaning you will often need only one of the three elements.

7.1 Registering as a Foreign Device

If you have a remote BB3-7101/MX-71 that needs to connect via router, including NAT router, to a local network, use Foreign Device Registration. There will typically be a master device, such as operator station or other front end, that includes BBMD. The IP address of this device is the one that should be given as the BBMD address for foreign device registration.



Broadcast Distribution Table (BDT)	Broadcast Address:Port	Broadcast Mask	
	---	---	

	Foreign Devices Registered Locally	Time to Live	
	---	---	

Local Device's Registration as a Foreign Device at Remote Location

Enable BBMD ☑

BBMD Time To Live (seconds) 900 (Zero disables foreign registration)

BBMD IP Address, Port 173.22.32.91 47808 **BBMD is registered.**

To enable BBMD processing, check the "Enable BBMD" box. This applies to foreign device registration. The broadcast distribution table functions regardless of whether foreign device registration is enabled.

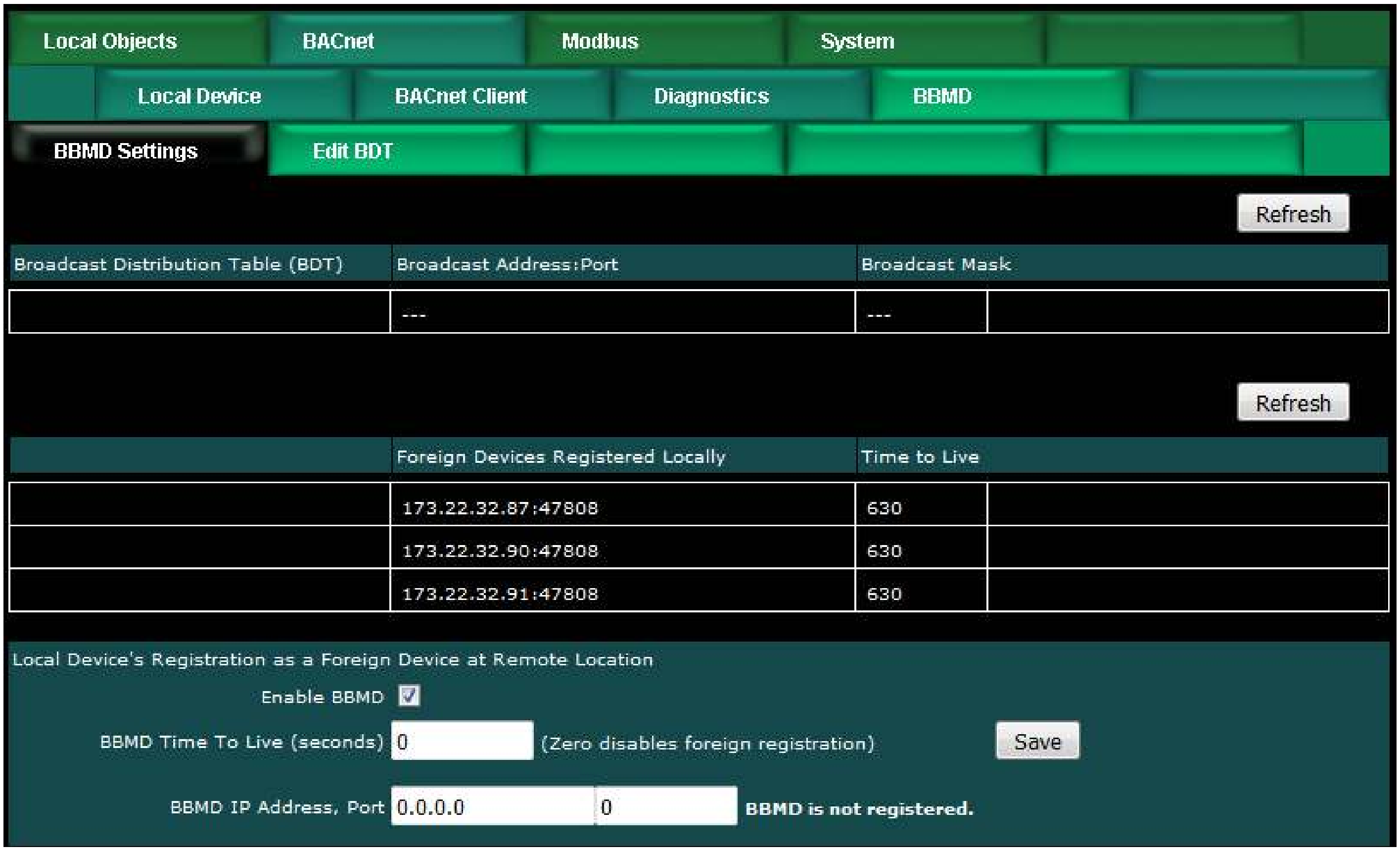
If the BB3-7101/MX-71 should register as a foreign device with another BBMD, then the port, time-to-live, and IP address of the remote BBMD must be given. The local BBMD will attempt to register with the remote BBMD whose address is given.

Disable this device's attempts to register elsewhere, but allow other devices to register here, by setting time to live to zero with BBMD enabled.

7.2 Allowing Other Devices to Register Locally

The BB3-7101/MX-71 can be the BBMD that other devices register with. The screen shot below shows that three other devices have registered with this BBMD, and broadcast messages will now be sent explicitly to these locations. In this case, there are NAT routers between this local device and the three remote devices. While they are all on physically separate local networks, they will appear as a single BACnet network even if the local networks are miles apart. The local BACnet client will be able to communicate with these remote BACnet devices as a result of the foreign registration.

Note that foreign registration only provides communication with a single remote device. If communicating with an entire remote network of BACnet devices is the intent, then a full BACnet router is required, and BBMD would be handled by the BACnet router (disable everything on this BBMD Settings page if connected via a BACnet router).



Broadcast Distribution Table (BDT)	Broadcast Address:Port	Broadcast Mask	
	---	---	

	Foreign Devices Registered Locally	Time to Live	
	173.22.32.87:47808	630	
	173.22.32.90:47808	630	
	173.22.32.91:47808	630	

Local Device's Registration as a Foreign Device at Remote Location

Enable BBMD ☑

BBMD Time To Live (seconds) 0 (Zero disables foreign registration)

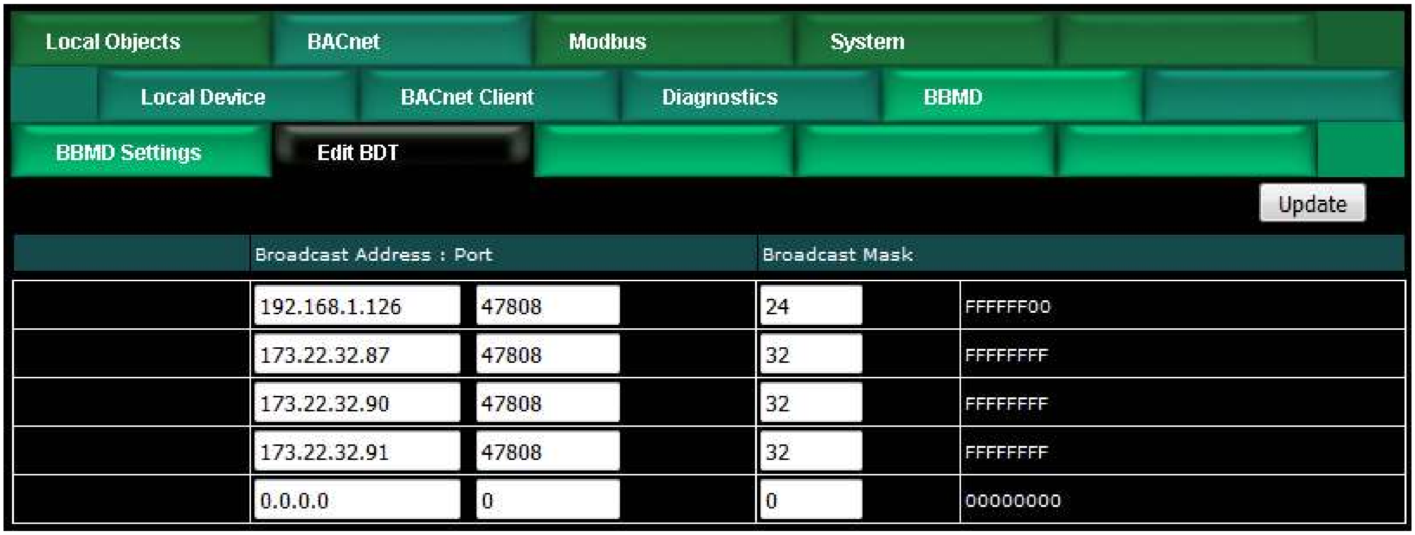
BBMD IP Address, Port 0.0.0.0 0 **BBMD is not registered.**

To allow foreign devices to register with this device, but not have this device register elsewhere, check Enable BBMD, but enter zero for BBMD Time To Live. This enables BBMD but disables this device's attempt to register somewhere else.

7.3 Broadcast Distribution Table

A Broadcast Distribution Table (BDT) defines a list of IP addresses that the BBMD should send broadcast messages to. It is important to note that a BBMD only forwards broadcast messages. It does not do full routing. If you are attempting to connect two networks across a NAT router, you must get a full BACnet Router to accomplish this. For this reason, the BDT has limited usefulness when only BBMD is present. The BB3-7101/MX-71 only includes BBMD, not full routing.

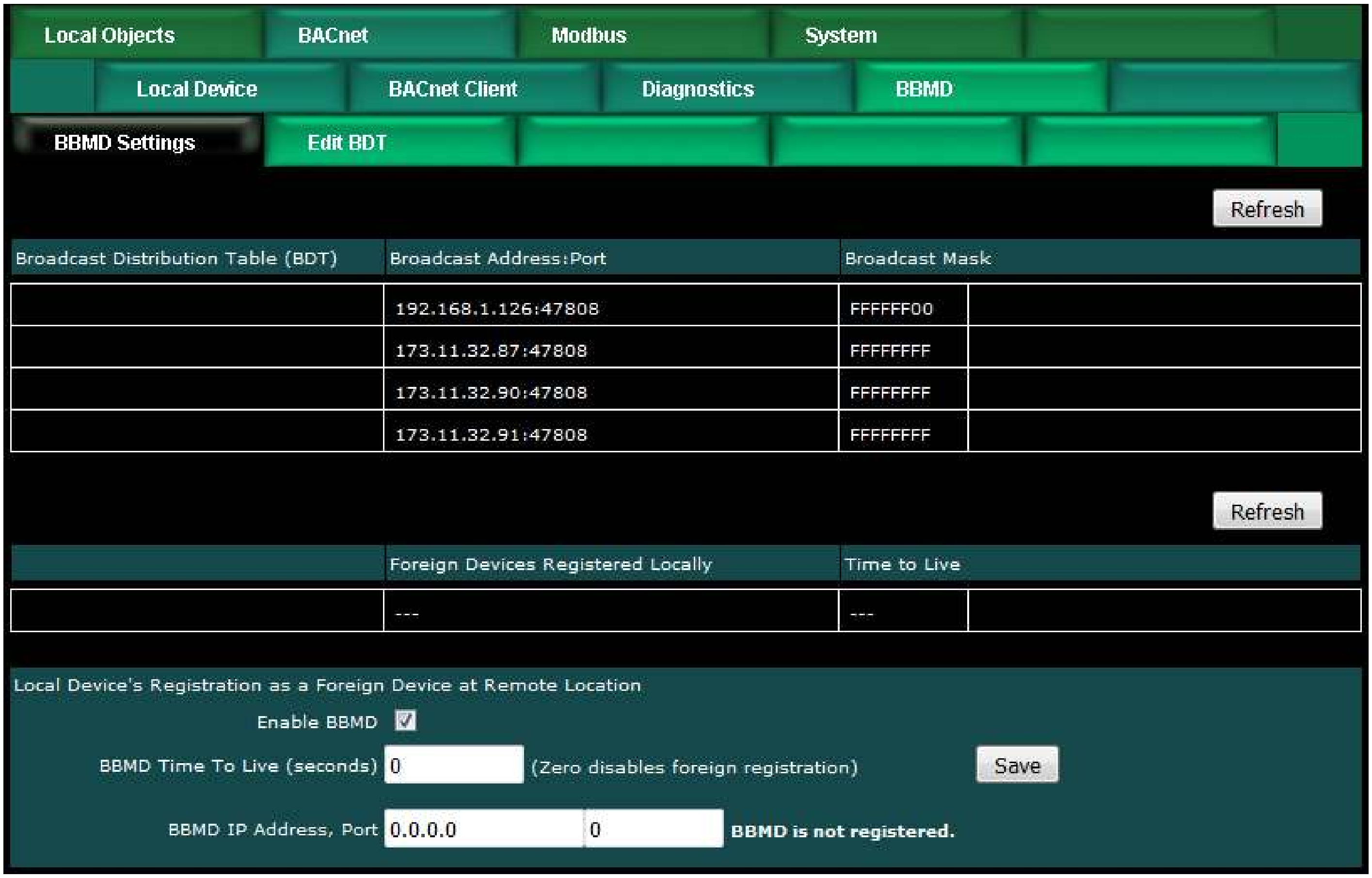
Broadcast distribution will result in device discovery, but you will not be able to read/write properties in the remote device without full routing. Foreign device registration via a router does result in being able to fully communicate with the foreign device from the local network.



	Broadcast Address : Port		Broadcast Mask	
	192.168.1.126	47808	24	FFFFFF00
	173.22.32.87	47808	32	FFFFFFFF
	173.22.32.90	47808	32	FFFFFFFF
	173.22.32.91	47808	32	FFFFFFFF
	0.0.0.0	0	0	00000000

Th Edit BDT page allows viewing of the broadcast distribution table that has been provided to the local device by an external network management tool capable of sending the BDT initialize. The BDT may also be edited on this page. Regardless of how the table is filled, it will be saved in the configuration file when saved, and reloaded upon restart.

Once the table has been initialized, it will appear on the BBMD Settings page as illustrated below.



Broadcast Distribution Table (BDT)	Broadcast Address:Port	Broadcast Mask	
	192.168.1.126:47808	FFFFFF00	
	173.11.32.87:47808	FFFFFFFF	
	173.11.32.90:47808	FFFFFFFF	
	173.11.32.91:47808	FFFFFFFF	

	Foreign Devices Registered Locally	Time to Live	
	---	---	

Local Device's Registration as a Foreign Device at Remote Location

Enable BBMD ☑

BBMD Time To Live (seconds) 0 (Zero disables foreign registration)

BBMD IP Address, Port 0.0.0.0 0 **BBMD is not registered.**

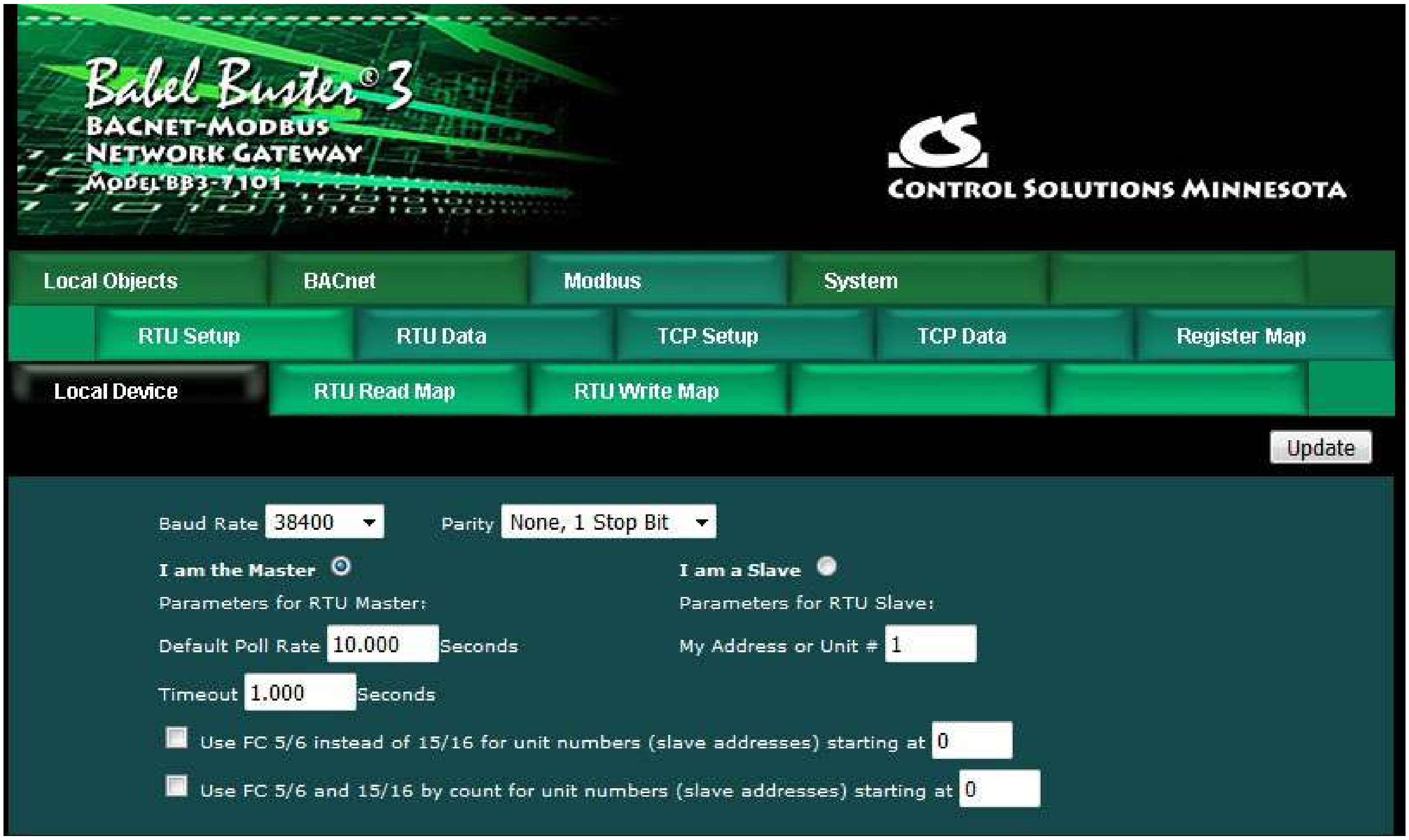
8. Configuring Gateway as a Modbus RTU Master

The BB3-7101/MX-71 can be a Modbus RTU master or slave. As a master you can read Modbus data from, or write Modbus data to, Modbus slave devices. The gateway will periodically poll the Modbus devices according to register maps you set up. To read from a remote Modbus device, configure a Read Map. To write to a remote Modbus device, configure a Write Map.

Data read from a remote device is stored in a local object when received. Data written to a remote device is taken from a local object when sent.

8.1 Modbus RTU Device Configuration

Modbus device configuration for RTU really consists of just port configuration.



Baud Rate 38400 Parity None, 1 Stop Bit

I am the Master ● I am a Slave ○

Parameters for RTU Master: Parameters for RTU Slave:

Default Poll Rate 10.000 Seconds My Address or Unit # 1

Timeout 1.000 Seconds

□ Use FC 5/6 instead of 15/16 for unit numbers (slave addresses) starting at 0

□ Use FC 5/6 and 15/16 by count for unit numbers (slave addresses) starting at 0

Select baud rate and parity from the drop down lists. Click either Master or Slave buttons to select type of operation. Enter timing parameters or address as applicable. Click update to register your changes.

The default poll rate entered here will be used for all Modbus RTU Read and Write maps unless a different number is entered in the expanded view of the map.

IMPORTANT: Set timeout to something long enough for the device. If too short, the gateway will not wait long enough for a response from the Modbus slave device, and the result will be a lot of "no response" errors from the device even though the device is perfectly functional.

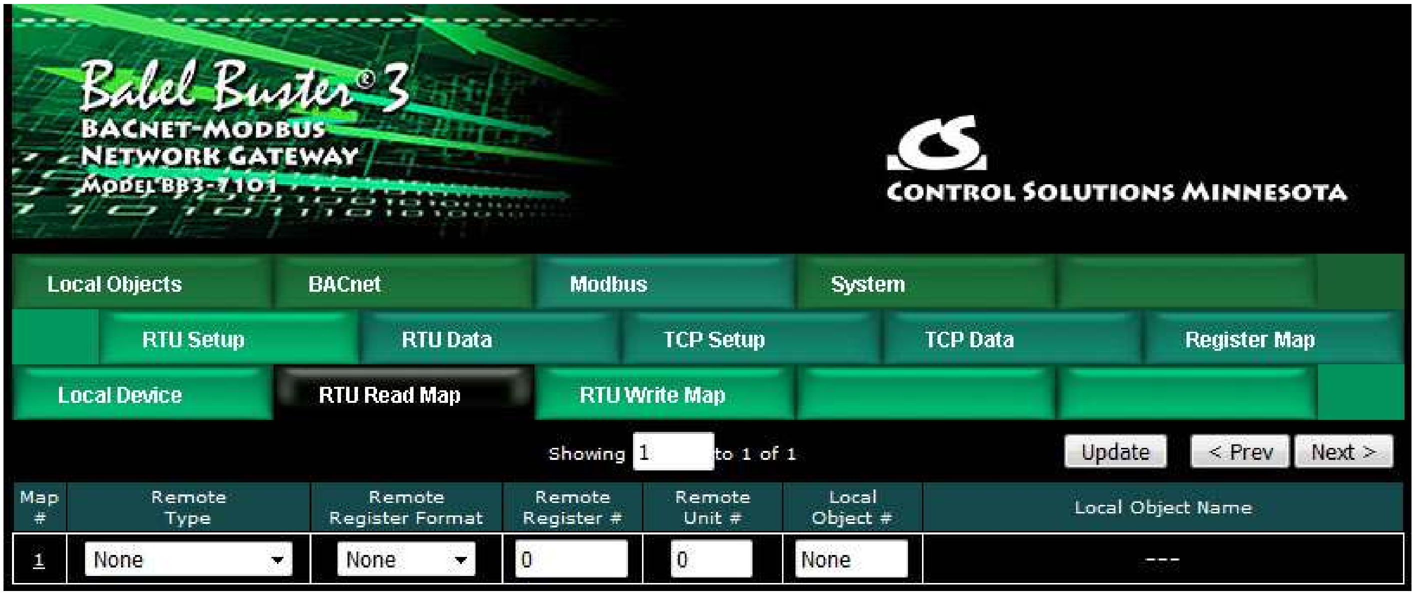
If your slave/server device only supports function codes 5 and 6 for writing coils and holding registers, check the Use FC 5/6 box. The default function codes are 15 and 16, which are most widely used. If you check the box, you should also enter a "starting at" unit # or slave address. This allows supporting both types of devices at the same time provided you assign slave addresses in two non-overlapping groups. (These settings do not apply if the gateway is the slave.)

If your slave/server device is especially picky, it may require function codes 5 and 6 for single writes, but yet expect function codes 15 and 16 for multiple register writes. Select "Use FC 5/6 by count..." if this is the case, and provide a starting unit number.

The starting unit number will most often be simply "1" unless you have a mix of device types that all follow different rules for their function codes. Note that the selection of FC 5/6 only pertains to write maps. Reading registers will always use the same function codes for reading.

8.2 Modbus RTU Master Read Maps

Getting the gateway to read registers from a Modbus device requires setting up a "Read Map" as shown here.

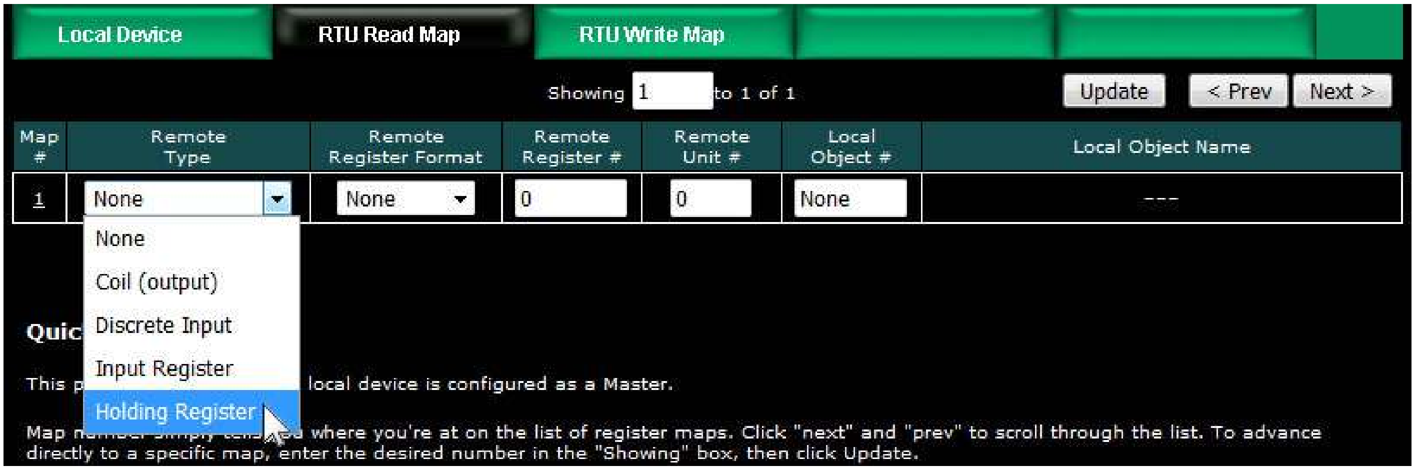


Map #	Remote Type	Remote Register Format	Remote Register #	Remote Unit #	Local Object #	Local Object Name
1	None	None	0	0	None	---

Map number simply tells you where you're at on the list of read maps. Click "next" and "prev" to scroll through the list. To advance directly to a specific map, enter the desired number in the "Showing" box, then click Update.

Maps entered on this page only read data from remote devices. Go to the RTU Write

Map to write data to those devices. The full parameter set is different for read versus write.



Showing 1 to 1 of 1

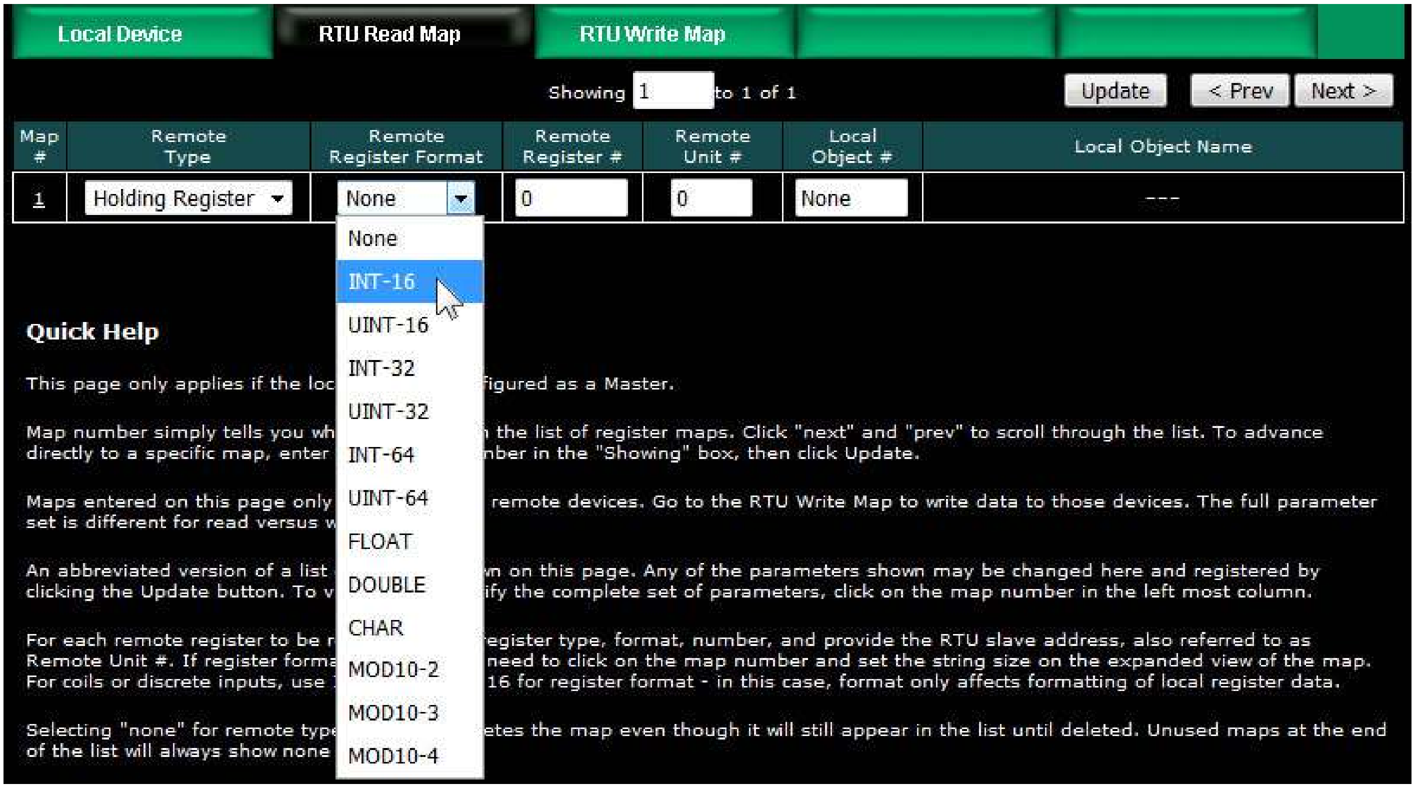
Map #	Remote Type	Remote Register Format	Remote Register #	Remote Unit #	Local Object #	Local Object Name
1	None	None	0	0	None	---

Quic

This p local device is configured as a Master.

Map n where you're at on the list of register maps. Click "next" and "prev" to scroll through the list. To advance directly to a specific map, enter the desired number in the "Showing" box, then click Update.

To create a Read Map, start by selecting the Modbus register type to read from the drop-down list.



Showing 1 to 1 of 1

Map #	Remote Type	Remote Register Format	Remote Register #	Remote Unit #	Local Object #	Local Object Name
1	Holding Register	None	0	0	None	---

Quick Help

This page only applies if the loc figured as a Master.

Map number simply tells you wh the list of register maps. Click "next" and "prev" to scroll through the list. To advance directly to a specific map, enter nber in the "Showing" box, then click Update.

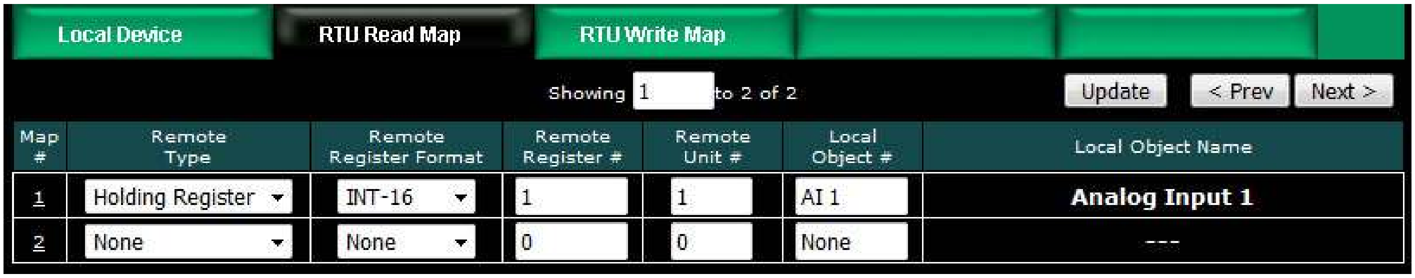
Maps entered on this page only remote devices. Go to the RTU Write Map to write data to those devices. The full parameter set is different for read versus w

An abbreviated version of a list n on this page. Any of the parameters shown may be changed here and registered by clicking the Update button. To v ify the complete set of parameters, click on the map number in the left most column.

For each remote register to be r egister type, format, number, and provide the RTU slave address, also referred to as Remote Unit #. If register forma need to click on the map number and set the string size on the expanded view of the map. For coils or discrete inputs, use 16 for register format - in this case, format only affects formatting of local register data.

Selecting "none" for remote type etes the map even though it will still appear in the list until deleted. Unused maps at the end of the list will always show none

Select the data format expected in the remote Modbus register. The abbreviation INT under Format means signed integer, while UINT represents unsigned integer. The INT and UINT are followed by the number of bits to be read (which translates into 1, 2, or 4 consecutive holding registers). The FLOAT format refers to 32-bit IEEE 754 format while DOUBLE refers to 64-bit IEEE 754 floating point. The MOD10 format is unique to Schneider Electric power meters, and is supported in 2, 3, and 4-register formats. (Note: Use INT-16 or UINT-16 for coils or discrete inputs - in this case format only affects local data conversion.)

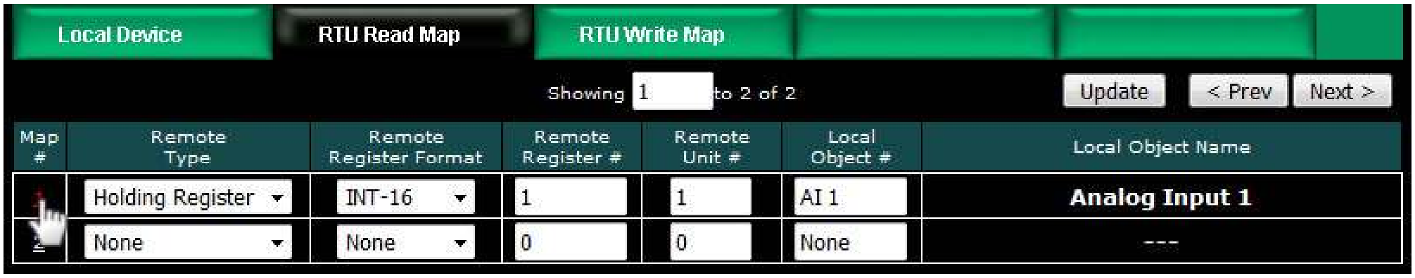


Showing 1 to 2 of 2

Map #	Remote Type	Remote Register Format	Remote Register #	Remote Unit #	Local Object #	Local Object Name
1	Holding Register	INT-16	1	1	AI 1	Analog Input 1
2	None	None	0	0	None	---

Enter the register number to read from the remote RTU slave and slave address of that RTU slave. Do not use Modicon numbers here. In other words, if your slave device's documentation says read register 40001, that is short hand (Modicon notation) for saying read holding register 1. Refer to the Modbus Reference Information section of this user guide for more discussion about register numbers like 40001. If you enter 40001 here to read the first holding register, you will get an exception error since the actual register number is not 40001.

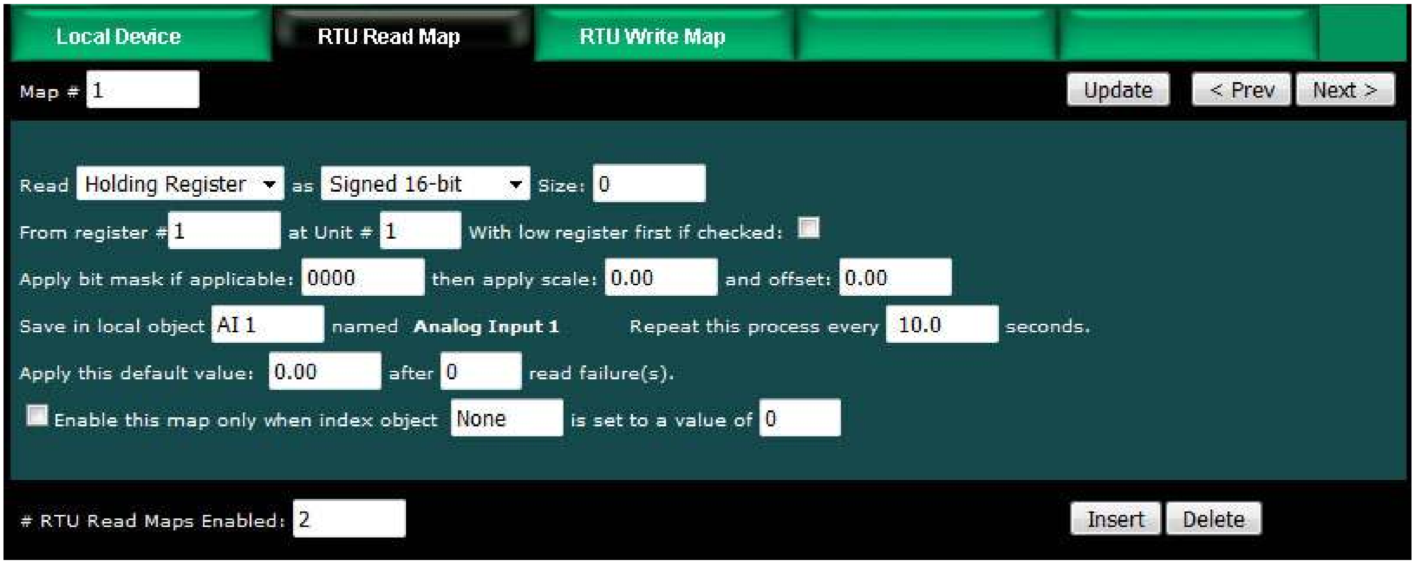
The Local Object is where data read from the remote Modbus RTU slave will be stored locally in the Babel Buster. If the local object data format does not match what you are reading from the Modbus slave, the data will be converted automatically when it is read.



Showing 1 to 2 of 2

Map #	Remote Type	Remote Register Format	Remote Register #	Remote Unit #	Local Object #	Local Object Name
1	Holding Register	INT-16	1	1	AI 1	Analog Input 1
2	None	None	0	0	None	---

Click on the Map number in the first column to access the expanded view of the Read Map.



Map # 1

Read Holding Register as Signed 16-bit Size: 0

From register # 1 at Unit # 1 With low register first if checked: □

Apply bit mask if applicable: 0000 then apply scale: 0.00 and offset: 0.00

Save in local object AI 1 named **Analog Input 1** Repeat this process every 10.0 seconds.

Apply this default value: 0.00 after 0 read failure(s).

□ Enable this map only when index object None is set to a value of 0

RTU Read Maps Enabled: 2

For each Modbus register to be read, select the register type, format, number, and remote unit (slave address). The optional bit mask and scaling are discussed with examples below.

Modbus protocol treats all input registers or holding registers as strictly 16-bit registers. To accommodate 32-bit or longer data, Modbus devices use multiple consecutive "registers" to hold the data. There is no standardization of whether the least significant part of the data comes first or last. Therefore, Babel Buster lets you set that according to whatever the slave device requires. If the least significant data is found in the first (or lower numbered) register in your slave device, then check the box after "With low register first".

The poll rate ("Repeat this process...") determines how often the Modbus register will be read. If zero is entered here, the rate will become the default poll rate given on the Devices page for the Modbus RTU port.

The default value will be stored into the local object after the given number of read failures if the fail count is non-zero. Setting the count to zero will disable the default, and the object will retain the most recent value obtained. If the default value does take effect, the actual data value read will be retained when communications are restored.

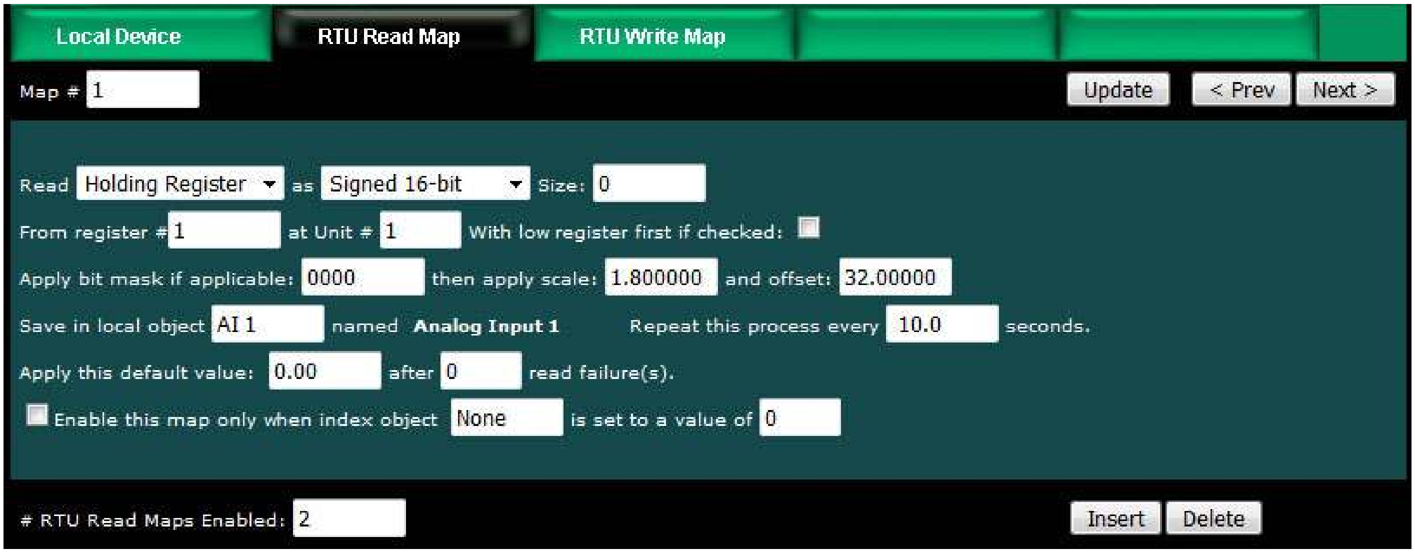
You have the option of making this Read Map conditional. If an index object is provided and the Enable box is checked, then this read map will only be executed when this local object contains the value given. This allows multiple read maps to supply data to the same local object based on the value of the index object. It also allows reading to simply be suspended if a single read map supplies data to the local object. In a more sophisticated scenario, you could potentially suspend reading of the slave if you know the slave is powered down.

Map number simply tells you where you're at on the list of read maps. Click "next" and "prev" to scroll through the list. To advance directly to a specific map, enter the desired number in the "Map #" box, then click Update.

Delete will remove the map number shown in the "Map #" box. Insert will insert a new map before the map number shown, and is used for placing maps between existing maps. It is not necessary to use Insert to add maps to the bottom of the list - simply add definition to the empty map already at the end of the list.

Selecting "none" for remote type effectively deletes the map even though it will still appear in the list until deleted. Unused maps at the end of the list will always show none as the type. If you wish to prevent these from being displayed, reduce the number of maps enabled.

The number of maps enabled simply limits the scope of map review so that you do not have to review a lot of unused maps. If the displayed maps are used up and you need more, increase the enabled number.



Map # 1

Read Holding Register as Signed 16-bit Size: 0

From register # 1 at Unit # 1 With low register first if checked: □

Apply bit mask if applicable: 0000 then apply scale: 1.800000 and offset: 32.00000

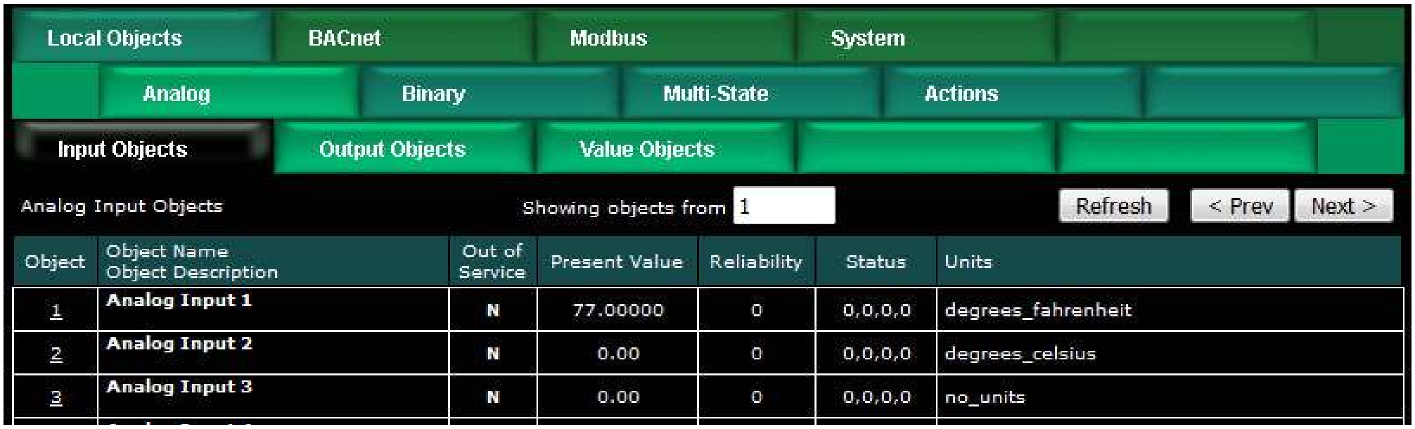
Save in local object AI 1 named **Analog Input 1** Repeat this process every 10.0 seconds.

Apply this default value: 0.00 after 0 read failure(s).

□ Enable this map only when index object None is set to a value of 0

RTU Read Maps Enabled: 2

You have the option of providing a scale and offset. A scale of zero will cause scale and offset to be ignored. If provided, the Modbus data will be treated as raw data. When the Modbus data is received, it will be multipled by scale, then added to offset, and then stored in the local object. If the Modbus slave was providing degrees Celsius, and the scale factors illustrated above were used, then a Modbus value of 25 would result in the local object receiving a value of 77 (degrees Fahrenheit).

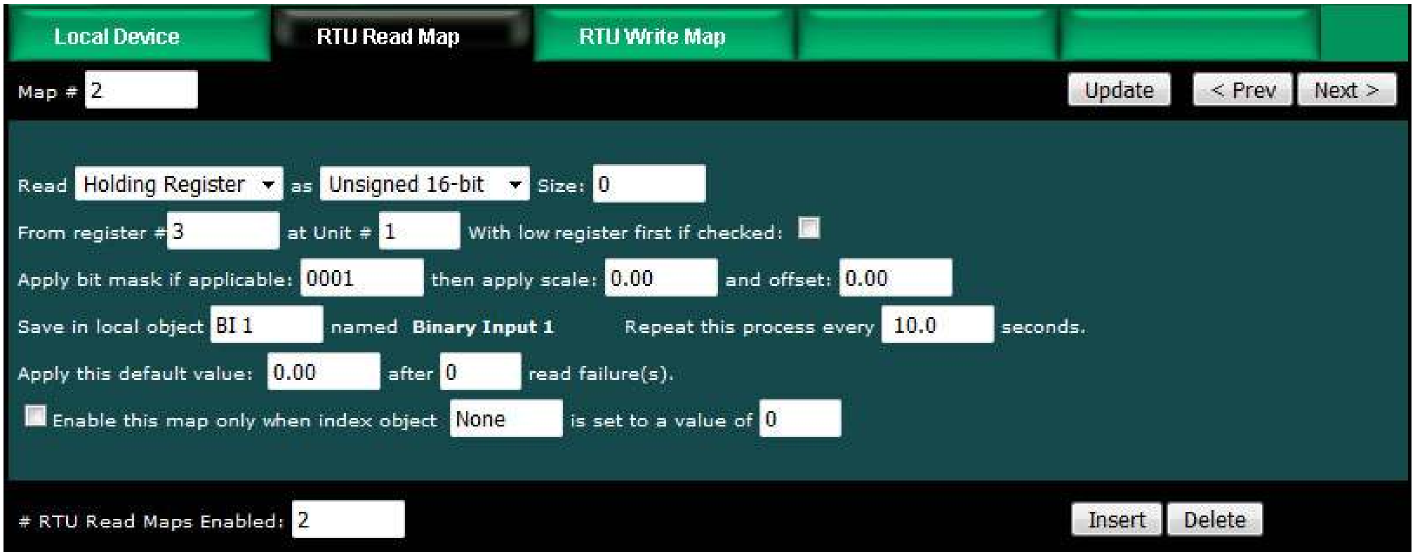


Analog Input Objects Showing objects from 1

Object	Object Name Object Description	Out of Service	Present Value	Reliability	Status	Units
1	**Analog Input 1**	N	77.00000	0	0,0,0,0	degrees_fahrenheit
2	**Analog Input 2**	N	0.00	0	0,0,0,0	degrees_celsius
3	**Analog Input 3**	N	0.00	0	0,0,0,0	no_units

It is common for Modbus devices to pack a number of status bits into a single holding register. In order to do meaningful things based on a single bit it is necessary to split that register into multiple local objects. Babel Buster supports this requirement by providing an optional bit mask.

If a bit mask is entered (in hexadecimal), and the remote register type is signed or unsigned integer (16-bit or 32-bit data), the mask will be bit-wise logical AND-ed with the Modbus data. The retained bits will be right justified in the result stored locally if the local object is Analog or Multi-State. The result will simply set a Binary object to Active if nonzero.



Map # 2

Read Holding Register as Unsigned 16-bit Size: 0

From register # 3 at Unit # 1 With low register first if checked: □

Apply bit mask if applicable: 0001 then apply scale: 0.00 and offset: 0.00

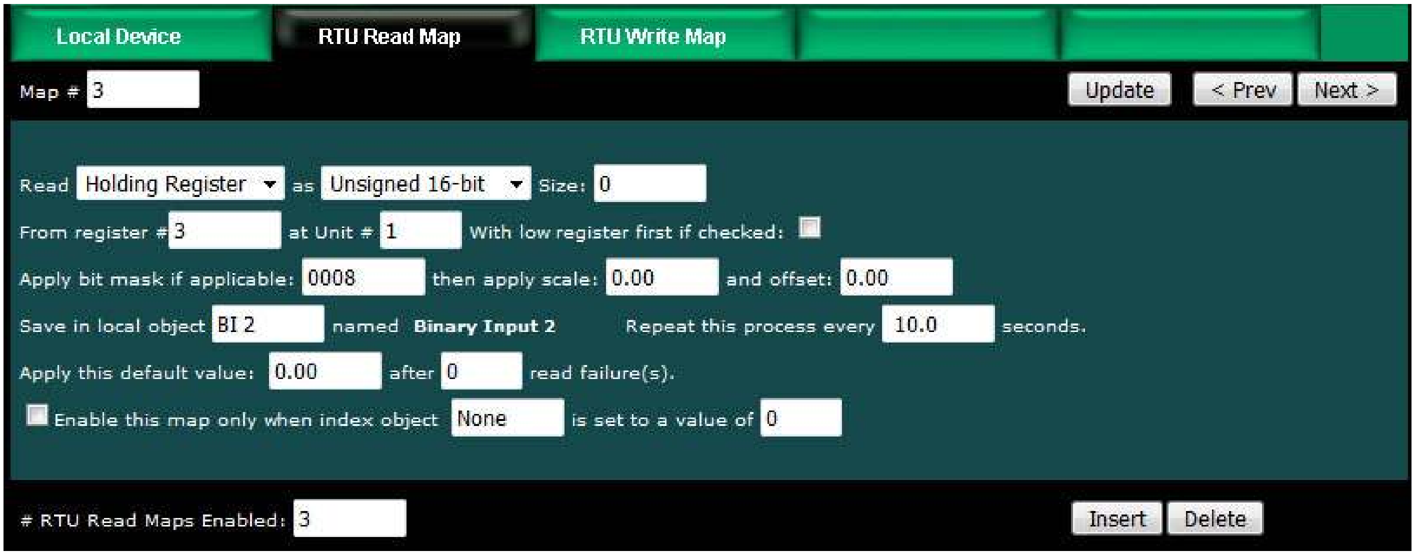
Save in local object BI 1 named **Binary Input 1** Repeat this process every 10.0 seconds.

Apply this default value: 0.00 after 0 read failure(s).

□ Enable this map only when index object None is set to a value of 0

RTU Read Maps Enabled: 2

Reading bit 0 is illustrated above while reading bit 3 is illustrated below (bit 3 mask in binary would be 1000 which is hexadecimal 8 as entered in the configuration page). Refer to the Modbus Reference section in this user guide for a list of all possible mask values.



Map # 3

Read Holding Register as Unsigned 16-bit Size: 0

From register # 3 at Unit # 1 With low register first if checked: □

Apply bit mask if applicable: 0008 then apply scale: 0.00 and offset: 0.00

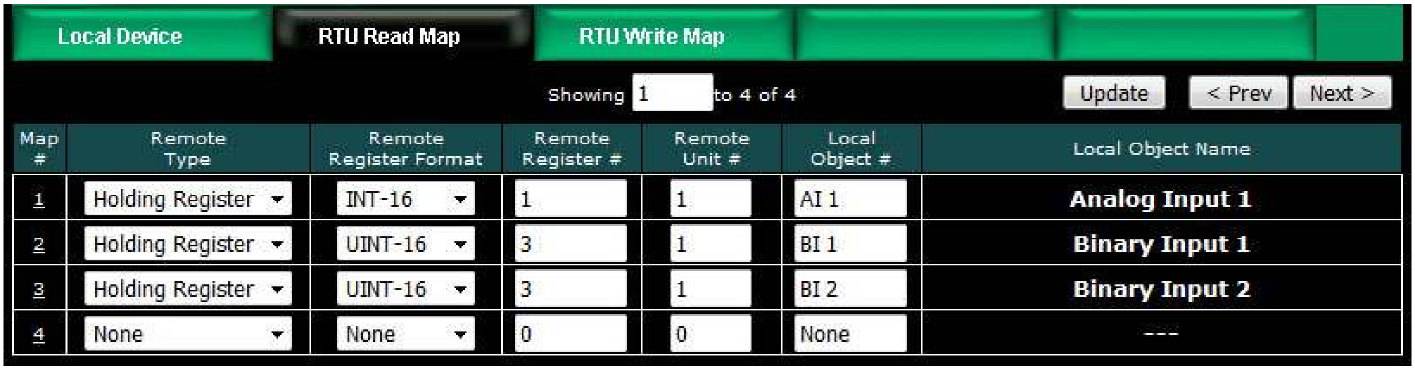
Save in local object BI 2 named **Binary Input 2** Repeat this process every 10.0 seconds.

Apply this default value: 0.00 after 0 read failure(s).

□ Enable this map only when index object None is set to a value of 0

RTU Read Maps Enabled: 3

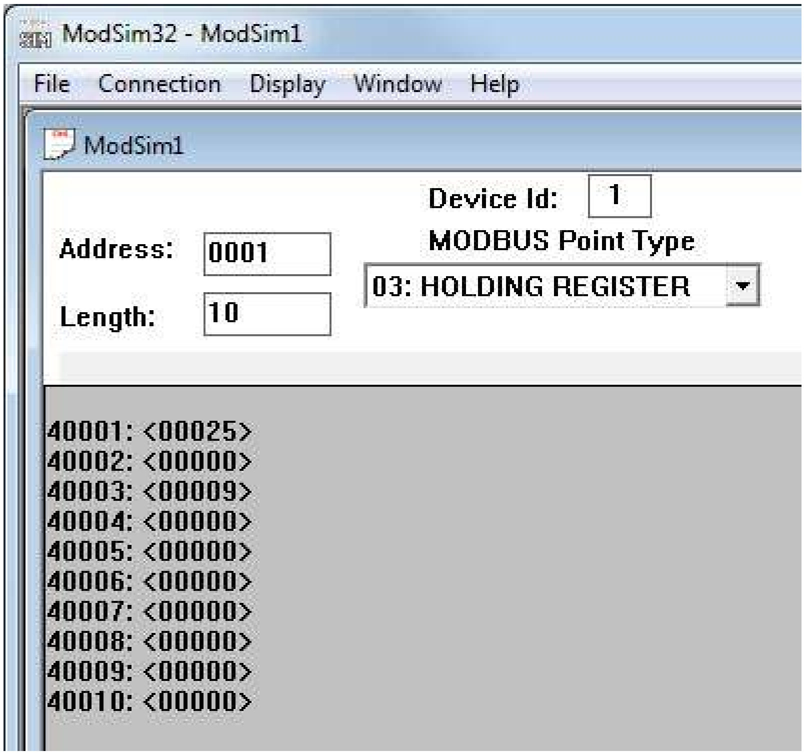
If the read maps referencing the same remote register are created in sequential contiguous order, the Babel Buster will optimize the RTU activity by reading the remote register once and then sharing the data with all of the read maps in the group. In the example illustrated here, two consecutive read maps reference the same Modbus register, each selecting a different bit.



Showing 1 to 4 of 4

Map #	Remote Type	Remote Register Format	Remote Register #	Remote Unit #	Local Object #	Local Object Name
1	Holding Register	INT-16	1	1	AI 1	Analog Input 1
2	Holding Register	UINT-16	3	1	BI 1	Binary Input 1
3	Holding Register	UINT-16	3	1	BI 2	Binary Input 2
4	None	None	0	0	None	---

To try out these read maps, we have set up ModSim with the remote register 3 containing a value of 9.



Device Id: 1

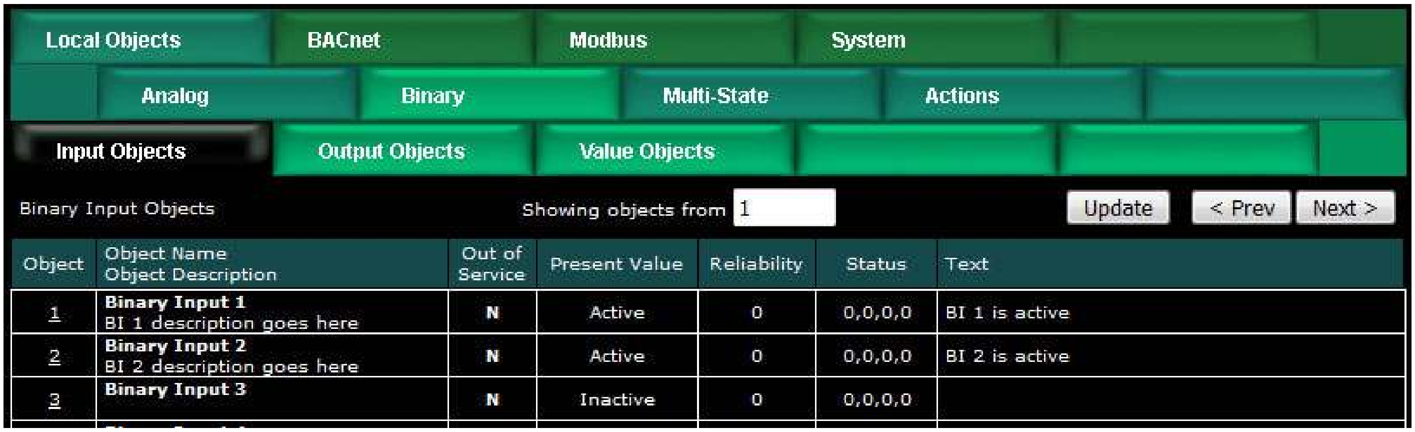
Address: 0001

Length: 10

MODBUS Point Type: 03: HOLDING REGISTER

```
40001: <00025>
40002: <00000>
40003: <00009>
40004: <00000>
40005: <00000>
40006: <00000>
40007: <00000>
40008: <00000>
40009: <00000>
40010: <00000>
```

The holding register value of 9 translates into the following local object values when the bit mask option is used as illustrated.



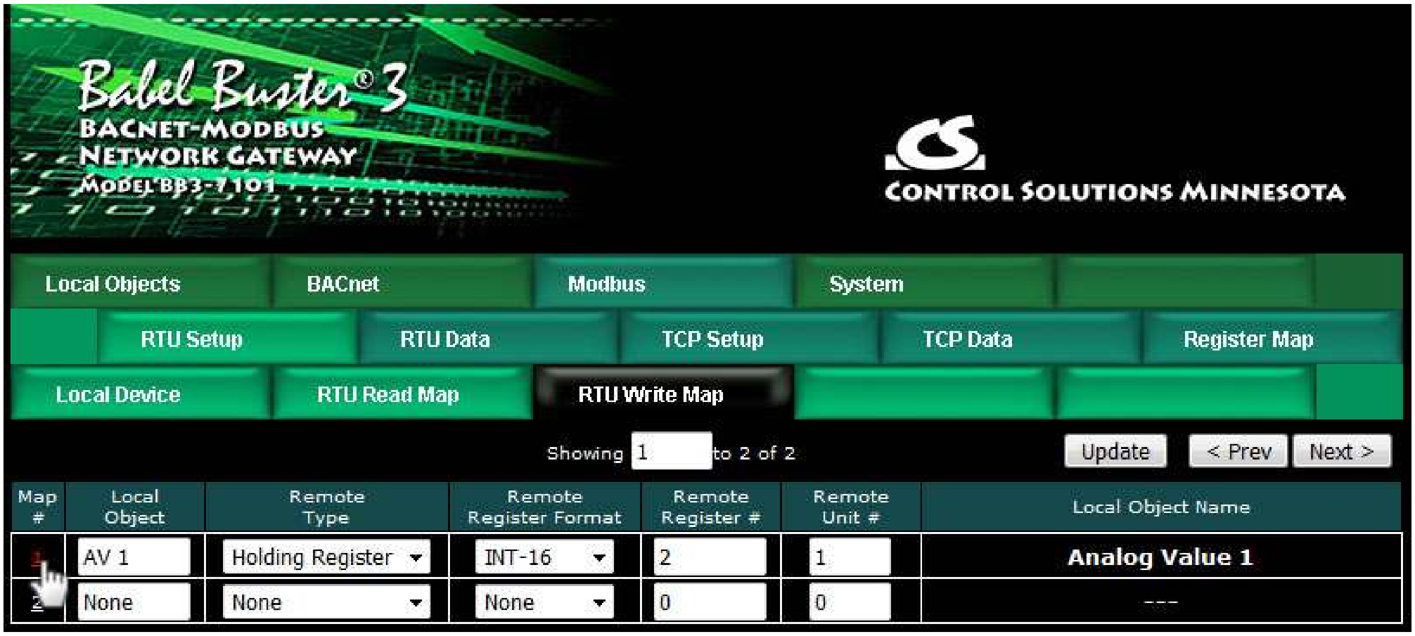
Binary Input Objects

Showing objects from 1

Object	Object Name Object Description	Out of Service	Present Value	Reliability	Status	Text
1	Binary Input 1 BI 1 description goes here	N	Active	0	0,0,0,0	BI 1 is active
2	Binary Input 2 BI 2 description goes here	N	Active	0	0,0,0,0	BI 2 is active
3	Binary Input 3	N	Inactive	0	0,0,0,0	

8.3 Modbus RTU Master Write Maps

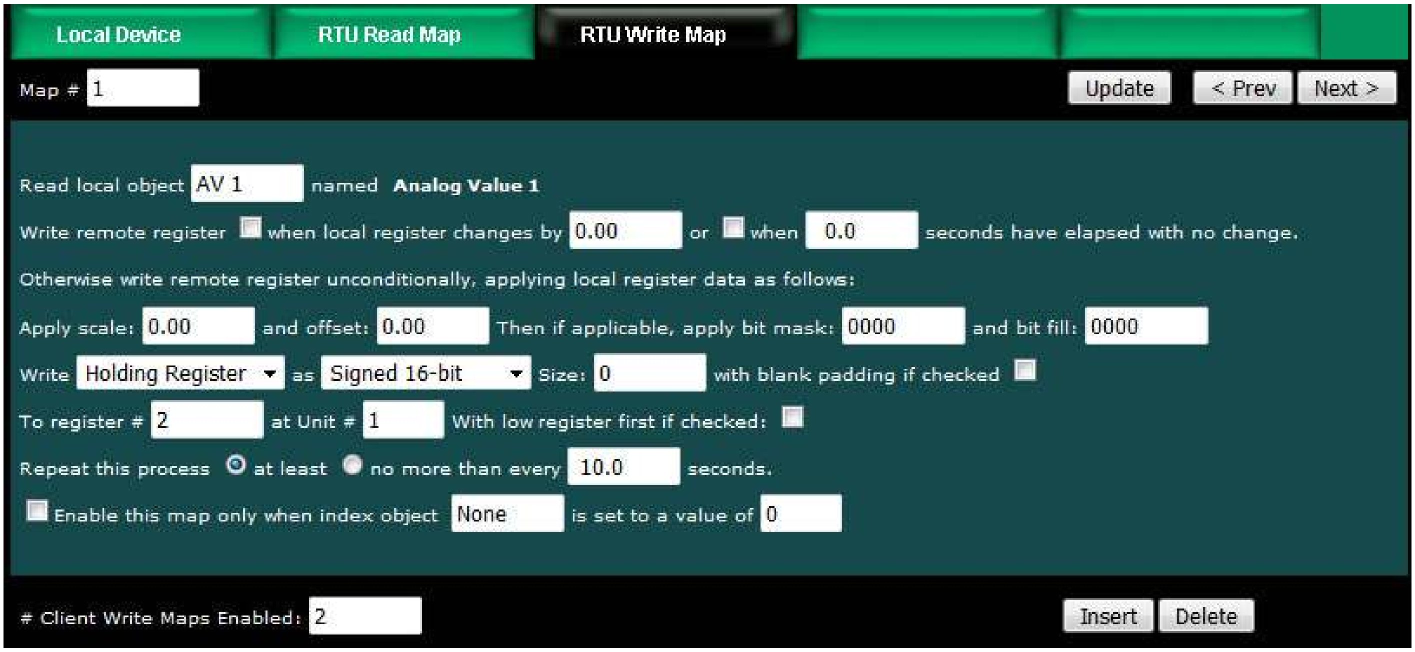
Getting the gateway to write registers to a Modbus device requires setting up a "Write Map" as shown here. Much of the Write Map is configured the same as a Read Map.



Showing 1 to 2 of 2

Map #	Local Object	Remote Type	Remote Register Format	Remote Register #	Remote Unit #	Local Object Name
1	AV 1	Holding Register	INT-16	2	1	Analog Value 1
2	None	None	None	0	0	---

The data direction is reversed but the same selections are still made. Select the local object that will be the source of data to write to the remote Modbus RTU slave. Select the register type, data format, and register number to be written to in that slave. Enter the slave address as Remote Unit. Click on the map number in the first column to access additional optional configuratino parameters.



Map # 1

Read local object AV 1 named **Analog Value 1**

Write remote register ☐ when local register changes by 0.00 or ☐ when 0.0 seconds have elapsed with no change.

Otherwise write remote register unconditionally, applying local register data as follows:

Apply scale: 0.00 and offset: 0.00 Then if applicable, apply bit mask: 0000 and bit fill: 0000

Write Holding Register as Signed 16-bit Size: 0 with blank padding if checked ☐

To register # 2 at Unit # 1 With low register first if checked: ☐

Repeat this process ● at least ○ no more than every 10.0 seconds.

☐ Enable this map only when index object None is set to a value of 0

Client Write Maps Enabled: 2

The local object data may be written to the remote Modbus slave periodically, or when it changes, or both. To send upon change (send on delta), check the first box and enter the amount by which the local object must change before being written to the remote Modbus device. To guarantee that the remote register will be written at least occasionally even if the data does not change, check the second box and enter some amount of time. This time period will be referred to as the "maximum quiet time".

Data from the local object may be manipulated before being written to the remote register. The local data is first multiplied by the scale factor. The offset is then added to it. If a bit mask is entered, and the remote register type is signed or unsigned integer (16-bit or 32-bit data), the mask will be bit-wise logical AND-ed with the data. The mask is right justified, then AND-ed with the data. The result is then left shifted back to the original position of the mask. In other words, the least significant bits of the original data will be stuffed at the position marked by the mask.

After the scaling and masking, the bit fill will be logically OR-ed into the result, but only if the mask was nonzero and was used. Both mask and fill are entered in hexadecimal. The effect of "fill" is that certain bits will always be set to 1 in the data written to the remote Modbus device.

Multiple local objects may be packed into a single remote Modbus register. To accomplish this, define two or more maps in sequence with the same remote destination. If the destination is the same, data types are 16 or 32-bit integer (signed or unsigned), bit masks are nonzero, and the maps are sequential, the results of all qualifying maps will be OR-ed together before being sent to the remote destination.

For the remote register to be written, select the register type, format, number, and remote unit (slave address). Data formats are the same as described above for Read Maps. Size is only specified for character strings. Use INT-16 or UINT-16 data format for coils - in this case format only affects local data conversion.

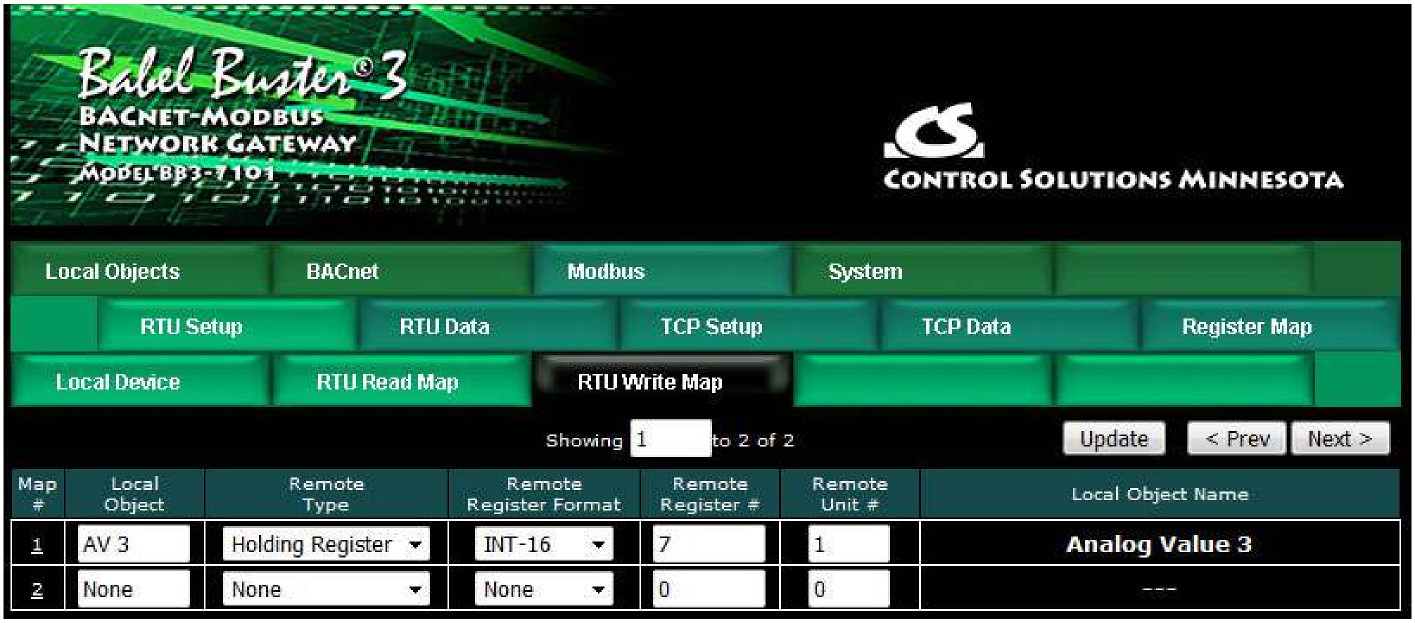
Modbus protocol treats all holding registers as strictly 16-bit registers. To accommodate 32-bit or longer data, Modbus devices use multiple consecutive "registers" to hold the data. There is no standardization of whether the least significant part of the data comes first or last. Therefore, Babel Buster lets you set that according to whatever the slave device requires. If the least significant data is found in the first (or lower numbered) register in your slave device, then check the box after "With low register first".

The repeat time may determine how often the remote register will be written. If send on delta and maximum quiet time are not checked above, clicking the "at least" button will establish a periodic update time. If send on delta is used and you wish to limit the network traffic when changes are too frequent, click the "no more than" button and enter the minumum time that should elapse before allowing another write to the remote device. It is valid to select "no more than every 0.0 seconds" if you want all changes to be sent, but no periodic writes.

You have the option of making this Write Map conditional. If an index object is provided and the Enable box is checked, then this write map will only be executed when this local object contains the value given. This allows multiple write maps to supply data to the same remote register based on the value of the local index object. It also allows writing to simply be suspended if a single write map supplies data to the remote register. In a more sophisticated scenario, you could potentially suspend writing of the slave if you know the slave is powered down.

Delete will remove the map number shown in the "Map #" box. Insert will insert a new map before the map number shown, and is used for placing maps between existing maps. It is not necessary to use Insert to add maps to the bottom of the list - simply add definition to the empty map already at the end of the list.

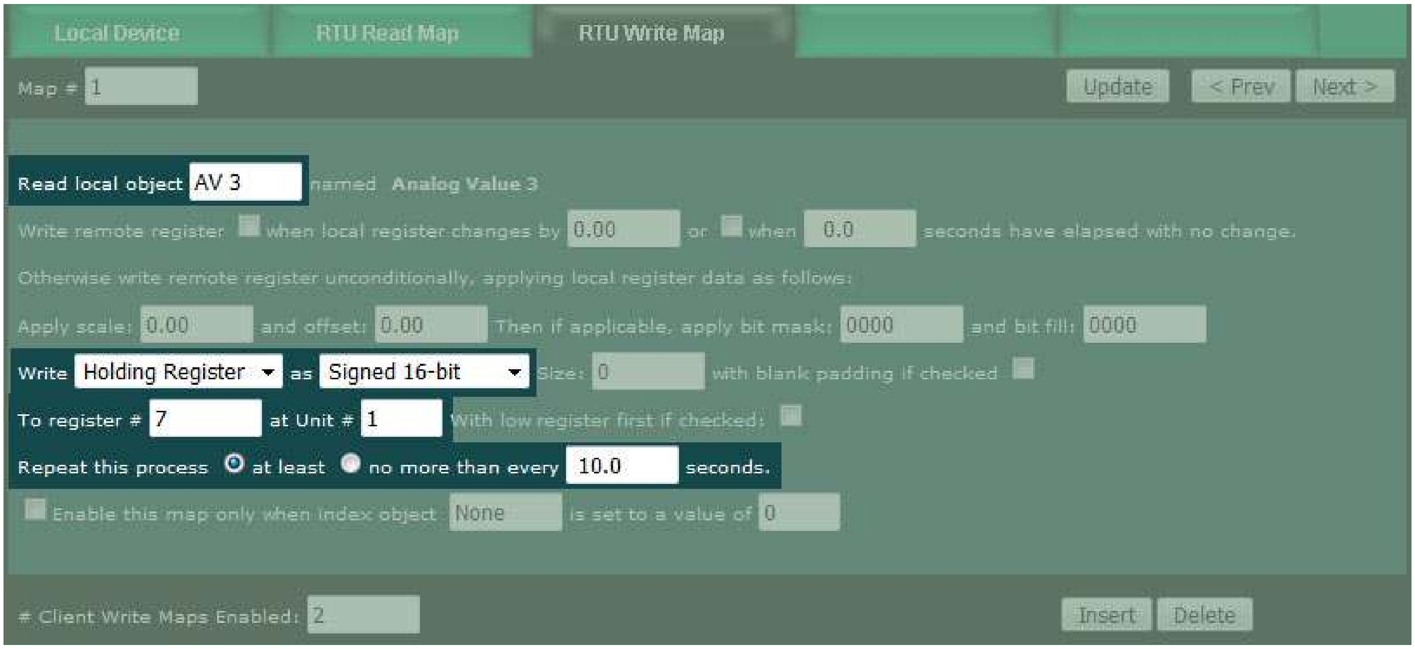
Selecting "none" for remote type effectively deletes the map even though it will still appear in the list until deleted. Unused maps at the end of the list will always show none as the type. If you wish to prevent these from being displayed, reduce the number of maps enabled.



Showing 1 to 2 of 2

Map #	Local Object	Remote Type	Remote Register Format	Remote Register #	Remote Unit #	Local Object Name
1	AV 3	Holding Register	INT-16	7	1	Analog Value 3
2	None	None	None	0	0	---

The expanded view of the read and write maps may look daunting at first. But many applications can be configured by entering information on the tabular map list page rather than the expanded view. The map entries made above appear as shown below in the expanded view. You only need to use the expanded view when you want to apply some of the more specialized optional parameters that are grayed out in this screen shot.



Map # 1

Read local object AV 3 named Analog Value 3

Write remote register □ when local register changes by 0.00 or □ when 0.0 seconds have elapsed with no change.

Otherwise write remote register unconditionally, applying local register data as follows:

Apply scale: 0.00 and offset: 0.00 Then if applicable, apply bit mask: 0000 and bit fill: 0000

Write Holding Register as Signed 16-bit Size: 0 with blank padding if checked □

To register # 7 at Unit # 1 With low register first if checked: □

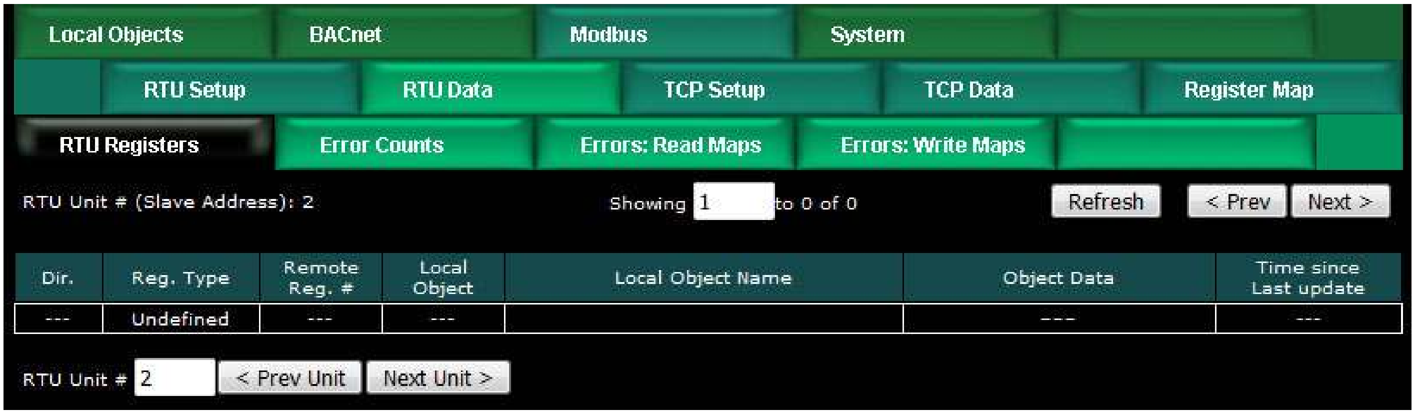
Repeat this process ● at least ○ no more than every 10.0 seconds.

□ Enable this map only when index object None is set to a value of 0

Client Write Maps Enabled: 2

8.4 Modbus RTU Master Data Displayed by Slave

The RTU Registers page shows a list of local objects mapped to RTU slave devices. The page will show only one device at a time, and may have no entries as illustrated below.



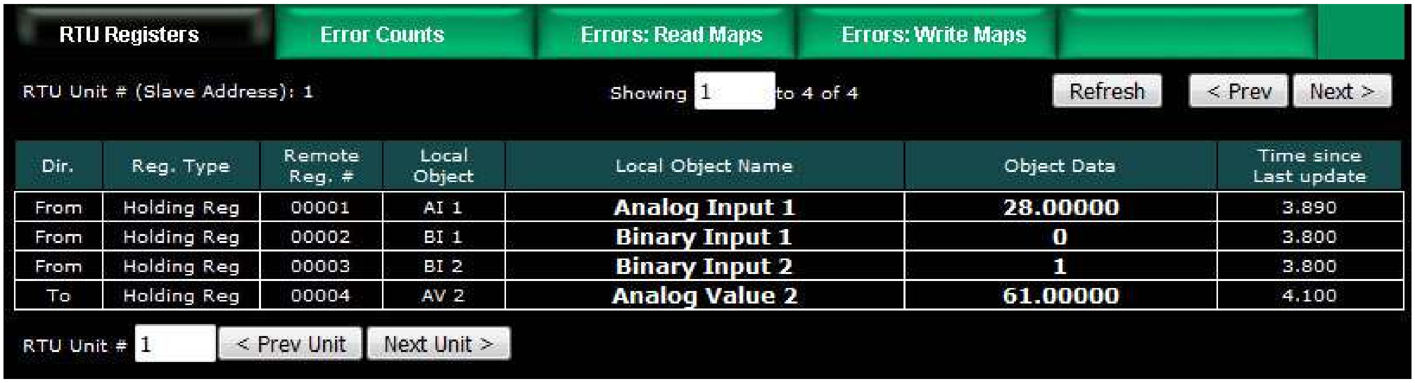
RTU Unit # (Slave Address): 2

Showing 1 to 0 of 0

Dir.	Reg. Type	Remote Reg. #	Local Object	Local Object Name	Object Data	Time since Last update
---	Undefined	---	---		---	---

RTU Unit # 2

Click the Next Unit button to go to the next RTU slave (or Prev Unit to back up), or simply enter a number in the RTU Unit # window and click Update. Registers for that slave will now be displayed. In addition to a summary of the map (both read and write maps are shown), the time since last update is displayed. This time should generally be less than the poll time. If the last update time is large, it means there is an error preventing the update.



RTU Unit # (Slave Address): 1

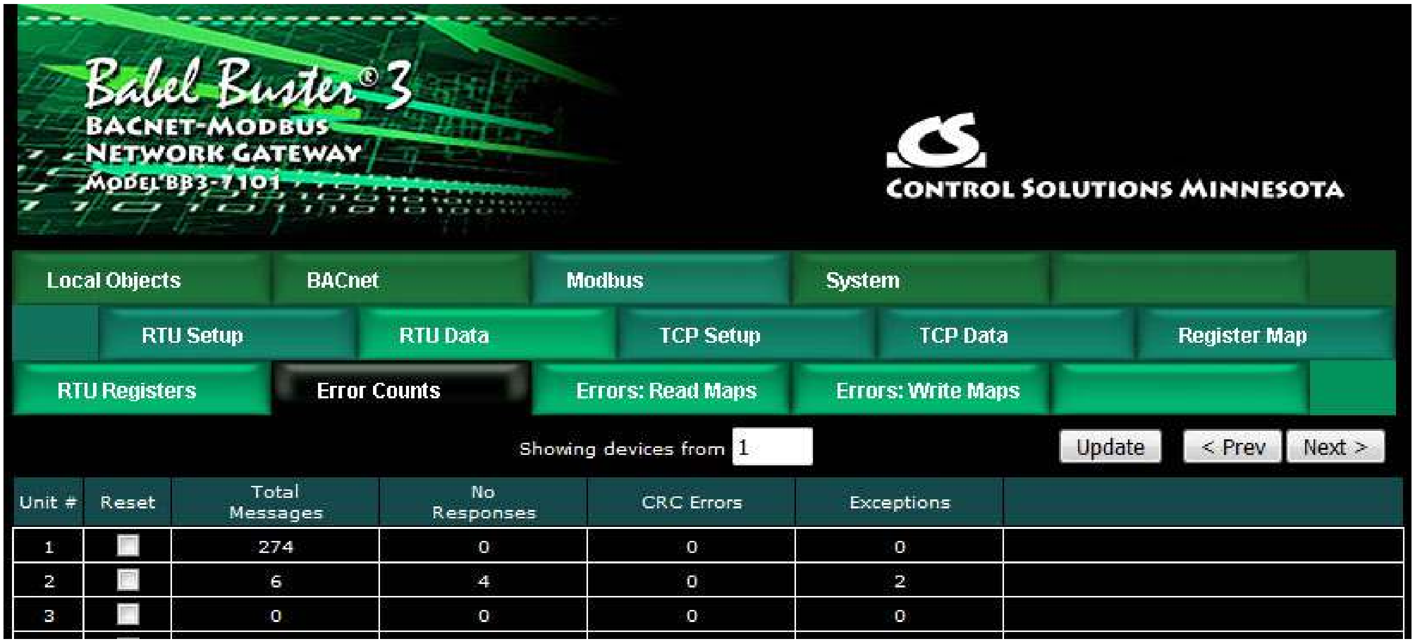
Showing 1 to 4 of 4

Dir.	Reg. Type	Remote Reg. #	Local Object	Local Object Name	Object Data	Time since Last update
From	Holding Reg	00001	AI 1	Analog Input 1	28.00000	3.890
From	Holding Reg	00002	BI 1	Binary Input 1	0	3.800
From	Holding Reg	00003	BI 2	Binary Input 2	1	3.800
To	Holding Reg	00004	AV 2	Analog Value 2	61.00000	4.100

RTU Unit # 1

8.5 Modbus RTU Errors

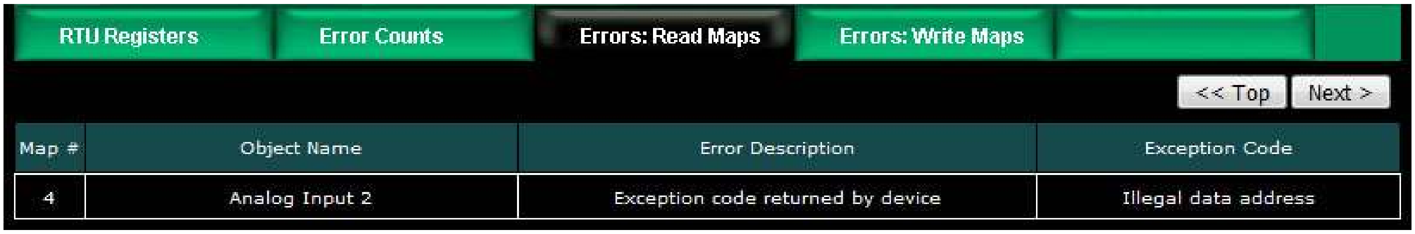
The Error Counts page shows a tabulation, by RTU slave, of all errors observed. In the example below, we can see that the RTU device at slave address 1 is running flawlessly while slave #2 has had some issues.



Showing devices from 1

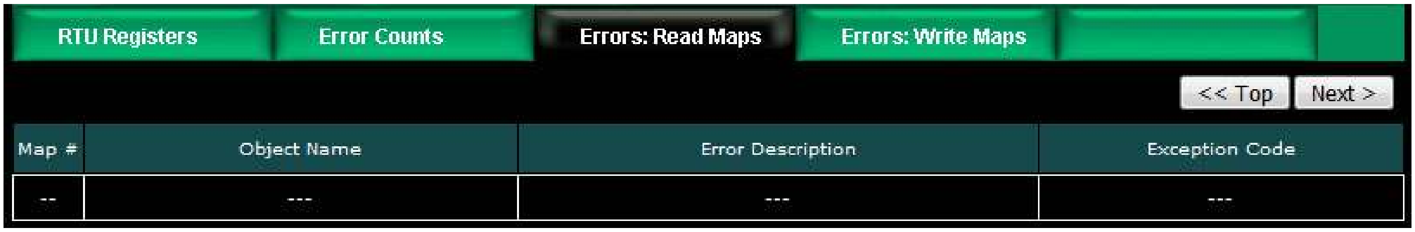
Unit #	Reset	Total Messages	No Responses	CRC Errors	Exceptions
1	□	274	0	0	0
2	□	6	4	0	2
3	□	0	0	0	0

If the counts show some problems, we can look for more detail on the Errors: Read Maps (or Errors: Write Maps) pages. These pages will tell us exactly which Read Map (or Write Map) the problem is occurring on, and what the error is, as illustrated below.



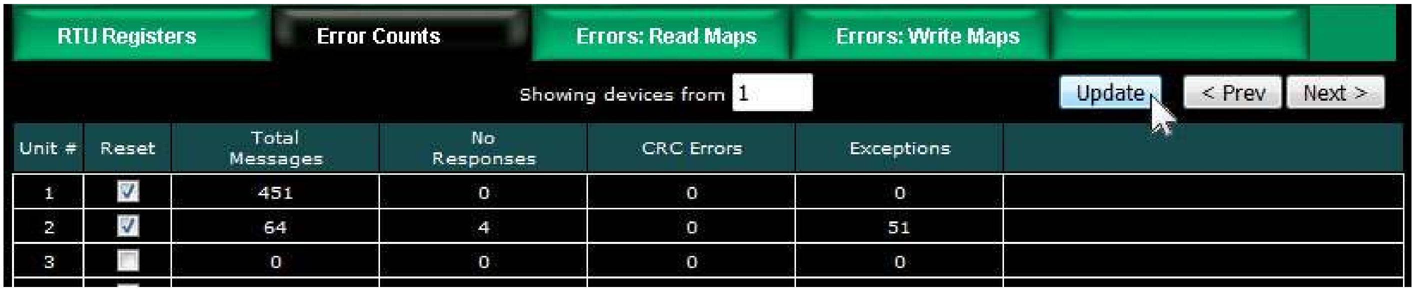
Map #	Object Name	Error Description	Exception Code
4	Analog Input 2	Exception code returned by device	Illegal data address

When the problem is resolved, it will be removed from this list.



Map #	Object Name	Error Description	Exception Code
--	---	---	---

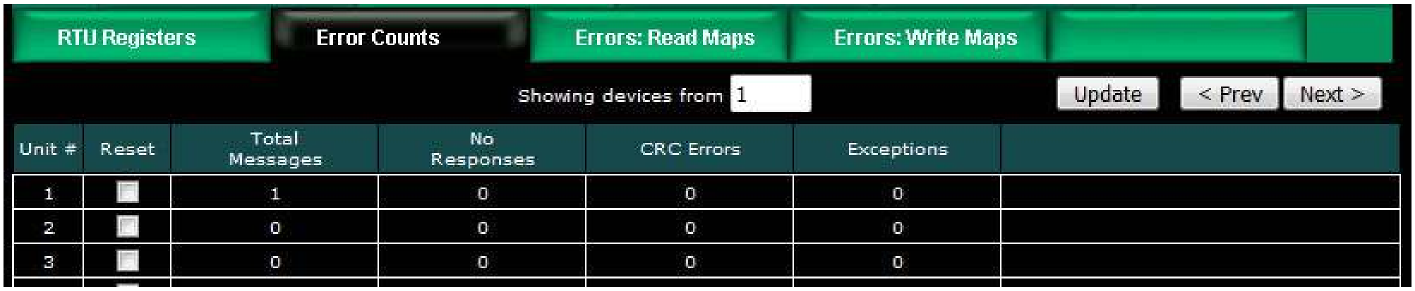
Once you have resolved problems, you can reset the error counts by checking the box in the Reset column and then clicking Update.



Showing devices from 1

Unit #	Reset	Total Messages	No Responses	CRC Errors	Exceptions
1	☑	451	0	0	0
2	☑	64	4	0	51
3	□	0	0	0	0

The counts will reset to zero, but in most cases, at least "Total Messages" will start incrementing again.



Showing devices from 1

Unit #	Reset	Total Messages	No Responses	CRC Errors	Exceptions
1	□	1	0	0	0
2	□	0	0	0	0
3	□	0	0	0	0

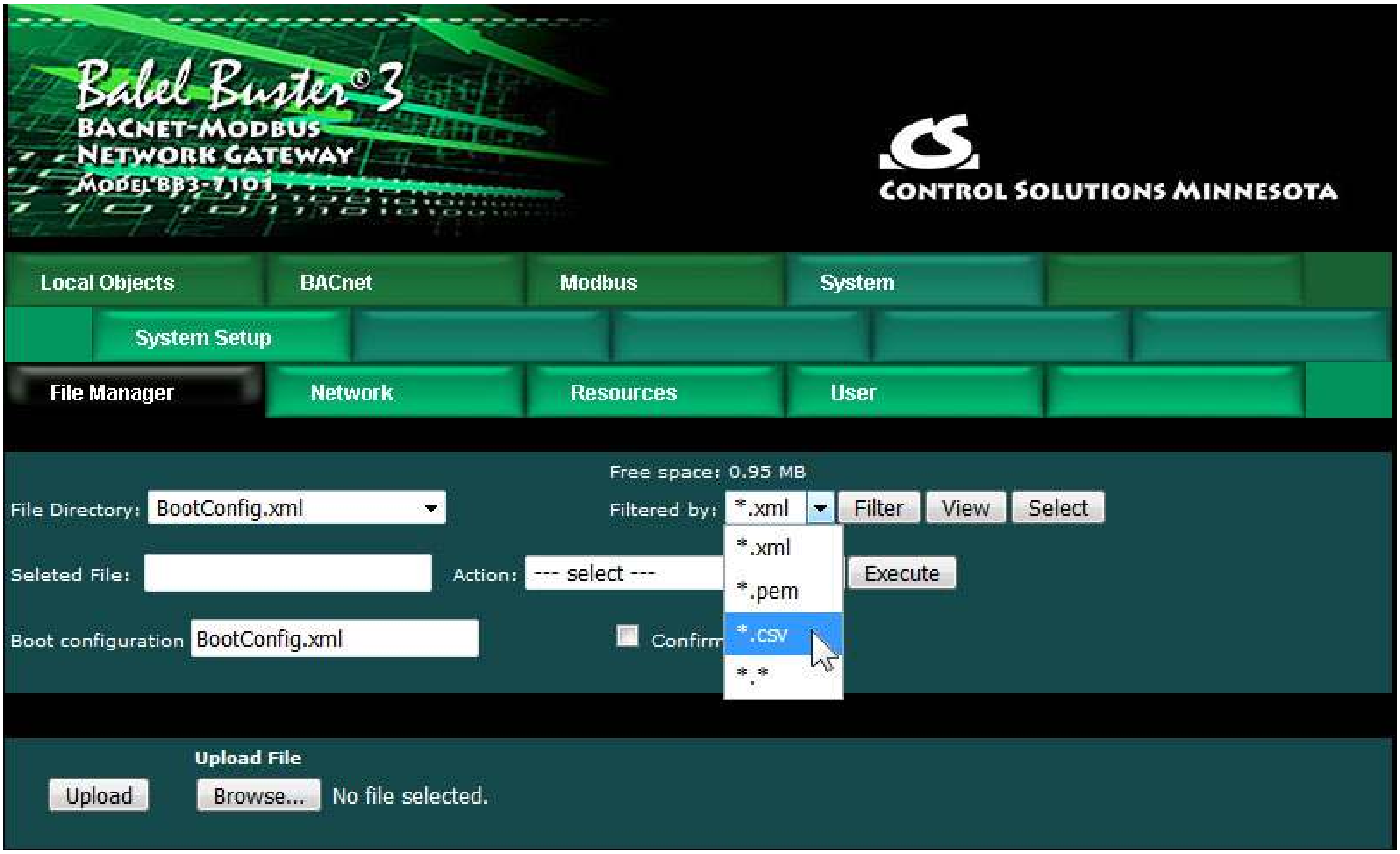
8.6 Importing Modbus RTU Maps from CSV File

The built-in web user interface is user friendly, but can get tedious if you have a lot of maps to enter. Quite often, Modbus register maps are available in spread sheet form. With a bit of editing, you can turn this into a CSV file that can be directly imported into the BB3-7101/MX-71 to quickly configure a lot of read and write maps. If you are proficient with spread sheets, you can probably create a rather large set of maps quickly and speed up the process of configuring the BB3-7101/MX-71.

There is more discussion about the File Manager in Section 11, but a summary of what you need to do to import maps from a CSV file is given here.

Start by uploading your CSV file. Use the Browse button to locate the file on your PC, then click Upload.

Select *.csv as the file filter. This will result in showing the list of CSV files currently stored in the BB3-7101/MX-71.



Free space: 0.95 MB

File Directory: BootConfig.xml

Filtered by: *.xml

Seleted File:

Action: --- select ---

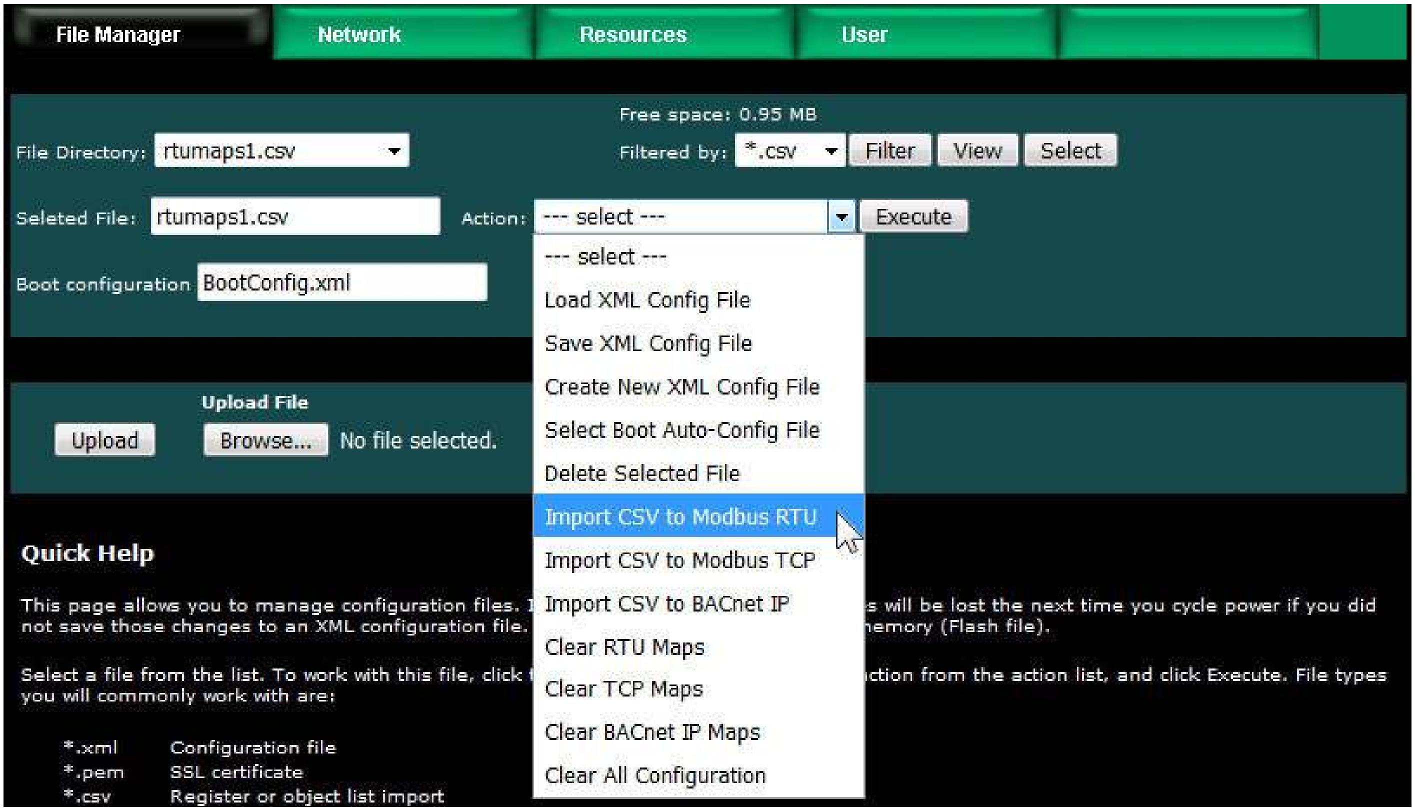
Boot configuration BootConfig.xml

□ Confirm

Upload File

No file selected.

Select your file from the File Directory drop-down list, then click the Select button on the right. Select "Import CSV to Modbus RTU" from the Action list, and click Execute.



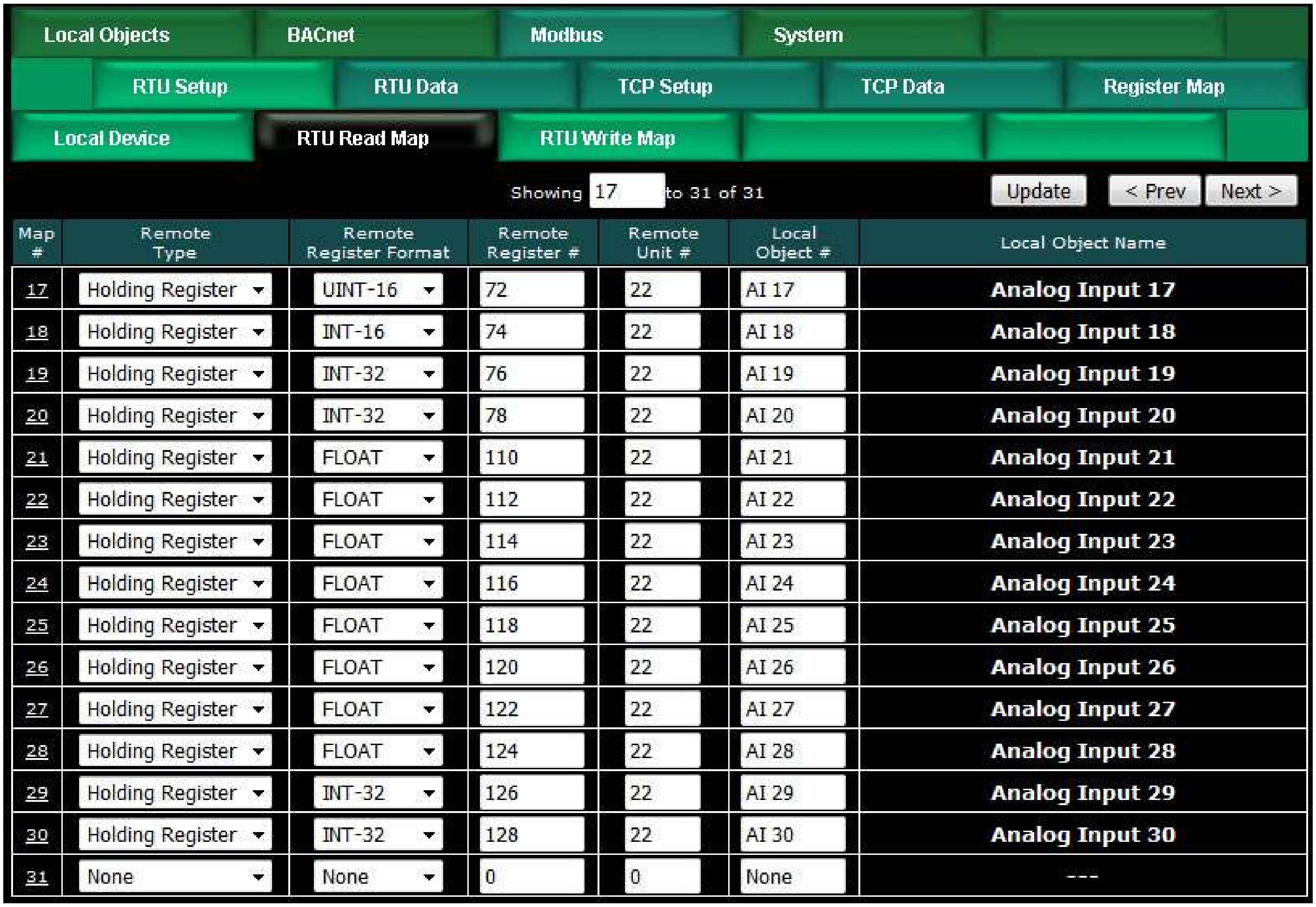
Quick Help

This page allows you to manage configuration files. I s will be lost the next time you cycle power if you did not save those changes to an XML configuration file. nemory (Flash file).

Select a file from the list. To work with this file, click t ction from the action list, and click Execute. File types you will commonly work with are:

*.xml	Configuration file
*.pem	SSL certificate
*.csv	Register or object list import

We imported 30 read maps in our example test case.

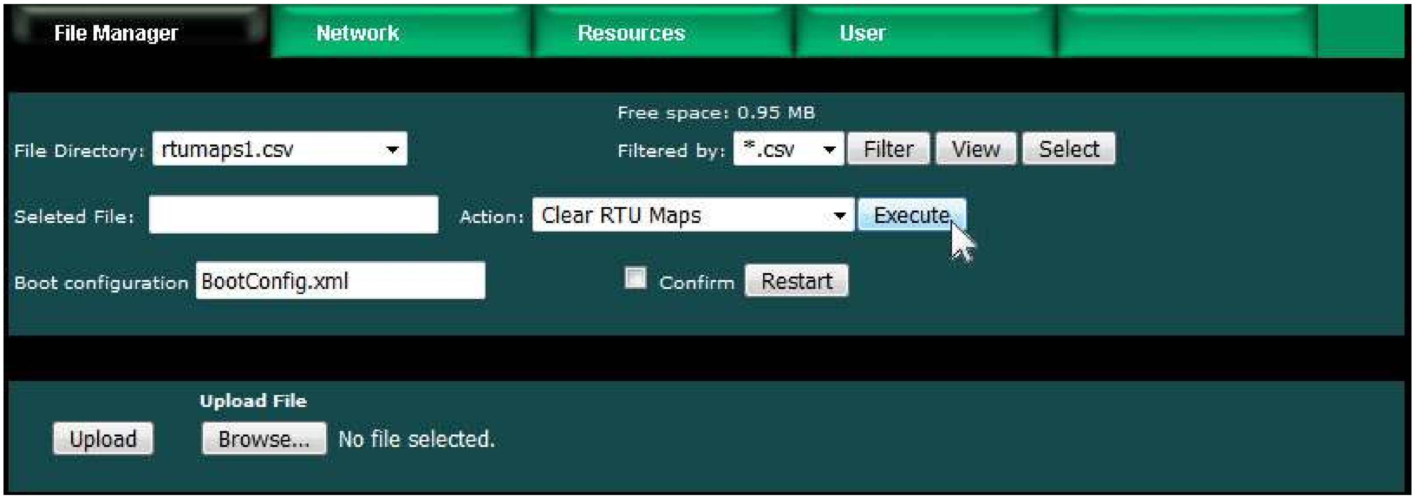


Showing 17 to 31 of 31

Map #	Remote Type	Remote Register Format	Remote Register #	Remote Unit #	Local Object #	Local Object Name
17	Holding Register	UINT-16	72	22	AI 17	Analog Input 17
18	Holding Register	INT-16	74	22	AI 18	Analog Input 18
19	Holding Register	INT-32	76	22	AI 19	Analog Input 19
20	Holding Register	INT-32	78	22	AI 20	Analog Input 20
21	Holding Register	FLOAT	110	22	AI 21	Analog Input 21
22	Holding Register	FLOAT	112	22	AI 22	Analog Input 22
23	Holding Register	FLOAT	114	22	AI 23	Analog Input 23
24	Holding Register	FLOAT	116	22	AI 24	Analog Input 24
25	Holding Register	FLOAT	118	22	AI 25	Analog Input 25
26	Holding Register	FLOAT	120	22	AI 26	Analog Input 26
27	Holding Register	FLOAT	122	22	AI 27	Analog Input 27
28	Holding Register	FLOAT	124	22	AI 28	Analog Input 28
29	Holding Register	INT-32	126	22	AI 29	Analog Input 29
30	Holding Register	INT-32	128	22	AI 30	Analog Input 30
31	None	None	0	0	None	---

8.7 Clearing Configuration

Read and write maps imported from a CSV file will be added to the list of maps already in place. If you wish to reload the list, you must first clear it. Clear the RTU maps by going to the File Manager page, then selecting "Clear RTU Maps" from the action list and clicking Execute.



Free space: 0.95 MB

File Directory: rtumaps1.csv

Filtered by: *.csv

Seleted File:

Action: Clear RTU Maps

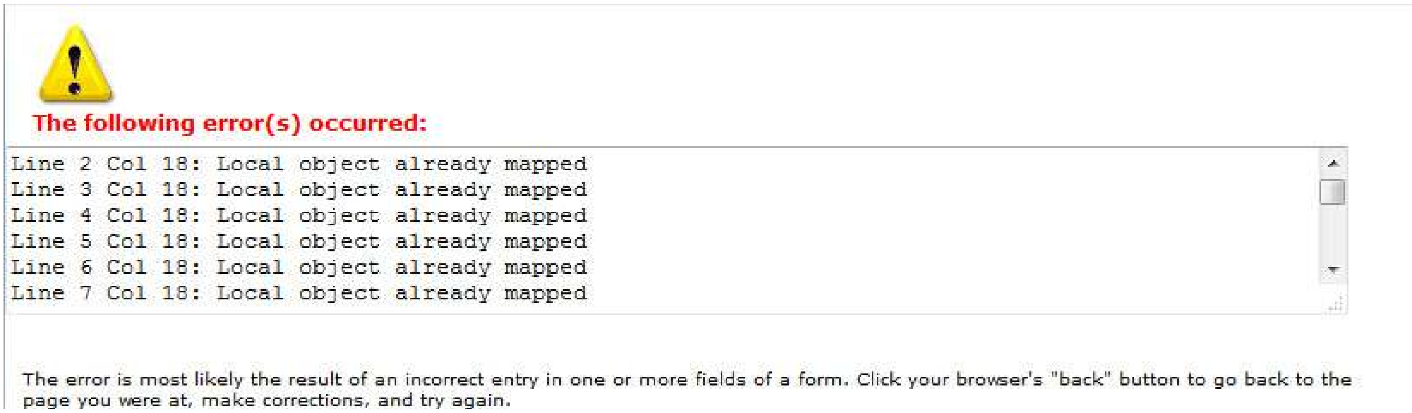
Boot configuration BootConfig.xml

□ Confirm

Upload File

No file selected.

If you forget to clear the maps before re-importing them, you will get an error notice something like this:



The following error(s) occurred:

```
Line 2 Col 18: Local object already mapped
Line 3 Col 18: Local object already mapped
Line 4 Col 18: Local object already mapped
Line 5 Col 18: Local object already mapped
Line 6 Col 18: Local object already mapped
Line 7 Col 18: Local object already mapped
```

The error is most likely the result of an incorrect entry in one or more fields of a form. Click your browser's "back" button to go back to the page you were at, make corrections, and try again.

9. Configuring Gateway as a Modbus TCP Client

The BB3-7101/MX-71 can be a Modbus TCP client and server. The terms client and server are more often used with Ethernet network devices, but for Modbus purposes, they still mean master and slave respectively. You must choose one or the other between master and slave for Modbus RTU, but Modbus TCP can be both simultaneously thanks to Ethernet.

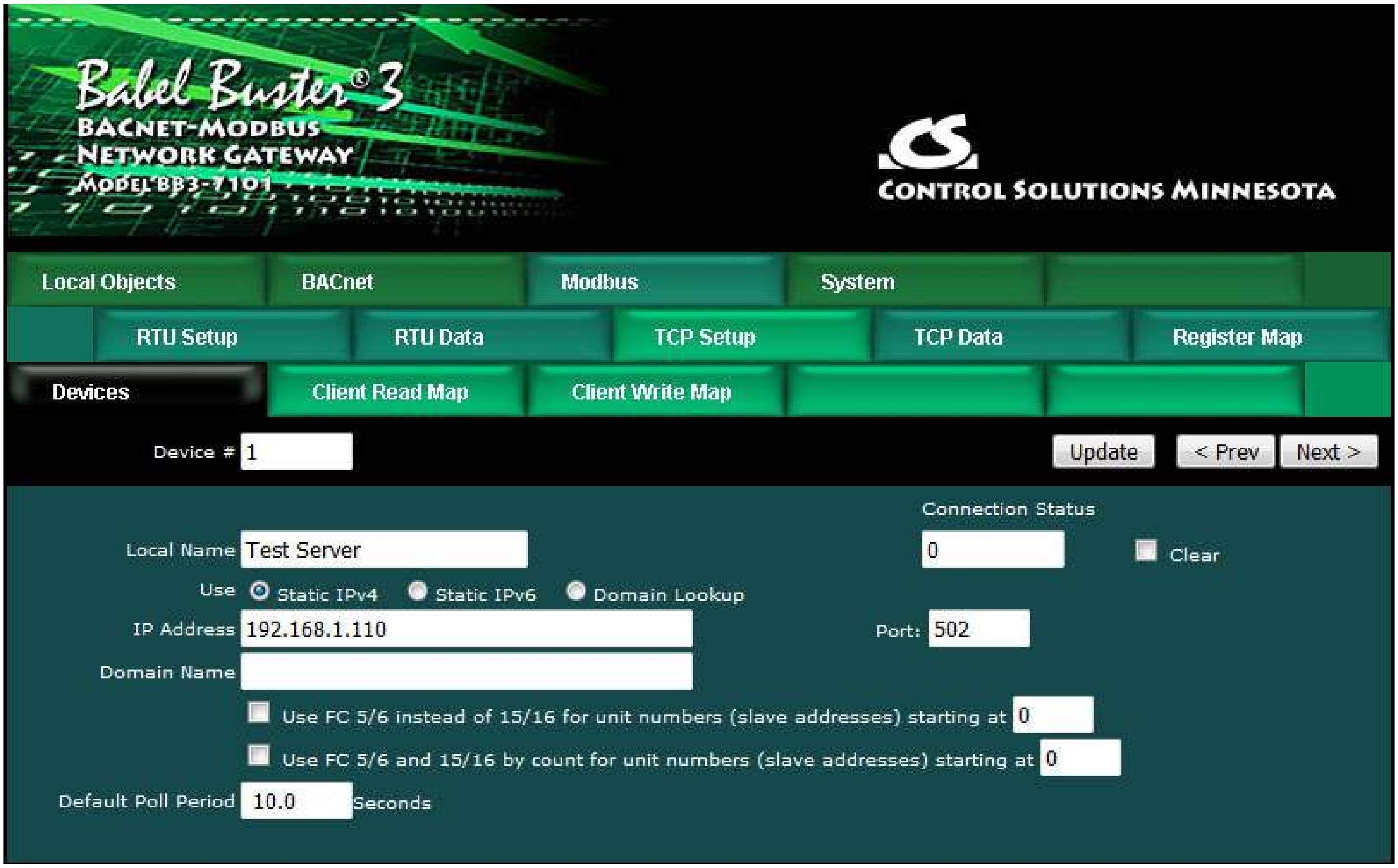
As a master (client) you can read Modbus data from, or write Modbus data to, other Modbus TCP devices. The gateway will periodically poll the other Modbus devices according to register maps you set up. To read from a remote Modbus device, configure a Read Map. To write to a remote Modbus device, configure a Write Map.

Data read from a remote device is stored in a local object when received. Data written to a remote device is taken from a local object when sent.

9.1 Modbus TCP Device Configuration

The Modbus TCP client is the Ethernet version of Modbus master. Modbus RTU requires a slave address. Modbus TCP also requires, in effect, a slave address. In the case of TCP, that slave address is an IP address (possibly along with a unit number). Since entering the IP address requires more effort than one simple slave number, and because internally a network socket is required per IP address, the TCP devices are set up in their own table.

For each Modbus TCP device that you wish to read and write, enter its IP address on the TCP Setup Devices page. The port is normally going to be 502 (the standard Modbus TCP port), but if it is different, enter that number here. You have the option of referring to a Modbus TCP device by domain name. If you use a domain name, be sure that domain can be found at the DNS servers provided on the Network setup page.



Device # 1

Connection Status 0 □ Clear

Local Name Test Server

Use ● Static IPv4 ○ Static IPv6 ○ Domain Lookup

IP Address 192.168.1.110 Port: 502

Domain Name

□ Use FC 5/6 instead of 15/16 for unit numbers (slave addresses) starting at 0

□ Use FC 5/6 and 15/16 by count for unit numbers (slave addresses) starting at 0

Default Poll Period 10.0 Seconds

A unit number is always included in each Modbus TCP packet. This is the equivalent of the RTU slave address. Some TCP devices pay no attention to unit and simply echo back whatever you had sent. However, if you are accessing RTU devices on the other side of a TCP to RTU router (gateway), then the unit does become the RTU slave address on the RTU side of the gateway and multiple RTU devices are accessed at the same TCP IP address. Unit numbers are entered on each individual read or write map.

The default poll rate entered here will be used for all Modbus TCP Read and Write maps unless a different number is entered in the expanded view of the map.

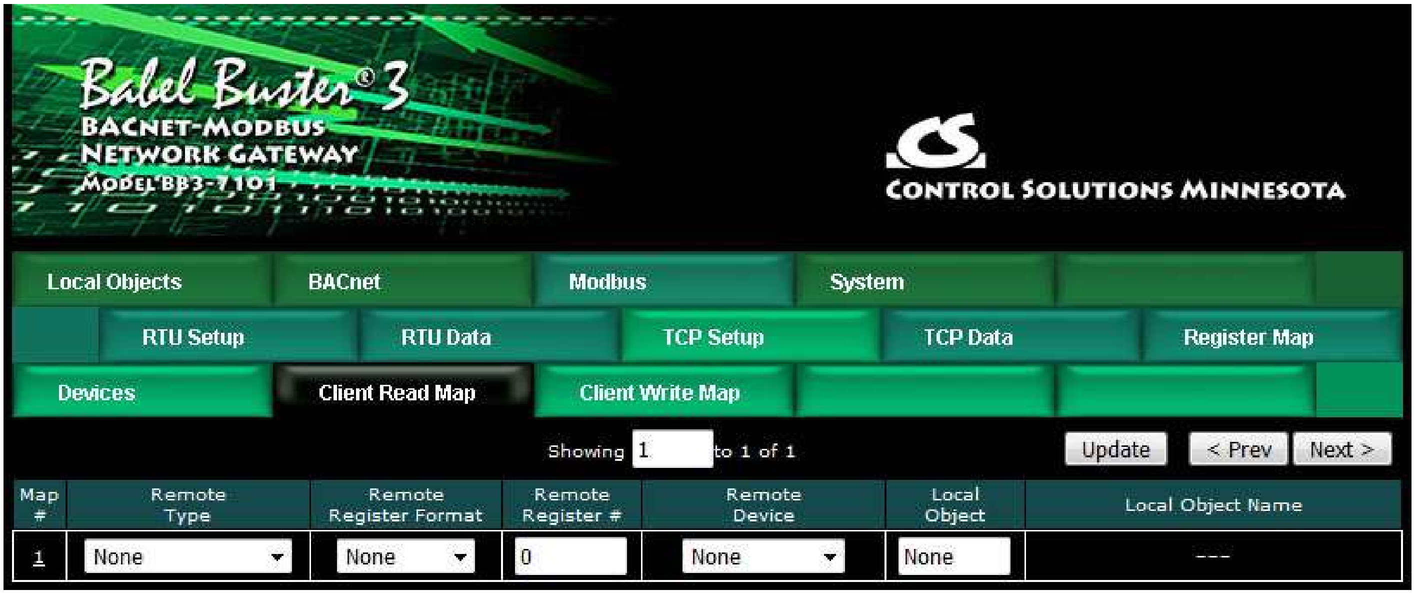
If your slave/server device only supports function codes 5 and 6 for writing coils and holding registers, check the Use FC 5/6 box. The default function codes are 15 and 16, which are most widely used. If you check the box, you should also enter a "starting at" unit # or slave address. This allows supporting both types of devices at the same time provided you assign slave addresses in two non-overlapping groups. (These settings do not apply if the gateway is the slave.)

If your slave/server device is especially picky, it may require function codes 5 and 6 for single writes, but yet expect function codes 15 and 16 for multiple register writes. Select "Use FC 5/6 by count..." if this is the case, and provide a starting unit number.

The starting unit number will most often be simply "1" unless you have a mix of device types that all follow different rules for their function codes. Note that the selection of FC 5/6 only pertains to write maps. Reading registers will always use the same function codes for reading.

9.2 Modbus TCP Client Read Maps

Getting the gateway to read registers from a Modbus TCP device requires setting up a "Read Map" as shown here.

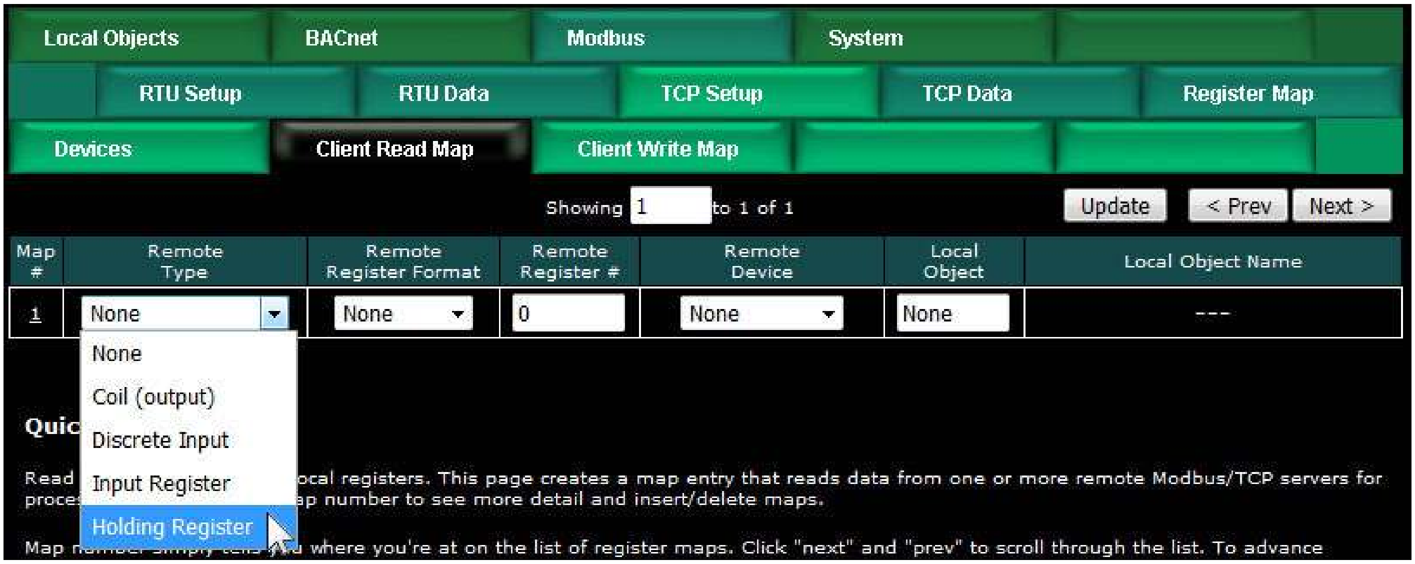


Showing 1 to 1 of 1

Map #	Remote Type	Remote Register Format	Remote Register #	Remote Device	Local Object	Local Object Name
1	None	None	0	None	None	---

Map number simply tells you where you're at on the list of read maps. Click "next" and "prev" to scroll through the list. To advance directly to a specific map, enter the desired number in the "Showing" box, then click Update.

Maps entered on this page only read data from remote devices. Go to the TCP Write Map to write data to those devices. The full parameter set is different for read versus write.



Showing 1 to 1 of 1

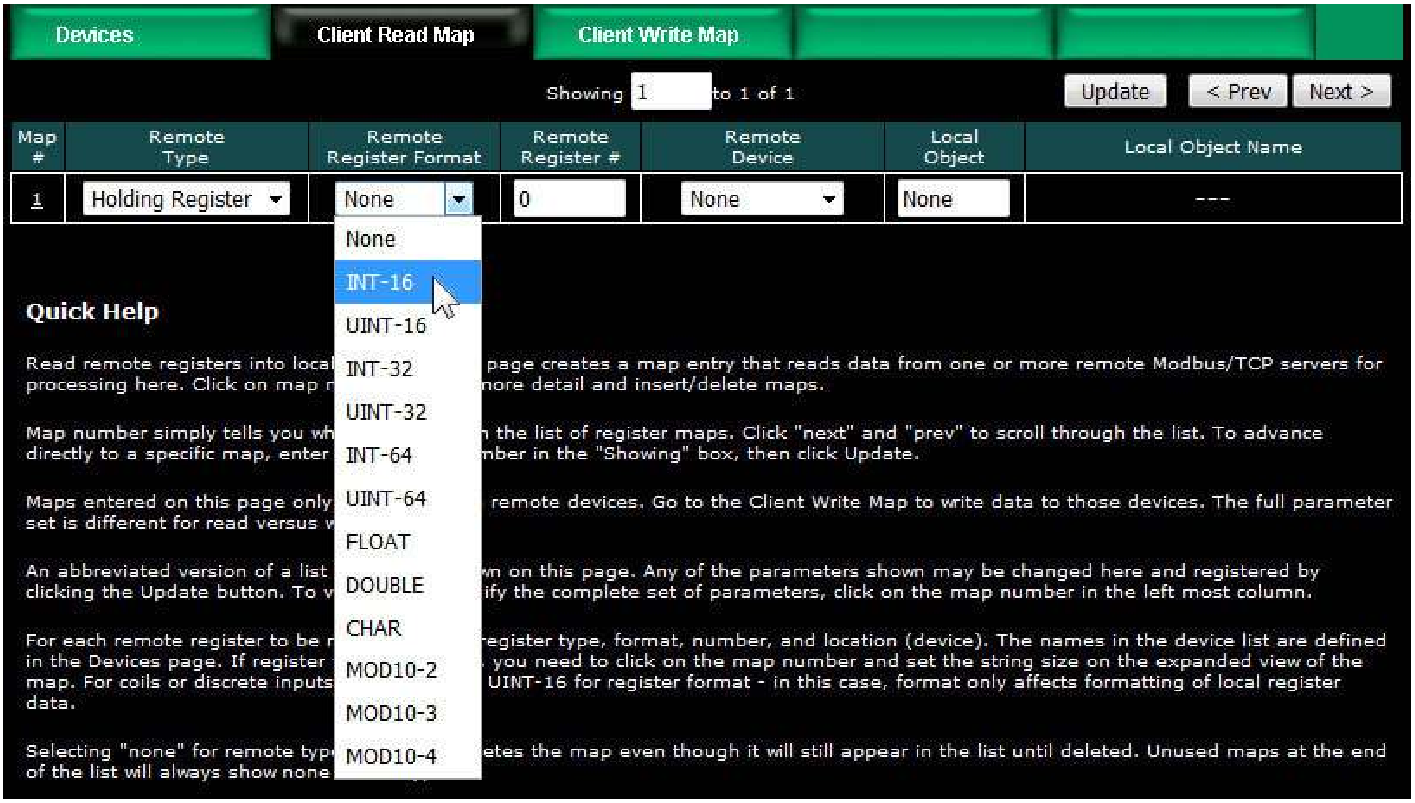
Map #	Remote Type	Remote Register Format	Remote Register #	Remote Device	Local Object	Local Object Name
1	None	None	0	None	None	---

Quic

Read ocal registers. This page creates a map entry that reads data from one or more remote Modbus/TCP servers for proce ap number to see more detail and insert/delete maps.

Map u where you're at on the list of register maps. Click "next" and "prev" to scroll through the list. To advance

To create a Read Map, start by selecting the Modbus register type to read from the drop-down list.



Showing 1 to 1 of 1

Map #	Remote Type	Remote Register Format	Remote Register #	Remote Device	Local Object	Local Object Name
1	Holding Register	None	0	None	None	---

Quick Help

Read remote registers into local page creates a map entry that reads data from one or more remote Modbus/TCP servers for processing here. Click on map n nore detail and insert/delete maps.

Map number simply tells you wh the list of register maps. Click "next" and "prev" to scroll through the list. To advance directly to a specific map, enter nber in the "Showing" box, then click Update.

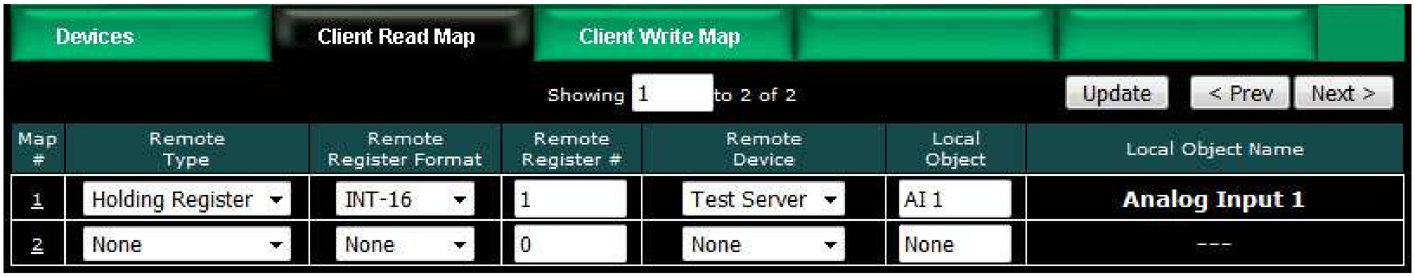
Maps entered on this page only remote devices. Go to the Client Write Map to write data to those devices. The full parameter set is different for read versus w

An abbreviated version of a list vn on this page. Any of the parameters shown may be changed here and registered by clicking the Update button. To v ify the complete set of parameters, click on the map number in the left most column.

For each remote register to be r egister type, format, number, and location (device). The names in the device list are defined in the Devices page. If register you need to click on the map number and set the string size on the expanded view of the map. For coils or discrete inputs UINT-16 for register format - in this case, format only affects formatting of local register data.

Selecting "none" for remote typ etes the map even though it will still appear in the list until deleted. Unused maps at the end of the list will always show none

Select the data format expected in the remote Modbus register. The abbreviation INT under Format means signed integer, while UINT represents unsigned integer. The INT and UINT are followed by the number of bits to be read (which translates into 1, 2, or 4 consecutive holding registers). The FLOAT format refers to 32-bit IEEE 754 format while DOUBLE refers to 64-bit IEEE 754 floating point. The MOD10 format is unique to Schneider Electric power meters, and is supported in 2, 3, and 4-register formats. (Note: Use INT-16 or UINT-16 for coils or discrete inputs - in this case format only affects local data conversion.)



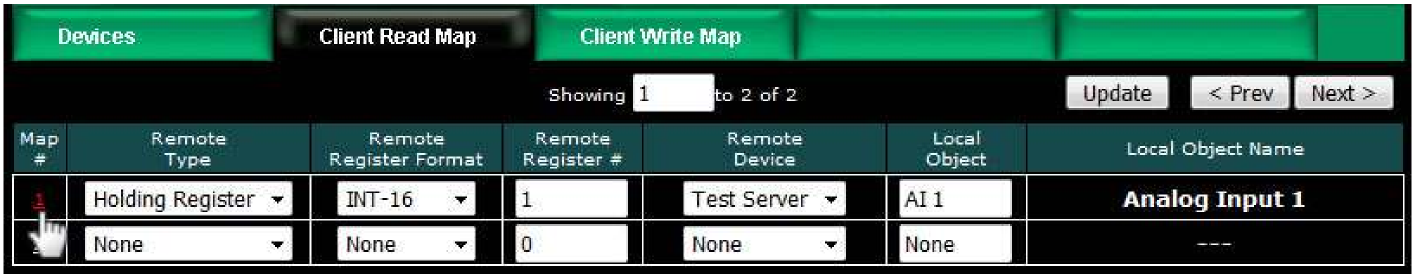
Showing 1 to 2 of 2

Map #	Remote Type	Remote Register Format	Remote Register #	Remote Device	Local Object	Local Object Name
1	Holding Register	INT-16	1	Test Server	AI 1	Analog Input 1
2	None	None	0	None	None	---

Enter the register number to read from the remote TCP server. Do not use Modicon numbers here. In other words, if your device's documentation says read register 40001, that is short hand (Modicon notation) for saying read holding register 1. Refer to the Modbus Reference Information section of this user guide for more discussion about register numbers like 40001. If you enter 40001 here to read the first holding register, you will get an exception error since the actual register number is not 40001.

Select a TCP device from the list that this register should be read from. Only devices entered on the Devices page will appear in the list.

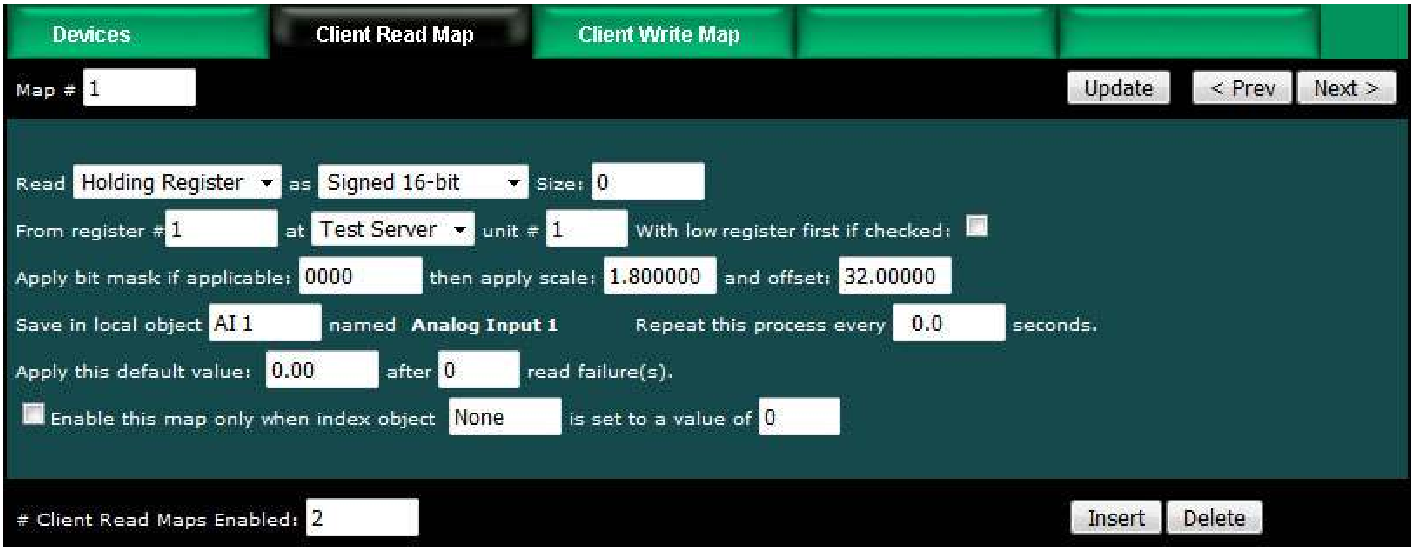
The Local Object is where data read from the remote Modbus TCP server will be stored locally in the Babel Buster. If the local object data format does not match what you are reading from the Modbus device, the data will be converted automatically when it is read.



Showing 1 to 2 of 2

Map #	Remote Type	Remote Register Format	Remote Register #	Remote Device	Local Object	Local Object Name
1	Holding Register	INT-16	1	Test Server	AI 1	**Analog Input 1**
2	None	None	0	None	None	---

Click on the Map number in the first column to access the expanded view of the Read Map.



Map # 1

Read Holding Register as Signed 16-bit Size: 0

From register # 1 at Test Server unit # 1 With low register first if checked: □

Apply bit mask if applicable: 0000 then apply scale: 1.800000 and offset: 32.00000

Save in local object AI 1 named **Analog Input 1** Repeat this process every 0.0 seconds.

Apply this default value: 0.00 after 0 read failure(s).

□ Enable this map only when index object None is set to a value of 0

Client Read Maps Enabled: 2

For each remote register to be read, select the register type, format, number. Select a TCP server device from the list to read from. Only devices entered on the Devices page will appear here. If a unit number other than the default unit entered for this TCP server on the Devices page should be used, enter that unit number here.

The optional bit mask and scaling are discussed with below.

Modbus protocol treats all input registers or holding registers as strictly 16-bit registers. To accommodate 32-bit or longer data, Modbus devices use multiple consecutive "registers" to hold the data. There is no standardization of whether the least significant part of the data comes first or last. Therefore, Babel Buster lets you set that according to whatever the remote Modbus device requires. If the least significant data is found in the first (or lower numbered) register in your Modbus device, then check the box after "With low register first".

The poll rate ("Repeat this process...") determines how often the remote register will be read. If zero is entered here, the rate will become the default poll rate given on the Devices page for the Modbus TCP device selected.

The default value will be stored into the local object after the given number of read failures if the fail count is non-zero. Setting the count to zero will disable the default, and the object will retain the most recent value obtained. If the default value does take effect, the actual data value read will be retained when communications are restored.

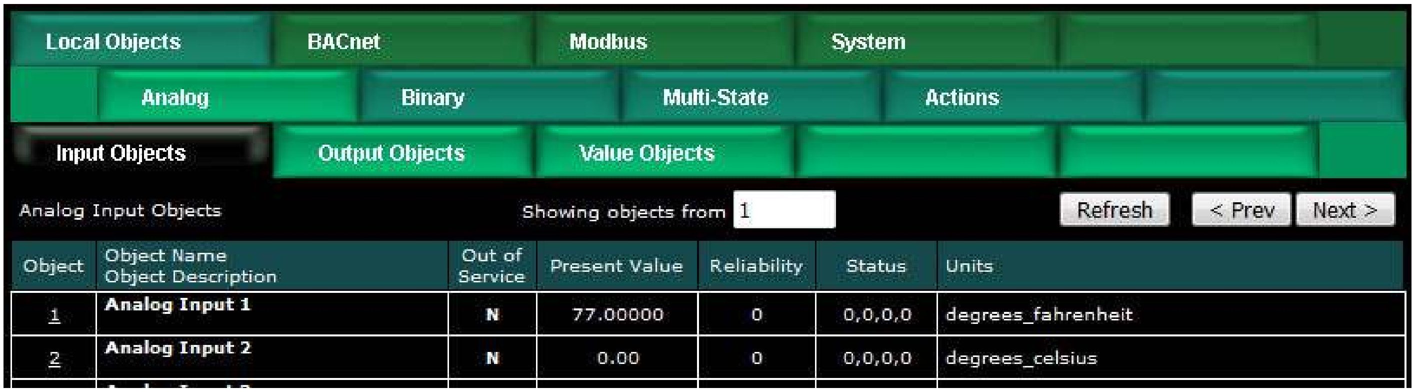
You have the option of making this Read Map conditional. If an index object is provided and the Enable box is checked, then this read map will only be executed when the index object contains the value given. This allows multiple read maps to supply data to the same local object based on the value of the index object. It also allows reading to simply be suspended if a single read map supplies data to the local object. In a more sophisticated scenario, you could potentially suspend reading of the remote Modbus device if you know the device is powered down.

Map number simply tells you where you're at on the list of read maps. Click "next" and "prev" to scroll through the list. To advance directly to a specific map, enter the desired number in the "Map #" box, then click Update.

Delete will remove the map number shown in the "Map #" box. Insert will insert a new map before the map number shown, and is used for placing maps between existing maps. It is not necessary to use Insert to add maps to the bottom of the list - simply add definition to the empty map already at the end of the list.

Selecting "none" for remote type effectively deletes the map even though it will still appear in the list until deleted. Unused maps at the end of the list will always show none as the type. If you wish to prevent these from being displayed, reduce the number of maps enabled.

The number of maps enabled simply limits the scope of map review so that you do not have to review a lot of unused maps. If the displayed maps are used up and you need more, increase the enabled number.



Analog Input Objects

Showing objects from 1

Object	Object Name Object Description	Out of Service	Present Value	Reliability	Status	Units
1	**Analog Input 1**	N	77.00000	0	0,0,0,0	degrees_fahrenheit
2	**Analog Input 2**	N	0.00	0	0,0,0,0	degrees_celsius

You have the option of providing a scale and offset. A scale of zero will cause scale and offset to be ignored. If provided, the Modbus data will be treated as raw data. When the Modbus data is received, it will be multipled by scale, then added to offset, and then stored in the local object. If the Modbus device was providing degrees Celsius, and the scale factors illustrated above were used, then a Modbus value of 25 would result in the local object receiving a value of 77 (degrees Fahrenheit).

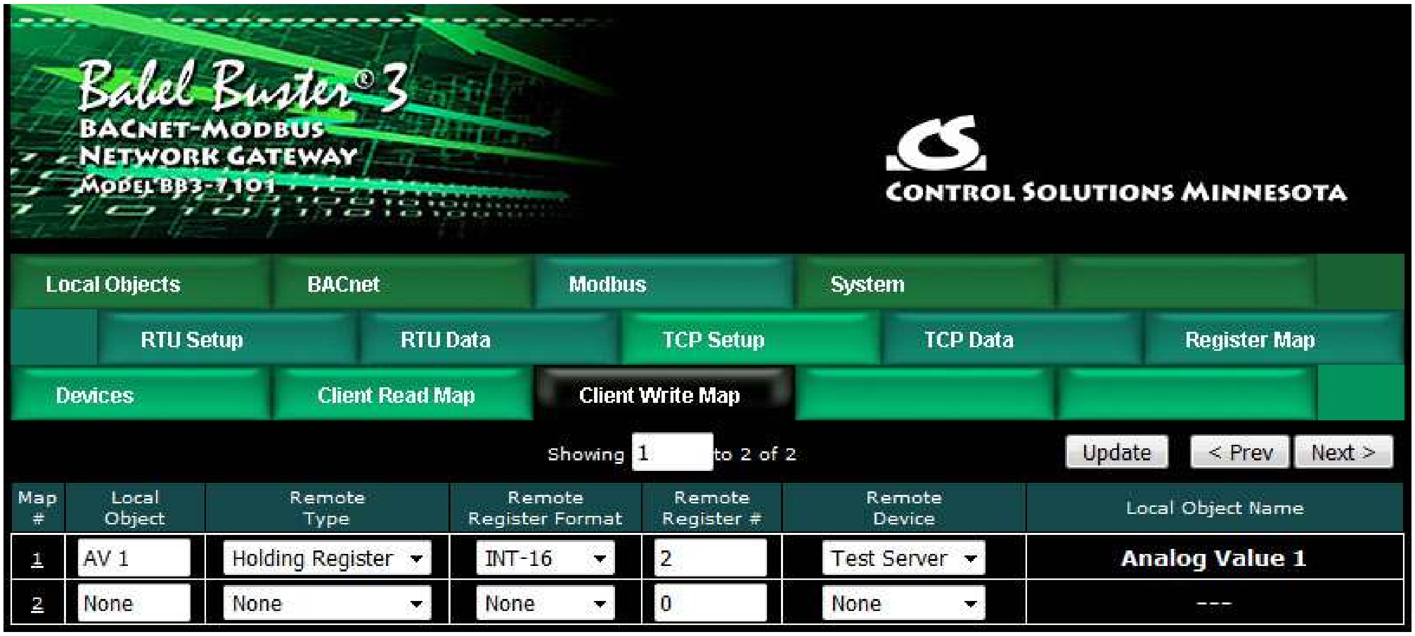
It is common for Modbus devices to pack a number of status bits into a single holding register. In order to do meaningful things based on a single bit, such as generate a COV on state change of a single bit, it is necessary to split that register into multiple local objects. Babel Buster supports this requirement by providing an optional bit mask.

If a bit mask is entered (in hexadecimal), and the remote register type is signed or unsigned integer (16-bit or 32-bit data), the mask will be bit-wise logical AND-ed with the Modbus data, and the retained bits will be right justified in the result stored locally. Refer to the Modbus Reference section in this user guide for a list of all possible mask values.

If the read maps referencing the same remote register are created in sequential contiguous order, the Babel Buster will optimize the TCP activity by reading the remote register once and then sharing the data with all of the read maps in the group. The example illustrated for Modbus RTU in section 8.2 works exactly the same for TCP. In that example, two consecutive read maps reference the same remote register, each selecting a different bit.

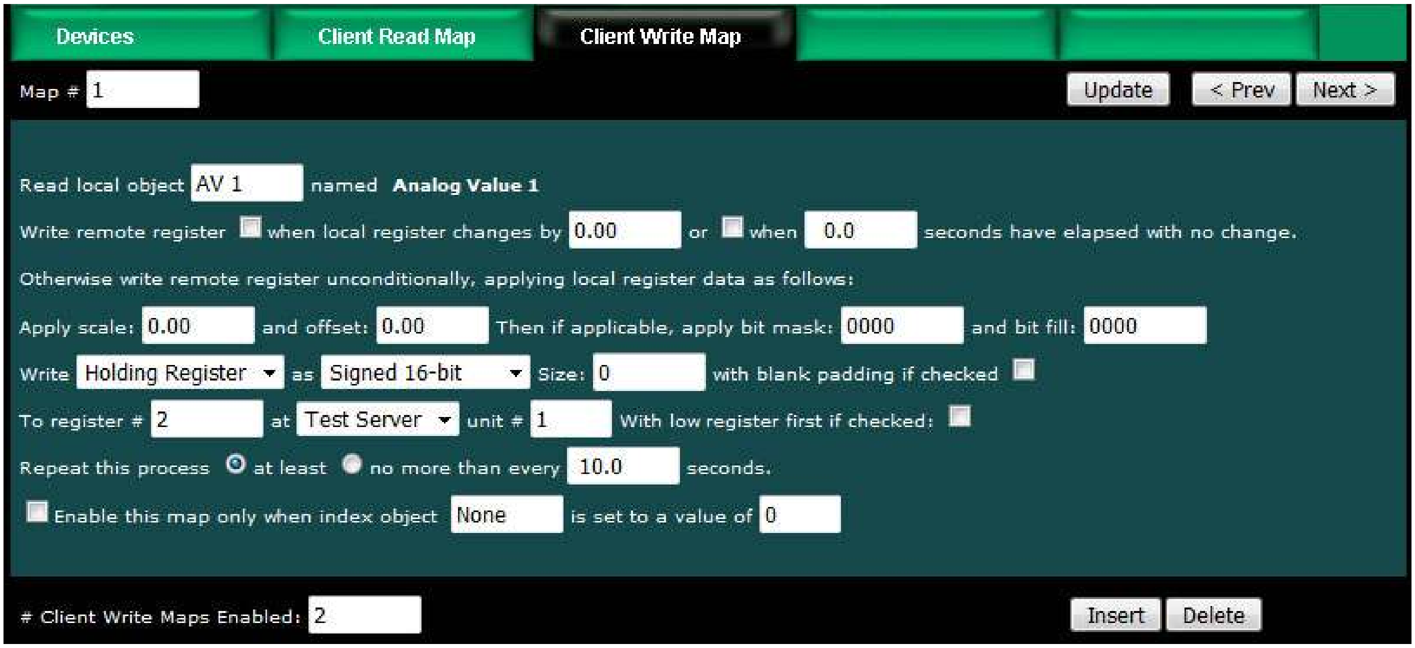
9.3 Modbus TCP Client Write Maps

Getting the gateway to write registers to a Modbus TCP device requires setting up a "Write Map" as shown here. Much of the Write Map is configured the same as a Read Map.



Map #	Local Object	Remote Type	Remote Register Format	Remote Register #	Remote Device	Local Object Name
1	AV 1	Holding Register	INT-16	2	Test Server	Analog Value 1
2	None	None	None	0	None	---

The data direction is reversed but the same selections are still made. Select the local object that will be the source of data to write to the remote Modbus TCP device. Select the register type, data format, and register number to be written to in that device. Select a TCP server device from the list to write to. Only devices entered on the Devices page will appear here. If a unit number other than the default unit entered for this TCP server on the Devices page should be used, enter that unit number here. Click on the map number in the first column to access additional optional configuration parameters.



Map # 1

Read local object AV 1 named **Analog Value 1**

Write remote register □ when local register changes by 0.00 or □ when 0.0 seconds have elapsed with no change.

Otherwise write remote register unconditionally, applying local register data as follows:

Apply scale: 0.00 and offset: 0.00 Then if applicable, apply bit mask: 0000 and bit fill: 0000

Write Holding Register as Signed 16-bit Size: 0 with blank padding if checked □

To register # 2 at Test Server unit # 1 With low register first if checked: □

Repeat this process ● at least ○ no more than every 10.0 seconds.

□ Enable this map only when index object None is set to a value of 0

Client Write Maps Enabled: 2

The local object data may be written to the remote device periodically, or when it changes, or both. To send upon change (send on delta), check the first box and enter the amount by which the local object must change before being written to the remote device. To guarantee that the remote register will be written at least occasionally even if the data does not change, check the second box and enter some amount of time. This time period will be referred to as the "maximum quiet time".

Data from the local object may be manipulated before being written to the remote register. The local data is first multiplied by the scale factor. The offset is then added to it. If a bit mask is entered, and the remote register type is signed or unsigned integer (16-bit or 32-bit data), the mask will be bit-wise logical AND-ed with the data. The mask is right justified, then AND-ed with the data. The result is then left shifted back to the original position of the mask. In other words, the least significant bits of the original data will be stuffed at the position marked by the mask.

After the scaling and masking, the bit fill will be logically OR-ed into the result, but only if the mask was nonzero and was used. Both mask and fill are entered in hexadecimal. The effect of "fill" is that certain bits will always be set to 1 in the data written to the remote Modbus device.

Multiple local objects may be packed into a single remote register. To accomplish this, define two or more maps in sequence with the same remote destination. If the destination is the same, data types are 16 or 32-bit integer (signed or unsigned), bit masks are nonzero, and the maps are sequential, the results of all qualifying maps will be OR-ed together before being sent to the remote destination.

For the remote register to be written, select the register type, format, number, select a remote TCP device from the list, and enter a unit number if the default unit number for that device should not be used. Data formats are the same as described above for Read Maps. Size is only specified for character strings. Use INT-16 or UINT-16 data format for coils - in this case format only affects local data conversion.

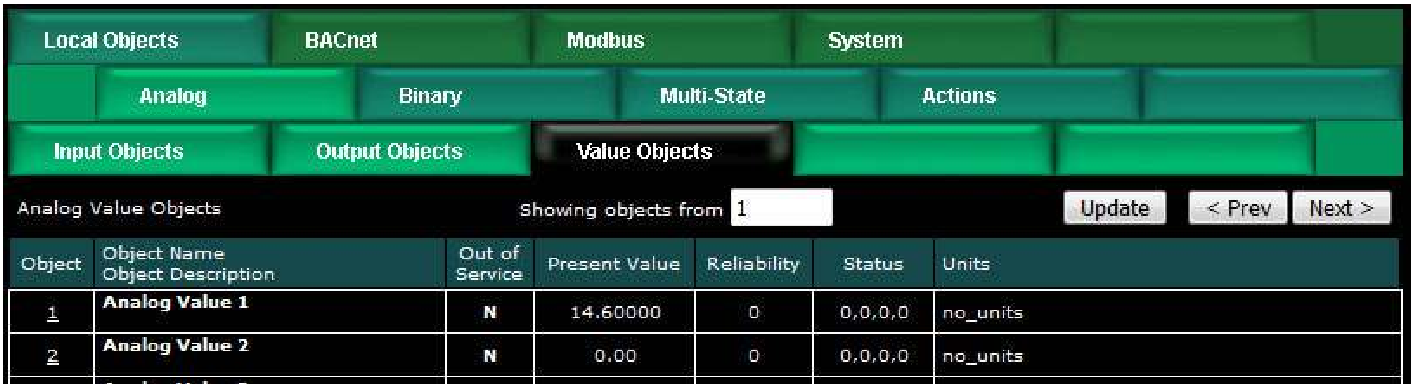
Modbus protocol treats all holding registers as strictly 16-bit registers. To accommodate 32-bit or longer data, Modbus devices use multiple consecutive "registers" to hold the data. There is no standardization of whether the least significant part of the data comes first or last. Therefore, Babel Buster lets you set that according to whatever the remote Modbus device requires. If the least significant data is found in the first (or lower numbered) register in your Modbus device, then check the box after "With low register first".

The repeat time may determine how often the remote register will be written. If send on delta and maximum quiet time are not checked above, clicking the "at least" button will establish a periodic update time. If send on delta is used and you wish to limit the network traffic in the event changes are frequent, click the "no more than" button and enter the minumum time that should elapse before another write to the remote device. It is valid to select "no more than every 0.0 seconds" if you want all changes to be sent, but no periodic writes.

You have the option of making this Write Map conditional. If an index object is provided and the Enable box is checked, then this write map will only be executed when the index object contains the value given. This allows multiple write maps to supply data to the same remote register based on the value of the local index object. It also allows writing to simply be suspended if a single write map supplies data to the remote register. In a more sophisticated scenario, you could potentially suspend writing of the remote device if you know the device is powered down.

Delete will remove the map number shown in the "Map #" box. Insert will insert a new map before the map number shown, and is used for placing maps between existing maps. It is not necessary to use Insert to add maps to the bottom of the list - simply add definition to the empty map already at the end of the list.

Selecting "none" for remote type effectively deletes the map even though it will still appear in the list until deleted. Unused maps at the end of the list will always show none as the type. If you wish to prevent these from being displayed, reduce the number of maps enabled.

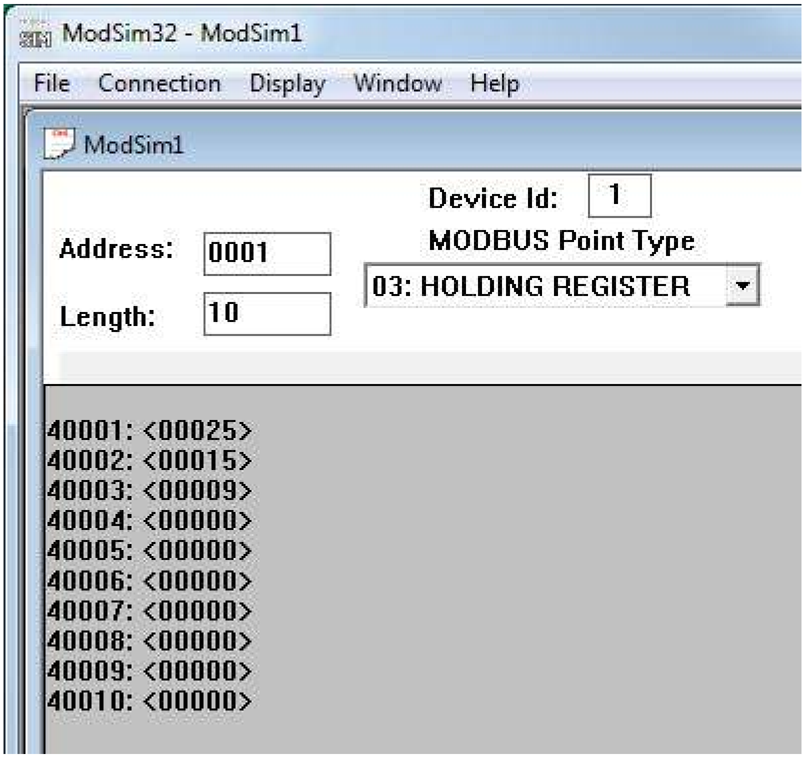


Analog Value Objects

Showing objects from 1

Object	Object Name Object Description	Out of Service	Present Value	Reliability	Status	Units
1	**Analog Value 1**	N	14.60000	0	0,0,0,0	no_units
2	**Analog Value 2**	N	0.00	0	0,0,0,0	no_units

If the local object and remote register are not the same format, then data is converted automatically when written. In the example above, the Write Map is Analog Value 1, a floating point value, to remote register 2, an unsigned 16-bit value. The data is converted and rounded up, as illustrated below by ModSim acting as our Modbus TCP server here.



Device Id: 1

MODBUS Point Type: 03: HOLDING REGISTER

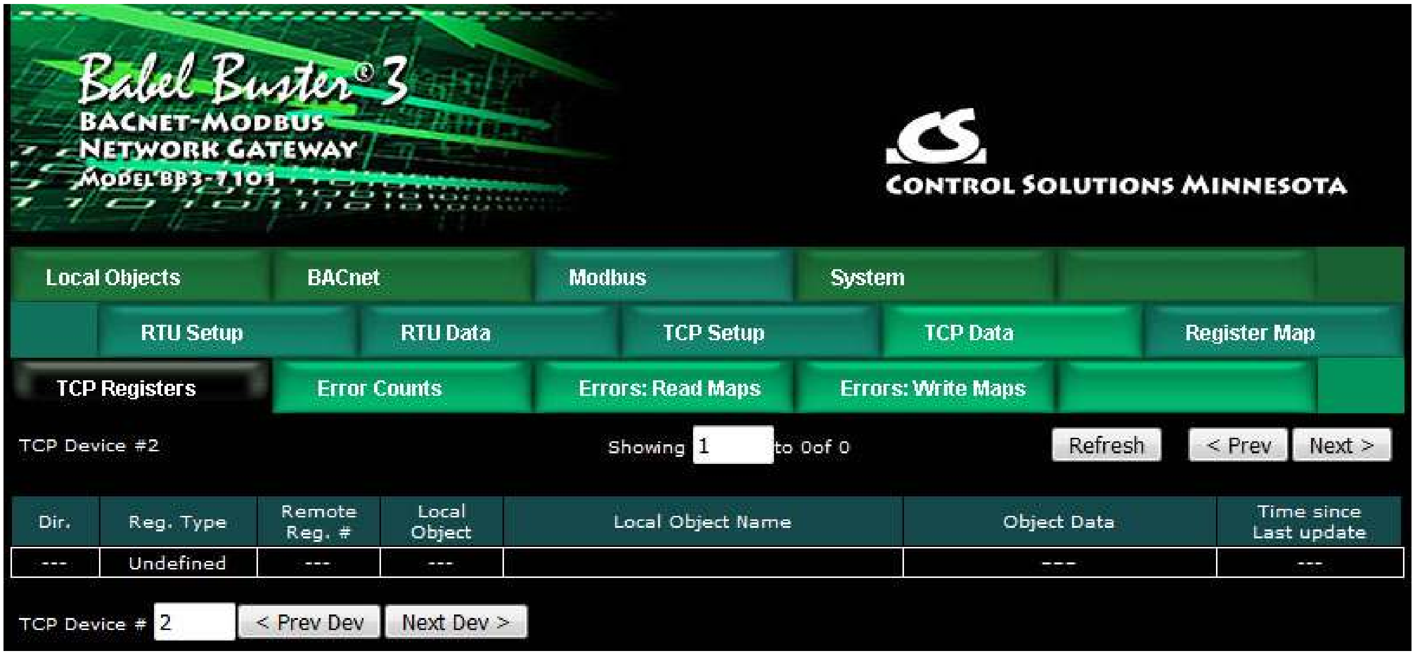
Address: 0001

Length: 10

```
40001: <00025>
40002: <00015>
40003: <00009>
40004: <00000>
40005: <00000>
40006: <00000>
40007: <00000>
40008: <00000>
40009: <00000>
40010: <00000>
```

9.4 Modbus TCP Client Data Displayed by Server

The TCP Registers page shows a list of local objects mapped to TCP server devices. The page will show only one device at a time, and may have no entries if there are no maps for that device.



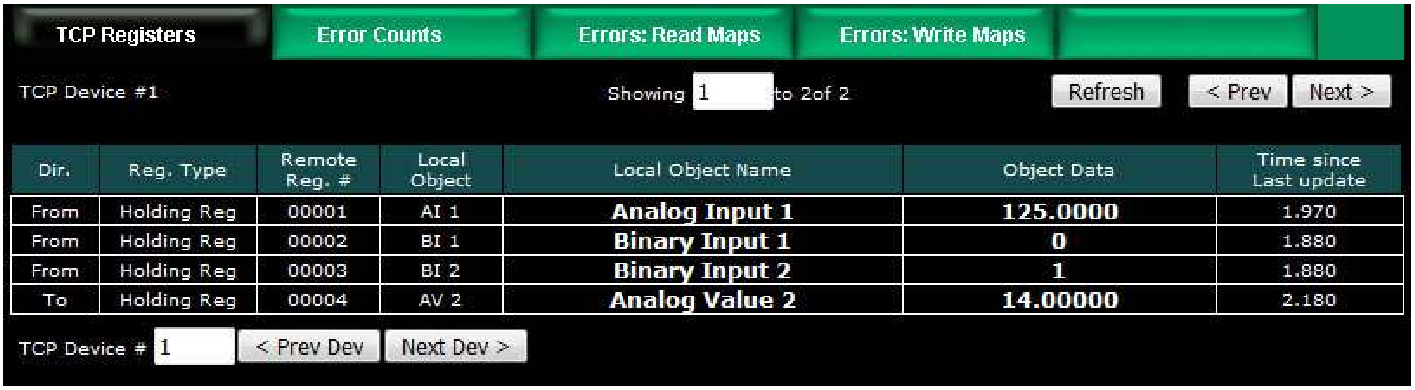
TCP Device #2

Showing 1 to 0of 0

Dir.	Reg. Type	Remote Reg. #	Local Object	Local Object Name	Object Data	Time since Last update
---	Undefined	---	---		---	---

TCP Device # 2

Click the Next Dev or Prev Dev buttons to go to the next or previous TCP device, or simply enter a number in the TCP Device # window and click Update. Registers for that server will now be displayed. In addition to a summary of the map (both read and write maps are shown), the time since last update is displayed. This time should generally be less than the poll time. If the last update time is large, it may mean there is an error preventing the update.



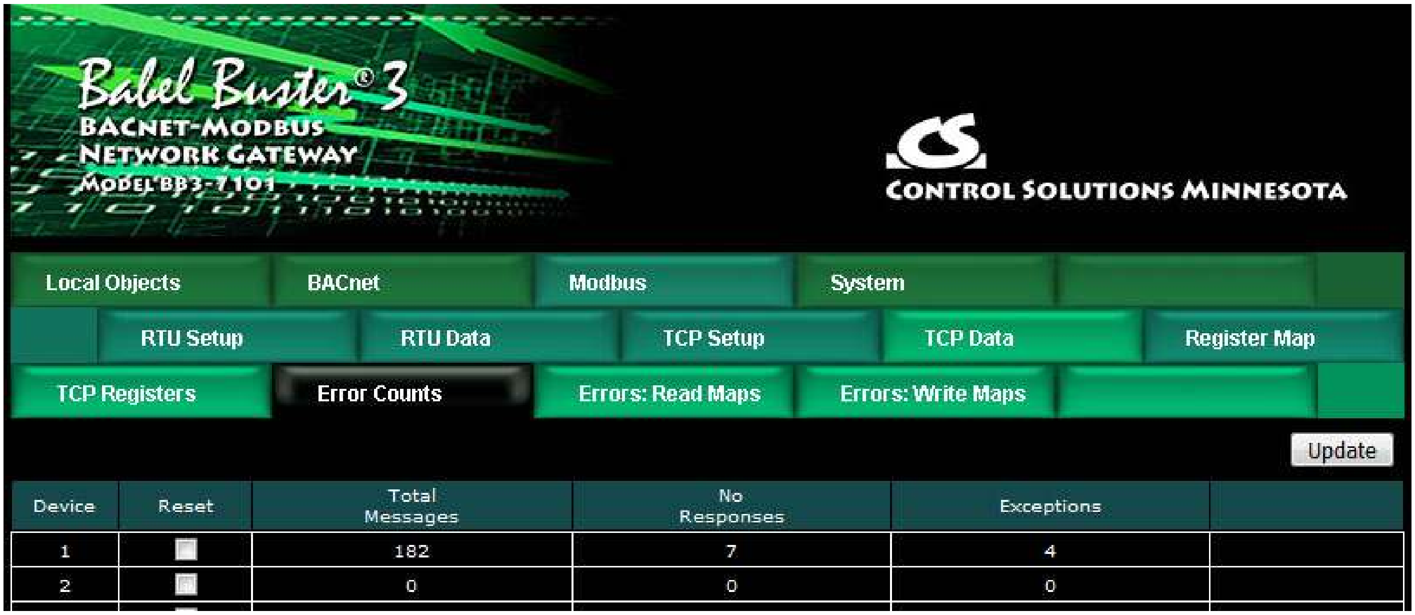
TCP Device #1 Showing 1 to 2of 2

Dir.	Reg. Type	Remote Reg. #	Local Object	Local Object Name	Object Data	Time since Last update
From	Holding Reg	00001	AI 1	Analog Input 1	125.0000	1.970
From	Holding Reg	00002	BI 1	Binary Input 1	0	1.880
From	Holding Reg	00003	BI 2	Binary Input 2	1	1.880
To	Holding Reg	00004	AV 2	Analog Value 2	14.00000	2.180

TCP Device # 1

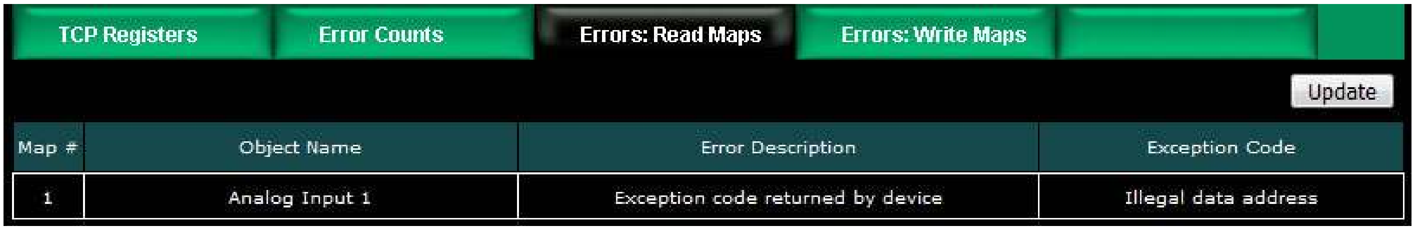
9.5 Modbus TCP Errors

The Error Counts page shows a tabulation, by TCP device, of all errors observed.



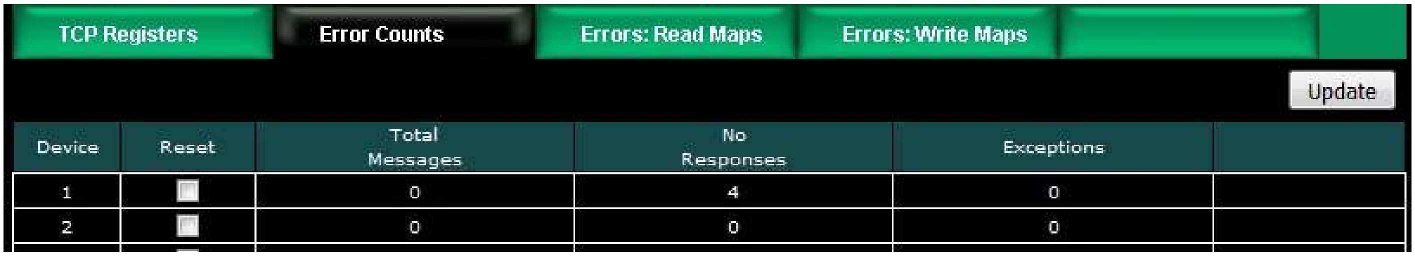
Device	Reset	Total Messages	No Responses	Exceptions	
1	□	182	7	4	
2	□	0	0	0	

If the counts show some problems, we can look for more detail on the Errors: Read Maps (or Errors: Write Maps) pages. These pages will tell us exactly which Read Map (or Write Map) the problem is occurring on, and what the error is, as illustrated below.



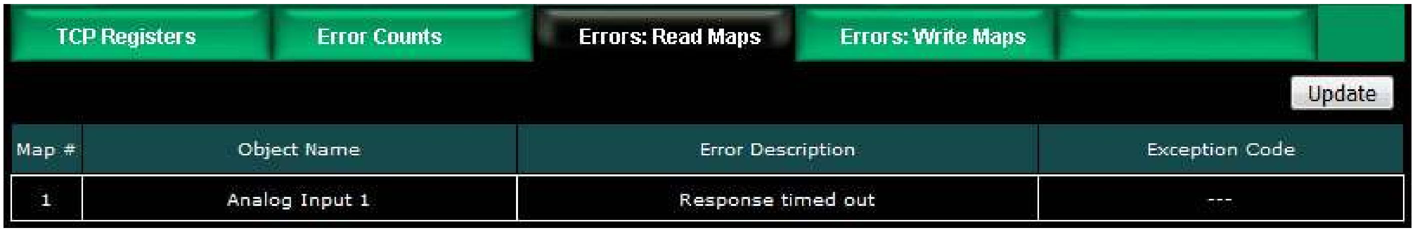
Map #	Object Name	Error Description	Exception Code
1	Analog Input 1	Exception code returned by device	Illegal data address

If you see total messages of zero and a "no responses" count greater than zero, it means the Babel Buster was not able to connect to the IP address of the TCP server. Without being able to connect at all, there was never an attempt to send a message, and hence zero total messages while "no responses" continues to increment.



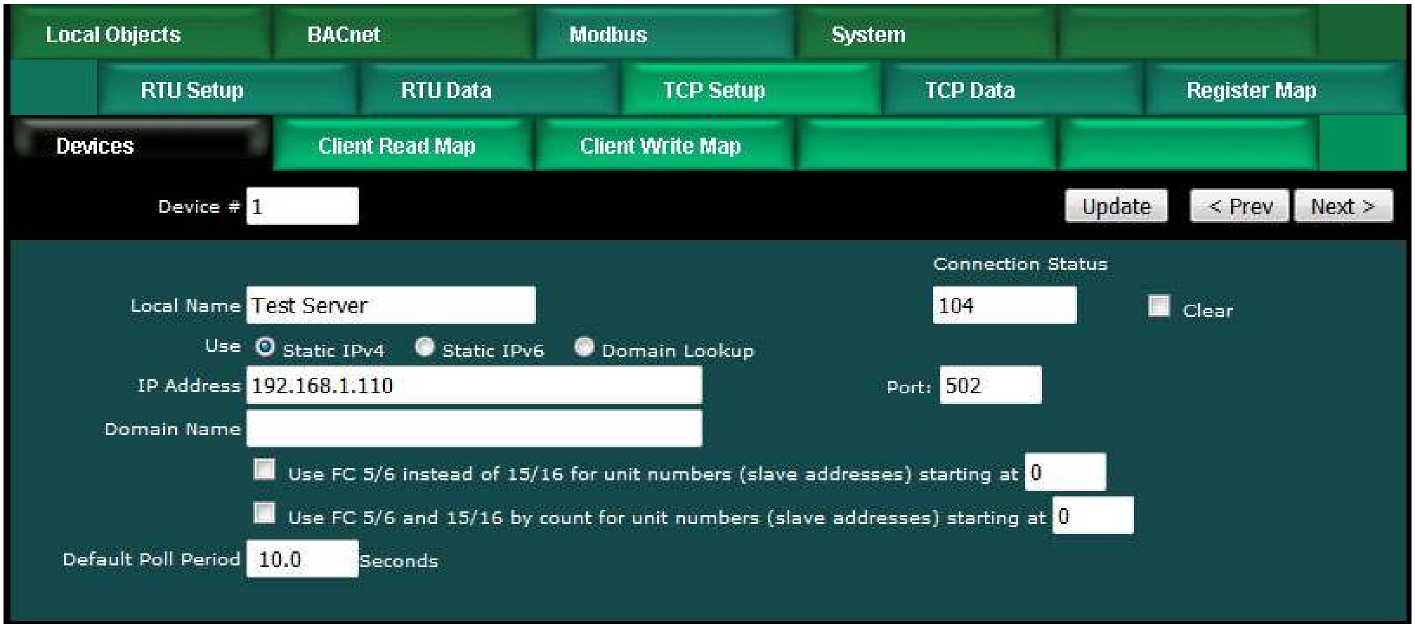
Device	Reset	Total Messages	No Responses	Exceptions	
1	□	0	4	0	
2	□	0	0	0	

When "no responses" is indicated, the Errors: Read Maps (or Write Maps as applicable) page will show that the response timed out, but this is typically a foregone conclusion when you see the "no responses" count for a TCP device.



Map #	Object Name	Error Description	Exception Code
1	Analog Input 1	Response timed out	---

When you get any type of connection related problem with a TCP device, the connection status will typically give you some clues.



Device # 1

Local Name Test Server

Connection Status 104 □ Clear

Use ● Static IPv4 ○ Static IPv6 ○ Domain Lookup

IP Address 192.168.1.110 Port: 502

Domain Name

□ Use FC 5/6 instead of 15/16 for unit numbers (slave addresses) starting at 0

□ Use FC 5/6 and 15/16 by count for unit numbers (slave addresses) starting at 0

Default Poll Period 10.0 Seconds

Connection status codes you may see include:

5 = Connection attempt timed out, unable to establish connection (usually means remote device not connected or not reachable)
104 = Connection reset by peer
111 = Connection refused
113 = Connection aborted
114 = Network is unreachable
115 = Network interface not configured
116 = Connection timed out
118 = Host is unreachable
125 = Address not available

205 = DNS error

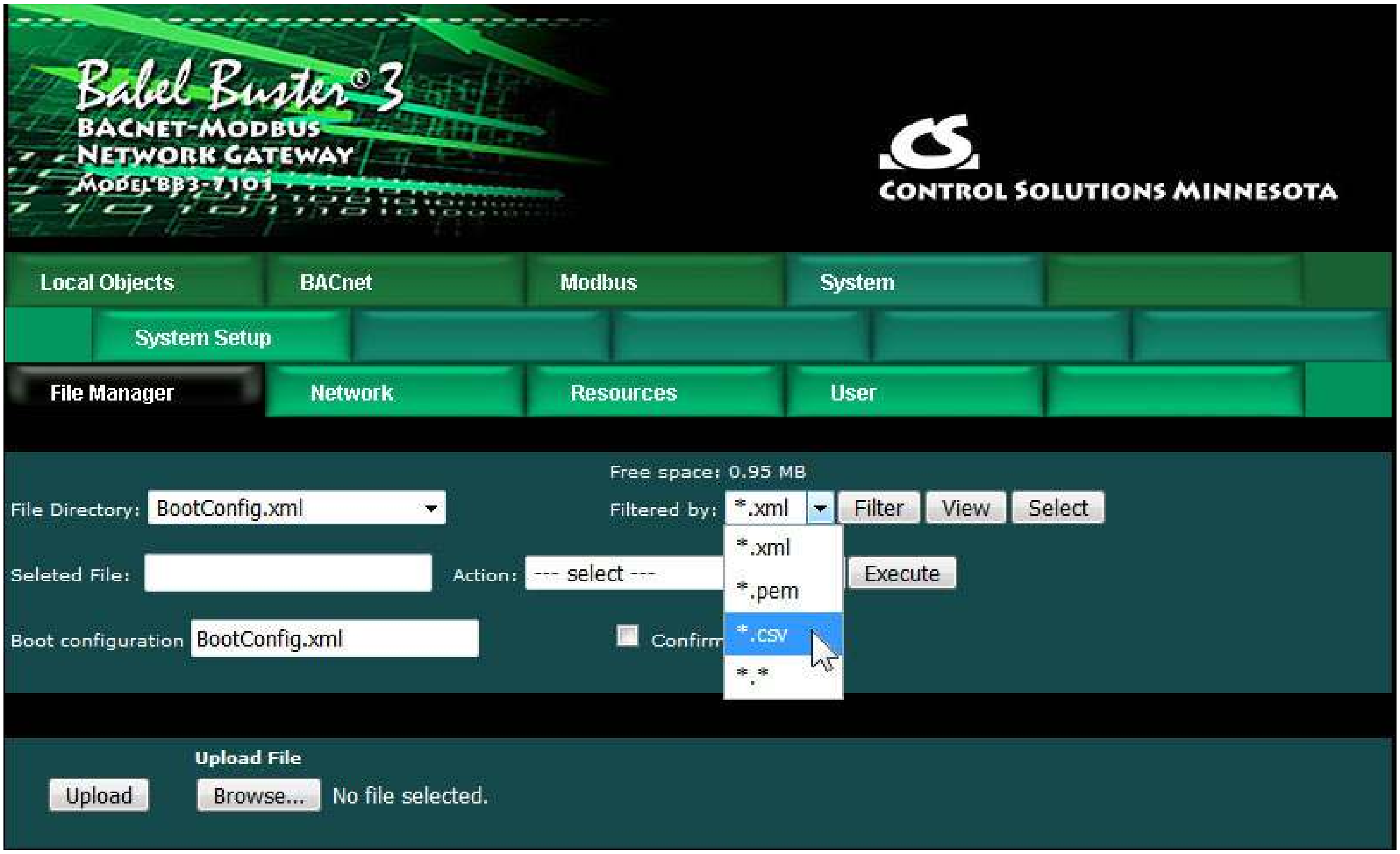
9.6 Importing Modbus TCP Maps from CSV File

The built-in web user interface is user friendly, but can get tedious if you have a lot of maps to enter. Quite often, Modbus register maps are available in spread sheet form. With a bit of editing, you can turn this into a CSV file that can be directly imported into the BB3-7101/MX-71 to quickly configure a lot of read and write maps. If you are proficient with spread sheets, you can probably create a rather large set of maps quickly and speed up the process of configuring the BB3-7101/MX-71.

There is more discussion about the File Manager in Section 11, but a summary of what you need to do to import maps from a CSV file is given here.

Start by uploading your CSV file. Use the Browse button to locate the file on your PC, then click Upload.

Select *.csv as the file filter. This will result in showing the list of CSV files currently stored in the BB3-7101/MX-71.



Free space: 0.95 MB

File Directory: BootConfig.xml

Filtered by: *.xml

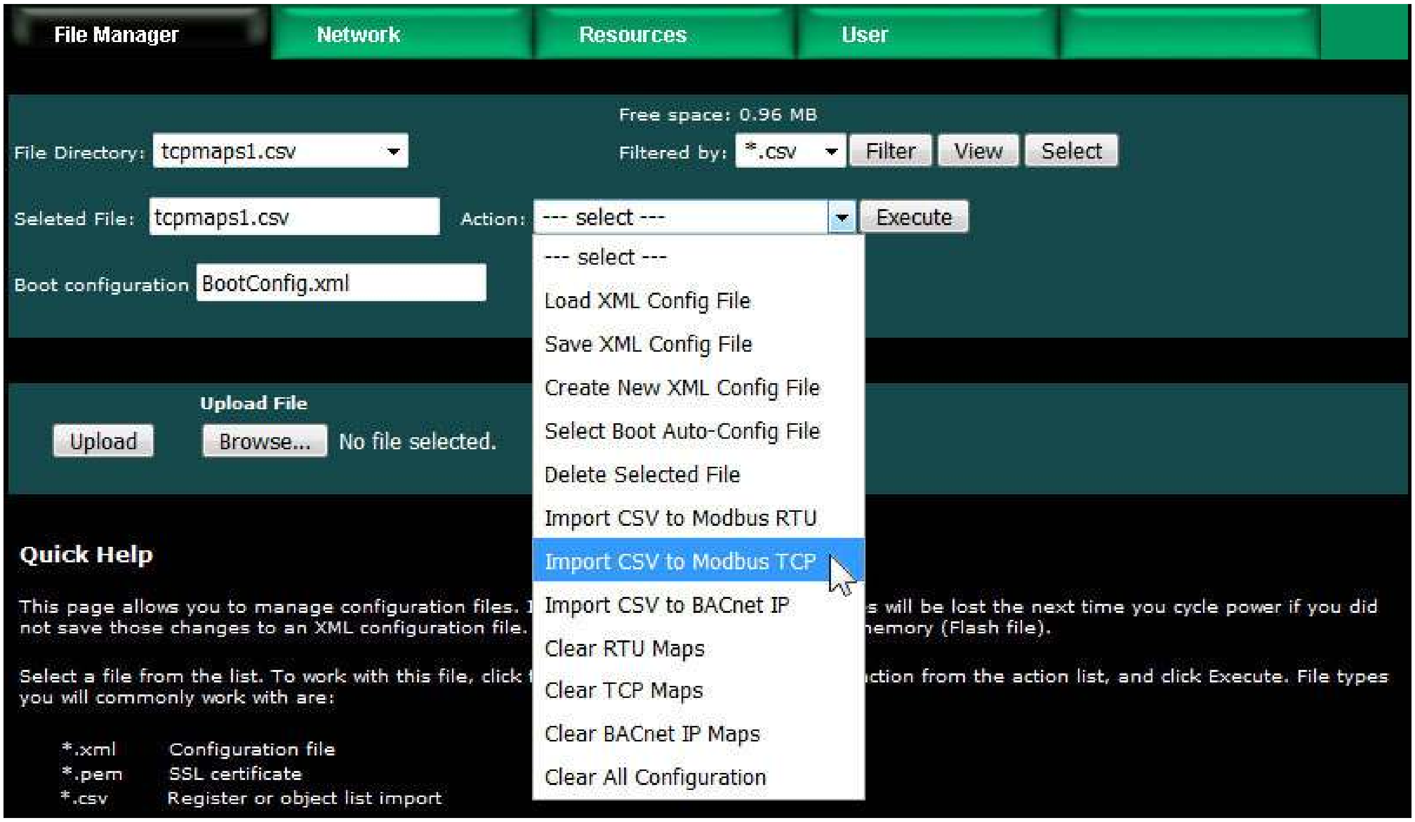
Seleted File:

Action: --- select ---

Boot configuration BootConfig.xml

No file selected.

Select your file from the File Directory drop-down list, then click the Select button on the right. Select "Import CSV to Modbus TCP" from the Action list, and click Execute.



Free space: 0.96 MB

File Directory: tcpmaps1.csv

Filtered by: *.csv

Seleted File: tcpmaps1.csv

Action: --- select ---

Boot configuration BootConfig.xml

Upload File

No file selected.

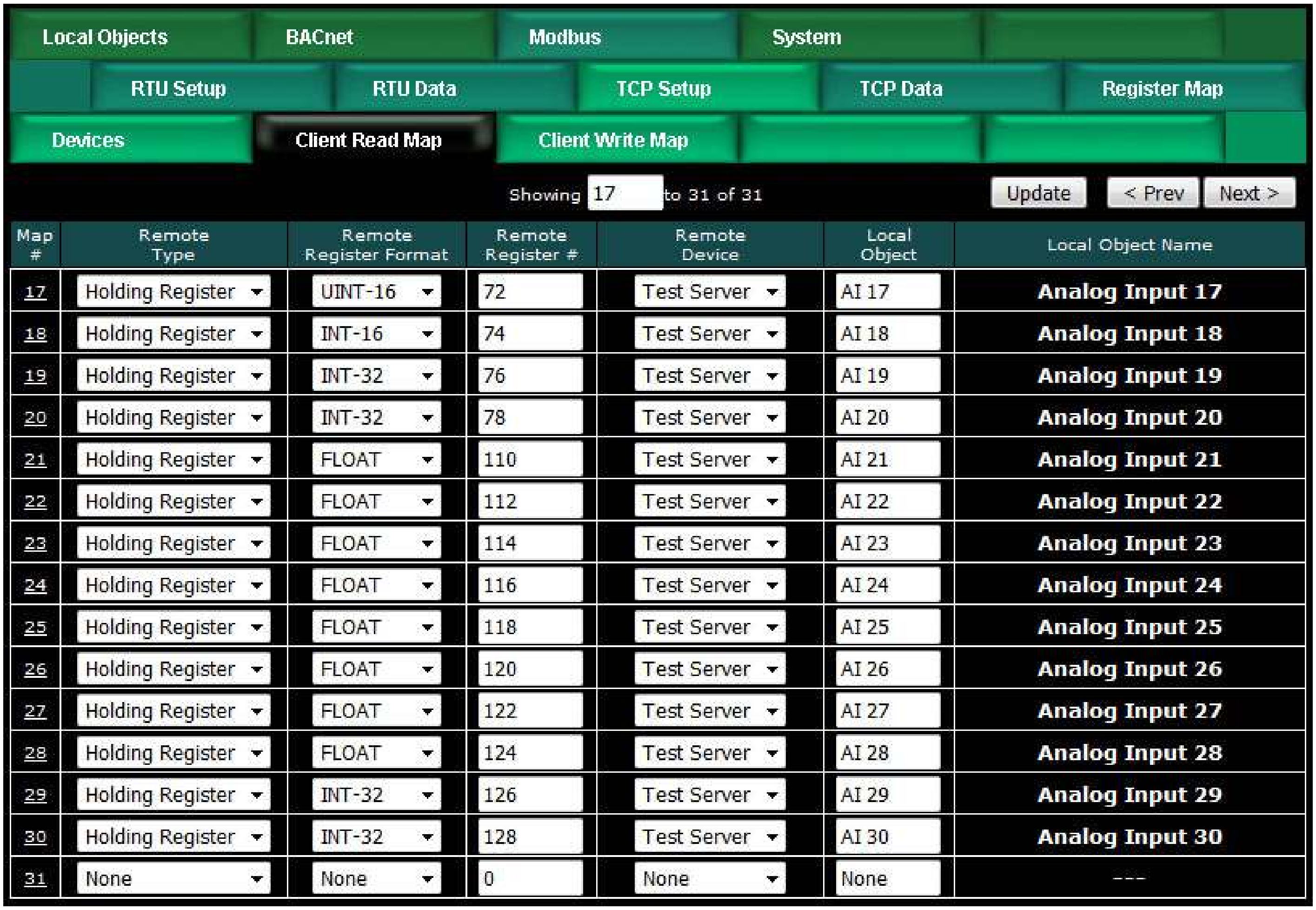
Quick Help

This page allows you to manage configuration files. I s will be lost the next time you cycle power if you did not save those changes to an XML configuration file. emory (Flash file).

Select a file from the list. To work with this file, click t ction from the action list, and click Execute. File types you will commonly work with are:

*.xml	Configuration file
*.pem	SSL certificate
*.csv	Register or object list import

We imported 30 read maps in our example test case.

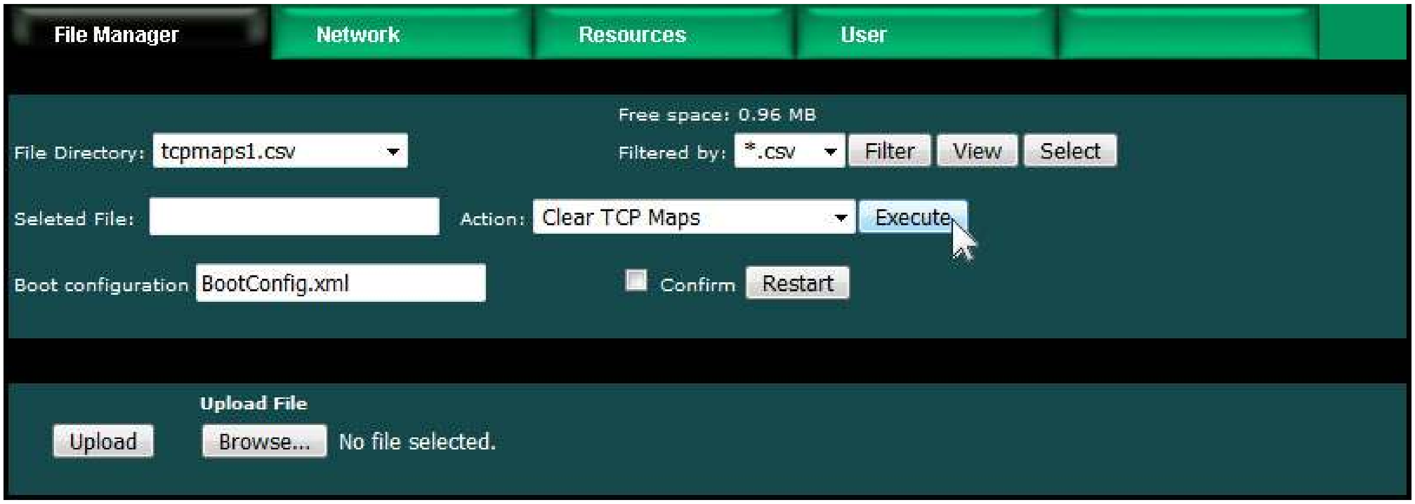


Showing 17 to 31 of 31

Map #	Remote Type	Remote Register Format	Remote Register #	Remote Device	Local Object	Local Object Name
17	Holding Register	UINT-16	72	Test Server	AI 17	**Analog Input 17**
18	Holding Register	INT-16	74	Test Server	AI 18	**Analog Input 18**
19	Holding Register	INT-32	76	Test Server	AI 19	**Analog Input 19**
20	Holding Register	INT-32	78	Test Server	AI 20	**Analog Input 20**
21	Holding Register	FLOAT	110	Test Server	AI 21	**Analog Input 21**
22	Holding Register	FLOAT	112	Test Server	AI 22	**Analog Input 22**
23	Holding Register	FLOAT	114	Test Server	AI 23	**Analog Input 23**
24	Holding Register	FLOAT	116	Test Server	AI 24	**Analog Input 24**
25	Holding Register	FLOAT	118	Test Server	AI 25	**Analog Input 25**
26	Holding Register	FLOAT	120	Test Server	AI 26	**Analog Input 26**
27	Holding Register	FLOAT	122	Test Server	AI 27	**Analog Input 27**
28	Holding Register	FLOAT	124	Test Server	AI 28	**Analog Input 28**
29	Holding Register	INT-32	126	Test Server	AI 29	**Analog Input 29**
30	Holding Register	INT-32	128	Test Server	AI 30	**Analog Input 30**
31	None	None	0	None	None	---

9.7 Clearing Configuration

Read and write maps imported from a CSV file will be added to the list of maps already in place. If you wish to reload the list, you must first clear it. Clear the TCP maps by going to the File Manager page, then selecting "Clear TCP Maps" from the action list and clicking Execute.



Free space: 0.96 MB

File Directory: tcpmaps1.csv

Filtered by: *.csv

Seleted File:

Action: Clear TCP Maps

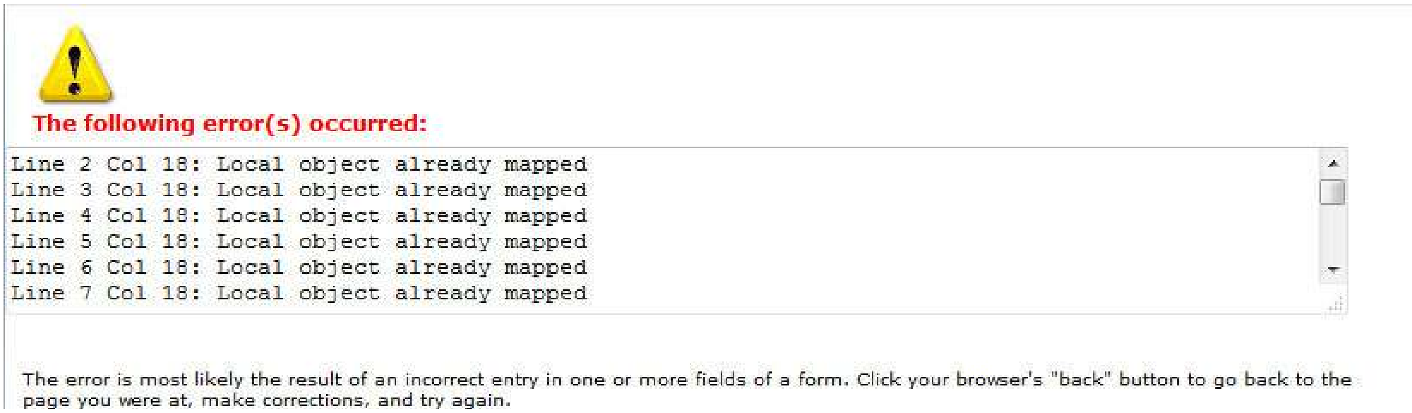
Boot configuration BootConfig.xml

□ Confirm

Upload File

No file selected.

If you forget to clear the maps before re-importing them, you will get an error notice something like this:



The following error(s) occurred:

```
Line 2 Col 18: Local object already mapped
Line 3 Col 18: Local object already mapped
Line 4 Col 18: Local object already mapped
Line 5 Col 18: Local object already mapped
Line 6 Col 18: Local object already mapped
Line 7 Col 18: Local object already mapped
```

The error is most likely the result of an incorrect entry in one or more fields of a form. Click your browser's "back" button to go back to the page you were at, make corrections, and try again.

Babel Buster® 3
MODEL BB3-7101-SP
MODEL MX-71-SP
NETWORK GATEWAY

CONTROL SOLUTIONS MINNESOTA

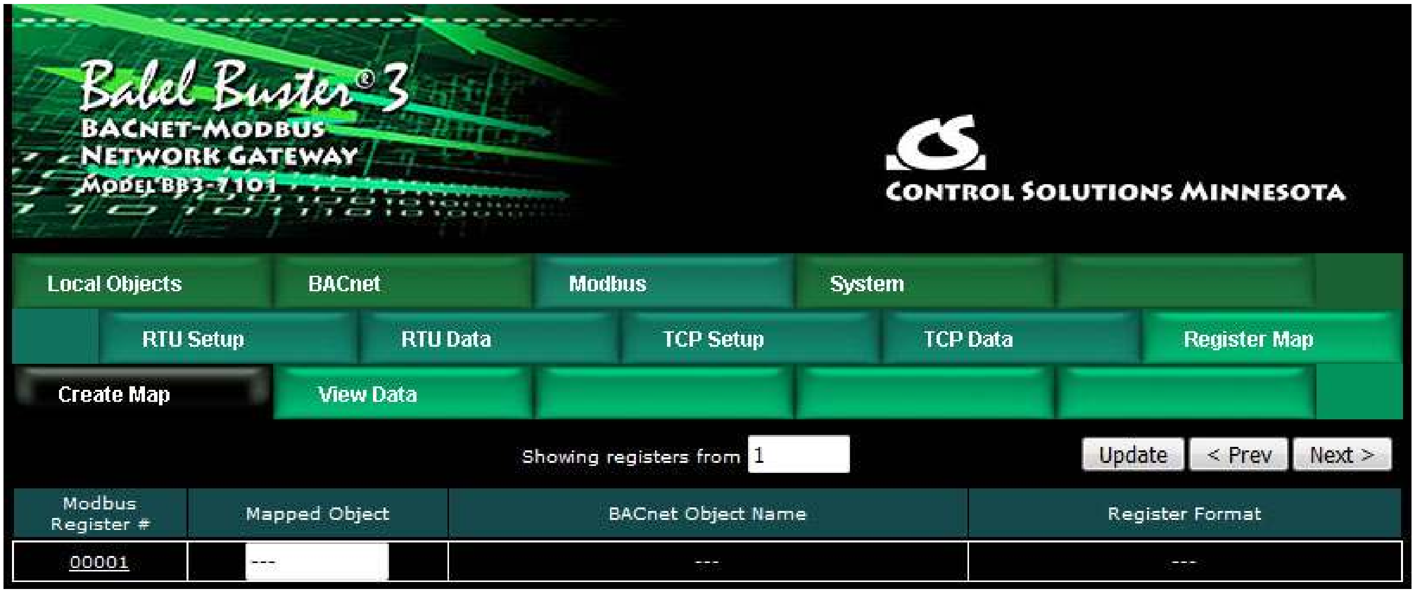
10. Configuring Gateway as a Modbus Server or Slave

10.1 Create Modbus Register Map

The previous generation of Babel Buster BACnet-Modbus gateways had a fixed set of Modbus registers when the gateway was acting as a Modbus slave. The Babel Buster 3 series provides a greater degree of flexibility by letting you create your own Modbus register map pretty much however you may like it.

The Modbus register map is created by literally "creating" Modbus registers and then mapping them to local BACnet objects. The Modbus registers can by any of several data formats (e.g. 16-bit integer, 32-bit floating point, etc) and data will be automatically converted on the fly if the BACnet object data format does not match the Modbus register format. BACnet objects may be read or written from a Modbus master once the register map is created.

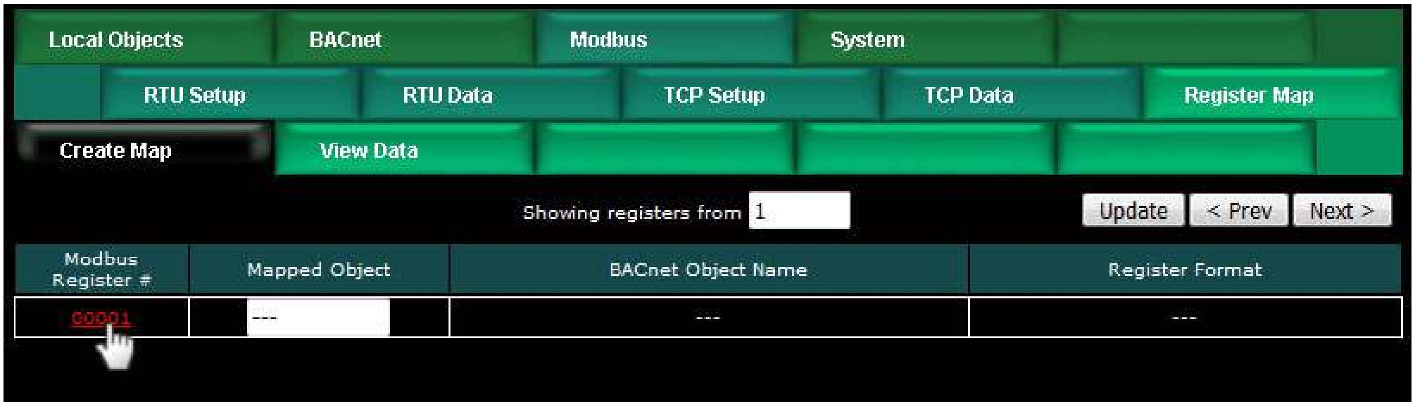
The Modbus register map page is found under the Modbus tab. Upon first visit, there will be no registers defined, which means any attempt made by your Modbus master to read registers will get an exception response.



Showing registers from 1

Modbus Register #	Mapped Object	BACnet Object Name	Register Format
00001	---	---	---

To begin the process of creating registers, click on the only register number available at this point. Later on, click on the register number at the end of the list to add more, or click anywhere in the middle of the list if there are gaps in the register number sequence that you would like to fill.



Showing registers from 1

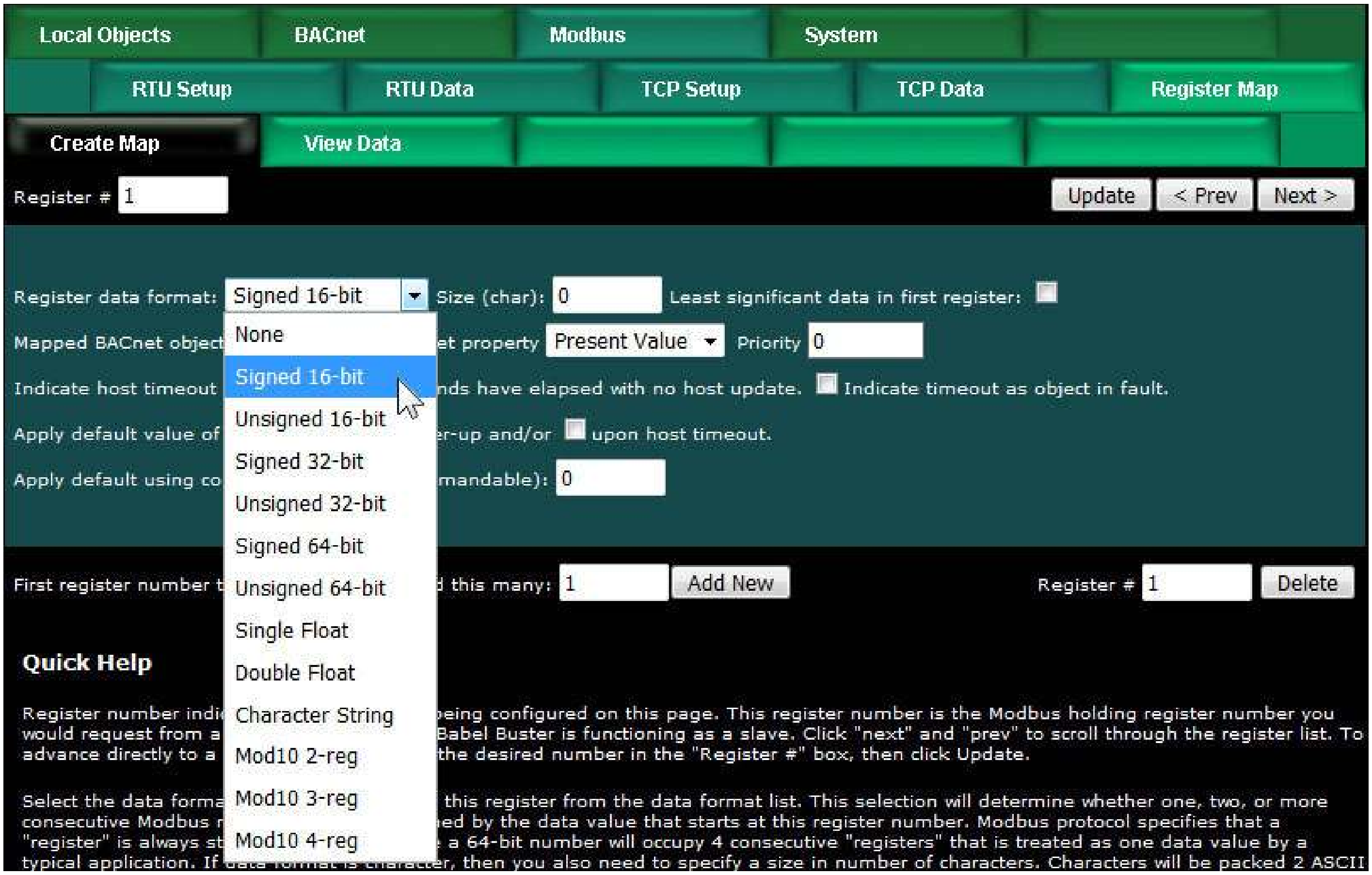
Modbus Register #	Mapped Object	BACnet Object Name	Register Format
00001	---	---	---

Upon clicking a register number, the register detail will be displayed. This can be either detailed configuration of an existing register, or detail about new registers you are about to add. The only time you can select the data format is when adding new registers. Once the registers are created, the data format cannot be changed because this impacts how many Modbus registers are actually used. If you had a set of 16-bit registers defined, and wanted to change one of them to 32-bit, it would cause all of the remaining registers to be renumbered if such a change was allowed. While this may seem harmless at first, it becomes a huge mess trying to keep track of where the rest of your registers moved to.

Select the data format for the new registers to be created. The designations Signed and Unsigned refer to integers. Single float refers to 32-bit IEEE 754 floating point format (same format used by BACnet Analog objects). Double float refers to 64-bit IEEE 754 floating point format. Character string refers to a series of registers with two ASCII characters per 16-bit register. Mod10 refers to a format unique to Schneider Electric power meters (refer to Schneider Electric documentation for a definition of those formats). The Size parameter is only used in conjunction with strings, and specifies character count.

IMPORTANT: Modbus protocol knows holding registers (or input registers) to simply be a 16-bit piece of data. The protocol knows nothing about signed or unsigned integer - that interpretation is up to you. The 16 bits may even be a collection of 16 status flags. If a register is defined in the Babel Buster as anything bigger than 16 bits, it is actually a pair (or series of 2 or more) 16-bit registers. Again, Modbus protocol does not know anything about floating point, 64-bit integer, etc. Modbus only knows how to send 16 bits of something at a time when function codes reference a holding register or input register. Modbus only knows how to send 1 bit at a time if referenced as a coil or discrete input. It is up to the Modbus master to be smart enough to ask for 2 registers at a time if it knows it wants to read a 32-bit value.

Modbus protocol is strict about 16-bit increments of data for holding registers. However, when register pairs (or quads) are used to hold a 32-bit (or 64-bit) value, the order in which those registers are interpreted is not defined by any standard. It is up to you to keep track of that. Babel Buster supports interoperability with other Modbus devices by letting you specify what order should be used internally. If the least significant data should appear in the first (or lower numbered) register, then check the box that says "Least significant data should be in first register" either when creating the register, or later by reconfiguring the existing register.



Register # 1

Register data format: Signed 16-bit Size (char): 0 Least significant data in first register: □

Mapped BACnet object ...et property Present Value Priority 0

Indicate host timeout ...nds have elapsed with no host update. □ Indicate timeout as object in fault.

Apply default value of ...er-up and/or □ upon host timeout.

Apply default using co...mandable): 0

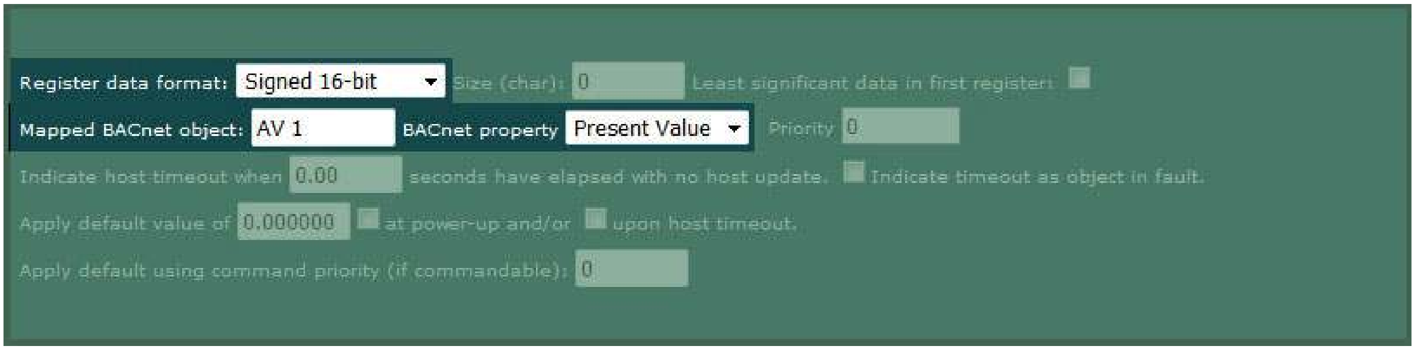
First register number t... this many: 1 Register # 1

Quick Help

Register number indi... being configured on this page. This register number is the Modbus holding register number you would request from a ... Babel Buster is functioning as a slave. Click "next" and "prev" to scroll through the register list. To advance directly to a ... the desired number in the "Register #" box, then click Update.

Select the data forma... this register from the data format list. This selection will determine whether one, two, or more consecutive Modbus r...ed by the data value that starts at this register number. Modbus protocol specifies that a "register" is always st... a 64-bit number will occupy 4 consecutive "registers" that is treated as one data value by a typical application. If data format is character, then you also need to specify a size in number of characters. Characters will be packed 2 ASCII

The registers you create are not useful until you map them to a BACnet object in your Babel Buster. There are only three items that you really must provide when creating a register: The register data format, the BACnet object it should be mapped to, and the property you want Modbus to have access to.



Register data format: Signed 16-bit Size (char): 0 Least significant data in first register: □

Mapped BACnet object: AV 1 BACnet property Present Value Priority 0

Indicate host timeout when 0.00 seconds have elapsed with no host update. □ Indicate timeout as object in fault.

Apply default value of 0.000000 □ at power-up and/or □ upon host timeout.

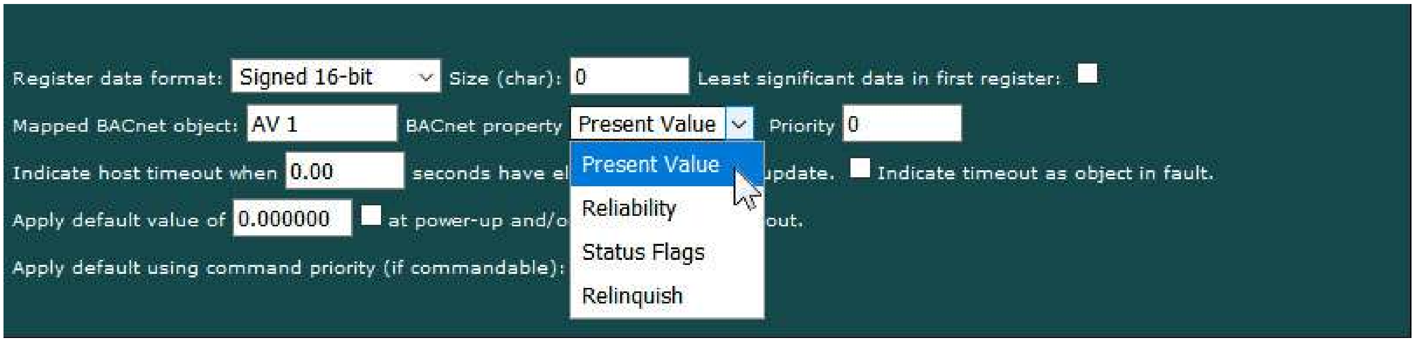
Apply default using command priority (if commandable): 0

The BACnet objects are designated by a 2-character abbreviation followed by an object number. The object abbreviations are as shown below. The object must exist within the Babel Buster before it is allowed to be referenced here.

Label	Object Type
AI	Analog Input
AO	Analog Output
AV	Analog Value
BI	Binary Input

BO	Binary Output
BV	Binary Value
MI	Multistate Input
MO	Multistate Output
MV	Multistate Value

The object properties that Modbus is allowed to access are Present Value, Reliability, and Status Flags. Only Present Value may be written by Modbus. Attempting to write to Reliability or Status Flags will result in a Modbus exception response. If the object is commandable, then Modbus writes to this register will apply the Priority shown next to BACnet property. Relinquish is a special case - see note at end of this section 10.1.



Register data format: Signed 16-bit Size (char): 0 Least significant data in first register: □

Mapped BACnet object: AV 1 BACnet property Present Value Priority 0

Indicate host timeout when 0.00 seconds have el update. □ Indicate timeout as object in fault.

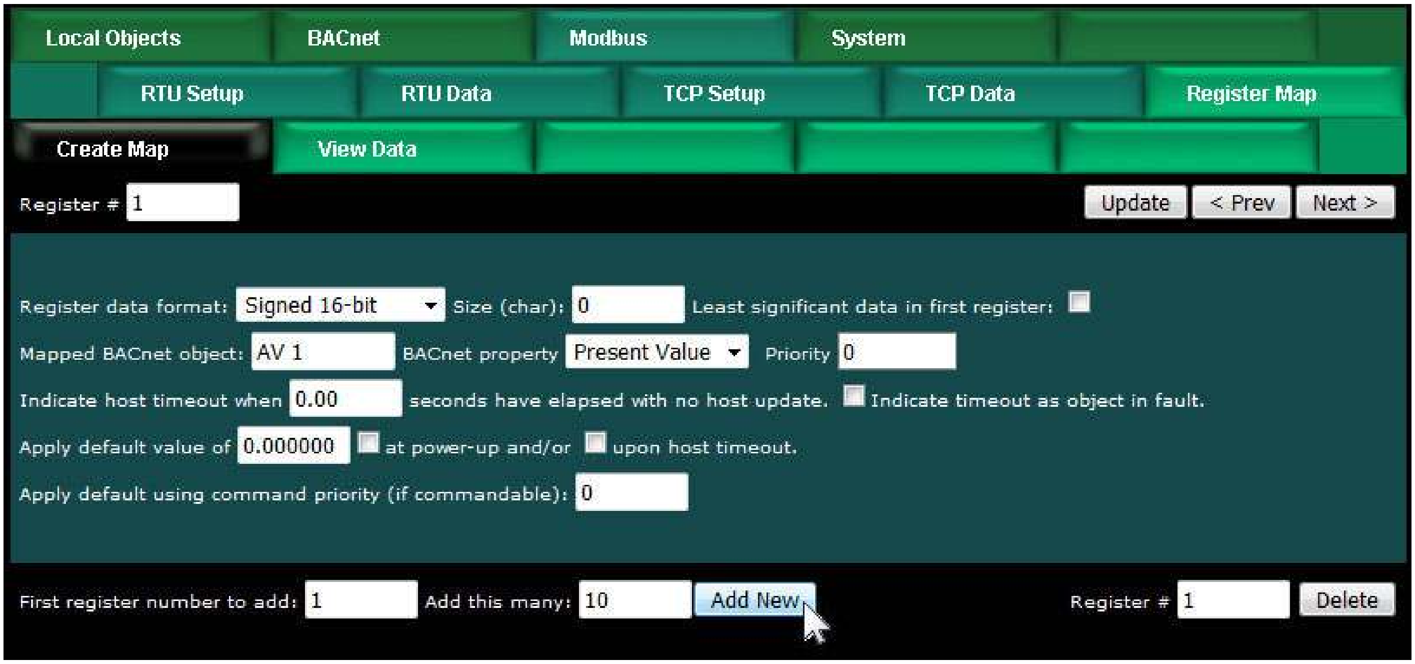
Apply default value of 0.000000 □ at power-up and/o out.

Apply default using command priority (if commandable):

The first register number to add, indicated at the bottom of the page, will be the first available register slot that is not yet assigned. You can enter some different number here. It is not required to create registers in contiguous order. You can jump around, as long as registers don't try to overlap. Select a number of registers to create, and click Add New. If you attempt to add registers that overlap existing registers, the allocation algorithm will find an available slot for you and fill that instead.

The only other restriction on Modbus register numbers created here is that they must fall within the range of the number of Modbus slave registers allocated at the bottom of the Resources page under System Setup.

You can create an entire series of registers with one click by setting some number in the "Add this many" window. If you have provided a BACnet object designation for Mapped BACnet object, that object number will be automatically incremented for each additional register in the series.



Register # 1

Register data format: Signed 16-bit Size (char): 0 Least significant data in first register: □

Mapped BACnet object: AV 1 BACnet property Present Value Priority 0

Indicate host timeout when 0.00 seconds have elapsed with no host update. □ Indicate timeout as object in fault.

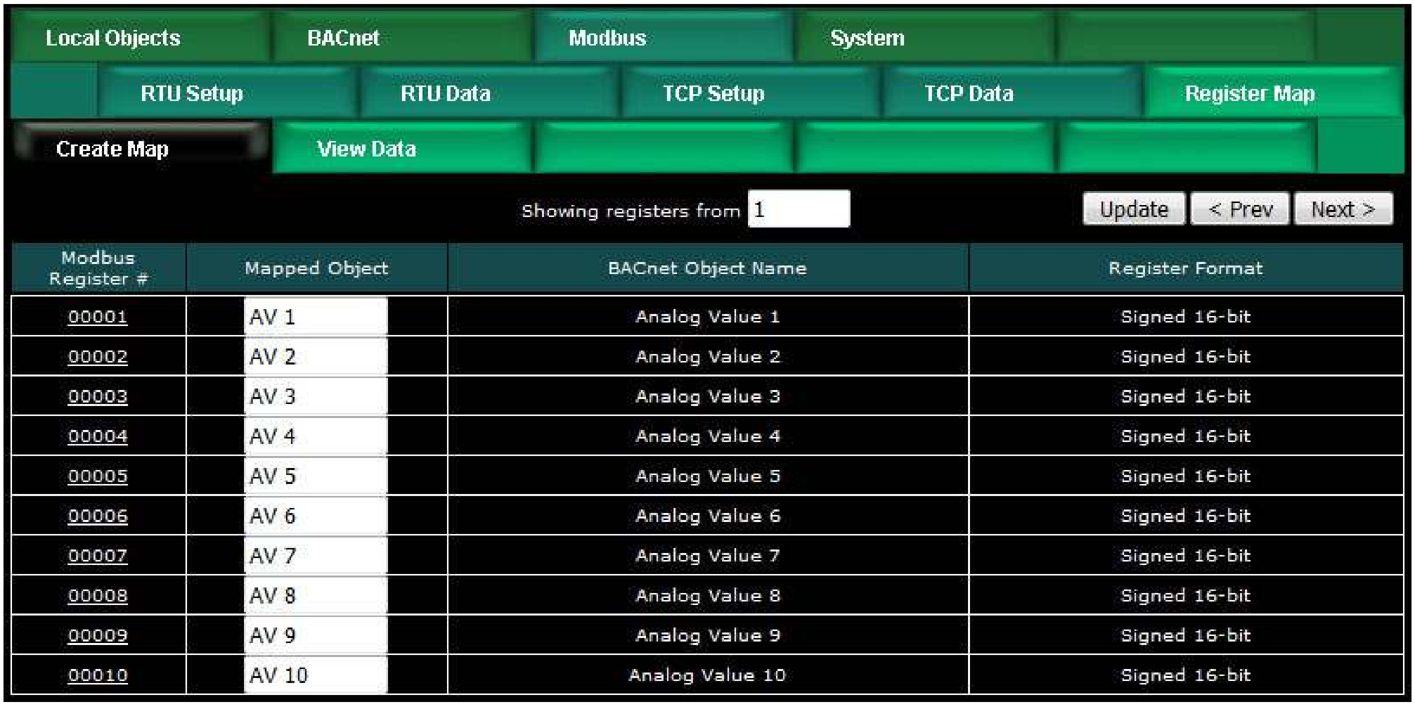
Apply default value of 0.000000 □ at power-up and/or □ upon host timeout.

Apply default using command priority (if commandable): 0

First register number to add: 1 Add this many: 10 Register # 1

Upon clicking Add New with the above screen showing, we will create 10 Modbus registers assigned one each to a series of Analog Value objects as illustrated below. Return to this tabular view by clicking the Create Map tab again.

You already know that Analog objects are 32-bit floating point data. What is important to observe here is that Modbus will be viewing them as 16-bit signed integer values. While this is more convenient for some Modbus applications, do be aware that the range of data will be more limited. To give Modbus the full vew of exactly what is in the Analog object, use Single Float instead since this is what an Analog Object uses internally.



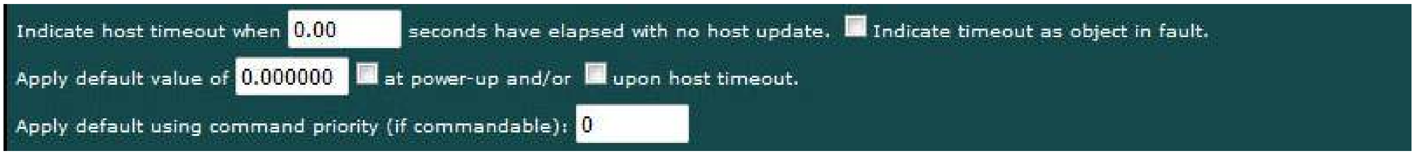
Showing registers from 1

Modbus Register #	Mapped Object	BACnet Object Name	Register Format
00001	AV 1	Analog Value 1	Signed 16-bit
00002	AV 2	Analog Value 2	Signed 16-bit
00003	AV 3	Analog Value 3	Signed 16-bit
00004	AV 4	Analog Value 4	Signed 16-bit
00005	AV 5	Analog Value 5	Signed 16-bit
00006	AV 6	Analog Value 6	Signed 16-bit
00007	AV 7	Analog Value 7	Signed 16-bit
00008	AV 8	Analog Value 8	Signed 16-bit
00009	AV 9	Analog Value 9	Signed 16-bit
00010	AV 10	Analog Value 10	Signed 16-bit

There are some optional parameters available when creating Modbus registers.

Errors are readily indicated when the Babel Buster is acting as master (or client). But if a remote master or client stops providing data while Babel Buster is operating as a slave (or server), the only way to detect that is absence of updates. Therefore, you have the option of setting a host timeout on the object mapped to the server here. Upon timeout, you have the option of indicating the timeout as a non-zero reliability code and fault in the BACnet object status, or simply setting a default value that may mean something as an off-normal value to Modbus.

The default value may be applied at start-up, or upon host timeout. If the default value is applied to a commandable (Output) object, the command priority given here will be used. If the Modbus register results in writing to a commandable object during normal operation, the Local Command Priority found on the BACnet :: Local Device page will be used.



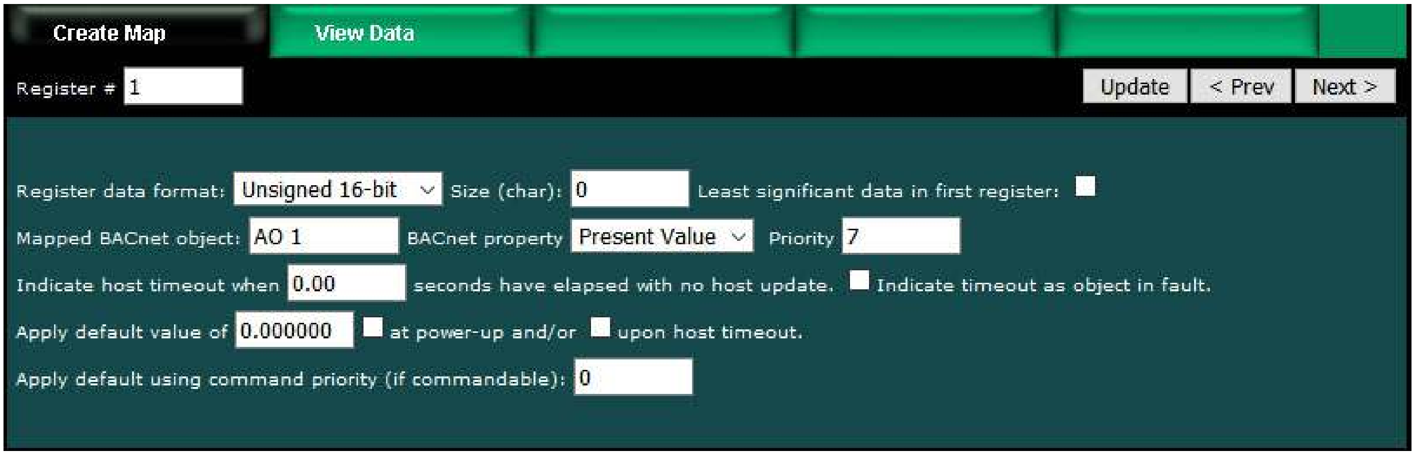
Indicate host timeout when 0.00 seconds have elapsed with no host update. □ Indicate timeout as object in fault.

Apply default value of 0.000000 □ at power-up and/or □ upon host timeout.

Apply default using command priority (if commandable): 0

The Relinquish "property" is a special case. If the mapped register will be writing to a commandable object, the Priority given here (1..16) will be used. Modbus has no special value that means "relinquish", and therefore Modbus cannot directly relinquish a commandable object. A special register mapping is provided for this purpose. The means of relinquishing the given command priority is to create a second Modbus register referencing the same object and same priority, but selecting the property "Relinquish". Writing a value of one to the relinquish register will result in the given command priority being relinquished (writing any other value has no effect). Reading this special "relinquish" register will return one if the given priority is relinquished, or zero if active.

Given the following two screen shots, writing any value to Modbus holding register 1 will result in that value being placed at priority 7 of Analog Output 1. Writing the value one (1) to Modbus holding register 2 will result in priority 7 of Analog Output 1 being relinquished.



Register # 1

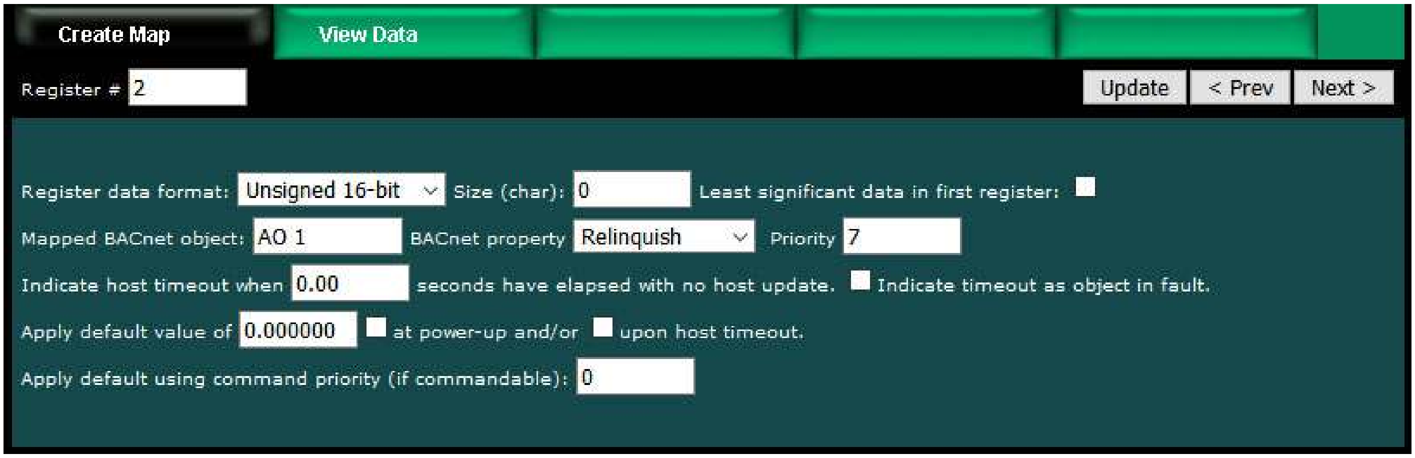
Register data format: Unsigned 16-bit Size (char): 0 Least significant data in first register: □

Mapped BACnet object: AO 1 BACnet property Present Value Priority 7

Indicate host timeout when 0.00 seconds have elapsed with no host update. □ Indicate timeout as object in fault.

Apply default value of 0.000000 □ at power-up and/or □ upon host timeout.

Apply default using command priority (if commandable): 0



Register # 2

Register data format: Unsigned 16-bit Size (char): 0 Least significant data in first register: □

Mapped BACnet object: AO 1 BACnet property Relinquish Priority 7

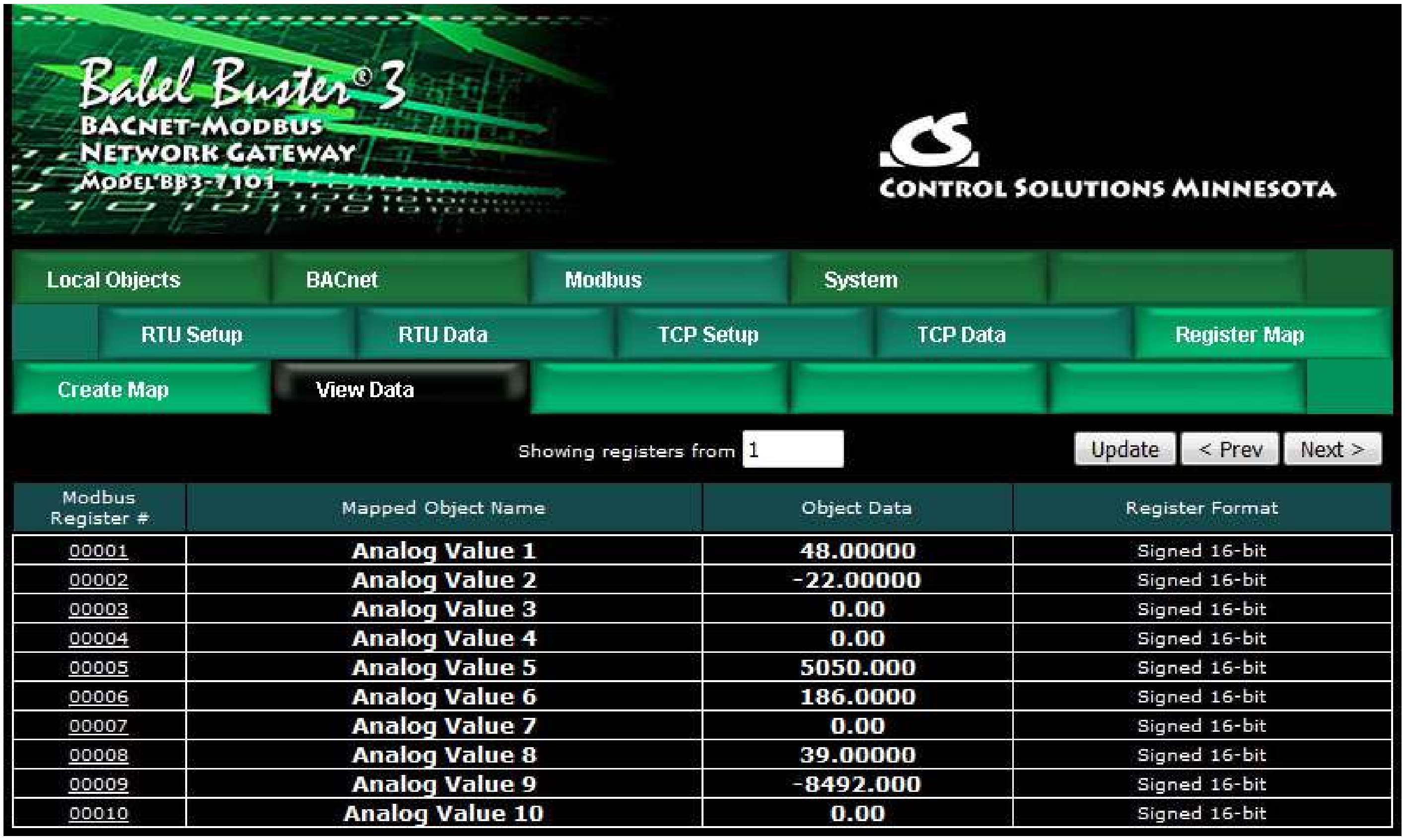
Indicate host timeout when 0.00 seconds have elapsed with no host update. □ Indicate timeout as object in fault.

Apply default value of 0.000000 □ at power-up and/or □ upon host timeout.

Apply default using command priority (if commandable): 0

10.2 View Modbus Register Data

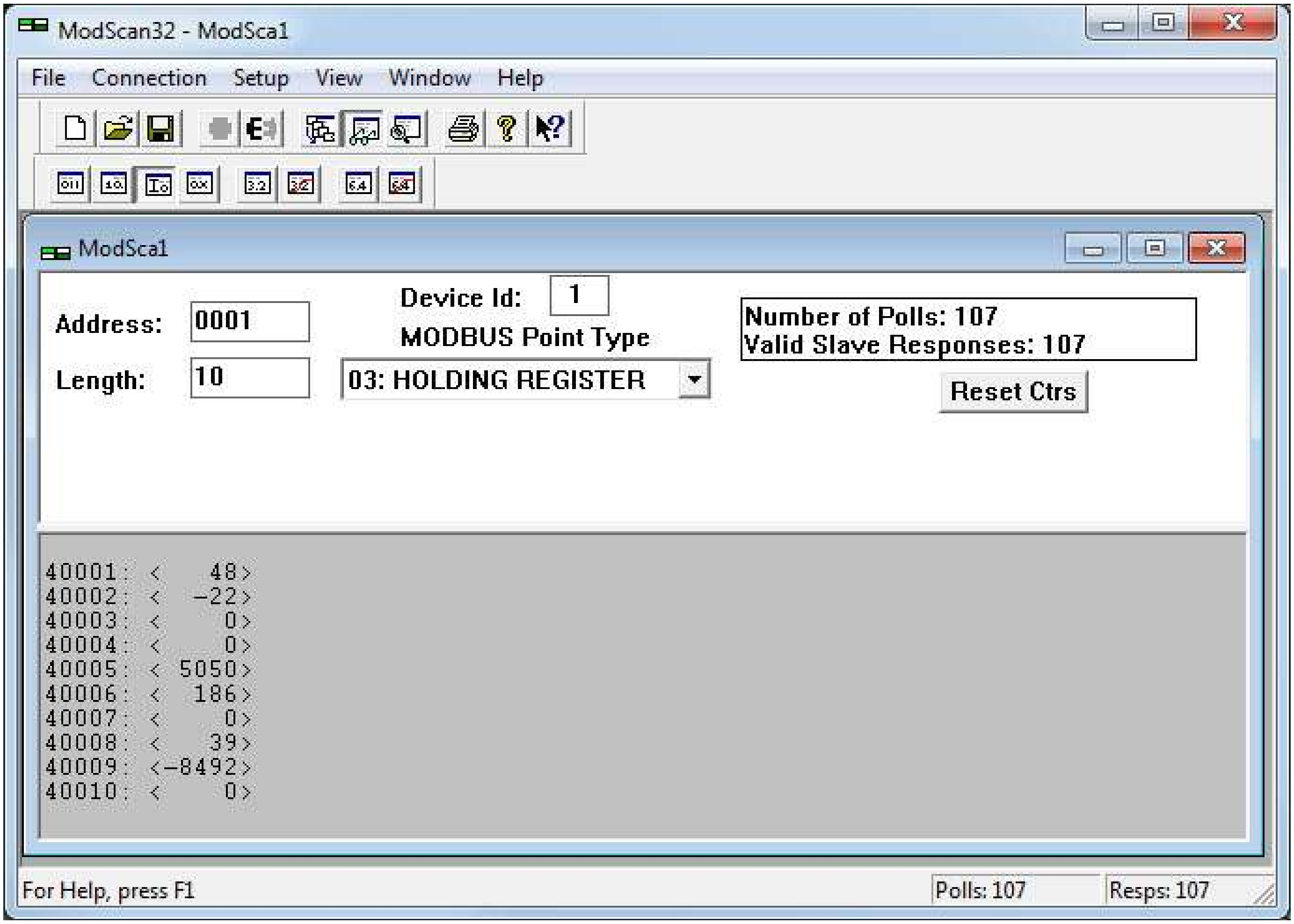
The View Data page under Modbus :: Register Map shows a table of Modbus registers, the BACnet objects they are mapped to, and the current value of those objects. This will usually be a good representation of what data should be showing up in your Modbus master device. However, do be aware that large analog values mapped to a 16-bit integer register will be bounded and therefore not necessarily accurate.



Showing registers from 1

Modbus Register #	Mapped Object Name	Object Data	Register Format
00001	Analog Value 1	48.00000	Signed 16-bit
00002	Analog Value 2	-22.00000	Signed 16-bit
00003	Analog Value 3	0.00	Signed 16-bit
00004	Analog Value 4	0.00	Signed 16-bit
00005	Analog Value 5	5050.000	Signed 16-bit
00006	Analog Value 6	186.0000	Signed 16-bit
00007	Analog Value 7	0.00	Signed 16-bit
00008	Analog Value 8	39.00000	Signed 16-bit
00009	Analog Value 9	-8492.000	Signed 16-bit
00010	Analog Value 10	0.00	Signed 16-bit

The screen shot below shows a Modbus scanner tool reading the registers mapped as illustrated above.



Address: 0001

Length: 10

Device Id: 1

MODBUS Point Type: 03: HOLDING REGISTER

Number of Polls: 107
Valid Slave Responses: 107

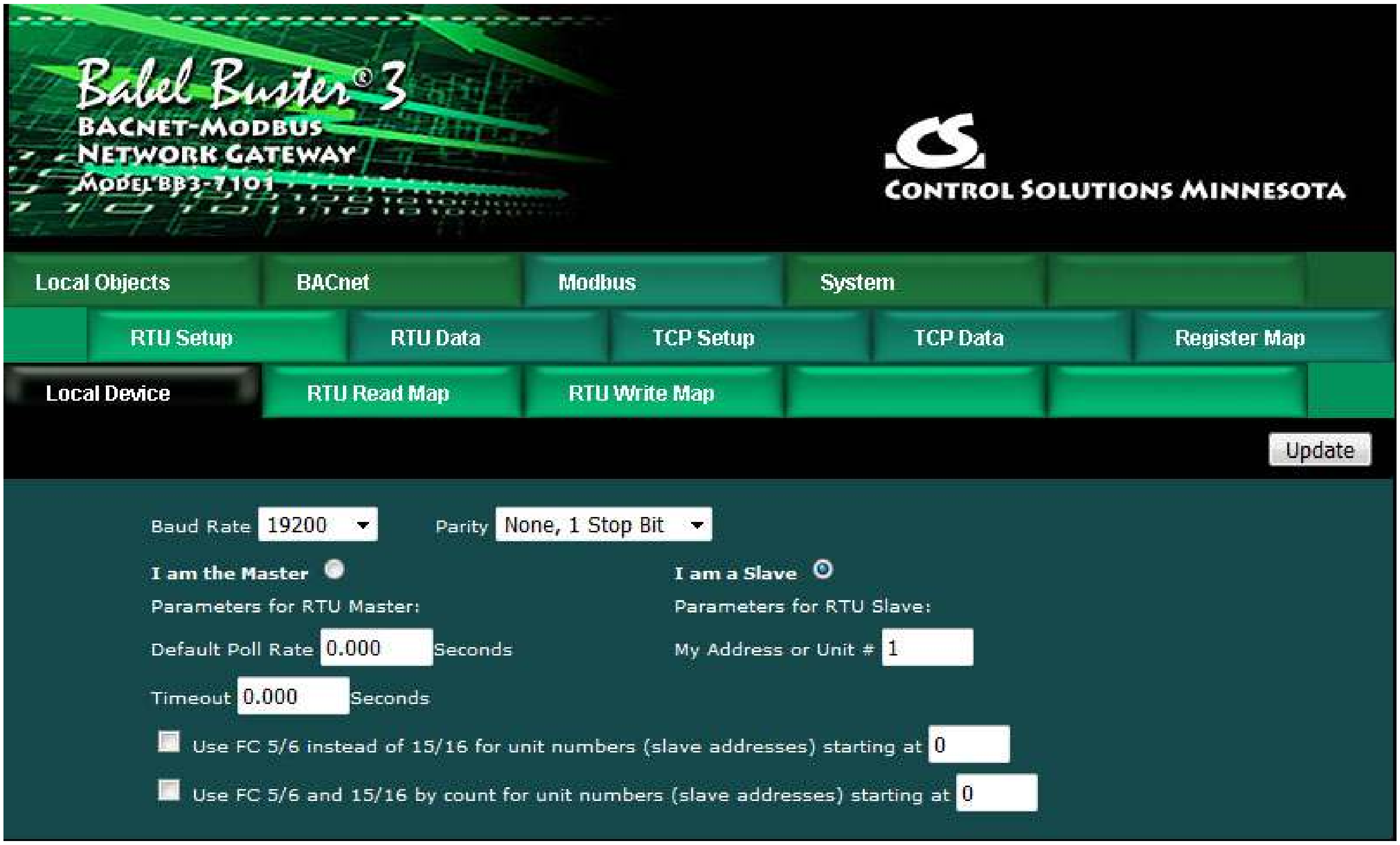
```
40001: <    48>
40002: <   -22>
40003: <     0>
40004: <     0>
40005: <  5050>
40006: <   186>
40007: <     0>
40008: <    39>
40009: <-8492>
40010: <     0>
```

For Help, press F1 | Polls: 107 | Resps: 107

10.3 Modbus RTU Device Configuration

Device configuration for RTU means configuring the serial port. Select the baud rate and parity as applicable. Select "I am a Slave". It is important to provide an address or unit number that is not used by any other slave on the RTU network. The poll rate, timeout, and "Use FC 5/6..." only apply when RTU is master.

These parameters are included in the XML configuration file along with all of your other configuration information about Modbus. After making changes, be sure to go to the File Manager page and save your configuration.



Baud Rate 19200

Parity None, 1 Stop Bit

I am the Master ○ I am a Slave ●

Parameters for RTU Master:

Default Poll Rate 0.000 Seconds

Timeout 0.000 Seconds

Parameters for RTU Slave:

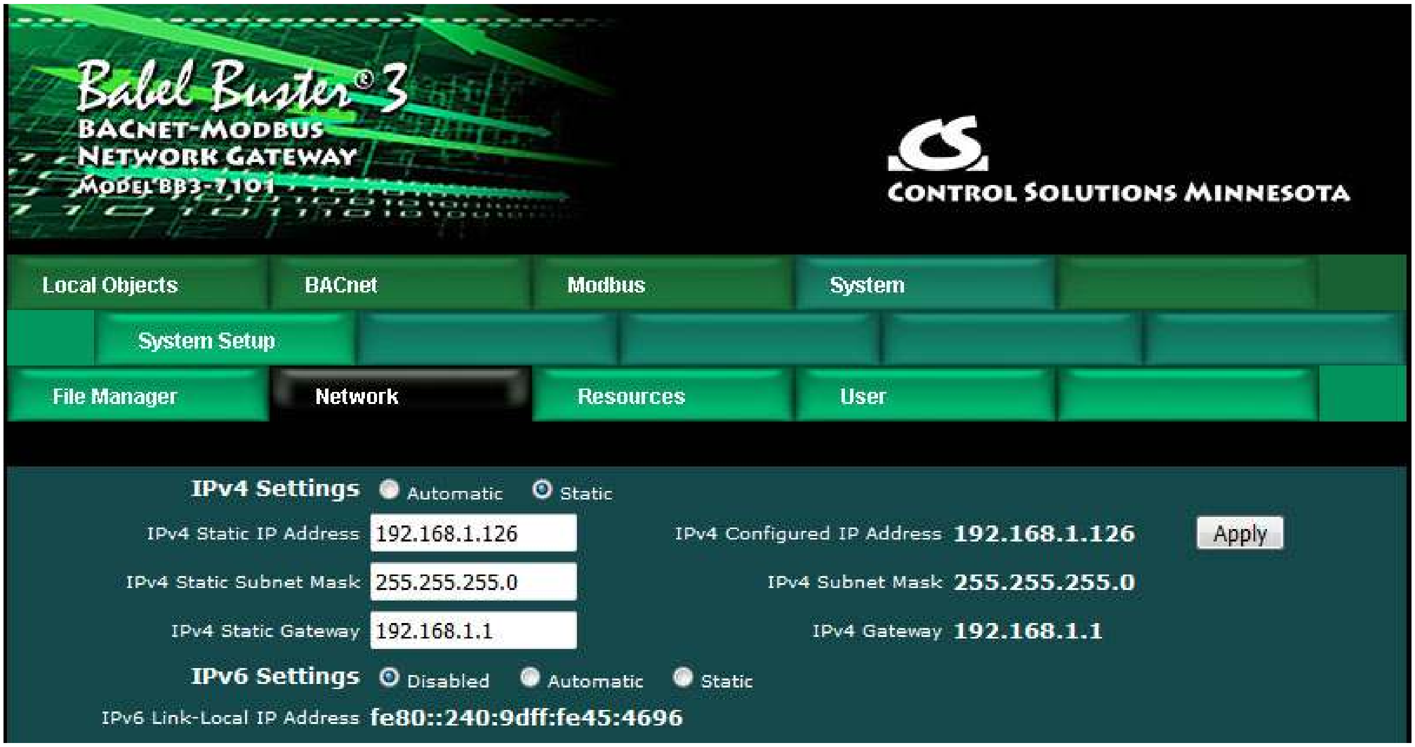
My Address or Unit # 1

☐ Use FC 5/6 instead of 15/16 for unit numbers (slave addresses) starting at 0

☐ Use FC 5/6 and 15/16 by count for unit numbers (slave addresses) starting at 0

10.4 Modbus TCP Device Configuration

Device configuration for Modbus TCP probably does not involve anything you haven't already done. All you really need to do is set the IP address on the Network set upage.



IPv4 Settings ○ Automatic ● Static

IPv4 Static IP Address 192.168.1.126

IPv4 Static Subnet Mask 255.255.255.0

IPv4 Static Gateway 192.168.1.1

IPv4 Configured IP Address 192.168.1.126

IPv4 Subnet Mask 255.255.255.0

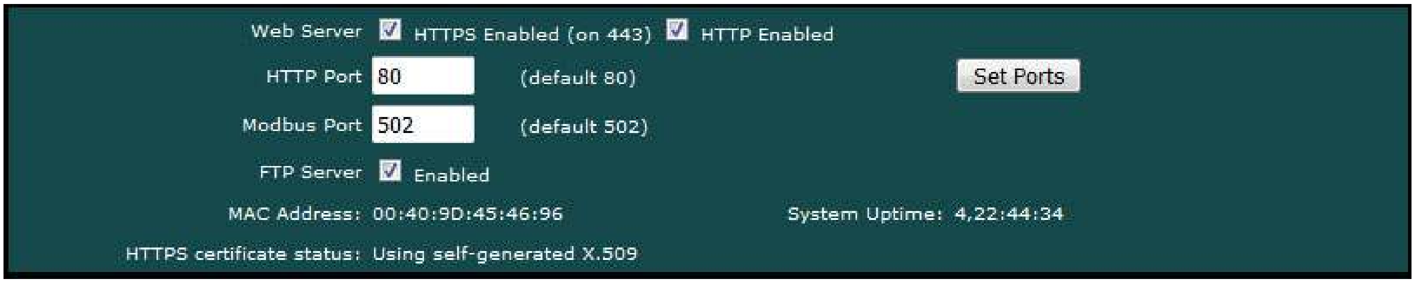
IPv4 Gateway 192.168.1.1

IPv6 Settings ● Disabled ○ Automatic ○ Static

IPv6 Link-Local IP Address fe80::240:9dff:fe45:4696

Aside from setting the IP address, the only other thing you need to confirm is that the

Modbus Port is set to 502 (standard Modbus TCP port), or whatever other non-standard port your devices use for Modbus TCP. If the Modbus Port is set to zero, the Modbus TCP server is disabled and port 502 is not accessible (as sometimes desired for network security).



Web Server ☑ HTTPS Enabled (on 443) ☑ HTTP Enabled

HTTP Port 80 (default 80)

Modbus Port 502 (default 502)

FTP Server ☑ Enabled

MAC Address: 00:40:9D:45:46:96

System Uptime: 4,22:44:34

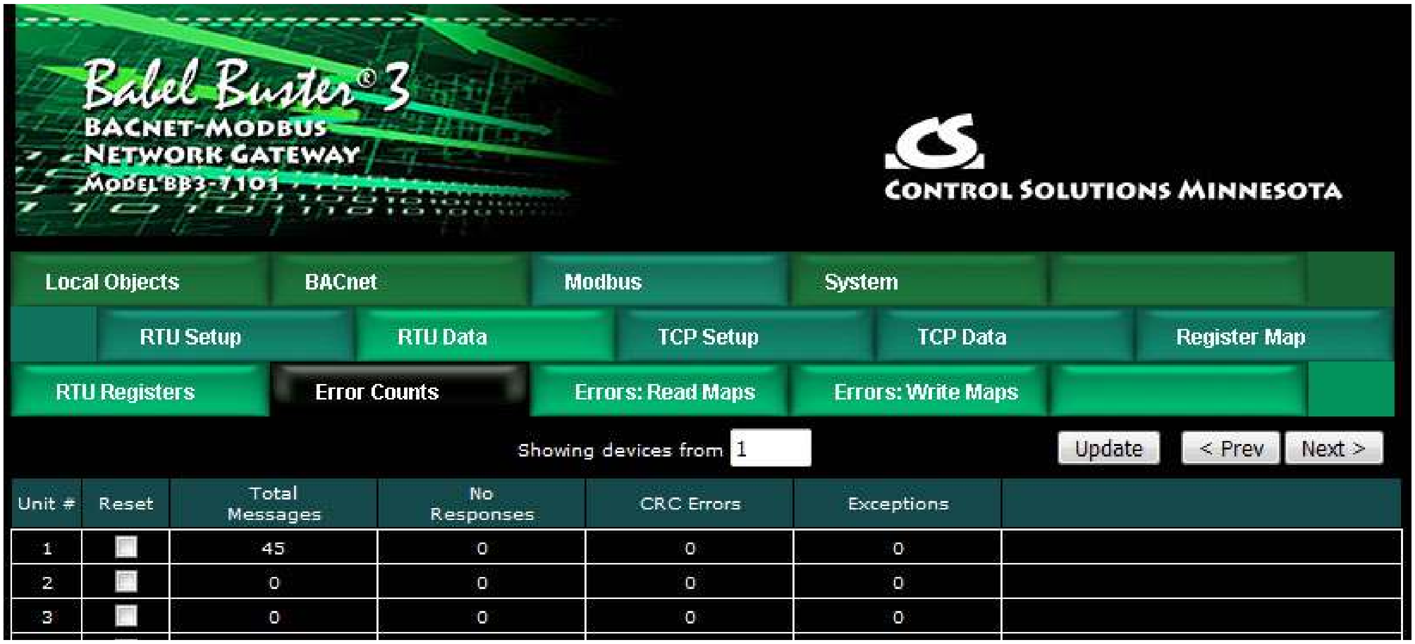
HTTPS certificate status: Using self-generated X.509

10.5 Modbus Slave Diagnostics

The Babel Buster BB3-7101/MX-71 brand new out of the box will have no registers configured, and the RTU side will be configured as Modbus RTU slave, 9600 baud, slave address 1. The TCP side will be configured to its default IP address.

Although no registers are configured, there will be a single holding register that may be read for diagnostic purposes, at register number 8801 (or 48801 if using Modicon notation). The content of this register will be firmware revision expressed as a 3-digit number "abbcc" where "a" is the major revision, "bb" is minor revision, and "cc" is build iteration. This should correspond to the firmware revision displayed on the home page (index.html) of the web user interface for the device, which is displayed as "a.bb.c".

When Modbus RTU is configured as a slave, received messages will be tabulated on the Error Counts page as unit #1 (regardless of which unit the BB3-7101/MX-71 is configured to be). Modbus TCP message counts for messages received as a slave (server) are not counted because the Error Counts page is normally used by the master (client). Since Modbus TCP can be both client and server at the same time, the Error Counts page for TCP will only apply to the client side to avoid confusion about where the counts are coming from. Modbus RTU can only be one or the other, so there is no confusion about where messages are originating.



Showing devices from 1

Unit #	Reset	Total Messages	No Responses	CRC Errors	Exceptions	
1	□	45	0	0	0	
2	□	0	0	0	0	
3	□	0	0	0	0	

Babel Buster® 3
MODEL BB3-7101-SP
MODEL MX-71-SP
NETWORK GATEWAY

CONTROL SOLUTIONS MINNESOTA

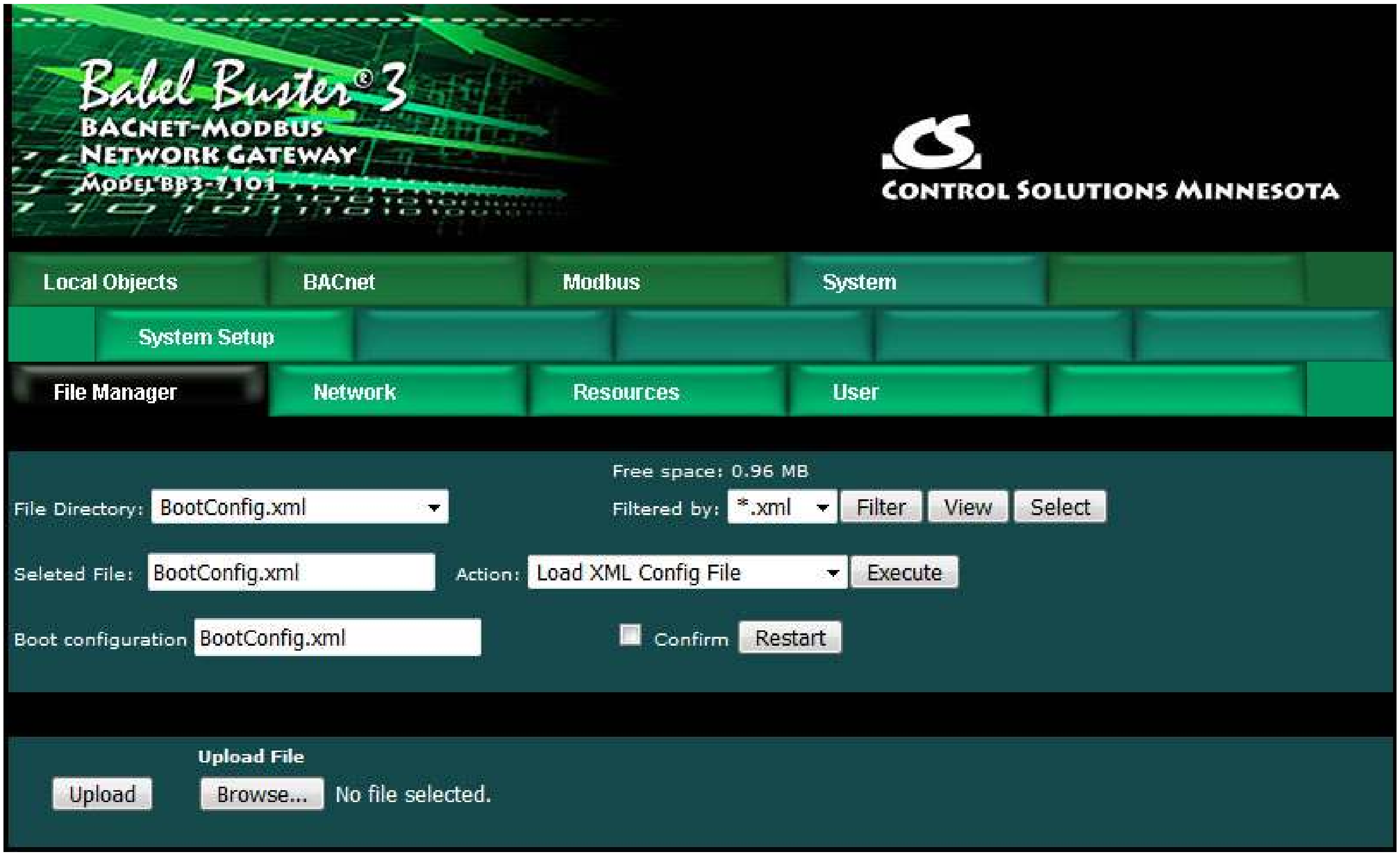
11. System Configuration and Resources

11.1 Using the File Manager

The File Manager page is probably one of the most important pages to know about. Among other things, this is where you tell the gateway to save all of the changes you have made. The various "Update" buttons on the many pages in the web user interface only copy your configuration from your PC's browser to temporary memory in the gateway. To retain those changes indefinitely (i.e. through restart or power cycle), you need to tell the gateway to save those changes in a configuration file.

The configuration files are stored in non-volatile (Flash) memory. The process of reprogramming the Flash takes a little time. It would be cumbersome to rewrite that file every time you made a minor change. Therefore, in the interest of being more responsive, and in the interest of extending the life of the Flash, configuration is only saved to Flash when you direct it to do so.

The File Manager is used in several other ways in addition to managing your XML configuration files. You upload SSL certificates here. You import CSV files for Modbus or BACnet client configuration here.



Free space: 0.96 MB

File Directory: BootConfig.xml

Filtered by: *.xml

Seleted File: BootConfig.xml

Action: Load XML Config File

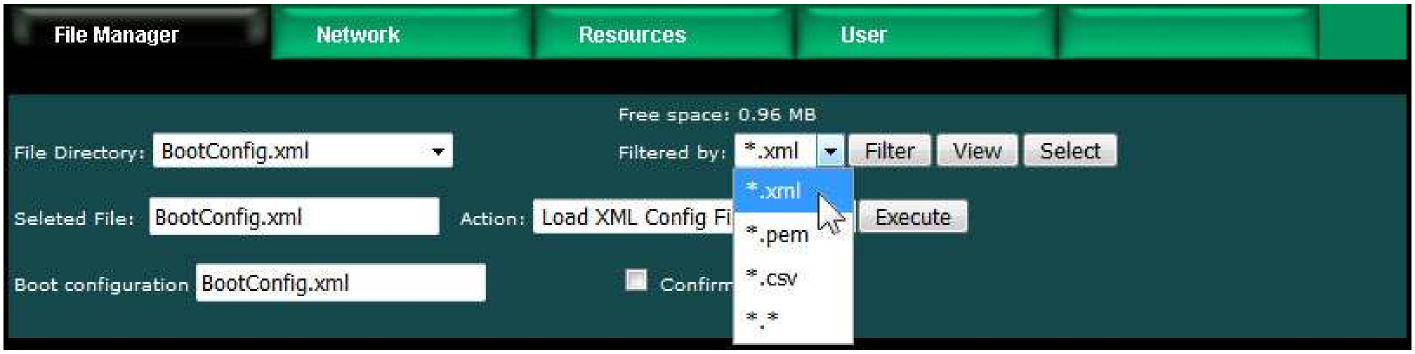
Boot configuration BootConfig.xml

□ Confirm

Upload File

No file selected.

The File Directory is a list of files that are currently stored in the Babel Buster's Flash file system. To filter files by type, select a type from the Filtered by list, and click Filter.



Free space: 0.96 MB

File Directory: BootConfig.xml

Filtered by: *.xml

Seleted File: BootConfig.xml

Action: Load XML Config Fi

Boot configuration BootConfig.xml

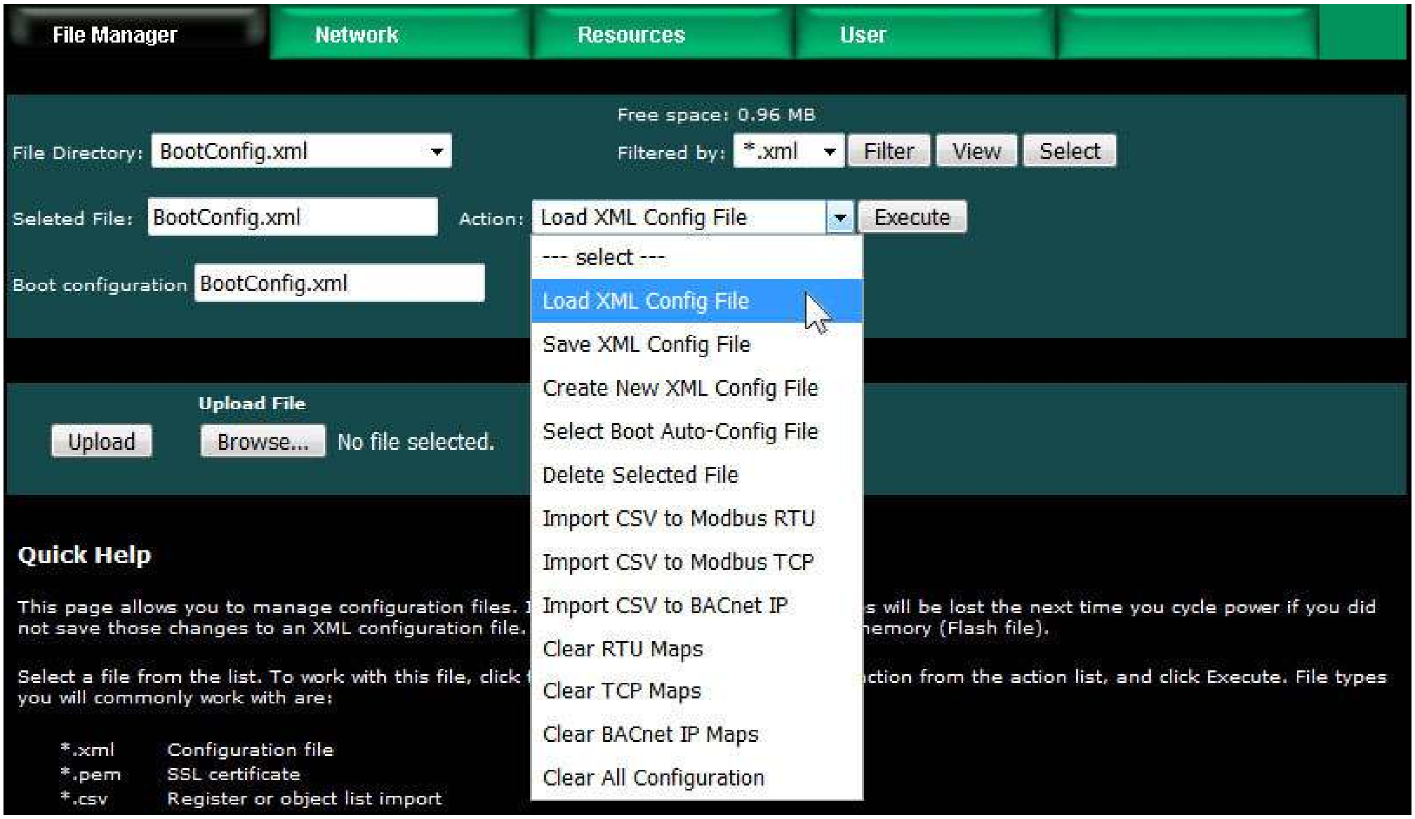
□ Confirm

File type filters are as follows:

*.xml	XML configuration files
*.pem	SSL certificates (for AWS IoT and/or HTTPS)
*.csv	CSV spreadsheet for Modbus register import
.	Display all files

There are several file related actions you may take. To take action with a certain file, select that file from the File Directory list, and click Select. That file should now show up in the Selected File window.

Once a file has been selected, choose your action from the Action list, and click Execute.



Free space: 0.96 MB

File Directory: BootConfig.xml

Filtered by: *.xml

Seleted File: BootConfig.xml

Action: Load XML Config File

Boot configuration BootConfig.xml

Upload File

No file selected.

Quick Help

This page allows you to manage configuration files. I s will be lost the next time you cycle power if you did not save those changes to an XML configuration file. emory (Flash file).

Select a file from the list. To work with this file, click t ction from the action list, and click Execute. File types you will commonly work with are:

*.xml	Configuration file
*.pem	SSL certificate
*.csv	Register or object list import

You must use the Select button to populate the Selected File window prior to executing any action from the list. Choose a file from the drop down list that shows all available files, then click the Select button. You may then act on that file.

You do not need to use the Select button to simply View a file. Clicking View will cause your browser to display the file chosen from the drop down list. If you attempt to View a CSV file, your PC will likely ask if you want to download the file or open it with your spread sheet program (e.g. Excel).

Upload File: To upload a file from your PC to this gateway, use the Browse button to find the file on your PC, open the file in the PC's file dialog box, and then click Upload.

NOTE: If you get a "File upload error: -1" message, click the browser's "back" button, then simply click the View button to view any file (does not matter which file), and then click browser's "back" button again to return to this page. This gets the browser and HTTP server back in sync, and this requirement generally happens only once following power-up.

Restart: To restart the gateway, check Confirm and click Restart. This is a hard reset that will accomplish the same thing as a power cycle without physically disconnecting and reconnecting power.

11.1.1 Load, Save, Create XML Configuration File

IMPORTANT: Configuration files from older gateways are not directly usable in the BB3-7101/MX-71, but the Babel Buster Configuration Builder program can be used to convert a BB2-7010 configuration into a BB3-7101/MX-71 configuration file. See Appendix H.

Load XML Config File: The configuration file shown in the "Boot configuration" window will be loaded automatically at startup. If you have uploaded a new configuration file and wish to use it without restarting, select that file (choose from list, click Select), select this action, and click Execute.

HINT: If you are loading a file generated externally and you get "parameter out of range" errors pertaining to defining objects or "table full" errors while loading maps or rules, you might not have sufficient resources allocated. You may need to increase some counts on the Resources page.

Save XML Config File: Any time you have made configuration changes that you want to retain as permanent, you need to come here, select the file from the directory list, and execute this Save action.

Create New XML Config File: You have the option to create a totally new configuration file. This is often suitable if you started with an existing configuration, made changes, and want to save your changes without replacing the original configuration. To create a new file, rather than selecting a file from the directory list, simply type a new name into the Selected file window. The name cannot contain spaces or special characters, and be sure to use the correct file suffix. Enter the name and execute this action.

11.1.2 Select Startup Configuration

Select Boot Auto-Config File: This is where you tell the Babel Buster what configuration to automatically load upon startup. To set the Boot configuration, select the XML file from the list, and execute this action. The name of the startup file, along with a few other important things like the gateway's own IP address, are stored in a different area of Flash that is not part of the file system.

When selecting a new Boot configuration file, it is a good idea to select the file, and execute Load XML Config File. If there are errors, they will be displayed. If there are errors in the file but you do not fix them, then the gateway will not fully start up the next time it restarts. The web user interface will be available, but it will not be talking to Modbus or BACnet devices.

11.1.3 Delete a File

Remove a file from the Flash file system by selecting it from the list and executing the Delete Selected File action.

11.1.4 Import CSV File

Import CSV to Modbus RTU: You can configure Modbus RTU read and write maps in bulk by importing the maps as a CSV file that you created using a standard spreadsheet program. Refer to Appendix B for details about the CSV format. Note that maps will be added to the existing map list. If you want to replace existing maps with imported maps, execute Clear RTU Maps first.

Import CSV to Modbus TCP: You can configure Modbus TCP read and write maps in bulk by importing the maps as a CSV file that you created using a standard spreadsheet program. Refer to Appendix B for details about the CSV format. Note that maps will be added to the existing map list. If you want to replace existing maps with imported maps, execute Clear TCP Maps first.

Import CSV to BACnet IP: You can configure BACnet IP read and write maps in bulk by importing the maps as a CSV file that you created using a standard spreadsheet program. Refer to Appendix B for details about the CSV format. Note that maps will be added to the existing map list. If you want to replace existing maps with imported maps, execute Clear BACnet IP Maps first.

HINT: If you get "table full" errors while importing CSV files, you might not have sufficient resources allocated. You may need to increase some counts on the Resources page.

11.1.5 Clear Configuration

Clear RTU Maps: Execute this action to clear (completely remove) all Modbus RTU read and write maps.

Clear TCP Maps: Execute this action to clear (completely remove) all Modbus TCP read and write maps. The Modbus TCP device table will be left intact.

Clear BACnet IP Maps: Execute this action to clear (completely remove) all BACnet IP read and write maps. The BACnet IP device table will be left intact.

Clear All Configuration: Execute this action to completely wipe out all configuration. This includes all Modbus maps and devices, all BACnet IP maps and devices, and all local objects. This will put you back to a "reset to factory" condition with the exception that your IP address is left unchanged. (See Appendix A, Section A.6, regarding forced hard configuration reset that includes IP address and root password.) If you want to make the now empty configuration permanent, select the file that is also selected as Boot configuration, and execute the Save XML Config File action.

The other means of completely wiping out all saved configuration is to simple delete the file named as the Boot configuration file, and then restart or power cycle the Babel Buster. Upon restart, a new empty configuration file will be created automatically.

11.2 Configuration Files and Restoring Default Settings

There is a means of restoring the Babel Buster to "manufacturer's default settings". First of all, make sure that the Boot configuration file is set to "BootConfig.xml". Then, after selecting this file as the boot file, delete it. Now restart the gateway. Upon restart, and upon finding that the boot configuration name is BootConfig.xml, and it does not exist, the gateway will automatically create one with default parameters. The automatic creation of a default file will not occur with any other file name.

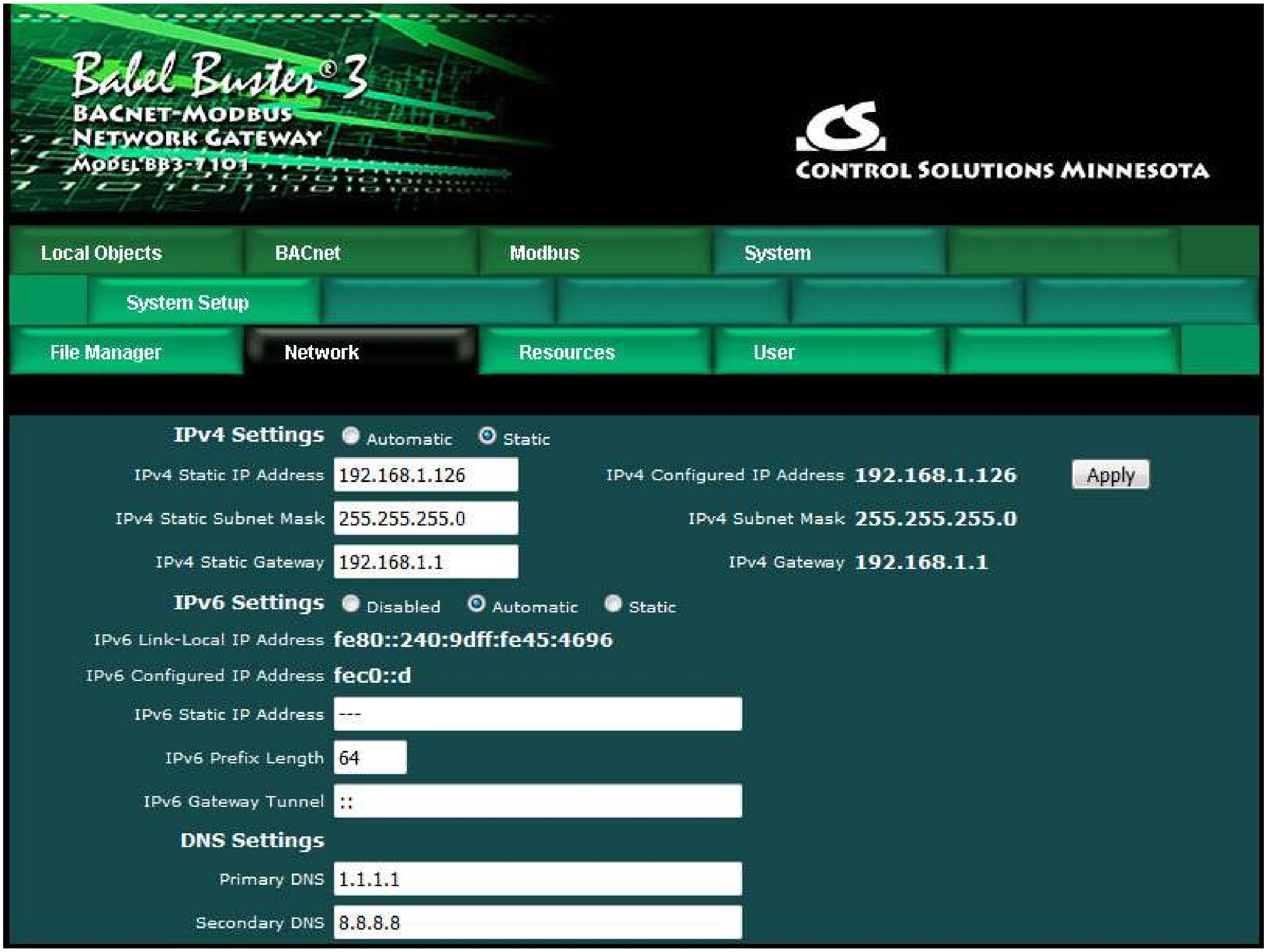
Manual Editing: It is possible to manually edit the XML file outside of the gateway. However, doing so is very prone to errors. If there are errors in the XML file, it will not load successfully on startup. If the configuration does not load on startup, none of the scanners will begin scanning. Because they are all blocked by configuration failure, entering new configuration via the web pages will not result in functionality being restored. You must successfully load a configuration file before the gateway will become functional. To check for errors, select the file here, select Load XML Config File, and click Execute. Error messages that would have been discarded by the automatic loading at startup will now be displayed on an error page if there are any.

Backup Copy of XML Config File: To save a copy of the configuration to your PC, select the file and click the View button. Your browser will now display the XML file. DO NOT do a text copy/paste to try to create an XML file - doing so will result in an invalid file format that cannot be loaded again. You must use the browser's "save as" or "save page" function. The browser should default to wanting to save a file with a .xml suffix. If correctly saved on your PC, you should be able to double click on the saved file and it will result in opening the file automatically in your browser. It was saved correctly if the browser does not give any error messages when displaying the XML (which should now look exactly as it did when you first clicked the View button). Saving the configuration file to your PC, and then uploading on a different device, is a quick and easy way to configure two Babel Busters the same way.

Note about caching: Your browser may cache files. If you view a file, make configuration changes, save the file, then view the file again, you may see the old file cached by the browser. To see the updated file, go to "Options" in your browser's tools menu, and delete temporary Internet files (or delete cache files). Also, if you upload a file, make changes on your PC, and re-upload the same file, the browser may send the old file. Again, you will need to find the button inside your browser options that lets you delete the cached files from your PC. To upload a configuration file from your PC to the gateway, use the Browse button to find the file on your PC, open the file in the PC's file dialog box, and then click Upload.

11.3 Network Configuration

The Network Configuration page is where you set the Babel Buster's IP address as well as a few other important things.



IPv4 Settings ○ Automatic ● Static

IPv4 Static IP Address	192.168.1.126	IPv4 Configured IP Address	192.168.1.126
IPv4 Static Subnet Mask	255.255.255.0	IPv4 Subnet Mask	255.255.255.0
IPv4 Static Gateway	192.168.1.1	IPv4 Gateway	192.168.1.1

IPv6 Settings ○ Disabled ● Automatic ○ Static

IPv6 Link-Local IP Address fe80::240:9dff:fe45:4696

IPv6 Configured IP Address fec0::d

IPv6 Static IP Address ---

IPv6 Prefix Length 64

IPv6 Gateway Tunnel ::

DNS Settings

Primary DNS 1.1.1.1

Secondary DNS 8.8.8.8

To change the IP address(es) of this device, make the applicable entries and click Apply. The "automatic" selection means DHCP. Changes to the IPv4 IP address will take effect upon the next system restart.

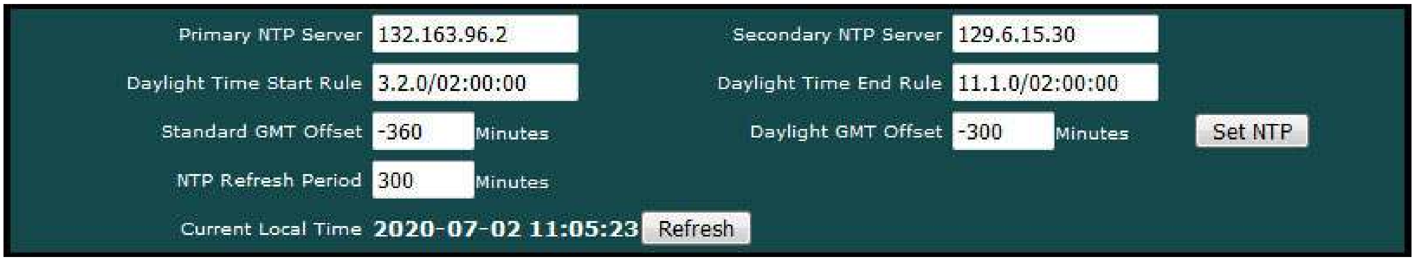
If IPv6 is enabled, IPv6 will always have a Link-Local address, plus one configured address. The configured address will be either the static IP address, or an IPv6 address obtained from an IPv6 DHCP server. If no configured address appears, the DHCP server may have been unreachable.

The IPv6 static IP address window is the configured static address. If "Static" is selected and a new IP address entered as the static address, this new address will not take effect until the next system restart.

The numbers shown to the right of the IPv4 input windows are the actual numbers currently in use. If static IP addresses have been entered but the gateway has not been restarted yet, these numbers will not be the same.

You may use domain names instead of static IP addresses in several instances. If domain names are used, you must supply the IP address of at least one DNS server here. The DNS server must be at a static IP address. These changes take effect immediately. Note: If you are using DHCP, the DNS addresses will be supplied by the DHCP server and should be set to 0.0.0.0 here.

The Babel Buster maintains time and date via SNTP services.



Primary NTP Server 132.163.96.2
Secondary NTP Server 129.6.15.30
Daylight Time Start Rule 3.2.0/02:00:00
Daylight Time End Rule 11.1.0/02:00:00
Standard GMT Offset -360 Minutes
Daylight GMT Offset -300 Minutes
NTP Refresh Period 300 Minutes
Current Local Time 2020-07-02 11:05:23

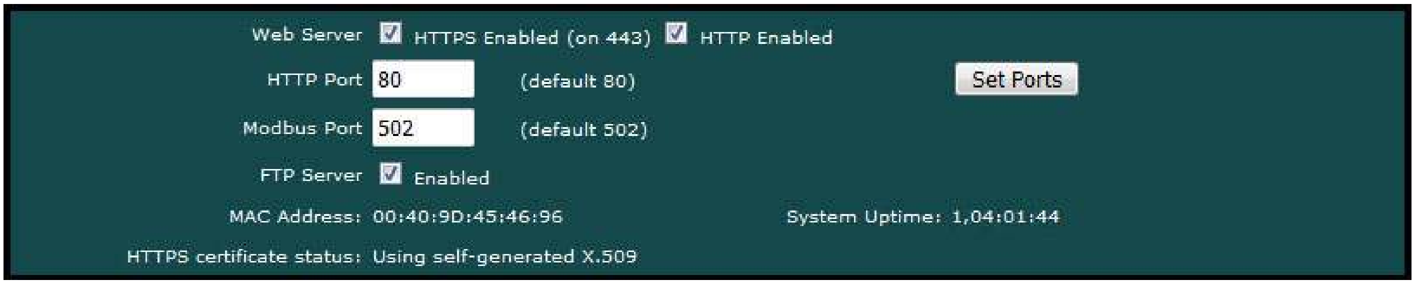
NTP setup: Enter a primary and secondary IP address of NTP servers, such as those found at www.nist.gov (go to http://tf.nist.gov/tf-cgi/servers.cgi to find more). Enter daylight start/end rules, and offset from GMT for both standard and daylight time. Offset is a negative number in the western hemisphere. Enter an NTP update time in minutes. Do not set NTP to update too frequently or you risk being denied service by the NTP server. Click the Set NTP button after all settings have been made. The Flash update will take several seconds. The initial update of local time may take a minute or two. You may need to restart the Babel Buster if NTP had never before been initialized.

Daylight savings time start/end rules consist of "date/time" where the date (m.n.d) indicates the day when summer time starts or ends, and time (hour:min:sec) is the current local time when summer time starts/ends. The date portion of the rule is formatted as follows:

- m indicates the month (1 <= m <= 12)
- n indicates which week of the month (1 <= n <= 5). 5 = the last week in the month.
- d indicates what day of the week (0 <= d <= 6). 0 = Sunday

For example: Start "4.1.0/02:00:00", end "10.5.0/02:00:00" means summer time starts at 2am on the first Sunday in April and ends at 2am on last Sunday in October. That was the old US rule. The new US rule is start "3.2.0/02:00:00" and end "11.1.0/02:00:00", which is start at 2am on the second Sunday in March, end at 2am on the first Sunday in November.

Note about time maintained here: Modbus and BACnet gateway functionality has no use for time and date. The only time you might have a need for valid time and date is when using SSL certificates for secure connections. If you are using a secure web connection and having trouble connecting, be sure NTP is set up here.



Web Server ☑ HTTPS Enabled (on 443) ☑ HTTP Enabled
HTTP Port 80 (default 80)
Modbus Port 502 (default 502)
FTP Server ☑ Enabled
MAC Address: 00:40:9D:45:46:96
System Uptime: 1,04:01:44
HTTPS certificate status: Using self-generated X.509

Secure browsing can be enabled here, and non-secure can be disabled. You cannot disable both, and a forced configuration reset will restore HTTP (non-secure) web browsing. In order to use HTTPS, you must first upload the necessary SSL certificates (see Appendix G) or allow the certificates to be self-generated by explicitly deleting existing certificates.

IMPORTANT: It is highly recommended that in making the transition from HTTP to HTTPS, you enable both until you confirm HTTPS is functional. If there is a problem with the SSL certificates provided for HTTPS, then HTTPS will not run and you will find an error message on the "HTTPS certificate status" line. If you disable standard HTTP without first verifying that HTTPS is functional, you may end up locked out and will then need to do a forced hard reset (Appendix A.6).

The HTTP port for browsing the user interface can be moved away from the default HTTP port 80. Select a different port, click Set Ports, and then restart the gateway to make that new port take effect. Don't forget to append the port number to the gateway's IP address when attempting to browse the web user interface if it has been moved away from port 80.

The Modbus port will be set to zero, meaning Modbus TCP is disabled, when the device is new. Enter the standard port 502 and click Set Ports to set the Modbus port. Set the port to some other port if you know that Modbus TCP operates on a non-standard port on your network. The device needs to be restarted after changing the Modbus TCP port.

FTP is enabled by default to allow firmware update uploads. It may be optionally disabled here. Just remember to enable it again before attempting a firmware update.

Any changes to this section (Set Ports button) require restarting the Babel Buster before they will take effect.

11.4 Resource Allocation

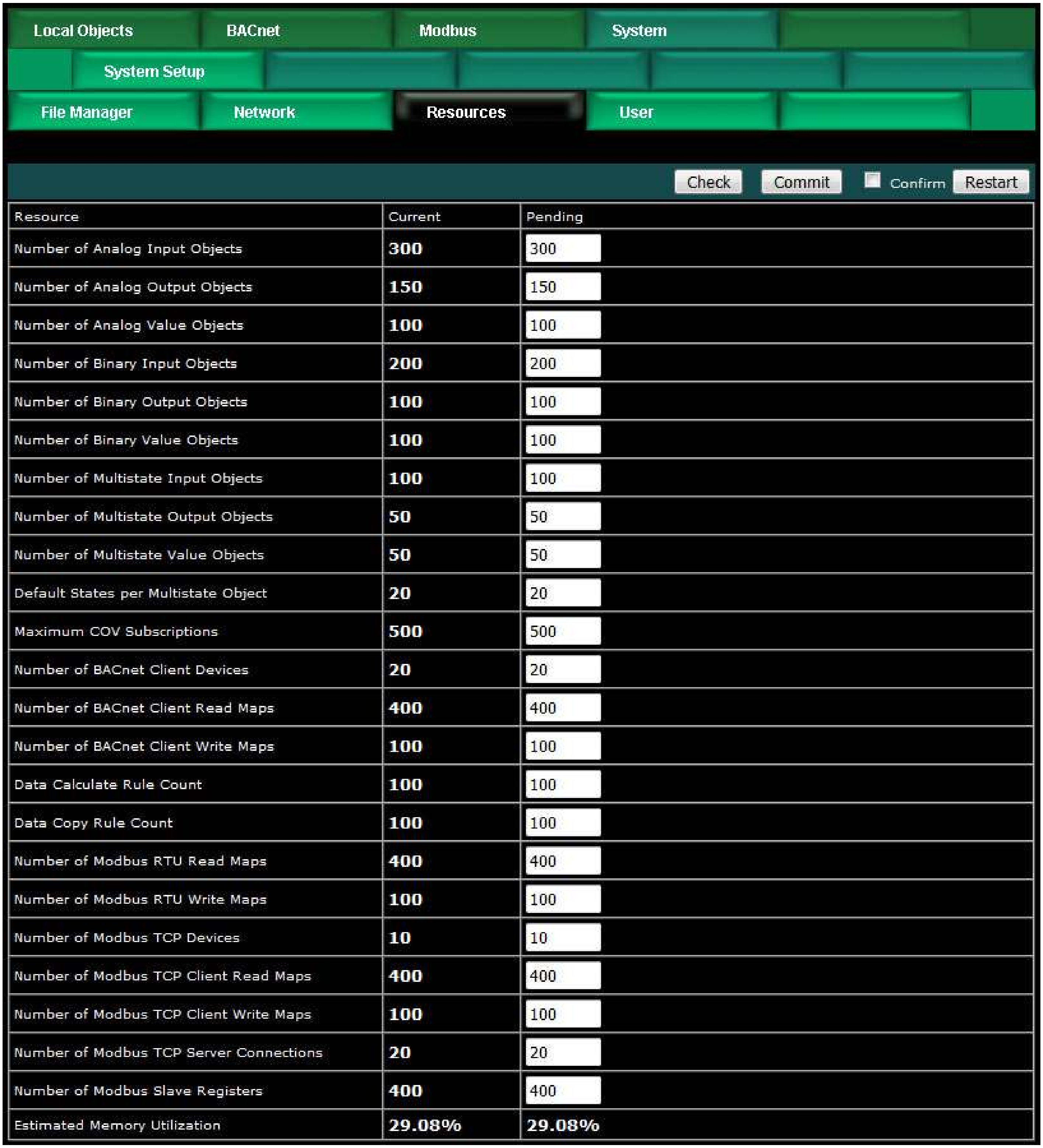
Historically, Control Solutions gateways had a fixed set of BACnet objects and other resources to work with. Invariably, there were always users that wanted less of this and more of that. Therefore, while there are still maximums imposed, you can now shift resources around as best suits your application. An example is shown below.

The values in the Pending column are those found in the most recently loaded XML configuration file. When saving or creating a new XML file, the numbers in the Current column will be written to the file. To change the allocations, change numbers in the Pending column. When you are ready to commit these changes, click the Commit button. To cause the changes to go into use, you must restart the device since memory allocation can occur only once at startup.

You can click the Check button prior to Commit to see if the values you have entered will be accepted. If adjustments need to be made, the values in the Pending column will be updated.

The first time you visit this page, you will see the initial default values. Should you change any of them, minimums and maximums currently defined in firmware will be imposed. If you see a value smaller than what you entered, it may be that you had exceeded the internal limit.

If you see that numbers toward the top of the list are large, and numbers near the bottom are all set to 1, it means the system has run out of free memory and you need to reallocate resources.



Resource	Current	Pending
Number of Analog Input Objects	300	300
Number of Analog Output Objects	150	150
Number of Analog Value Objects	100	100
Number of Binary Input Objects	200	200
Number of Binary Output Objects	100	100
Number of Binary Value Objects	100	100
Number of Multistate Input Objects	100	100
Number of Multistate Output Objects	50	50
Number of Multistate Value Objects	50	50
Default States per Multistate Object	20	20
Maximum COV Subscriptions	500	500
Number of BACnet Client Devices	20	20
Number of BACnet Client Read Maps	400	400
Number of BACnet Client Write Maps	100	100
Data Calculate Rule Count	100	100
Data Copy Rule Count	100	100
Number of Modbus RTU Read Maps	400	400
Number of Modbus RTU Write Maps	100	100
Number of Modbus TCP Devices	10	10
Number of Modbus TCP Client Read Maps	400	400
Number of Modbus TCP Client Write Maps	100	100
Number of Modbus TCP Server Connections	20	20
Number of Modbus Slave Registers	400	400
Estimated Memory Utilization	29.08%	29.08%

The estimated memory utilization shown at the bottom gives you an indication of how close you are to running out of memory. You will not be allowed to commit a resource allocation greater than 100%.

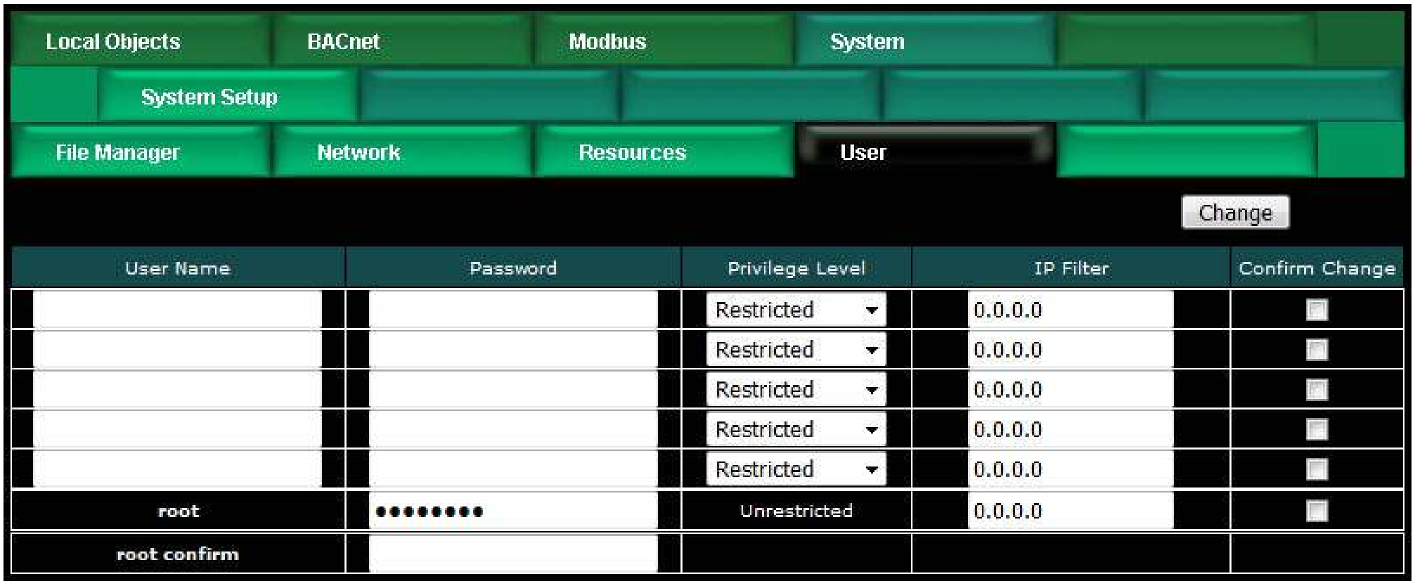
The object count limits for BACnet objects are set to 5,000 objects. However, if you were to try to allocate 5,000 of each type of object, you would never get there due to running out of resources. Control Solutions has tested a configuration in which 5,000 Modbus registers were being read via Modbus TCP (using 5,000 Modbus TCP read maps) and distributed to 5,000 BACnet Analog Input objects. This process was repeatable every 7 seconds in this particular benchmark.

11.5 User Login Passwords

There is only one default login provided initially, namely the username "root" with a unique password generated specifically for your particular Babel Buster. This password is provided to you in either external documentation included with the gateway, or it may be found on a label attached to the gateway. Network security laws in some jurisdictions require that Internet connected (or connectable) devices be shipped with unique default passwords, and the BB3-7101/MX-71 complies with this requirement.

Additional user logins may be created. The privilege level Administrator lets that user see and change anything. The privilege level Maintenance allows the user to log in and see (and change) values in the local objects via the Local Objects page, but cannot access any other pages. The Restricted level has no meaning in the BB3-7101/MX-71 (other than block access to everything) since it does not operate as a user defined web server.

You also have the option of IP filtering. If set, then the user can only access Babel Buster's web pages from that IP address. Leave set to 0.0.0.0 to disable filtering.



User Name	Password	Privilege Level	IP Filter	Confirm Change
		Restricted	0.0.0.0	☐
		Restricted	0.0.0.0	☐
		Restricted	0.0.0.0	☐
		Restricted	0.0.0.0	☐
		Restricted	0.0.0.0	☐
root	●●●●●●●●	Unrestricted	0.0.0.0	☐
root confirm				

12. Trouble Shooting

12.1 BACnet IP Trouble Shooting

12.1.1 Most Common Problems

BACnet IP is typically easier to get running than MS/TP just because Ethernet is pretty straight forward. The most frequent problem is "no response" or timeout. The most common cause of this problem for BACnet IP is a network configuration problem, such as incorrect IP address or IP address that cannot be reached as configured. The problem sometimes lies outside the Babel Buster and may require consulting with the IT personnel responsible for the network if on a large network.

The subnet mask determines what part of the IP address constitutes the domain, and all devices on the same network must be on the same domain before they can communicate.

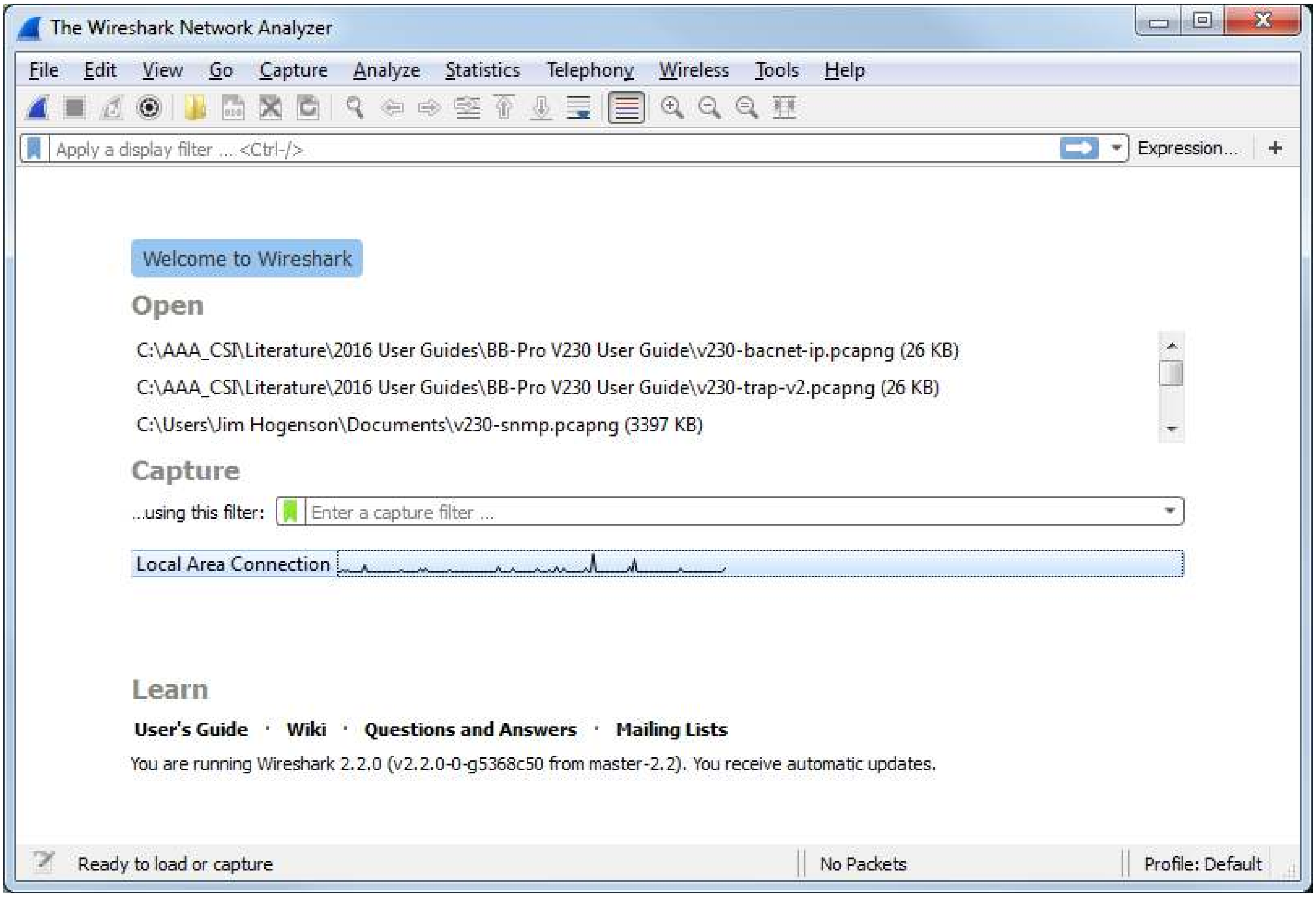
Obviously two devices being assigned the same IP address is going to cause trouble. If you can communicate at all with a device having a duplicate IP address, it will be intermittent, and potentially erratic as the other device having the same IP address may be responding to your queries.

If you are connecting via one or more routers, then everything that applies to routing issues will apply to your device. A complete discussion of NAT routing, BACnet routing, etc, is beyond the scope of this document - you should refer these questions to your IT administrator when applicable.

Once the BB3-7101/MX-71 is communicating BACnet IP, then next area for possible concern is with the BACnet client. If the gateway is supposed to be polling other IP devices, but the data does not appear correct, the first thing to check is the reliability code. Any reliability code other than zero is a problem. Refer to the list at the bottom of any of the Data Objects pages for explanation of the non-zero codes. If the reliability code indicates that an error was returned by the server (meaning the other BACnet device you are trying to query), then refer to the BACnet Diagnostics page for additional error information.

12.1.2 Using Wireshark

One of the most useful tools for diagnosing BACnet IP problems is Wireshark. You can get a free copy at www.wireshark.org. Additional important information about Wireshark can be found in Appendix F of this user guide. When you start Wireshark, the startup screen appears as follows (as of this writing). Click on Local Area Connection to begin captureing traffic.



Open

C:\AAA_CSI\Literature\2016 User Guides\BB-Pro V230 User Guide\v230-bacnet-ip.pcapng (26 KB)

C:\AAA_CSI\Literature\2016 User Guides\BB-Pro V230 User Guide\v230-trap-v2.pcapng (26 KB)

C:\Users\Jim Hogenson\Documents\v230-snmp.pcapng (3397 KB)

Capture

...using this filter: Enter a capture filter ...

Local Area Connection

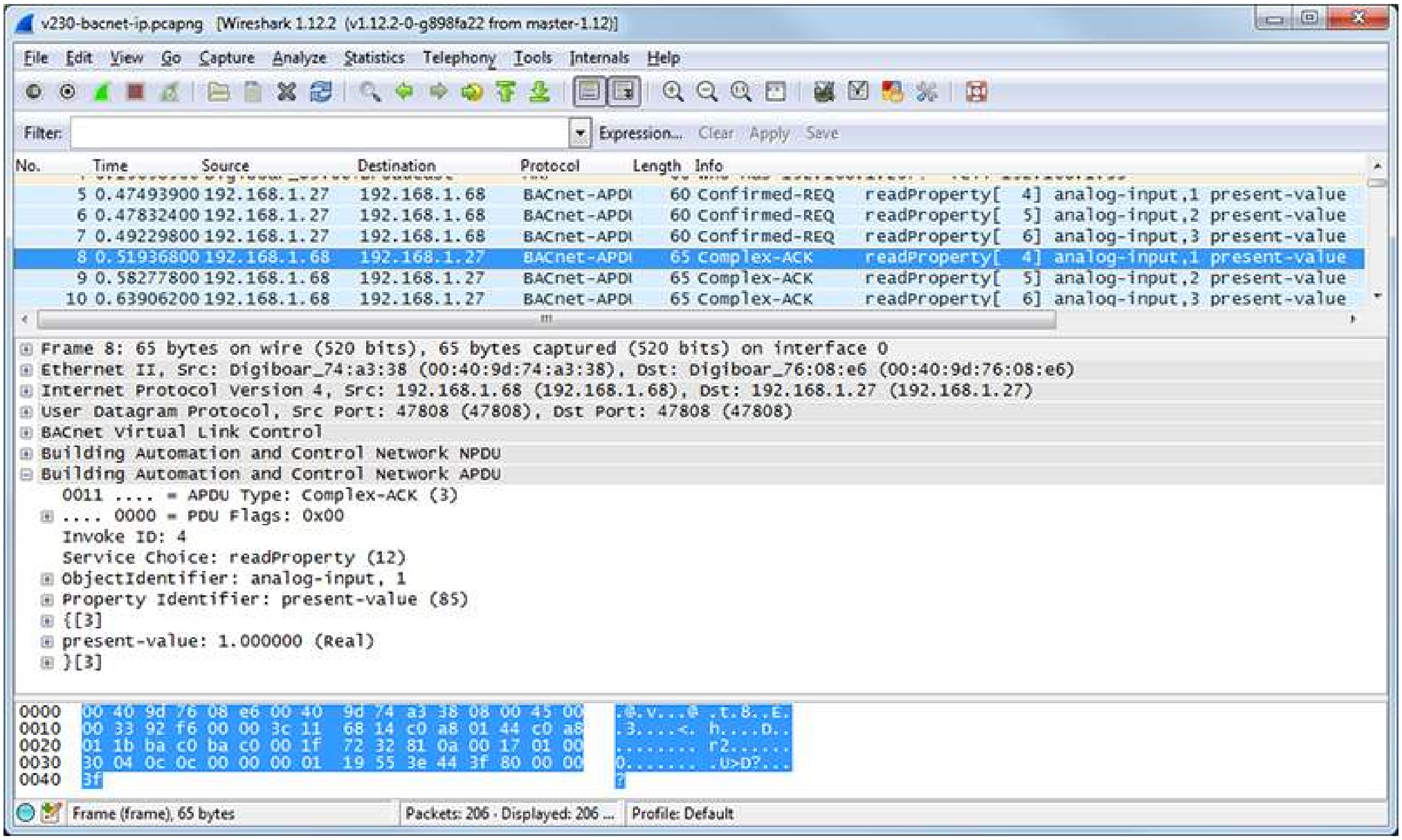
Learn

User's Guide · Wiki · Questions and Answers · Mailing Lists

You are running Wireshark 2.2.0 (v2.2.0-0-g5368c50 from master-2.2). You receive automatic updates.

Ready to load or capture | No Packets | Profile: Default

Network packet capture will be live with Ethernet. Click on any packet of interest, and you can expand the tree structure to see the full content of the packet. Wireshark includes complete decoding for BACnet protocol - a very useful feature.


v230-bacnet-ip.pcapng [Wireshark 1.12.2 (v1.12.2-0-g898fa22 from master-1.12)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter:
Expression... Clear Apply Save

```
No.   Time        Source        Destination   Protocol     Length Info
  5 0.47493900 192.168.1.27  192.168.1.68  BACnet-APDU  60 Confirmed-REQ  readProperty[  4] analog-input,1 present-value
  6 0.47832400 192.168.1.27  192.168.1.68  BACnet-APDU  60 Confirmed-REQ  readProperty[  5] analog-input,2 present-value
  7 0.49229800 192.168.1.27  192.168.1.68  BACnet-APDU  60 Confirmed-REQ  readProperty[  6] analog-input,3 present-value
  8 0.51936800 192.168.1.68  192.168.1.27  BACnet-APDU  65 Complex-ACK    readProperty[  4] analog-input,1 present-value
  9 0.58277800 192.168.1.68  192.168.1.27  BACnet-APDU  65 Complex-ACK    readProperty[  5] analog-input,2 present-value
 10 0.63906200 192.168.1.68  192.168.1.27  BACnet-APDU  65 Complex-ACK    readProperty[  6] analog-input,3 present-value
```

```
⊞ Frame 8: 65 bytes on wire (520 bits), 65 bytes captured (520 bits) on interface 0
⊞ Ethernet II, Src: Digiboar_74:a3:38 (00:40:9d:74:a3:38), Dst: Digiboar_76:08:e6 (00:40:9d:76:08:e6)
⊞ Internet Protocol Version 4, Src: 192.168.1.68 (192.168.1.68), Dst: 192.168.1.27 (192.168.1.27)
⊞ User Datagram Protocol, Src Port: 47808 (47808), Dst Port: 47808 (47808)
⊞ BACnet Virtual Link Control
⊞ Building Automation and Control Network NPDU
⊟ Building Automation and Control Network APDU
    0011 .... = APDU Type: Complex-ACK (3)
  ⊞ .... 0000 = PDU Flags: 0x00
    Invoke ID: 4
    Service Choice: readProperty (12)
  ⊞ ObjectIdentifier: analog-input, 1
  ⊞ Property Identifier: present-value (85)
  ⊞ {[3]
  ⊞ present-value: 1.000000 (Real)
  ⊞ }[3]
```

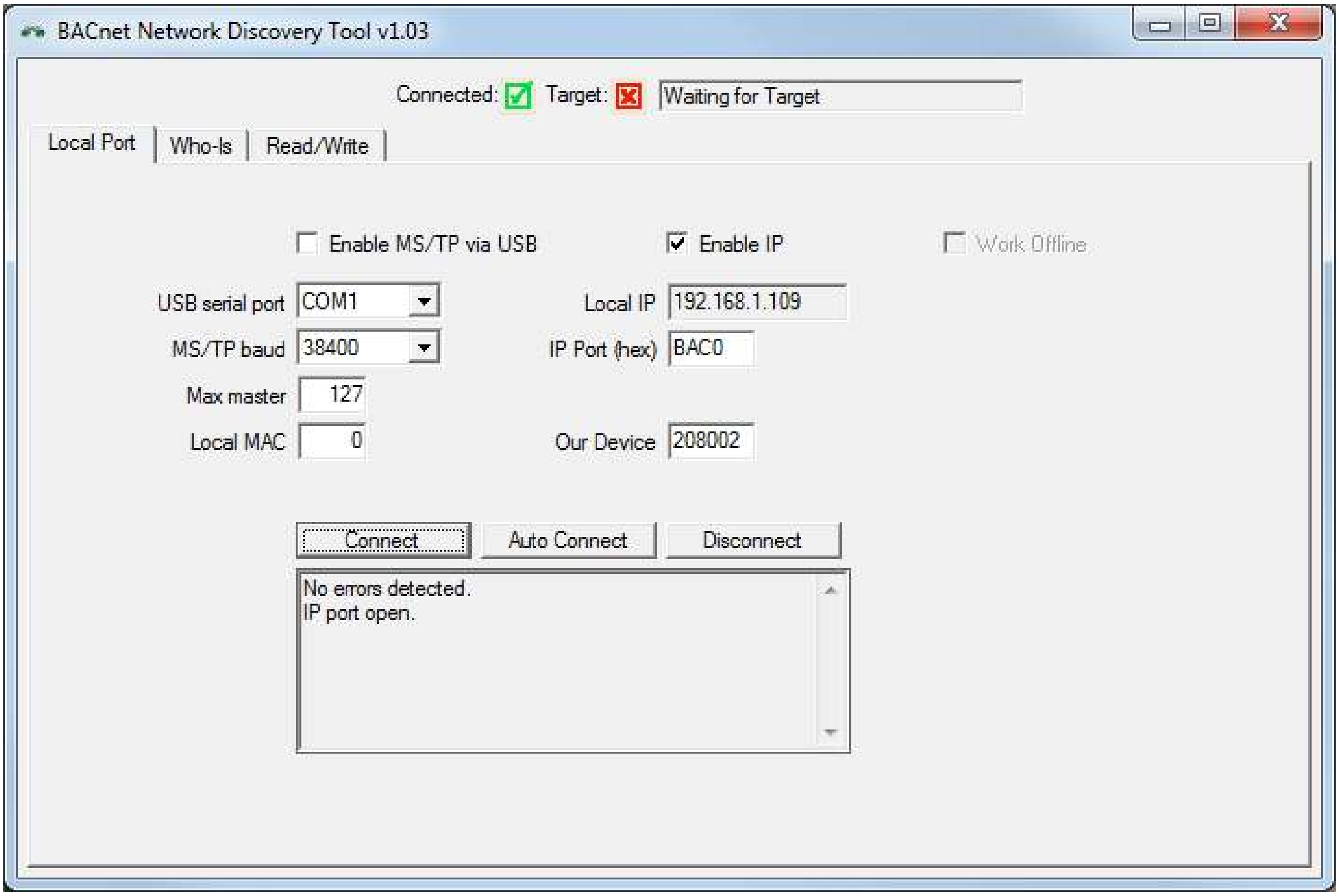
```
0000  00 40 9d 76 08 e6 00 40  9d 74 a3 38 08 00 45 00   .@.v...@ .t.8..E.
0010  00 33 92 f6 00 00 3c 11  68 14 c0 a8 01 44 c0 a8   .3....<. h....D..
0020  01 1b ba c0 ba c0 00 1f  72 32 81 0a 00 17 01 00   ........ r2......
0030  30 04 0c 0c 00 00 00 01  19 55 3e 44 3f 80 00 00   0....... .U>D?...
0040  3f                                                 ?
```

Frame (frame), 65 bytes | Packets: 206 · Displayed: 206 ... | Profile: Default

12.1.3 Using Network Discovery Tool

Control Solutions has created a Network Discovery Tool to perform simple diagnostics on BACnet devices and networks. It works with BACnet IP using your PC's Ethernet connection - assuming your PC is connected to the BACnet IP network.

Simple check Enable IP, and click Connect to begin.



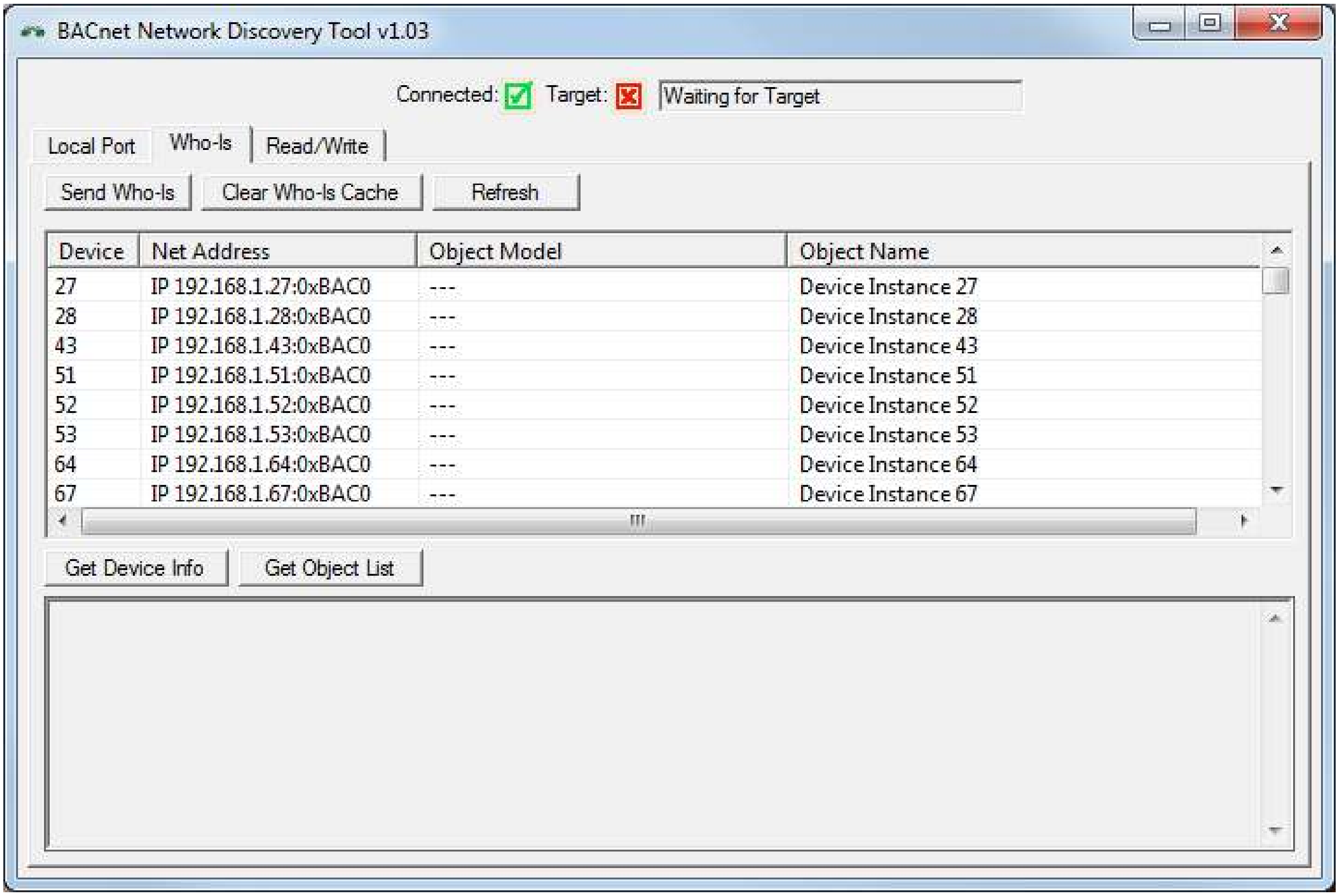
Connected: ☑ Target: ☒ Waiting for Target

☐ Enable MS/TP via USB ☑ Enable IP ☐ Work Offline

USB serial port COM1
MS/TP baud 38400
Max master 127
Local MAC 0
Local IP 192.168.1.109
IP Port (hex) BAC0
Our Device 208002

```
No errors detected.
IP port open.
```

Once connected, go to the Who-Is page. Usually, by the time you get there, the results of the first automatic Who-Is are already displayed.

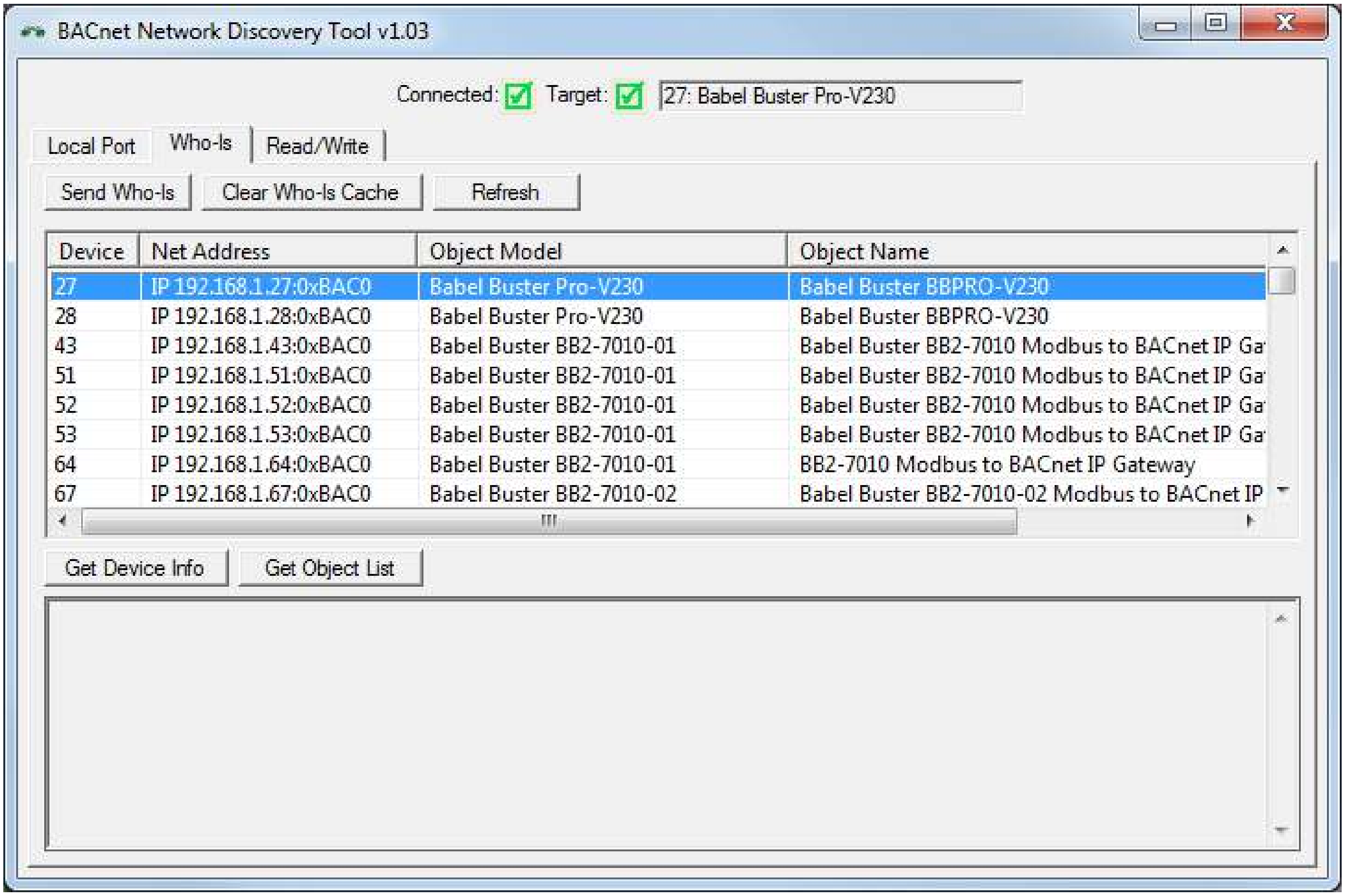


Connected: ☑ Target: ☒ Waiting for Target

Device	Net Address	Object Model	Object Name
27	IP 192.168.1.27:0xBAC0	---	Device Instance 27
28	IP 192.168.1.28:0xBAC0	---	Device Instance 28
43	IP 192.168.1.43:0xBAC0	---	Device Instance 43
51	IP 192.168.1.51:0xBAC0	---	Device Instance 51
52	IP 192.168.1.52:0xBAC0	---	Device Instance 52
53	IP 192.168.1.53:0xBAC0	---	Device Instance 53
64	IP 192.168.1.64:0xBAC0	---	Device Instance 64
67	IP 192.168.1.67:0xBAC0	---	Device Instance 67

Click the Refresh button to cause the discovery tool to query every responding device to read object model and device oblect name from each of them.

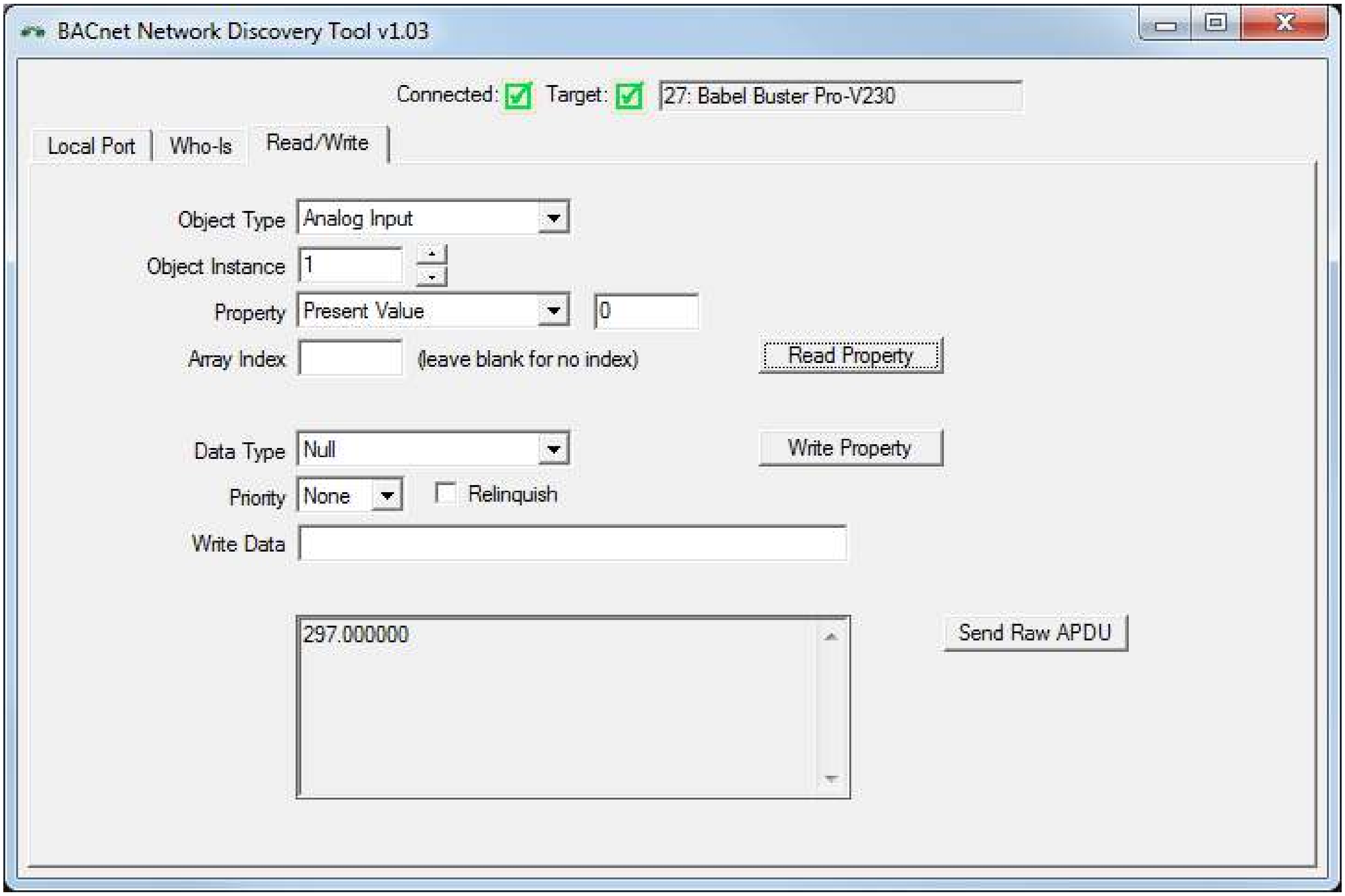
Double click on the device you with to query further. It will now appear as the Target.



Connected: ☑ Target: ☑ 27: Babel Buster Pro-V230

Device	Net Address	Object Model	Object Name
27	IP 192.168.1.27:0xBAC0	Babel Buster Pro-V230	Babel Buster BBPRO-V230
28	IP 192.168.1.28:0xBAC0	Babel Buster Pro-V230	Babel Buster BBPRO-V230
43	IP 192.168.1.43:0xBAC0	Babel Buster BB2-7010-01	Babel Buster BB2-7010 Modbus to BACnet IP Ga
51	IP 192.168.1.51:0xBAC0	Babel Buster BB2-7010-01	Babel Buster BB2-7010 Modbus to BACnet IP Ga
52	IP 192.168.1.52:0xBAC0	Babel Buster BB2-7010-01	Babel Buster BB2-7010 Modbus to BACnet IP Ga
53	IP 192.168.1.53:0xBAC0	Babel Buster BB2-7010-01	Babel Buster BB2-7010 Modbus to BACnet IP Ga
64	IP 192.168.1.64:0xBAC0	Babel Buster BB2-7010-01	BB2-7010 Modbus to BACnet IP Gateway
67	IP 192.168.1.67:0xBAC0	Babel Buster BB2-7010-02	Babel Buster BB2-7010-02 Modbus to BACnet IP

You can read and write properties in any of the standard objects typically used in any Control Solutions device and in most other devices. Select object type, instance, and property to read data by clicking the Read Property button.



Connected: ☑ Target: ☑ 27: Babel Buster Pro-V230

Object Type: Analog Input

Object Instance: 1

Property: Present Value 0

Array Index: (leave blank for no index)

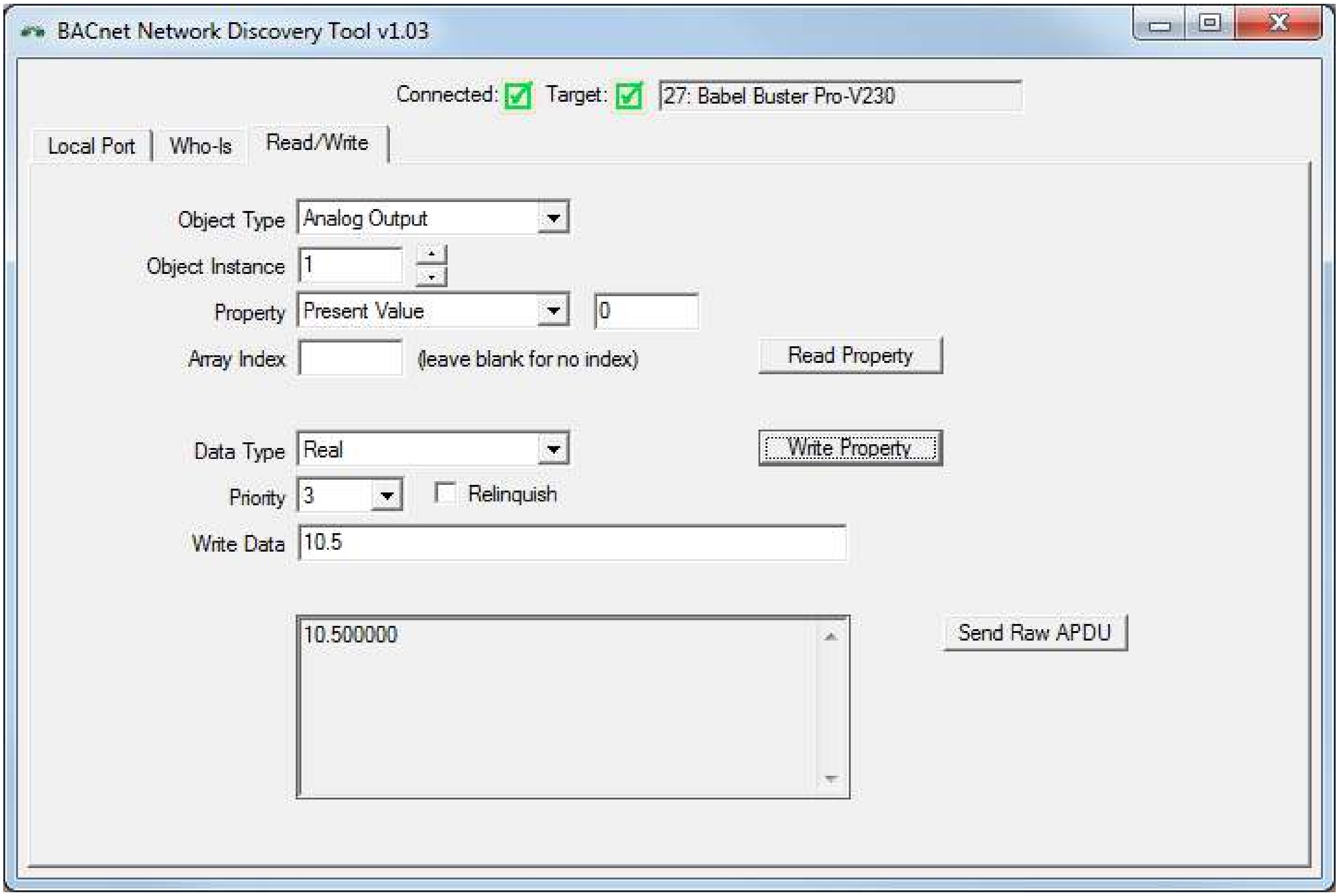
Data Type: Null

Priority: None □ Relinquish

Write Data:

```
297.000000
```

In addition to selecting the same parameters you would for reading, select data type, priority if writing to a commandable object, and data value to write that property by clicking the Write Property button.



BACnet Network Discovery Tool v1.03

Connected: ☑ Target: ☑ 27: Babel Buster Pro-V230

Object Type: Analog Output

Object Instance: 1

Property: Present Value 0

Array Index: (leave blank for no index)

Data Type: Real

Priority: 3 □ Relinquish

Write Data: 10.5

```
10.500000
```

12.2 Modbus RTU Trouble Shooting

You will find message and error counters listed on the Error Counts page under RTU Data. If the Babel Buster is configured as Modbus master, then the Error Counts page will list counts by slave address. If the Babel Buster is configured as Modbus slave, then errors show up on the first line (Unit # 1) regardless of what address the Babel Buster is configured to be.

The Errors: Read Maps and Errors: Write Maps pages will tell you exactly which maps are getting errors when the Babel Buster is configured as Modbus Master.

The most frequent problem is "no response" or timeout. This means the master and slave are not connecting for any of several possible reasions: (a) There is a wiring problem; (b) Port parameters are not configured the same (baud rate, etc); (c) Master's timeout setting is too short.

When it comes to wiring, remember that RS-458 is NOT truly a 2-wire interface as it is commonly referred to. Refer to the wiring diagram in Appendix A.1, and note that there must be a ground path between the Babel Buster and the Modbus RTU device. Refer to the RS-485 FAQ under Support at csimn.com if you have questions or concerns about wiring.

If you are getting CRC errors, that is almost always a wiring problem, but can be a port problem such as mismatched parity setting. A CRC error will not be caused by incorrect configuration of a Read Map or Write Map.

If you are getting exception errors, that is somewhat good news - it means that at least you are successfully communicating. An exception error most often means the master is asking the slave for a register that the slave does not have. If the Babel Buster is configured as Modbus master, this means the Read Map or Write Map is not configured correctly.

12.3 Modbus TCP Trouble Shooting

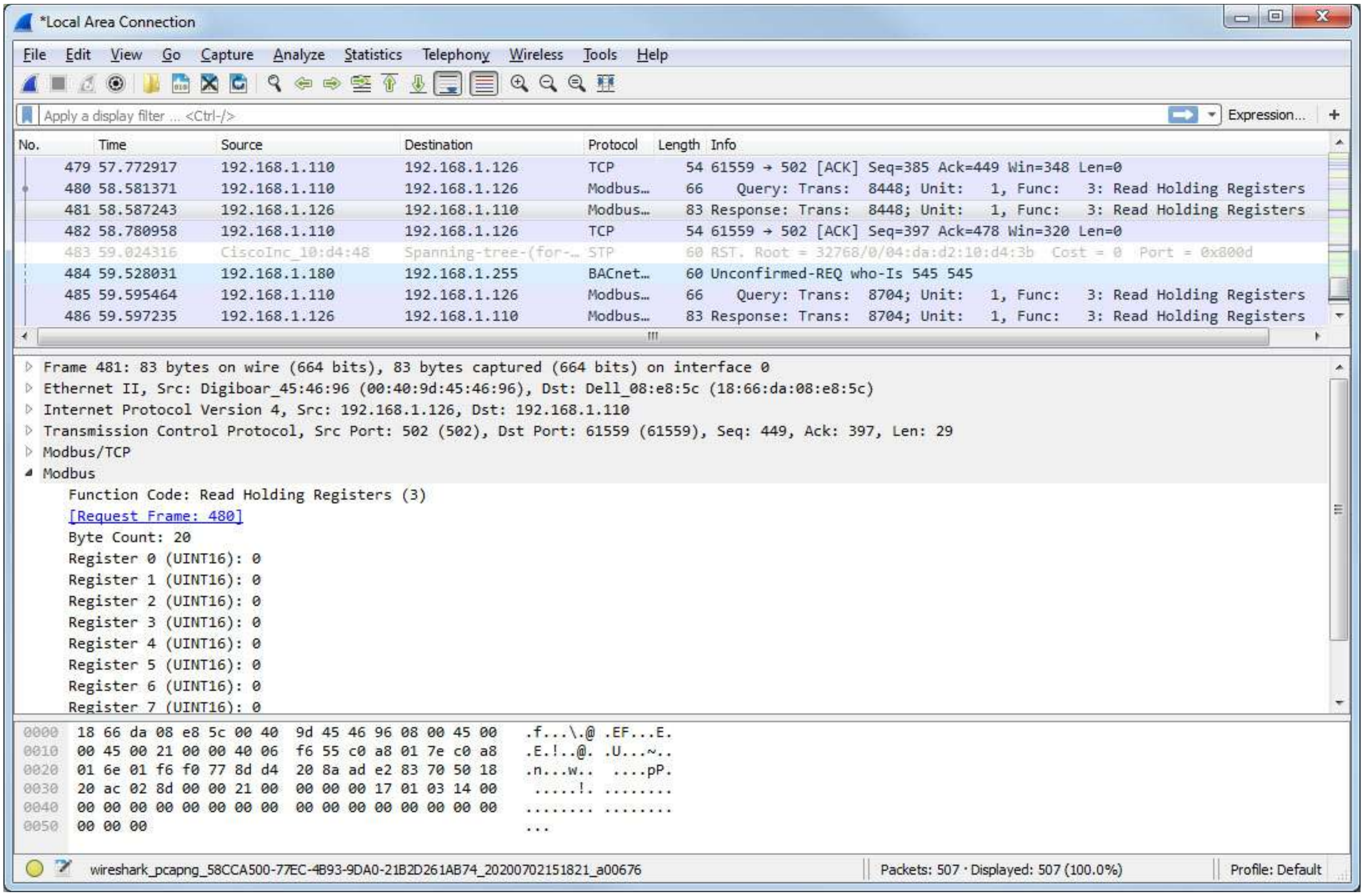
You will find message and error counters listed on the Error Counts page under TCP Data for Modbus client activity. Counts will be listed by device number for those devices found on the TCP Setup Devices page.

The Errors: Read Maps and Errors: Write Maps pages will tell you exactly which maps are getting errors when the Babel Buster is operating as Modbus TCP client (master).

The most frequent problem is "no response" or timeout. The most common cause of this problem for Modbus TCP is a network configuration problem, such as incorrect IP address or IP address that cannot be reached as configured. The problem sometimes lies outside the Babel Buster and may require consulting with the IT personnel responsible for the network if on a large network.

If you are getting exception errors, that is somewhat good news - it means that at least you are successfully communicating. An exception error most often means the master is asking the slave for a register that the slave does not have. If the Babel Buster is configured as Modbus master, this means the Read Map or Write Map is not configured correctly.

Wireshark can be a useful tool for analyzing Modbus TCP issues. Refer to Appendix F for more about Wireshark. The screen shot below illustrates a Modbus TCP response captured on the network.


*Local Area Connection
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Expression...
+

```
No.   Time       Source          Destination          Protocol  Length Info
  479 57.772917  192.168.1.110   192.168.1.126        TCP         54 61559 → 502 [ACK] Seq=385 Ack=449 Win=348 Len=0
  480 58.581371  192.168.1.110   192.168.1.126        Modbus…     66    Query: Trans:  8448; Unit:   1, Func:   3: Read Holding Registers
  481 58.587243  192.168.1.126   192.168.1.110        Modbus…     83 Response: Trans:  8448; Unit:   1, Func:   3: Read Holding Registers
  482 58.780958  192.168.1.110   192.168.1.126        TCP         54 61559 → 502 [ACK] Seq=397 Ack=478 Win=320 Len=0
  483 59.024316  CiscoInc_10:d4:48 Spanning-tree-(for-… STP       60 RST. Root = 32768/0/04:da:d2:10:d4:3b  Cost = 0  Port = 0x800d
  484 59.528031  192.168.1.180   192.168.1.255        BACnet…     60 Unconfirmed-REQ who-Is 545 545
  485 59.595464  192.168.1.110   192.168.1.126        Modbus…     66    Query: Trans:  8704; Unit:   1, Func:   3: Read Holding Registers
  486 59.597235  192.168.1.126   192.168.1.110        Modbus…     83 Response: Trans:  8704; Unit:   1, Func:   3: Read Holding Registers
```

```
▷ Frame 481: 83 bytes on wire (664 bits), 83 bytes captured (664 bits) on interface 0
▷ Ethernet II, Src: Digiboar_45:46:96 (00:40:9d:45:46:96), Dst: Dell_08:e8:5c (18:66:da:08:e8:5c)
▷ Internet Protocol Version 4, Src: 192.168.1.126, Dst: 192.168.1.110
▷ Transmission Control Protocol, Src Port: 502 (502), Dst Port: 61559 (61559), Seq: 449, Ack: 397, Len: 29
▷ Modbus/TCP
◢ Modbus
    Function Code: Read Holding Registers (3)
    [Request Frame: 480]
    Byte Count: 20
    Register 0 (UINT16): 0
    Register 1 (UINT16): 0
    Register 2 (UINT16): 0
    Register 3 (UINT16): 0
    Register 4 (UINT16): 0
    Register 5 (UINT16): 0
    Register 6 (UINT16): 0
    Register 7 (UINT16): 0
```

```
0000  18 66 da 08 e8 5c 00 40  9d 45 46 96 08 00 45 00   .f...\.@ .EF...E.
0010  00 45 00 21 00 00 40 06  f6 55 c0 a8 01 7e c0 a8   .E.!..@. .U...~..
0020  01 6e 01 f6 f0 77 8d d4  20 8a ad e2 83 70 50 18   .n...w..  ....pP.
0030  20 ac 02 8d 00 00 21 00  00 00 00 17 01 03 14 00    .....!. ........
0040  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0050  00 00 00                                           ...
```

wireshark_pcapng_58CCA500-77EC-4B93-9DA0-21B2D261AB74_20200702151821_a00676 | Packets: 507 · Displayed: 507 (100.0%) | Profile: Default

12.4 File Upload Errors

If you get a "File upload error: -1" message, click the browser's "back" button, then simply click the View button to view any file (does not matter which file), and then click browser's "back" button again to return to the File Manager page. This gets the browser and HTTP server back in sync, and this requirement generally happens only once following power-up.

If you get a different persistent file upload error, check the space available versus the size of the file you are trying to upload. Available file space is displayed on the File Manager page as "Free space". The free space indicated is approximate. If close to zero, try deleting some files first.

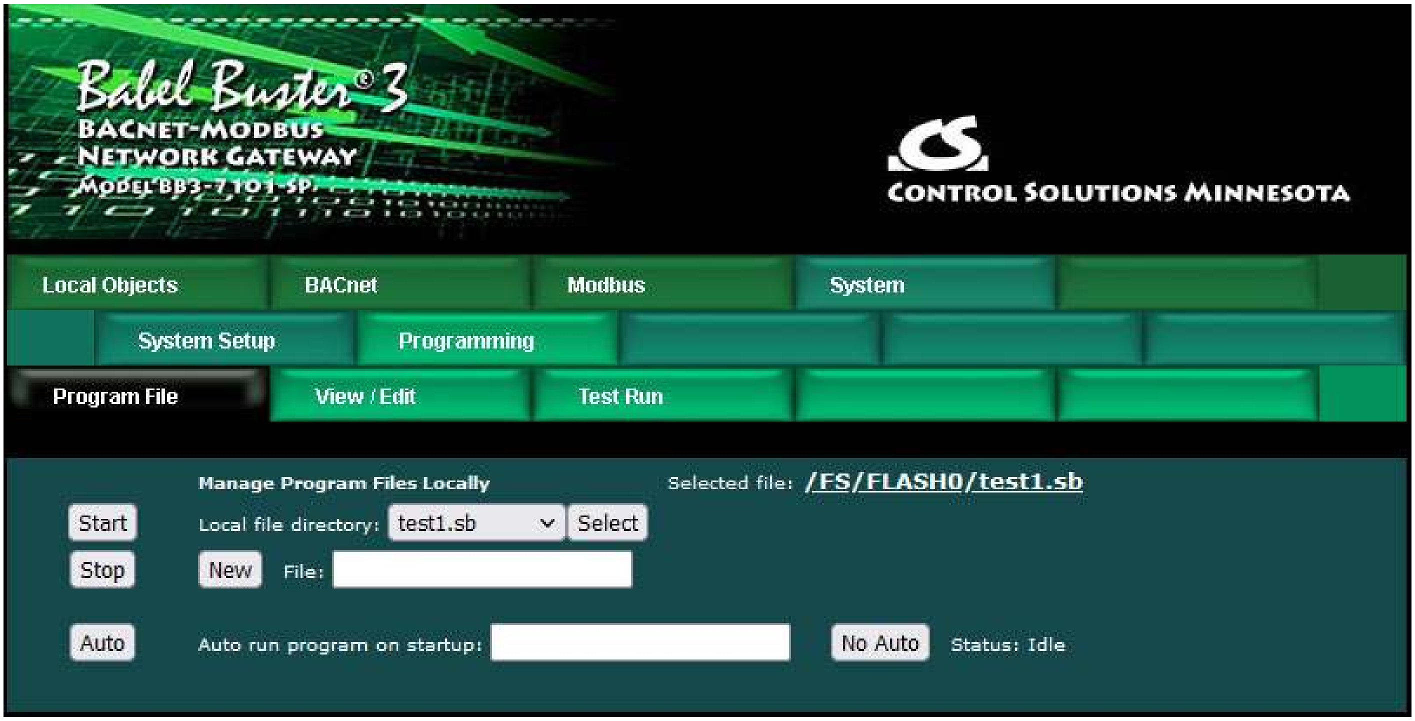
Babel Buster® 3
MODEL BB3-7101-SP
MODEL MX-71-SP
NETWORK GATEWAY

CONTROL SOLUTIONS MINNESOTA

13. Programming with Script Basic

13.1 Creating a Program

To create a new program, enter a new file name for your program ending in ".sb" in the File window next to the New button. The program should use only alphanumeric characters and be limited to 20 characters. As soon as you click the New button, the file will be created and automatically selected. If you are returning to edit a previously created program. select that file from the File list, and click Select.



Manage Program Files Locally

Selected file: /FS/FLASH0/test1.sb

Local file directory: test1.sb

File:

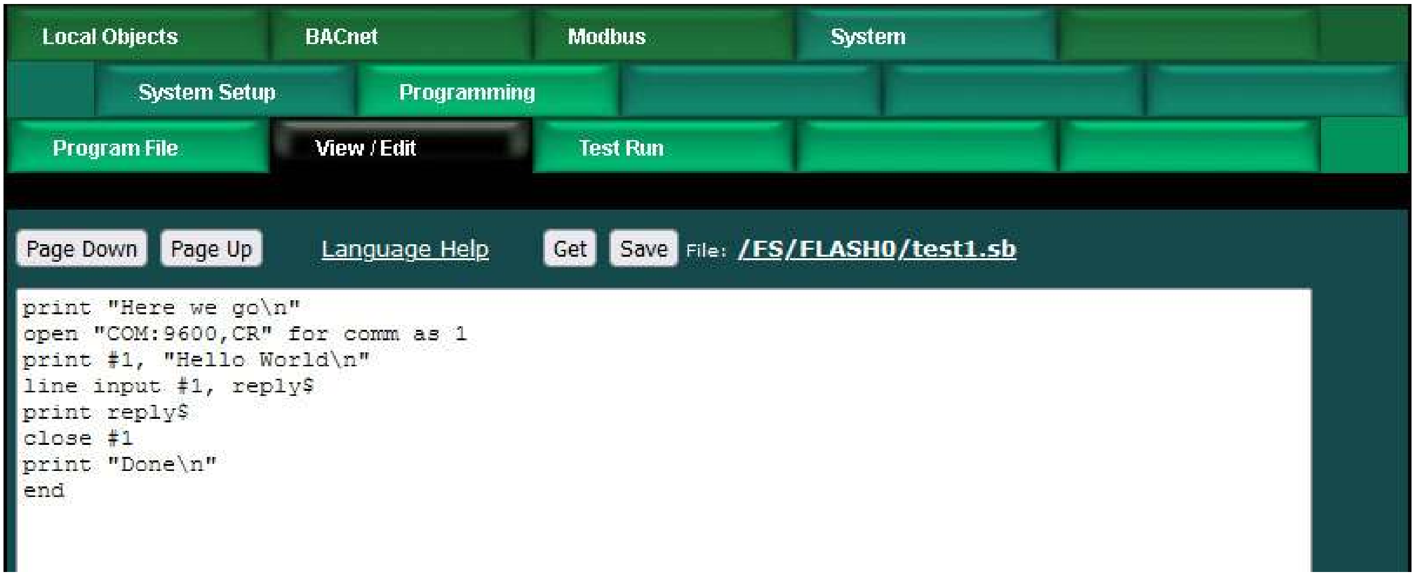
Auto run program on startup:

Status: Idle

There is a local, very simple text editor available via the web View/Edit page. To edit an existing file, start by clicking Get. After entering a new program, or editing an existing program, click Save.

It is recommended that you use an external text editor for large programs, and simply upload it on the Program File page.

The Language Help link provides a summary of Script Basic. For a complete reference, use the external Script Basic compiled help available for download at csimn.com.



```
print "Here we go\n"
open "COM:9600,CR" for comm as 1
print #1, "Hello World\n"
line input #1, reply$
print reply$
close #1
print "Done\n"
end
```

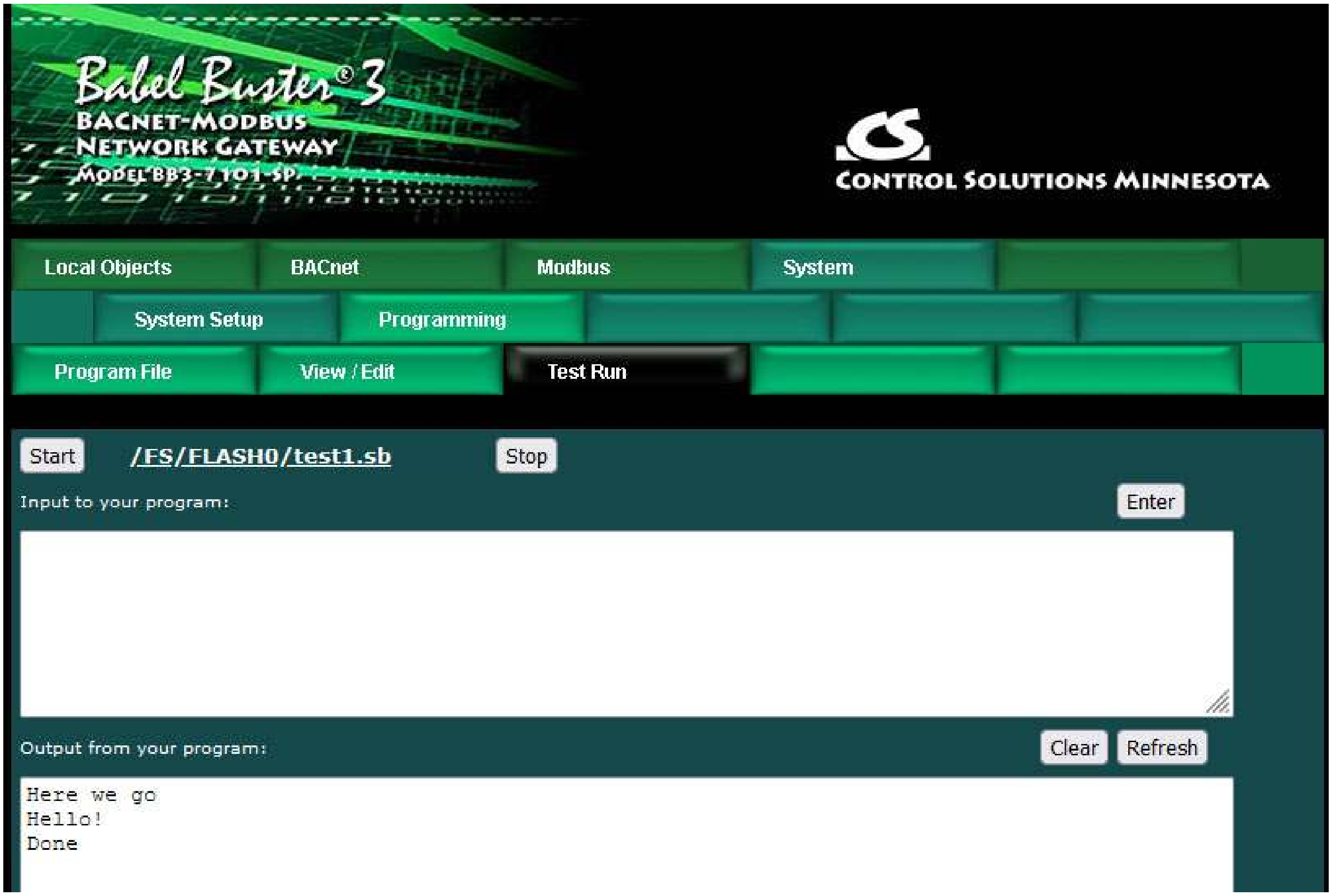
13.2 Testing the Program

The virtual terminal on the Test page can be used while testing programs. Any "print" statement (without a file number) will send its output to the Output window, and any "line input" statement will take input from the Input window. The print and line input statements will have no effect when running in the background (i.e. running as Auto run from startup).

The Test page does not auto refresh, therefore any output produced by a print statement will not appear until you click the Refresh button. Some initial output may appear immediately since the program began running faster than the initial page refresh that occurs after clicking Start, but you will need to click Refresh to see additional output.

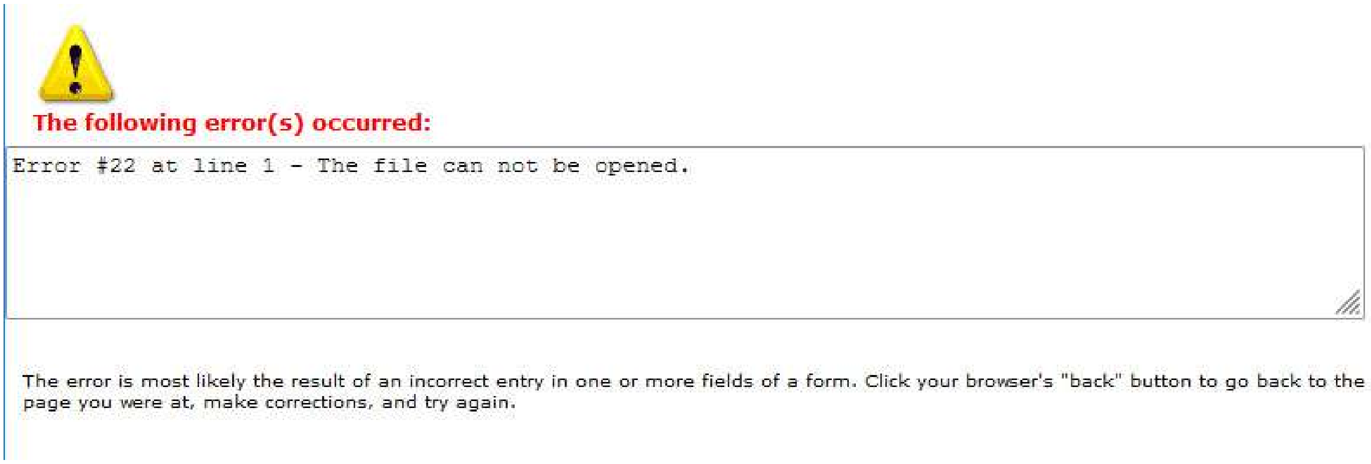
WARNING: If you are testing a program, you should select No Auto (see below), then restart the Babel Buster if an auto-run program had previously been selected. You will have two programs running at the same time if the auto run program is running and you are also running a program here. The results can be unpredictable if you are running two copies of the same or similar program at the same time.

An abbreviated screen shot of the Virtual Terminal is illustrated below. In this example, output from the above test program is illustrated. The string "Hello World" was sent to the serial port, and the terminal on the serial port replied with "Hello!" which was then echoed to the virtual terminal.



```
Here we go
Hello!
Done
```

If you get an error message that looks like the following, the usual cause is forgetting to select Disabled on the Modbus RTU Setup Local Device page and Modbus RTU has already claimed the comm port for its use, thus blocking Basic.



The following error(s) occurred:

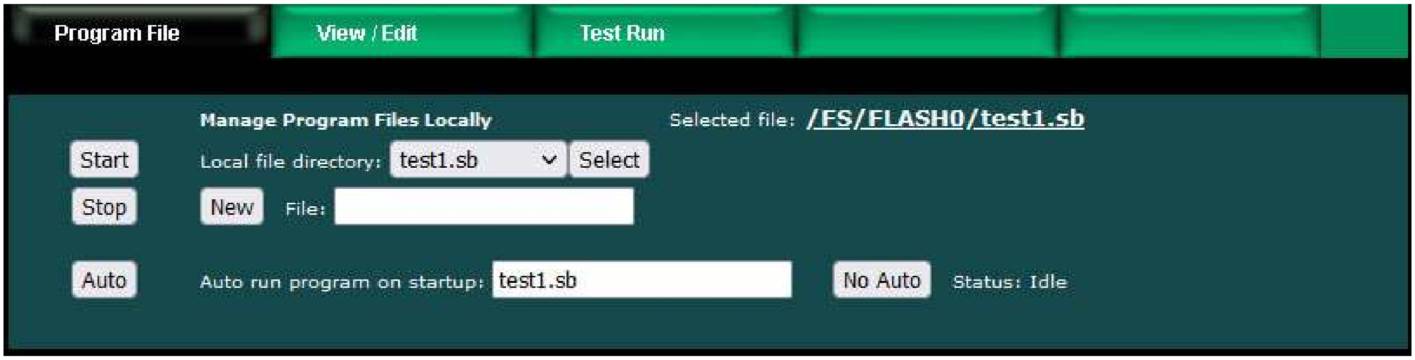
```
Error #22 at line 1 - The file can not be opened.
```

The error is most likely the result of an incorrect entry in one or more fields of a form. Click your browser's "back" button to go back to the page you were at, make corrections, and try again.

13.3 Setting the Program to Auto-Run on Startup

Your program will not be very useful if it does not automatically start up when the Babel Buster boots up. You cause that to happen by selecting your file from the list, and then clicking the Auto button. After selecting the auto-run program, go to the File Manager page and save your configuration. The auto-run program is saved in your configuration file.

From this point on, your program named in the "Auto run program on startup" window will be automatically started upon bootup. Of course, if the program has errors, it might not keep on running. Be sure to test your program ahead of time, and also consider use of the "On Error" feature of Basic. What exactly you might do upon error is entirely up to you, but one potentially useful option is to set an error number of your own making in a specific data object that can be read via Modbus or SNMP.



Manage Program Files Locally

Selected file: /FS/FLASH0/test1.sb

Local file directory: test1.sb

File:

Auto run program on startup: test1.sb

Status: Idle

To eliminate the auto run program, simply click the No Auto button. This will stop the program and clear the auto-run program name. To retain this change, be sure to go to the File Manager page and save your configuration file again.

13.4 Serial Port Functions

Opening Comm Port:

To open the communication port as file #2 for example, you would use:

```
open "COM:9600" for comm as 2
```

The line end character is otherwise known as line feed or "\n". The carriage return will be ignored, unless it is specified to be the line end instead. To open the comm port using the carriage return as line end instead of the Linux line end, you would use:

```
open "COM:9600,CR" for comm as 2
```

Many devices return both carriage return and line feed at the ends of lines. The sole line end character recognized as the end of line for the comm port will end the line while the other will be discarded and not returned in the string that gets placed in a variable for Basic.

Any of the baud rates valid for typical serial port usage may be specified in place of the 9600 baud used in the above examples. The baud rate may be anything from 1200 to 115,200.

```
open "COM:9600,EVEN" for comm as 2
open "COM:9600,ODD" for comm as 2
open "COM:9600,2STOP" for comm as 2
```

Port settings will default to no parity and one stop bit (8 data bits). You can change parity by using the notations illustrated above. You may still add ",CR" to the string when parity is included.

IMPORTANT: If you get an error trying to open the comm port, check to see that "Disabled" is selected as Baud Rate on the Modbus RTU Setup Local Device page. If you do not select Disabled there, then the port is already in use by Modbus and you cannot open it in Basic.

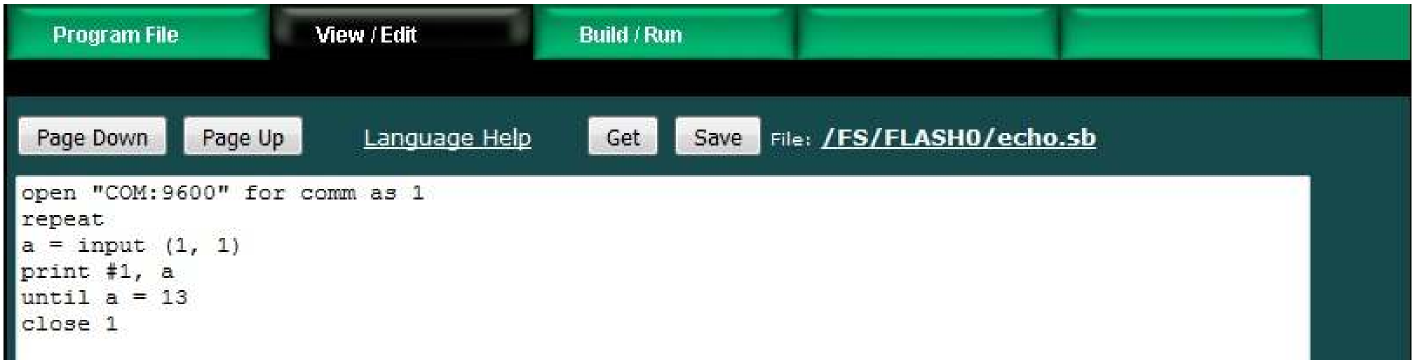
Input from Comm Port:

The "line input" in Basic is used to receive entire lines from the comm port. For example,

```
line input #2, myLine
```

will accept a line up to the line end character into the variable myLine.

The "line input" in Basic expects to receive a full line terminated by a line end character. If you want to capture one character at a time and not be concerned with whole lines or line end characters, the following exmple illustrates capturing one character at a time and outputing one character at a time to the virtual terminal screen, until that one character is a carriage return - then the program closes the port and termiantes in this simple example.



```
open "COM:9600" for comm as 1
repeat
a = input (1, 1)
print #1, a
until a = 13
close 1
```

Output to Comm Port:

Any variation on the file print referencing the comm port file number will send output to the comm port. For example,

```
print #2, "You typed ",myLine,"\n"
```

will echo the line received above right back to the comm port along with the comment 'You typed '.

Comm Port Timeout:

```
timeout (n)
```

Sets timeout in seconds that the "line input #n" request for input from the communiation port will wait before returning an empty string if nothing was received on the communication port. Otherwise, the line input will wait indefinitely for a line that ends in a line end character.

13.5 Special Functions

13.5.1 Register (Object) Access

You may read any of the local BACnet objects using the getreg function, and write them using the setreg command. Note that getreg is a function while setreg is not; it is a command and therefore requires no parenthesis. The object type and instance are encoded into a single "register" number for use by the Basic functions.

Usage is:

```
    data = getreg (x)
    setreg x, data
```

where 'x' is the BACnet object number encoded as indicated below, and 'data' is the Present Value of the BACnet object. Consider the following example:

```
    MyData = getreg (22)
    MyData = MyData * 2.5
    setreg 20024, MyData
```

The above example will get the Present Value of AI 22 and place it in the variable MyData, then multiply it by 2.5, then place the resulting value in the Present Value of AV 24. Variables may be used in place of the constants used in the above example.

BACnet objects are accessed via the register interface. Register numbers are BACnet object type multiplied times 10,000 plus object number starting at #1. Register numbers corresponding to BACnet objects are as follows:

Object Type	Object Number	Register Number
Analog Input	AI 1	1
Analog Output	AO 1	10001
Analog Value	AV 1	20001
Binary Input	BI 1	30001
Binary Output	BO 1	40001
Binary Value	BV 1	50001
Multi-State Input	MI 1	130001
Multi-State Output	MO 1	140001
Multi-State Value	MV 1	190001

13.5.1.1 LED Control via Phantom Register Access

A set of "phantom" registers exists for the purpose of allowing your Script Basic program to turn on and off the Request and Reply LEDs at will. There are two sets of registers. One group will cause an automatically timed "flash" of the respective LED while the other group will provide static on/off control of the LEDs. Static on/off means once turned on, it will remain on until you explicitly turn it off again. Flash and static cannot both be used at the same time. The LEDs are in the BB3-7101 are bi-color, meaning they can each be turned on to one of two colors, but of course not at the same time.

The phantom registers are only accessible from Script Basic, but use the same setreg command that is used to place values into the local BACnet objects. To "flash" an LED, write a non-zero value to the respective register. For static control, write a non-zero value to the static register to turn the LED on, and write zero to the same register to turn the LED off.

Register No.	LED function
640001	Request LED Flashes Green (n/a on MX)
640002	Request LED Flashes Yellow (MX Yellow)
640003	Reply LED Flashes Green (MX Green)
640004	Reply LED Flashes Red (MX Red)
640005	Request LED Green Static On/Off (n/a on MX)
640006	Request LED Yellow Static On/Off (MX Yellow On/Off)
640007	Reply LED Green Static On/Off (MX Green On/Off)
640008	Reply LED Red Static On/Off (MX Red On/Off)

For example, to flash the Reply LED Red, you would use:

```
setreg 640004,1
```

The LED registers can be written but not read. Using getreg on the LED registers will not return the LED state - your program needs to remember what it did.

13.5.1.2 BACnet Object Command Priority Access via Phantom Register Access

The Local Command Priority found on the BACnet Device Settings page will be used any time Script Basic writes to a commandable object. However, if you wish to override that default setting, you may do so by using the setreg function and the phantom register assigned for this purpose. Command priority must be 1 through 16, excluding 6.

Register No.	Priority function
640099	Set Command Priority 1..16

For example, to set the local command priority to 3, you would use the following line:

```
setreg 640099,3
```

Then for example, to set Analog Output 1 priority 3 to a level of 100, you would use the following line after the above line:

```
setreg 10001,100
```

There is also a means of relinquishing the command priority that had been set above. To relinquish Analog Output 1 priority 3, assuming the two lines above had been previously executed, you would use this line:

```
setreg -10001,0
```

Using the setreg function with the encoded register given as negative will have the effect of relinquishing the command priority currently set (or default command priority from BACnet Device Settings page if none had been set in the Basic program). Using a negative register number for a non-commandable object will have no effect.

13.5.1.3 Program Watchdog Timer via Phantom Register Access

A watchdog timer is available to Script Basic via the following phantom register.

Register No.	Watchdog function
641000	Set/Reset Watchdog Timeout in Seconds (zero to disable)

The following example will set the watchdog timer to half a second.

```
setreg 641000,0.5
```

If your program does not call setreg to set/reset the watchdog timer again withing this amount of time, the gateway will be restarted (as if power cycled).

13.5.1.4 Register Access Status

You may optionally check to see if your most recent getreg or setreg call resulted in finding a valid register number (or encoded object instance and number). Simply use the getreg function with a register number of zero to check the previous getreg or setreg call.

```
n = getreg (0)
```

The value of 'n' will be 1 if the previous operation was successul, or 0 if it failed.

13.5.2 Error Information Retrieval

```
n = geterr (x,y)
```

Returns n=0 for no error, otherwise returns error code or status for item specified by 'x', item number 'y' (1..N). Error codes returned are those displayed on the respective web pages as error codes or device status.

Item 'x' may be:
1 = BACnet client device (count of errors for device)
2 = BACnet client read map (class * 10,000 + code)
3 = BACnet client write map (class * 10,000 + code)
4 = Modbus TCP device
5 = Modbus TCP read map (error * 100 + exception)
6 = Modbus TCP write map (error * 100 + exception)
7 = Modbus RTU device
8 = Modbus RTU read map (error * 100 + exception)
9 = Modbus RTU write map (error * 100 + exception)

BACnet error class and code are as defined by BACnet protocol, and these are listed in the Quick Help section of the BACnet Diagnostics page in the gateway.

Modbus TCP device code returned will be the connection status as listed in the Quick Help section of the Modbus TCP Devices page in the gateway

Modbus RTU device code will simply be an indication of whether any errors have been tabulated for that slave address, with the returned code indicating as follows:
1 = 'No response' errors have been tabulated
2 = CRC errors have been tabulated
3 = Exception errors have been tabulated for this RTU device.

Modbus 'error' for read/write maps is as follows, with exception being as defined by Modbus protocol when 'error' is Exception. Exception codes are as defined for Modbus protocol.
1 = TCP transaction ID mismatch
2 = Exception (see exception codes)
3 = Function code mismatch
4 = Insufficient data received
5 = No response (time-out)
6 = CRC error
7 = BACnet client timeout
8 = BACnet error code received
9 = Host time-out

13.5.3 IP Address Retrieval

```
    ip$ = getipaddr (x,y)
```

Returns IP address as a string for item type 'x', item number 'y' (1..N), formatted as character string.

Item 'x' may be:
0 = Our own IP address
1 = BACnet client device
2 = Modbus TCP device

13.5.4 BACnet Object Status and Reliability Codes

```
    n = getobjstatus (x, y)
```

Check status for BACnet object 'x' (encoded as for getreg/setreg), returning 1 if status is true/active, or 0 if false/inactive.

The status types to check as 'y' may be:
1 = Out Of Service
4 = In Fault
16 = New data since last call of function for this object

The status codes 'y' are used as a bit mask. Other values will return indeterminate results. As an example, getobjstatus(1,16) will return a value of 1 if AI 1 has received new data since the last function call.

```
    n = getobjrel (x)
```

Returns reliability code for BACnet object 'x'. A value of 0 means there are no faults. The object number 'x' is encoded as for the 'setreg' command. The meaning of the reliability codes in the range of 1-63 is defined by the BACnet protocol standard. The meaning of codes 64-100 are displayed as applicable on the various rule pages in the web UI for this device. The meaning of reliability codes 101-199 are defined by the user's Script Basic program if applicable.

```
    setobjrel x, y
```

Sets the reliability code for BACnet object 'x'. A value of 0 means there are no faults. The object number 'x' is encoded as for the 'setreg' command. The meaning of the reliability codes in the range of 1-63 is defined by the BACnet protocol standard. The meaning of codes 64-100 are displayed as applicable on the various rule pages in the web UI for this device. The meaning of reliability codes 101-199 are defined by the user's Script Basic program if applicable.

The value 'y' must be in the range of 101-199 when your program uses this command, or it will be ignored. If you set the reliability code to any non-zero value, the object will be considered to be in fault by any BMS system. If you use this feature to indicate the fact that your program has detected a problem, you must also be sure to set the value back to zero when your program detects that the problem no longer exists.

Note that 'setobjrel' is a command, not a function, and therefore you do not enclose the parameters in parenthesis.

13.5.5 Free Memory

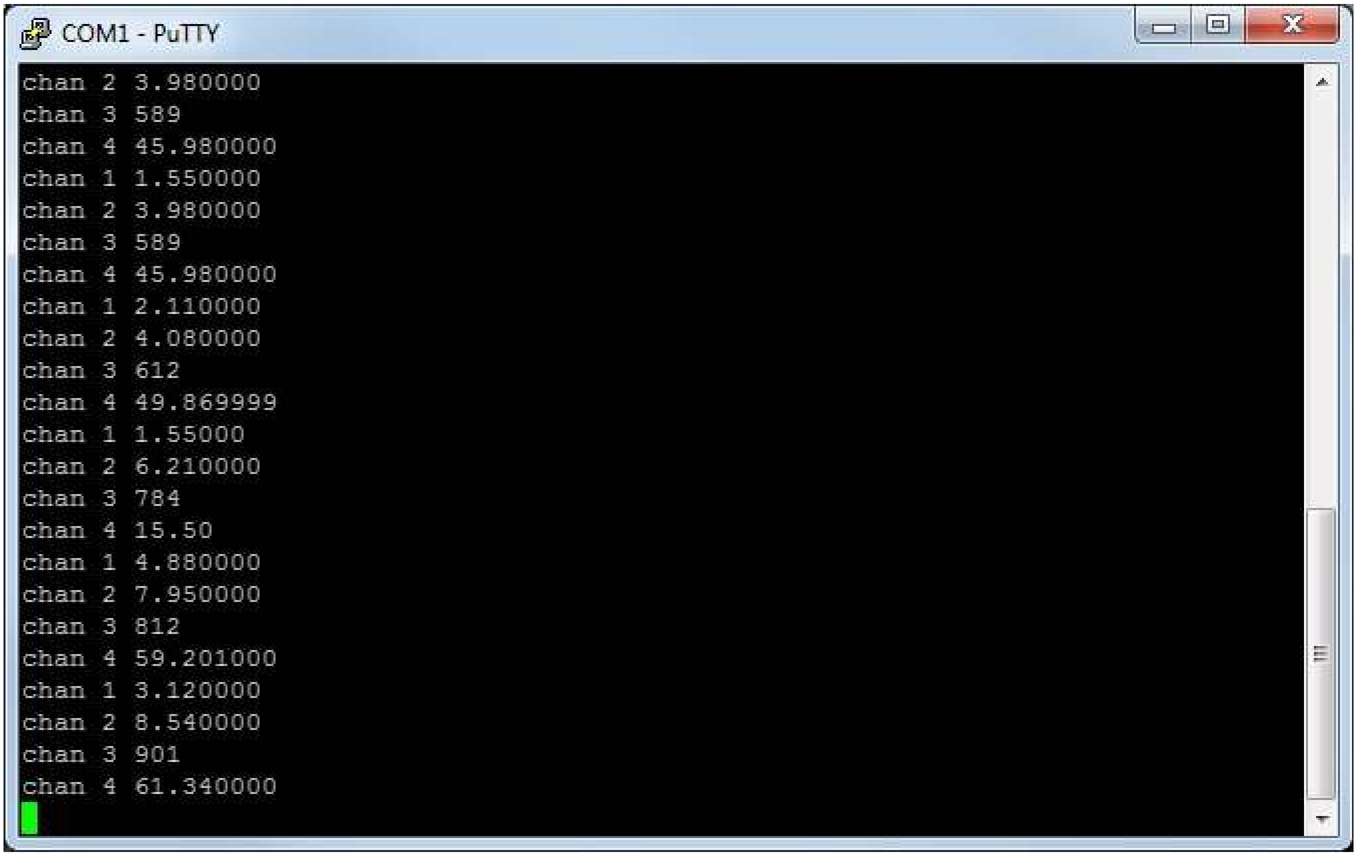
You can find out within your program how many bytes of free memory remain. In the line below, 'n' will be the number of bytes. It will typically start out over 1,500,000. When it drops to hear zero, the system will stop functioning because memory allocation from the heap has run up against stack space.

```
    n = freemem()
```

This function should only be used for diagnostics while developing your program, and not be used in a production program that is expected to run indefinitely.

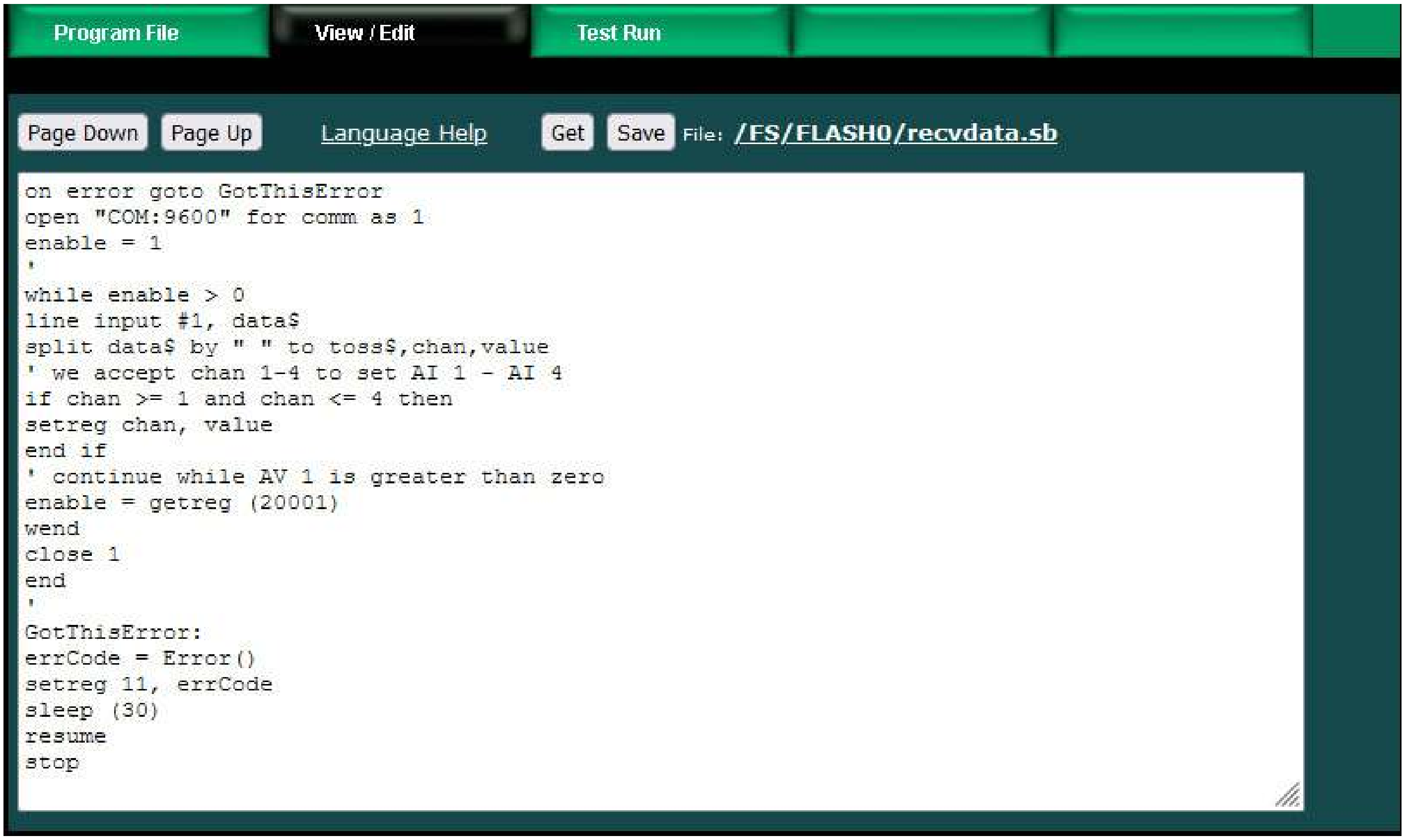
13.6 Example: Data Logger Capture

Our first example will capture data from a data logger type device that periodically sends strings of data. In this case, it is simply a channel number and that channel's data value. This device is a 4-channel device, and its output is illustrated here by simply connecting it to a PC (via RS485 to RS232 adapter) and running PuTTY to see its output.



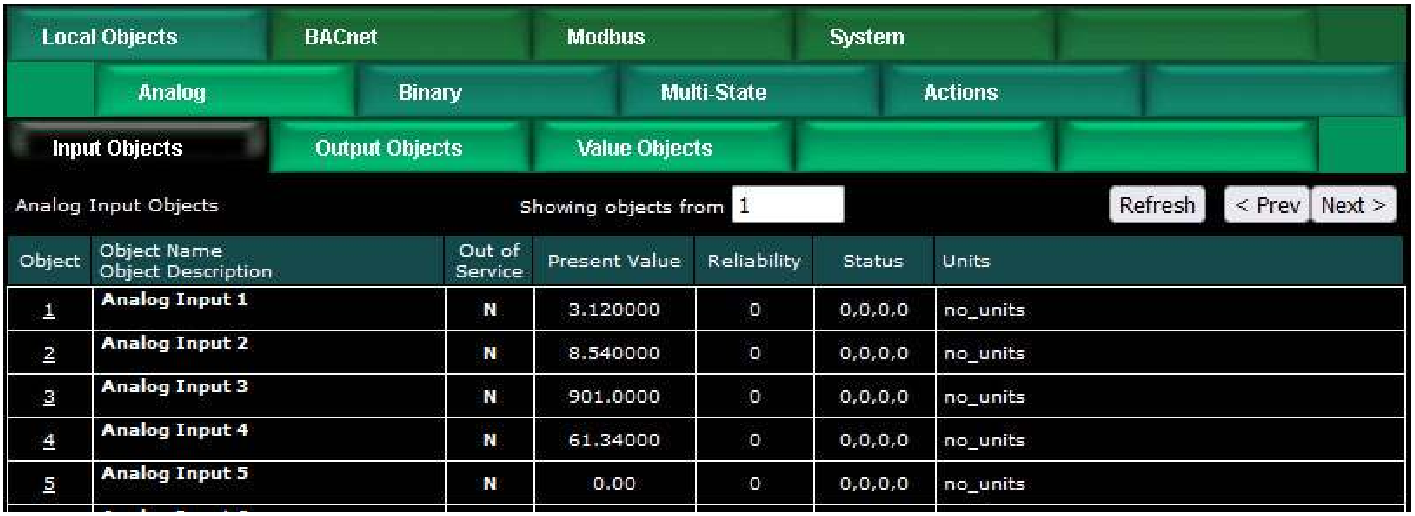
```
chan 2 3.980000
chan 3 589
chan 4 45.980000
chan 1 1.550000
chan 2 3.980000
chan 3 589
chan 4 45.980000
chan 1 2.110000
chan 2 4.080000
chan 3 612
chan 4 49.869999
chan 1 1.55000
chan 2 6.210000
chan 3 784
chan 4 15.50
chan 1 4.880000
chan 2 7.950000
chan 3 812
chan 4 59.201000
chan 1 3.120000
chan 2 8.540000
chan 3 901
chan 4 61.340000
```

The program to capture data from this device and store the results in local objects is illustrated below. The key line to notice here is the "split" where the data line is effectively parsed. The "toss$" is going to end up holding the string "chan" which we don't care about. The numbers representing channel number and that channel's data value will end up in the variables "chan" and "value". Then, based on channel number, we save the data value to that object.



```
on error goto GotThisError
open "COM:9600" for comm as 1
enable = 1
'
while enable > 0
line input #1, data$
split data$ by " " to toss$,chan,value
' we accept chan 1-4 to set AI 1 - AI 4
if chan >= 1 and chan <= 4 then
setreg chan, value
end if
' continue while AV 1 is greater than zero
enable = getreg (20001)
wend
close 1
end
'
GotThisError:
errCode = Error()
setreg 11, errCode
sleep (30)
resume
stop
```

The local object values are going to continue changing as new data is received. The screen shot below illustrates the most recently received data in this example.



Object	Object Name Object Description	Out of Service	Present Value	Reliability	Status	Units
1	Analog Input 1	N	3.120000	0	0,0,0,0	no_units
2	Analog Input 2	N	8.540000	0	0,0,0,0	no_units
3	Analog Input 3	N	901.0000	0	0,0,0,0	no_units
4	Analog Input 4	N	61.34000	0	0,0,0,0	no_units
5	Analog Input 5	N	0.00	0	0,0,0,0	no_units

The above example is always just receiving data that is sent automatically. If you needed to query a device in order to receive data, you can also do that from Script Basic. A query program is illustrated in the next section.

13.6.1 Program Code - Capture

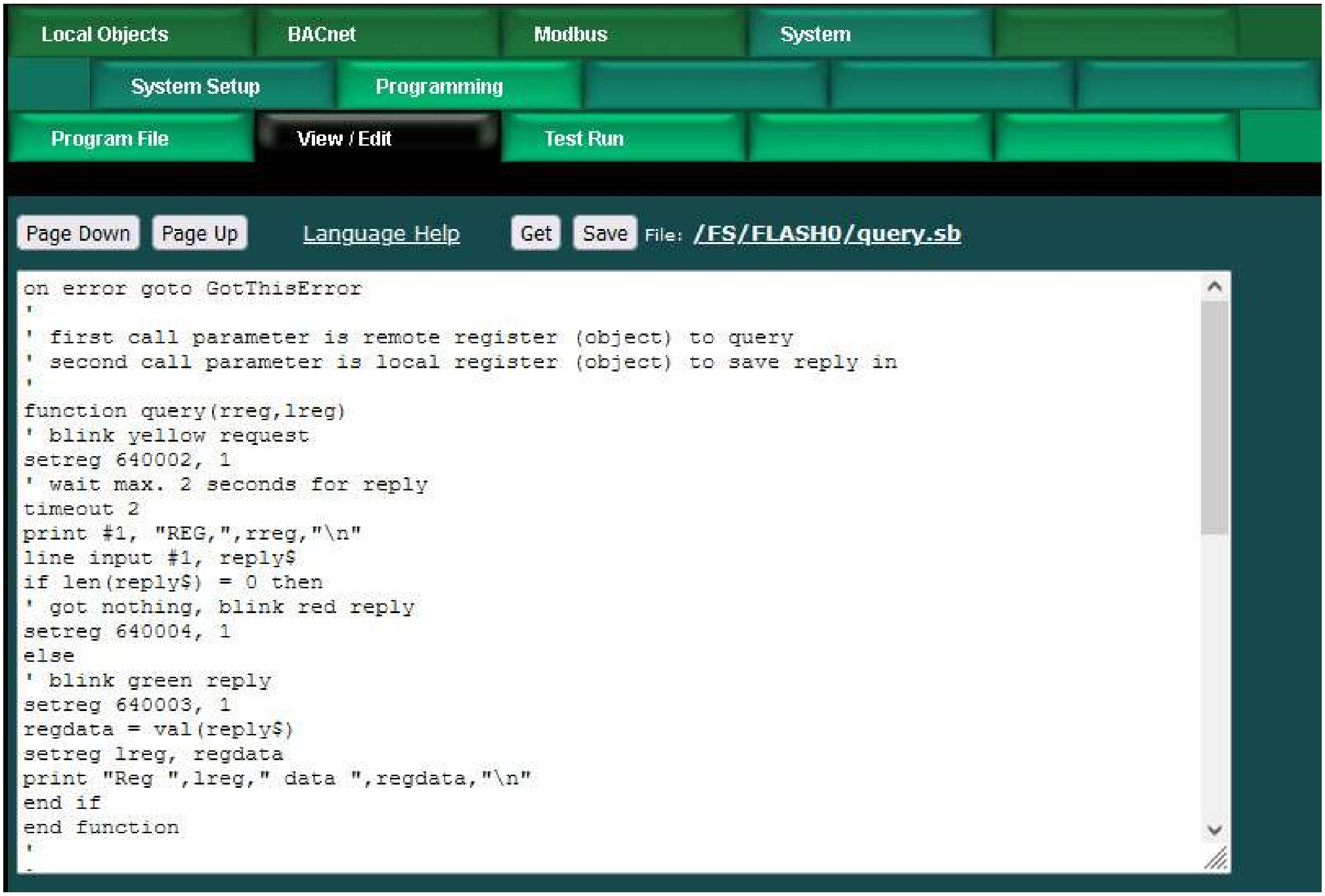
Here is the source code for the example program in this section.

```
on error goto GotThisError
open "COM:9600" for comm as 1
```

```
enable = 1
'
while enable > 0
line input #1, data$
split data$ by " " to toss$,chan,value
' we accept chan 1-4 to set AI 1 - AI 4
if chan >= 1 and chan <= 4 then
setreg chan, value
end if
' continue while AV 1 is greater than zero
enable = getreg (20001)
wend
close 1
end
'
GotThisError:
errCode = Error()
setreg 11, errCode
sleep (30)
resume
stop
```

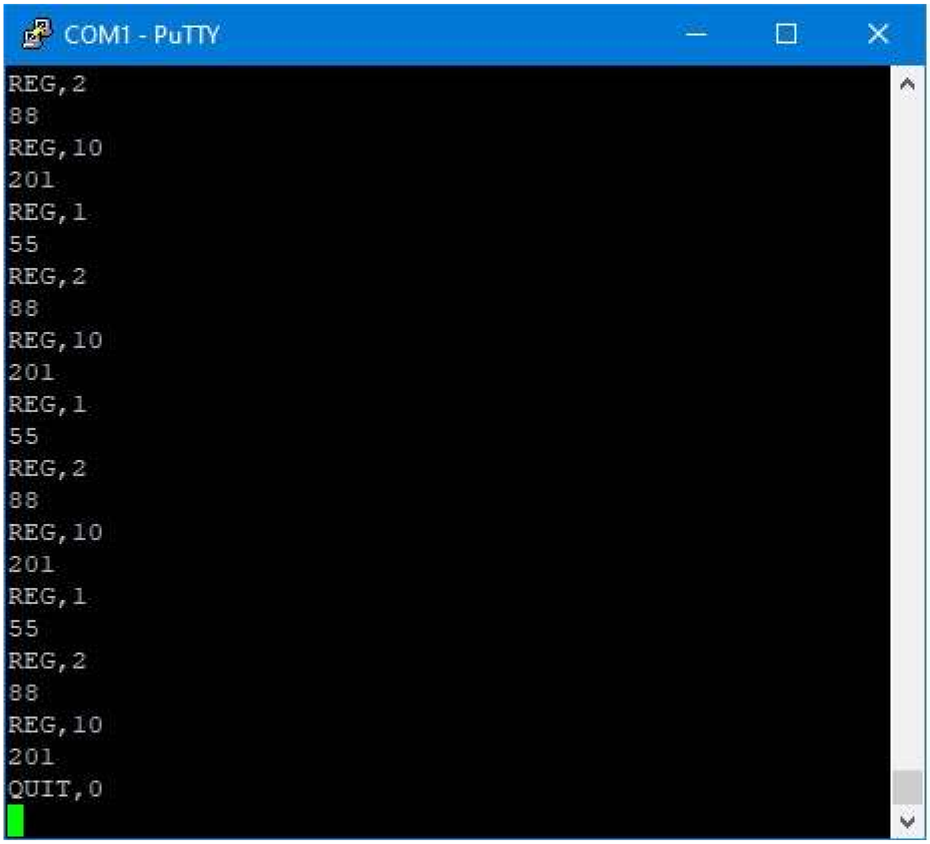
13.7 Example: Querying Serial Device

The following example program illustrates the need to query a device in order to receive data from it. Our protocol here is very simple. We send "REG,X" where X is an object number (encoded register number) to request the data value for object X in the remote device. We will receive simply a number that we then put in one of our local objects.



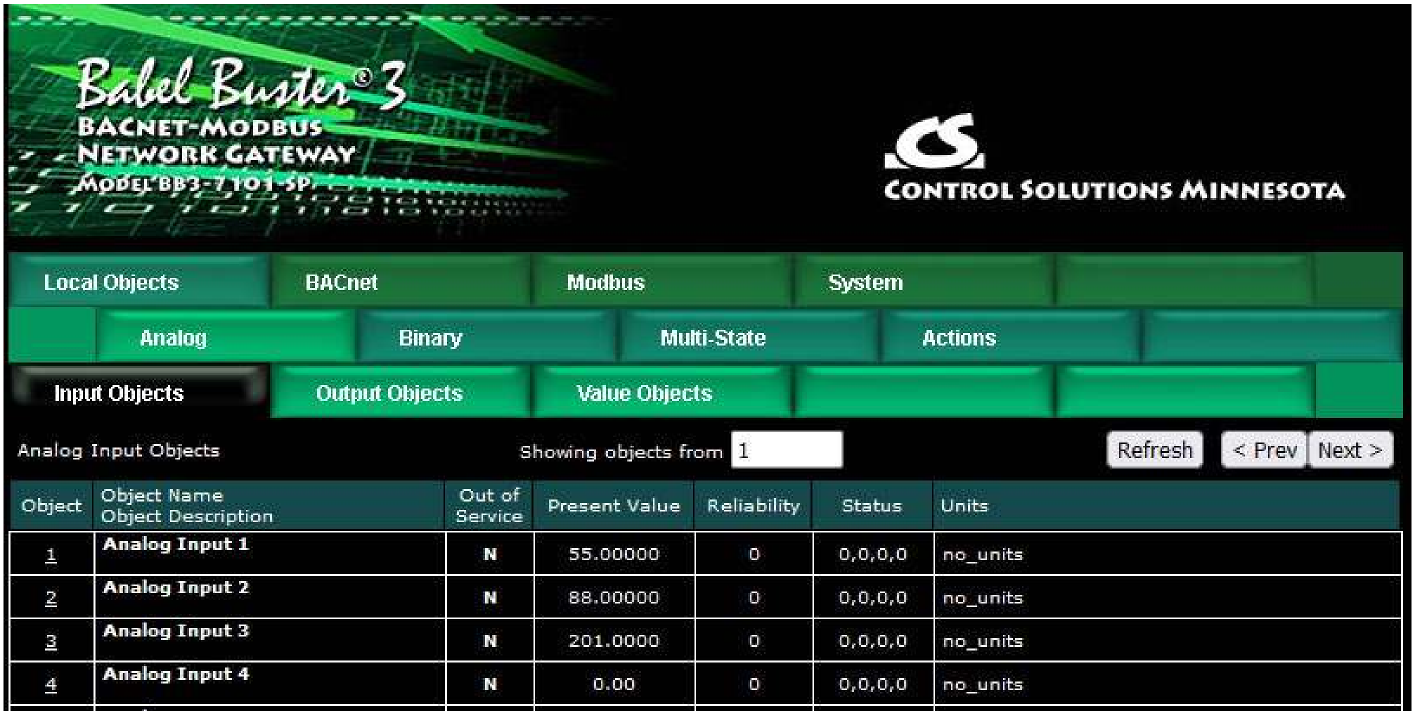
```
on error goto GotThisError
'
' first call parameter is remote register (object) to query
' second call parameter is local register (object) to save reply in
'
function query(rreg,lreg)
' blink yellow request
setreg 640002, 1
' wait max. 2 seconds for reply
timeout 2
print #1, "REG,",rreg,"\n"
line input #1, reply$
if len(reply$) = 0 then
' got nothing, blink red reply
setreg 640004, 1
else
' blink green reply
setreg 640003, 1
regdata = val(reply$)
setreg lreg, regdata
print "Reg ",lreg," data ",regdata,"\n"
end if
end function
'
```

To illustrate this simple query program, a second BB3-7101-SP was programmed to be the repsonding side of this interaction. The communication between the two was monitored using PuTTY.



```
REG,2
88
REG,10
201
REG,1
55
REG,2
88
REG,10
201
REG,1
55
REG,2
88
REG,10
201
REG,1
55
REG,2
88
REG,10
201
QUIT,0
```

The local object values following the last set of queries illustrated above is shown in the following screen shot.



Analog Input Objects Showing objects from 1

Object	Object Name Object Description	Out of Service	Present Value	Reliability	Status	Units
1	Analog Input 1	N	55.00000	0	0,0,0,0	no_units
2	Analog Input 2	N	88.00000	0	0,0,0,0	no_units
3	Analog Input 3	N	201.0000	0	0,0,0,0	no_units
4	Analog Input 4	N	0.00	0	0,0,0,0	no_units

13.7.1 Program Code - Query

Here is the source code for the example query program in this section. You will see in our examples that we sometimes use variable names like reply$ for string variables.

This is an old Basic convention that is not required by Script Basic. Any variable can contain any data type in Script Basic. You do not need to use "$" to designate the variable as a string variable in this version of Basic.

```
on error goto GotThisError
'
' first call parameter is remote register (object) to query
' second call parameter is local register (object) to save reply in
'
function query(rreg,lreg)
' blink yellow request
setreg 640002, 1
' wait max. 2 seconds for reply
timeout 2
print #1, "REG,",rreg,"\n"
line input #1, reply$
if len(reply$) = 0 then
' got nothing, blink red reply
setreg 640004, 1
else
' blink green reply
setreg 640003, 1
regdata = val(reply$)
setreg lreg, regdata
print "Reg ",lreg," data ",regdata,"\n"
end if
end function
'
' query program
'
open "COM:9600,CR" for comm as 1
print "Here we go\n"
enable = 1
do while enable > 0
' expand this list to query whatever you wish
' this example sets AI 1 - AI 3
query (1, 1)
query (2, 2)
query (3, 3)
sleep (1)
' continue until AV 1 is zero
enable = getreg (20001)
loop
print #1, "QUIT,0","\n"
close #1
print "Done\n"
end
'
GotThisError:
errCode = Error()
print "Error: ",errcode,"\n"
setreg 20002, 1
resume
```

```
stop
```

13.7.2 Program Code - Reply

Here is the source code for the reply test program in this section.

```
on error goto GotThisError
'
print "Here we go\n"
open "COM:9600" for comm as 1
enable = 1
do while enable > 0
line input #1, query$
split query$ by "," to command$,regnum
if command$ like "REG" then
regdata = getreg (regnum)
print #1, regdata,"\n"
end if
if command$ like "QUIT" then enable = 0
loop
close #1
print "Done\n"
end
'
GotThisError:
errCode = Error()
print "Error: ",errcode,"\n"
setreg 20002, 1
resume
stop
```

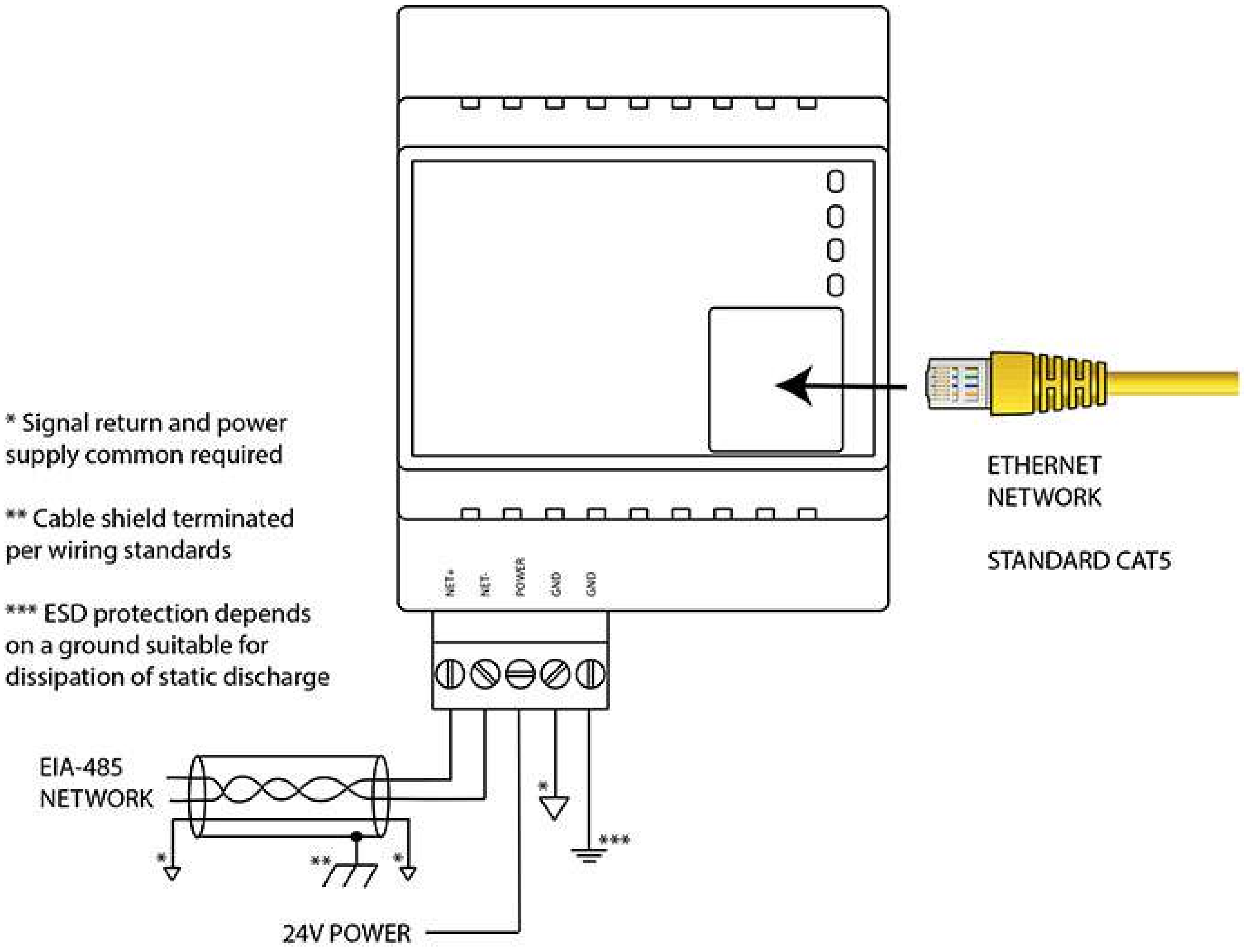
Babel Buster® 3
MODEL BB3-7101-SP
MODEL MX-71-SP
NETWORK GATEWAY

CONTROL SOLUTIONS MINNESOTA

Appendix A Hardware Details

A.1 Wiring

Wiring for the Babel Buster BB3-7101 is illustrated below.

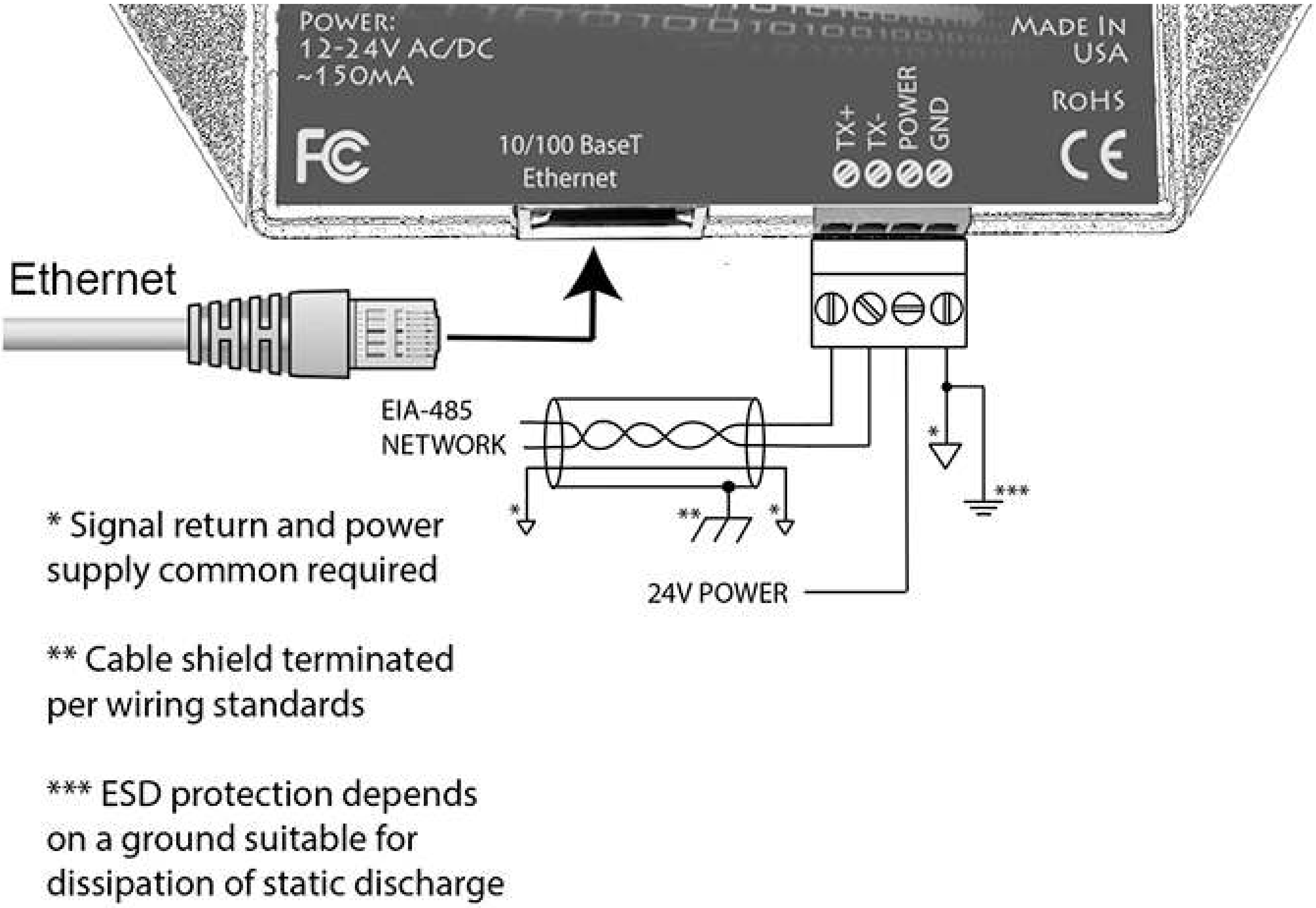


* Signal return and power supply common required

** Cable shield terminated per wiring standards

*** ESD protection depends on a ground suitable for dissipation of static discharge

Wiring for the MX-71 is illustrated below.



* Signal return and power supply common required

** Cable shield terminated per wiring standards

*** ESD protection depends on a ground suitable for dissipation of static discharge

Wire the gateway as illustrated. Follow all conventional standards for wiring of EIA-485 networks when connecting the Modbus RTU EIA-485 (RS485) network. This includes use and termination of shield, termination of the network, and grounding.

IMPORTANT: Although EIA-485 (RS485) is thought of as a 2-wire network, you MUST include a third conductor connected to GND or common at each device so that all devices are operating at close to the same ground potential. Proper grounding of equipment should ensure proper operation without the third conductor; however, proper grounding often cannot be relied upon. If large common mode voltages are present, you may even need to insert optically isolated repeaters between EIA-485 devices.

Use standard CAT5 cables for Ethernet connections. Use control wire as applicable for local electrical codes for connecting the 24V (AC or DC) power supply.

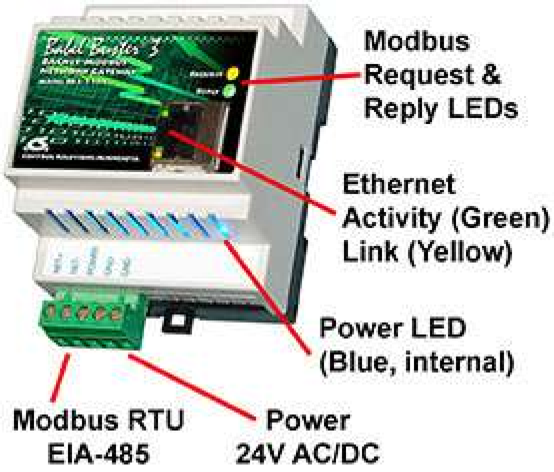
Note that in addition to connecting power supply common to a GND terminal, you must also connect a GND terminal to earth ground in order to ensure proper ESD protection.

BB3-7101-232: The standard BB3-7101 Modbus RTU port uses RS-485. The RS-232 version replaces the RS-485 transceiver with an RS-232 transceiver. The NET+/NET- terminals are replaced by TXD and RXD on the -232 version. TXD is data out from the BB3-7101-232, and RXD is data in to the gateway. Hardware handshake is not supported.

A.2 Front Panel LED Indicators

A.2.1 BB3-7101 LED Indicators

Power-up LED behavior: On power up, the Reply LED will remain on solid red for about 20 seconds, then the Request and Reply LEDs will do a "lamp test" where Request is yellow and Reply is Red simultaneously for about 1 second, and then both Request and Reply turn green simultaneously for about 1 second. The LEDs will then begin to operate according to their normal functionality.


Modbus Request & Reply LEDs
Ethernet Activity (Green) Link (Yellow)
Power LED (Blue, internal)
Modbus RTU EIA-485
Power 24V AC/DC

Babel Buster BB3-7101 Request and Reply LEDs reflect Modbus RTU traffic, and the Ethernet activity LED will indicate network traffic in general. If Modbus RTU is not being used at all, then the Request and Reply LEDs will indicate TCP traffic. If Modbus RTU is in use, then the Request and Reply LEDs will indicate Modbus RTU traffic while the Ethernet LEDs will be the only indication of TCP traffic.

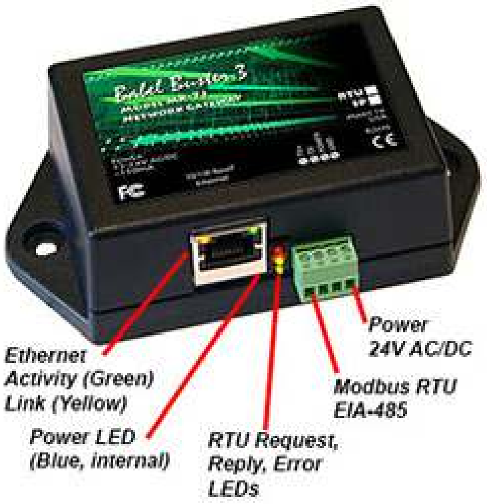
Babel Buster BB3-7101 LEDs indicate as follows (LEDs are bi-color):

REQUEST	Flashes yellow each time a request is sent when operating as Modbus Master, or each time a request is received when operating as Modbus Slave.
REPLY	Operating as Modbus Master, flashes green each time a good response is received, or red when an error code is received, the request times out, or there is a flaw in the response such as CRC error. Operating as Modbus Slave, flashes green each time a good response is sent, or red if an exception code is sent (meaning the received request resulted in an error).
Ethernet Activity	Green LED is on solid during portions of the boot-up process, and then flashes briefly when Ethernet network traffic is detected.

Ethernet Link	Yellow LED indicates an Ethernet link is present. This indicator will light if a link is present regardless of processor or network activity. If not lit, check network wiring.
Status	Blue LED (internal) on any time power is present and internal power supply is functioning.

A.2.2 MX-71 LED Indicators

Power-up LED behavior: On power up, the Request, Reply and Error LEDs will remain off for about 20 seconds, then all three LEDs will do a "lamp test" where they all turn on simultaneously for about 1 second. The LEDs will then begin to operate according to their normal functionality.



Babel Buster MX-71 Request, Reply and Error LEDs reflect Modbus RTU traffic, and the Ethernet activity LED will indicate network traffic in general.

Babel Buster MX-71 LEDs indicate as follows (LEDs are each a single color):

Error (red)	Operating as Modbus Master, flashes red when an error code is received, the request times out, or there is a flaw in the response such as CRC error. Operating as Modbus Slave, flashes red if an exception code is sent (meaning the received request resulted in an error).
Request (yellow)	Flashes yellow each time a request is sent when operating as Modbus Master, or each time a request is received when operating as Modbus Slave.
Reply (green)	Operating as Modbus Master, flashes green each time a good response is received. Operating as Modbus Slave, flashes green each time a good response is sent.

Ethernet Activity	Green LED is on solid during portions of the boot-up process, and then flashes briefly when Ethernet network traffic is detected.
Ethernet Link	Yellow LED indicates an Ethernet link is present. This indicator will light if a link is present regardless of processor or network activity. If not lit, check network wiring.
Status	Blue LED (internal) on any time power is present and internal power supply is functioning.

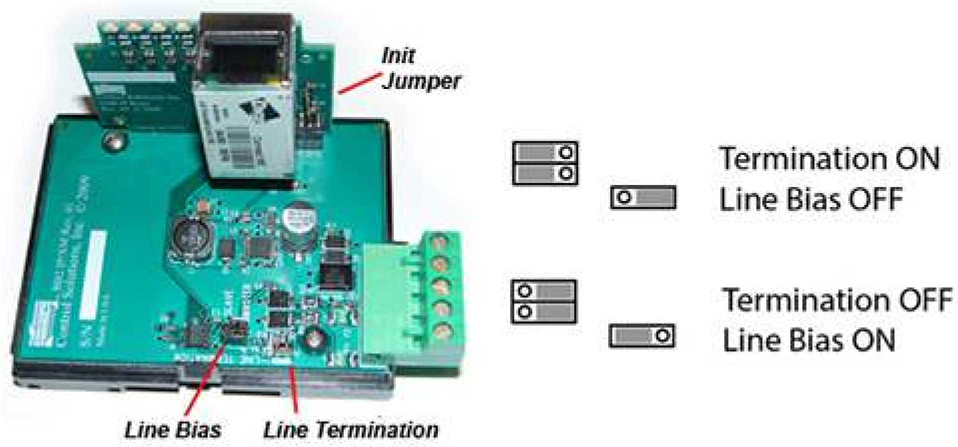
A.3 RS-485 Line Termination & Bias

Enable line termination only when this device is placed at the end of the network. Termination should only be enabled at two points on the network, and these two points must be specifically the end points.

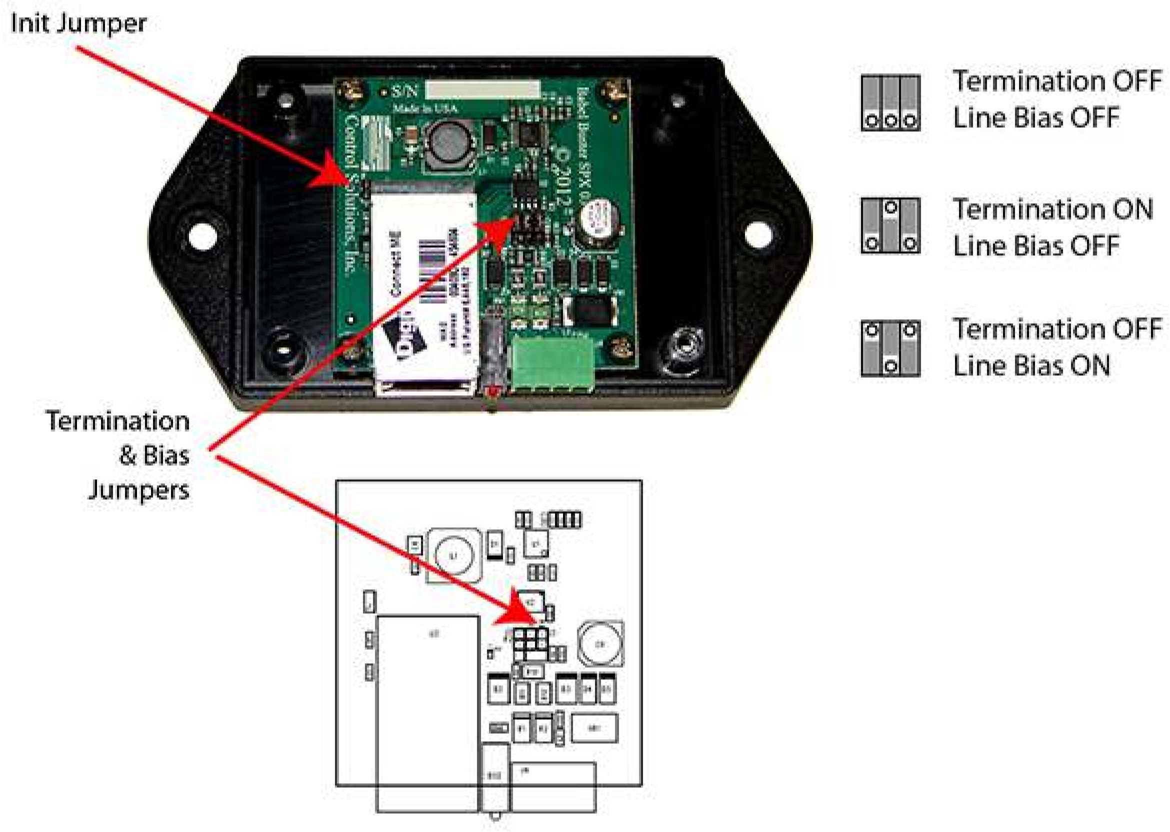
Enable line bias when needed. Line bias should only be enabled at one point on the network, and does not have to be the end point. Line bias holds the line in a known neutral state when no devices are transmitting. Without bias, the transition from offline to online by a transmitter can look like a false start bit and cause loss of communication.

The line conditioning options are enabled when the respective shunt is moved to the position indicated by the diagrams below.

Jumper locations for Babel Buster BB3-7101:



Jumper locations for Babel Buster MX-71:



A.4 Soft Configuration Reset

Soft reset should be used to remove all configuration information any time you do have the ability to connect to the gateway's web user interface. The "Clear Configuration" action is described in Section 11.1.5. Using the forced hard reset should only be used as a last resort if you are unable to connect to the gateway because the SSL certificates are invalid for a secure connection or you are unable to recover the lost IP address.

A.5 Discovering Lost IP Address

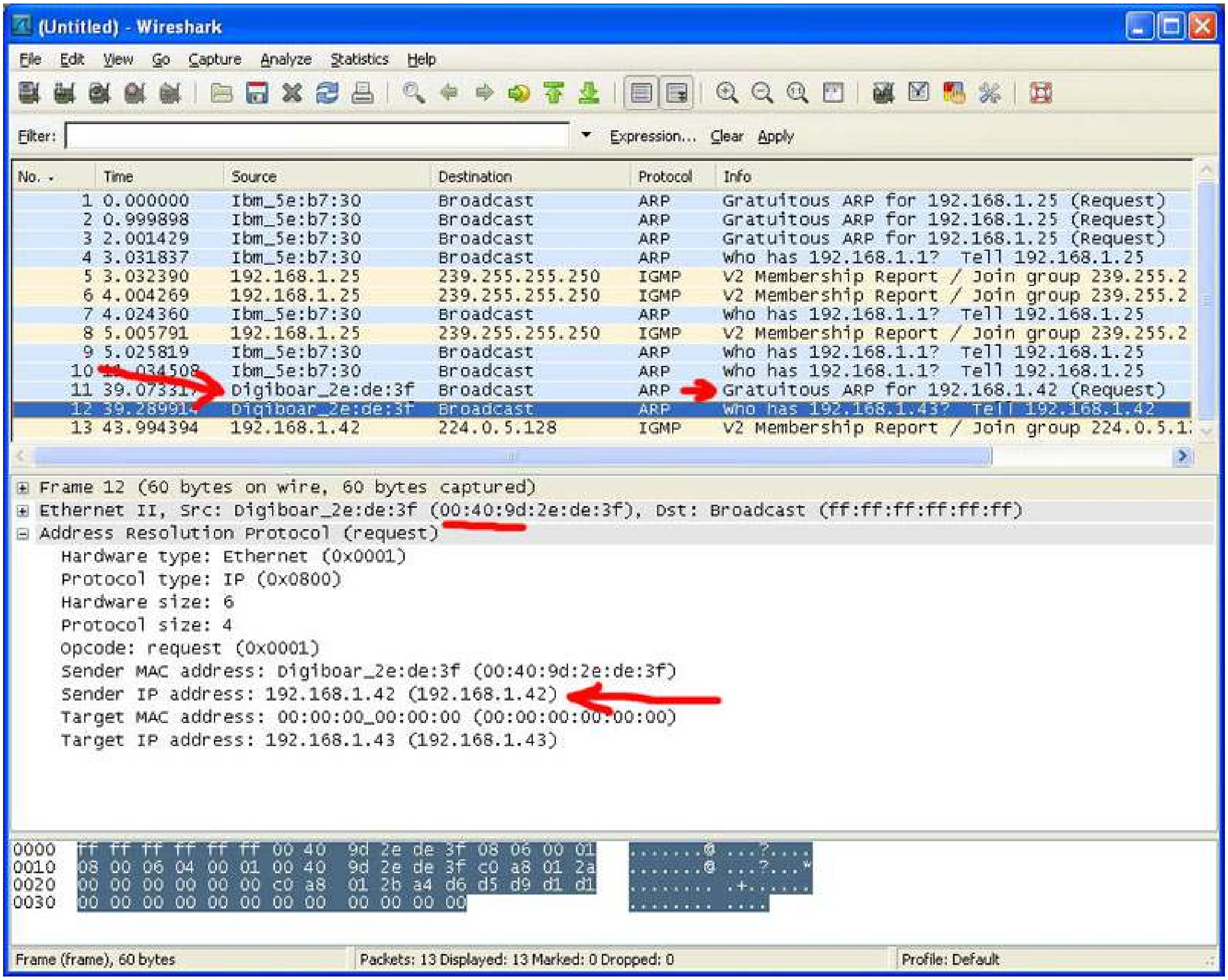
You can use Wireshark to discover a lost IP address if the gateway is still functional. Connect the gateway directly to your PC running Wireshark using a cross-over cable (or standard CAT5 cable if your PC supports auto-MDX). With Wireshark running, power up the gateway.

Upon power up, BB3-7101/MX-71 will ping its own IP address one or more times. This is part of its duplicate address resolution mechanism. If it finds another device with its own IP address, it will set its own IP address to a default pseudo-random address generally starting with 192.

Wait until you are certain BB3-7101/MX-71 has booted up, or wait 2-3 minutes to be sure if you don't recognize the bootup LED sequence. Now look for the ARP packets and note what IP address they came from. This is your device. (To make sure it is your device, connect only the BB3-7101/MX-71 to your PC while doing this exercise.)

Your device will have a MAC address that starts with 00:40:9D, also labeled with a source that starts with “Digiboar_”. This label comes from the fact that the server modules used on Control Solutions IP products are made by Digi International, previously known as “Digiboard”.

There will usually be one or more "pings" or ARP packets to the device's own IP address, and one last ping to its own address plus one. In the illustration here, the BB3-7101/MX-71 is located at 192.168.1.42.


(Untitled) - Wireshark
File Edit View Go Capture Analyze Statistics Help
Filter:
Expression... Clear Apply

```
No. -  Time        Source              Destination       Protocol  Info
    1  0.000000    Ibm_5e:b7:30        Broadcast         ARP       Gratuitous ARP for 192.168.1.25 (Request)
    2  0.999898    Ibm_5e:b7:30        Broadcast         ARP       Gratuitous ARP for 192.168.1.25 (Request)
    3  2.001429    Ibm_5e:b7:30        Broadcast         ARP       Gratuitous ARP for 192.168.1.25 (Request)
    4  3.031837    Ibm_5e:b7:30        Broadcast         ARP       Who has 192.168.1.1?  Tell 192.168.1.25
    5  3.032390    192.168.1.25        239.255.255.250   IGMP      V2 Membership Report / Join group 239.255.2
    6  4.004269    192.168.1.25        239.255.255.250   IGMP      V2 Membership Report / Join group 239.255.2
    7  4.024360    Ibm_5e:b7:30        Broadcast         ARP       Who has 192.168.1.1?  Tell 192.168.1.25
    8  5.005791    192.168.1.25        239.255.255.250   IGMP      V2 Membership Report / Join group 239.255.2
    9  5.025819    Ibm_5e:b7:30        Broadcast         ARP       Who has 192.168.1.1?  Tell 192.168.1.25
   10  [illegible].034508   Ibm_5e:b7:30        Broadcast         ARP       Who has 192.168.1.1?  Tell 192.168.1.25
   11  39.07331    Digiboar_2e:de:3f   Broadcast         ARP       Gratuitous ARP for 192.168.1.42 (Request)
   12  39.28991    Digiboar_2e:de:3f   Broadcast         ARP       Who has 192.168.1.43?  Tell 192.168.1.42
   13  43.994394   192.168.1.42        224.0.5.128       IGMP      V2 Membership Report / Join group 224.0.5.1
```

```
⊞ Frame 12 (60 bytes on wire, 60 bytes captured)
⊞ Ethernet II, Src: Digiboar_2e:de:3f (00:40:9d:2e:de:3f), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
⊟ Address Resolution Protocol (request)
    Hardware type: Ethernet (0x0001)
    Protocol type: IP (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: request (0x0001)
    Sender MAC address: Digiboar_2e:de:3f (00:40:9d:2e:de:3f)
    Sender IP address: 192.168.1.42 (192.168.1.42)
    Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
    Target IP address: 192.168.1.43 (192.168.1.43)
```

```
0000  ff ff ff ff ff ff 00 40  9d 2e de 3f 08 06 00 01   .......@ ...?....
0010  08 00 06 04 00 01 00 40  9d 2e de 3f c0 a8 01 2a   .......@ ...?...*
0020  00 00 00 00 00 00 c0 a8  01 2b a4 d6 d5 d9 d1 d1   ........ .+......
0030  00 00 00 00 00 00 00 00  00 00 00 00               ........ ....
```

Frame (frame), 60 bytes | Packets: 13 Displayed: 13 Marked: 0 Dropped: 0 | Profile: Default

A.6 Forced Hard Configuration Reset

IMPORTANT: Before considering the forced hard reset, be sure you have considered soft configuration reset, or discovering lost IP address if applicable.

The "Init" jumper inside the BB3-7101/MX-71 serves two purposes, and what it does depends on whether you apply the jumper before or after the BB3-7101/MX-71 boots up.

Hard Configuration Reset:

Installing the jumper after bootup causes the BB3-7101/MX-71 to do a hard reset on its configuration memory. The IPv4 address will be reset to 10.0.0.101. The root password will be reset to the original default password. After clearing all configuration, the BB3-7101/MX-71 will automatically restart. Remove the jumper when you see the indication of restart after about 30 seconds, which is both LEDs coming on solid on the RJ45 Ethernet connector and remaining on for a couple of seconds. If you miss the start of reboot, both LEDs on the RJ45 will come on and stay on. It will now be attempting the firmware update, but you can abort that by simply powering down the BB3-7101/MX-71. If both LEDs on the RJ45 jack come on and remain on, remove the jumper and then power cycle the BB3-7101/MX-71.

Once you have regained access to the device, go to the File Manager page, execute the Clear All configuration action, then select the file named as "Boot configuration" and execute the Save XML Config File action to wipe out any configuration normally saved in the XML configuration file.

Note: The forced hard reset will restore HTTP web access and disable HTTPS web access. The forced hard reset will also restore FTP access to allow FTP firmware uploads if needed.

Note: The hard reset of configuration also means all of your resource allocations are reset to original factory defaults. If you want resource allocations that are different, you will need to repeate the allocation setup as described in Section 15.3.

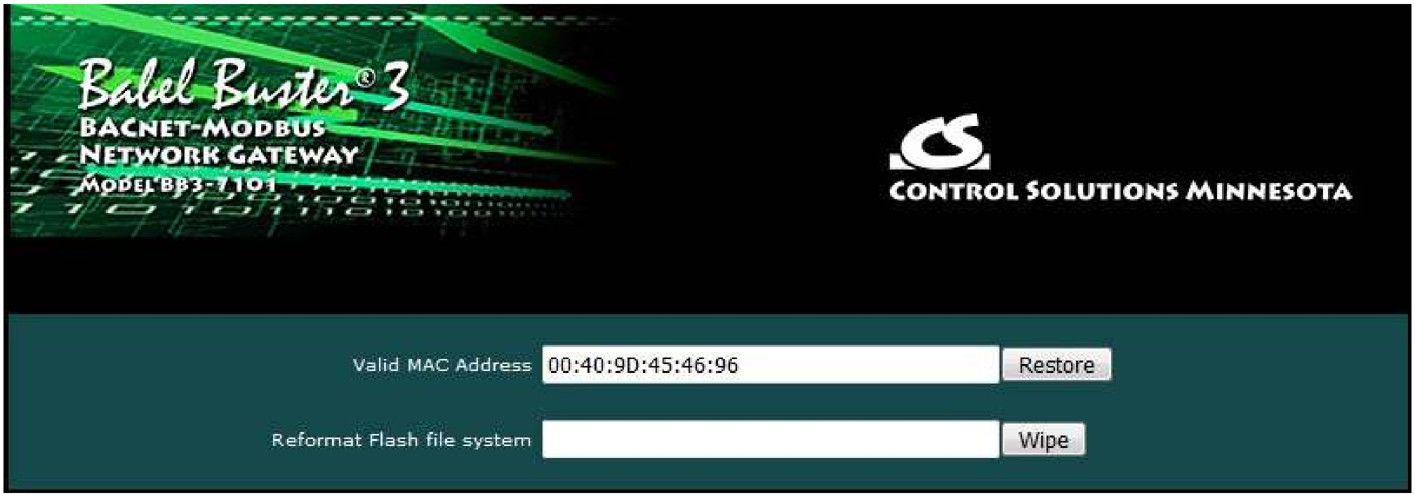
Firmware Update Recovery:

Installing this jumper prior to power-up causes the server to go into TFTP firmware update mode. Normally you would perform a firmware update by simply uploading a new image.bin file (provided by Control Solutions tech support) using the BB3-7101/MX-71's internal FTP server and a command line FTP session on your PC (Linux or Windows command line). Detailed instructions are included in the zip file that also contains the applicable image.bin file.

Should the FTP upload fail for some reason, then you need to resort to the TFTP upload method as the fallback method. Full details on how to go about this can be found under the topic "Restoring a corrupt application image" at https://info.csimn.com.

Additional maintenance page:

Go to http(s)://10.0.0.101/html/pgRestoreAddr.html to find the following page (substituting your IP address). It serves two purposes as noted below, which ideally you will never have a use for.



Valid MAC Address 00:40:9D:45:46:96 Restore

Reformat Flash file system Wipe

File System Wipe:

On rare occasion, the Flash file system has been observed to get corrupted as a result of losing power while a write operation was in progress. This is most effectively confirmed by opening a command prompt FTP session (Windows 10 PowerShell) to try to view the files in the Flash file system. If FTP fails to show any files, in addition to other problems saving or loading files, it may be that the file system has gotton corrupted. If this happens, go to the page pictured above, and enter the Reformat key, then click Wipe, and then power cycle the device (or restart from the File Manager page). The reformat key is 55AAAA55. Simply type that into the window next to the Wipe button.

MAC Address Restore:

In the event the MAC address has been reset due to NVRAM checksum failure, this page will permit restoring the MAC address to its original address as printed on the component label internal to this device, or on the default password label found on the outside or on external documentation included with the device.

If the MAC address is deemed to be valid, the window will be labeled "Valid MAC Address" and you will not be allowed to change it. If the MAC address is deemed to be invalid, the window will be labeled "Restore MAC Address" and you should then enter the correct MAC address and click Restore. A restart is then needed.

A.7 Firmware Update Notes

The most up to date firmware is shipped with all new devices. This isn't like a new laptop where you spent the first half a day updating software on a computer you thought was brand new. If you believe you have discovered an issue that you believe a firmware update might fix, contact technical support first to confirm whether that is the case, and then to get a login to the firmware update support site.

The brute force approach to updating firmware using TFTP as noted in the section above is always available, but the more graceful approach is to use FTP to upload the new image.bin file. There is one minor problem: The upload wants to buffer the entire file in RAM while it procedes to reprogram the Flash memory. **If the memory**

utilization indicated on the Resources page in your device is above about 30%, the FTP upload will fail, and thus the firmware update will not take place.

You have two choices: (1) Use the TFTP approach, or (2) Temporarily reconfigure your gateway to use a minimum of resources to free up space to temporarily buffer the image.bin file upload.

More detailed instructions for the FTP upload are included in the zip file you will download to obtain the firmware update. Instructions for the TFTP upload are available in our knowledgebase at https://info.csimn.com.

Babel Buster® 3
MODEL BB3-7101-SP
MODEL MX-71-SP
NETWORK GATEWAY

CONTROL SOLUTIONS MINNESOTA

Appendix B CSV File Formats

HINT: If you get "table full" errors while importing CSV files, you might not have sufficient resources allocated. You may need to increase some counts on the Resources page.

B.1 Modbus RTU Master Read/Write Maps

The CSV file for configuring Modbus TCP client read and write maps should contain a single header line with the labels indicated below, and content as applicable.

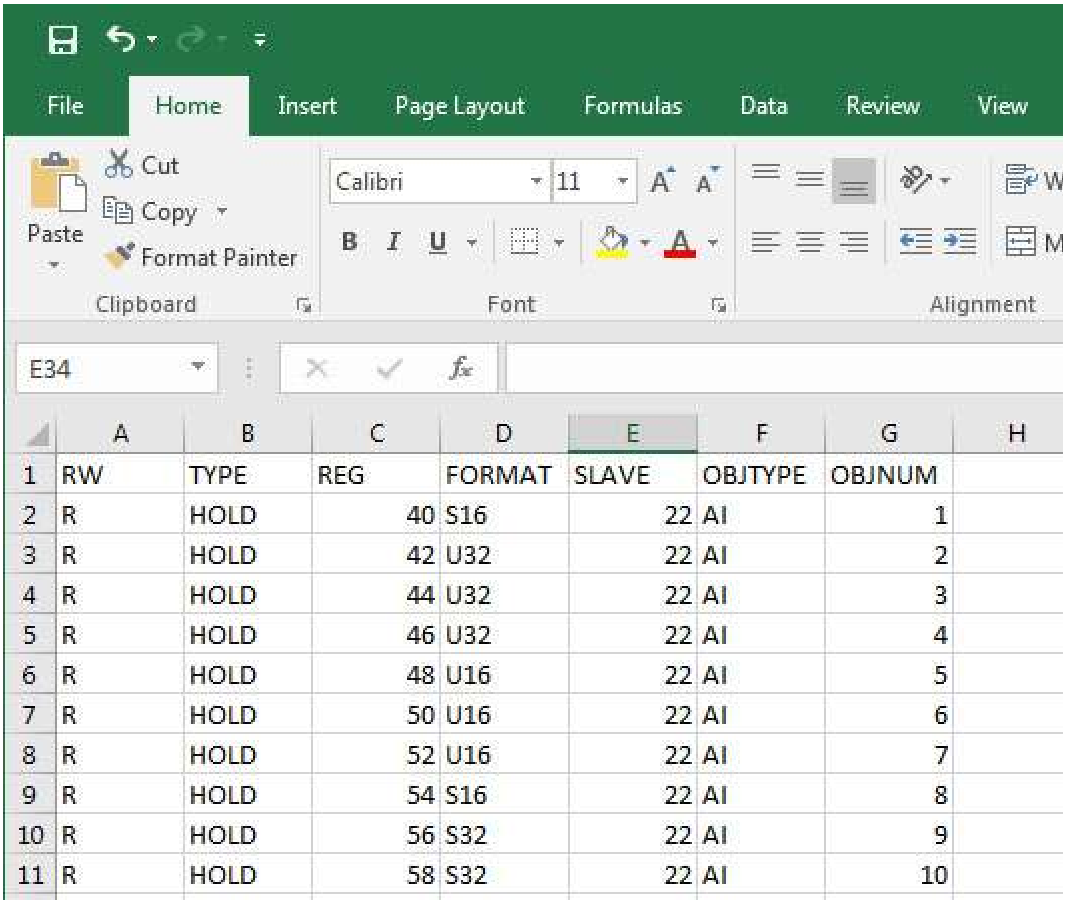
Header Line Label	Notes	Description of Use
RW	-	Enter 'R' to Read from a remote device, or 'W' to Write to a remote device.
TYPE	-	Use this column to specify remote registers by type (see B.4)
REG	-	Use this column in conjunction with Type to specify remote register numbers of the selected type. Note that register numbers are 1-indexed, meaning raw address 0 should be entered as register #1.
FORMAT	-	Specify the format of the remote register to be read or written (see B.5)
SLAVE	-	Provide the slave address, ID, or unit number, of the Modbus RTU slave to be polled.
SWAP	-	Any data item that occupies more than one Modbus register, e.g. 32-bit or Float, needs to have the register order defined since this is not standardized by Modbus protocol. The Babel Buster gateways default to the high order register first. If the remote Modbus slave has its registers ordered with low order first, then select 'T' (True) to "swap" the register order (or 'F' to keep the default order).
SCALE	-	Data is multiplied by this scale factor after read from a remote device or before being written to a remote device.
OFFSET	-	This offset is added to the data value after read from a remote device or before being written to a remote device.
POLL	-	Specify a periodic poll time in seconds (fractions of sections are recognized).

OBJTYPE	-	Indicate the type of local BACnet object (see B.6) where data read from a remote device will be placed, or where data written to a remote device will be taken from.
OBJNUM	-	Indicate the object number that goes along with object type in the previous column.
MASK	-	When READING: If a bit mask is entered (in hexadecimal), and the remote register type is signed or unsigned integer, the mask will be bit-wise logical AND-ed with the data, and the retained bits will be right justified in the result. When WRITING: If a bit mask is entered, and the remote register type is signed or unsigned, the mask will be bit-wise logical AND-ed with the data. The mask is right justified, then AND-ed with the data. The result is then left shifted back to the original position of the mask. In other words, the least significant bits of the original data will be stuffed at the position marked by the mask.
DEFAULT	1	When READING: The default value will be stored into the local object/register after the given number of read failures if the fail count (MAXFAIL) is non-zero.
MAXFAIL	1	If non-zero, sets the maximum number of times that a read attempt may fail before the default value will be placed in the local object/register. Setting the count to zero will disable the default, and the object/register will retain the most recent value obtained.
FILL	2	When WRITING: The bit fill will be logically OR-ed into the result, but only if the mask was nonzero and was used. Both mask and fill are entered in hexadecimal.
MAXQUIET	2	If using 'send on delta', to guarantee that the remote device will be written at least occasionally even if the data does not change, enter a maximum quiet time (in seconds).
MINQUIET	2	If using 'send on delta', and the delta increment is small, the result can be a large amount of network traffic. To limit network traffic, provide a MINQUIET time (in seconds) that must elapse between transmission of changed values.
DELTA	2	The local object/register data may be written to the remote device periodically, or when the local value changes, or both. To send upon change (send on delta), provide a DELTA value as the amount by which the local object/register must change before being written to the remote device. Leave blank if send on delta should not be used.

Notes:
1) Applies only to Read maps (enter zero as place holder for Write maps)
2) These apply only to Write maps (enter zero as place holder for Read maps)

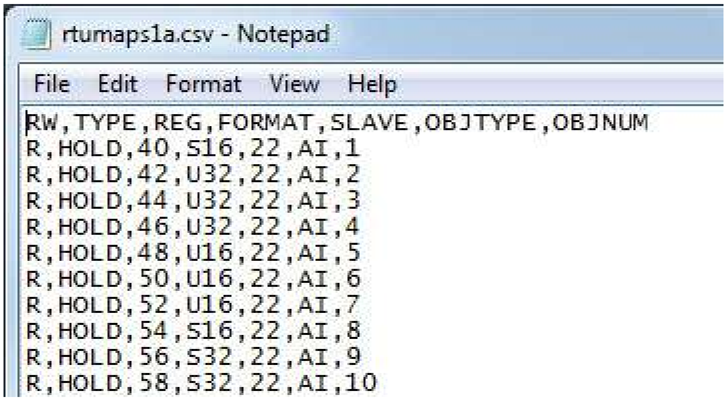
The minimum required header line for Modbus RTU must include RW, TYPE, REG, FORMAT, SLAVE, OBJTYPE, OBJNUM. All other columns are optional.

This is an example of a minimum CSV file as it would appear in a spread sheet program:



	A	B	C	D	E	F	G	H
1	RW	TYPE	REG	FORMAT	SLAVE	OBJTYPE	OBJNUM	
2	R	HOLD	40	S16	22	AI	1	
3	R	HOLD	42	U32	22	AI	2	
4	R	HOLD	44	U32	22	AI	3	
5	R	HOLD	46	U32	22	AI	4	
6	R	HOLD	48	U16	22	AI	5	
7	R	HOLD	50	U16	22	AI	6	
8	R	HOLD	52	U16	22	AI	7	
9	R	HOLD	54	S16	22	AI	8	
10	R	HOLD	56	S32	22	AI	9	
11	R	HOLD	58	S32	22	AI	10	

This is an example of how the minimum CSV file looks as just plain text:



```
RW,TYPE,REG,FORMAT,SLAVE,OBJTYPE,OBJNUM
R,HOLD,40,S16,22,AI,1
R,HOLD,42,U32,22,AI,2
R,HOLD,44,U32,22,AI,3
R,HOLD,46,U32,22,AI,4
R,HOLD,48,U16,22,AI,5
R,HOLD,50,U16,22,AI,6
R,HOLD,52,U16,22,AI,7
R,HOLD,54,S16,22,AI,8
R,HOLD,56,S32,22,AI,9
R,HOLD,58,S32,22,AI,10
```

B.2 Modbus TCP Client Read/Write Maps

The CSV file for configuring Modbus TCP client read and write maps should contain a single header line with the labels indicated below, and content as applicable.

The only difference between TCP and RTU formats is DEVNUM and UNIT in TCP, versus just SLAVE in RTU. Everything else is identical.

Header Line Label	Notes	Description of Use
RW	-	Enter 'R' to Read from a remote device, or 'W' to Write to a remote device.
TYPE	-	Use this column to specify remote registers by type (see B.4)
REG	-	Use this column in conjunction with Type to specify remote register numbers of the selected type. Note that register numbers are 1-indexed, meaning raw address 0 should be entered as register #1.
FORMAT	-	Specify the format of the remote register to be read or written (see B.5)
DEVNUM	-	Specify the device number where the remote register is to be found. This number is used to look up a device in the Modbus TCP Client Device table which contains the device's IP address, etc.
UNIT	-	Unit number is optional, and may be 1 to 247.
SCALE	-	Data is multiplied by this scale factor after read from a remote device or before being written to a remote device.
OFFSET	-	This offset is added to the data value after read from a remote device or before being written to a remote device.
POLL	-	Specify a periodic poll time in seconds (fractions of sections are recognized).
OBJTYPE	-	Indicate the type of local BACnet object (see B.6) where data read from a remote device will be placed, or where data written to a remote device will be taken from.
OBJNUM	-	Indicate the object number that goes along with object type in the previous column.
MASK	-	When READING: If a bit mask is entered (in hexadecimal), and the remote register type is signed or unsigned integer, the mask will be bit-wise logical AND-ed with the data, and the retained bits will be right justified in the result. When WRITING: If a bit mask is entered, and the remote register type is signed or unsigned, the mask will be bit-wise logical AND-ed with the data. The mask is right justified, then AND-ed with the data. The result is then left shifted back to the original position of the mask. In other words, the least significant bits of the original data will be stuffed at the position marked by the mask.
DEFAULT	1	When READING: The default value will be stored into the local object/register after the given number of read failures if the fail count (MAXFAIL) is non-zero.
MAXFAIL	1	If non-zero, sets the maximum number of times that a read attempt may fail before the default value will be placed in the local object/register. Setting the count to zero will disable the

		default, and the object/register will retain the most recent value obtained.
FILL	2	When WRITING: The bit fill will be logically OR-ed into the result, but only if the mask was nonzero and was used. Both mask and fill are entered in hexadecimal.
MAXQUIET	2	If using 'send on delta', to guarantee that the remote device will be written at least occasionally even if the data does not change, enter a maximum quiet time (in seconds).
MINQUIET	2	If using 'send on delta', and the delta increment is small, the result can be a large amount of network traffic. To limit network traffic, provide a MINQUIET time (in seconds) that must elapse between transmission of changed values.
DELTA	2	The local object/register data may be written to the remote device periodically, or when the local value changes, or both. To send upon change (send on delta), provide a DELTA value as the amount by which the local object/register must change before being written to the remote device. Leave blank if send on delta should not be used.

Notes:
1) Applies only to Read maps (enter zero as place holder for Write maps)
2) These apply only to Write maps (enter zero as place holder for Read maps)

The minimum required header line for Modbus TCP must include RW, TYPE, REG, FORMAT, DEVNUM, OBJTYPE, OBJNUM. All other columns are optional.

B.3 BACnet IP Client Read/Write Maps

The CSV file for configuring BACnet IP client read and write maps should contain a single header line with the labels indicated below, and content as applicable.

Header Line Label	Notes	Description of Use
RW	-	Enter 'R' to Read from a remote device, or 'W' to Write to a remote device.
REMOTEOBJTYPE	-	Indicate the type of local BACnet object (see B.6) that should be read or written at the remote device. In addition to the object types recognized as local objects, the client may read remote Accumulator objects referenced as type "AC".
REMOTEOBJNUM	-	Indicate the remote object number that goes along with object type in the previous column.
PROPERTY	-	Specify by BACnet code the object property (see Appendix D) that should be read. The most common is Present Value, whose code is 85.
INDEX	-	If the property to be read/written is an array, then an array index is needed. Specify "no index" by entering zero in the CSV column. Otherwise enter 1 or greater, and note that

		actual index values will be offset by -1 when applied by the BACnet client.
DEVNUM	-	Specify the device number where the remote object is to be found. This number is used to look up a device in the BACnet Client Device table which contains the device's BACnet Device Instance, or static binding if applicable, etc.
SCALE	-	Data is multiplied by this scale factor after read from a remote device or before being written to a remote device.
OFFSET	-	This offset is added to the data value after read from a remote device or before being written to a remote device.
POLL	-	Specify a periodic poll time in seconds (fractions of sections are recognized).
OBJTYPE	-	Indicate the type of local BACnet object (see B.6) where data read from a remote device will be placed, or where data written to a remote device will be taken from.
OBJNUM	-	Indicate the object number that goes along with object type in the previous column.
DEFAULT	1	When READING: The default value will be stored into the local object after the given number of read failures if the fail count (MAXFAIL) is non-zero.
MAXFAIL	1	If non-zero, sets the maximum number of times that a read attempt may fail before the default value will be placed in the local object. Setting the count to zero will disable the default, and the object will retain the most recent value obtained.
DATATYPE	2	Provide the data type code that the remote object being written expects to receive. 1=Boolean, 2=Unsigned Integer, 3=Signed Integer, 4=Real, 9=Enumerated (note that 5, 6, 7, 8 are not used here)
PRIORITY	2	If writing to a commandable object, then a priority (1-16) must be provided.
MAXQUIET	2	If using 'send on delta', to guarantee that the remote device will be written at least occasionally even if the data does not change, enter a maximum quiet time (in seconds).
MINQUIET	2	If using 'send on delta', and the delta increment is small, the result can be a large amount of network traffic. To limit network traffic, provide a MINQUIET time (in seconds) that must elapse between transmission of changed values.
DELTA	2	The local object/register data may be written to the remote device periodically, or when the local value changes, or both. To send upon change (send on delta), provide a DELTA value as the amount by which the local object must change before being written to the remote device. Leave blank if send on delta should not be used.

Notes:
1) Applies only to Read maps (enter zero as place holder for Write maps)

2) These apply only to Write maps (enter zero as place holder for Read maps)

The minimum required header line for BACnet IP must include RW, REMOTEOBJTYPE, REMOTEOBJNUM, PROPERTY, DEVNUM, OBJTYPE, OBJNUM. All other columns are optional.

B.4 Modbus Register Types

The content of the TYPE column should contain one of the following CSV Labels:

CSV Label	Modbus Register Type	Function Code for Read	Function Code for Write
COIL	Coil (1 bit)	1	5 or 15
DISC	Discrete Input (1 bit)	2	n/a
INPUT	Input Register (16 bits)	4	n/a
HOLD	Holding Register (16 bits)	3	6 or 16

B.5 Modbus Register Formats

The content of the FORMAT column should contain one of the following CSV Labels:

CSV Label	Modbus Register Data Format	Occupies # Registers
S16	Signed 16-bit Integer	1
U16	Unsigned 16-bit Integer	1
S32	Signed 32-bit Integer	2
U32	Unsigned 32-bit Integer	2
S64	Signed 64-bit Integer	4
U64	Unsigned 64-bit Integer	4
FP	Floating Point, IEEE 754 32-bit	2
DP	Double Precision Floating Point, IEEE 754 64-bit	4
MOD102	Mod-10, 2-register	2
MOD103	Mod-10, 3-register	3
MOD104	Mod-10, 4-register	4

Note: For Coil or Discrete Input, use S16 or U16 as register format. These 1-bit register types do not really have any format so this is just a place-holder in these instances.

B.6 BACnet Object Types

CSV Label	BACnet Object Type
AI	Analog Input
AO	Analog Output
AV	Analog Value

BI	Binary Input
BO	Binary Output
BV	Binary Value
MI	Multistate Input
MO	Multistate Output
MV	Multistate Value

Babel Buster® 3
MODEL BB3-7101-SP
MODEL MX-71-SP
NETWORK GATEWAY

CONTROL SOLUTIONS MINNESOTA

Appendix C BACnet Object Properties

C.1 Data Object Properties (Analog, Binary, Multi-state)

The following properties are found in the Analog, Binary, and Multi-state types of Input, Output, and Value objects. Some properties apply only to certain object types as noted where applicable.

Property	Encoding
Object_Identifier (75)	BACnetObjectIdentifier
Object_Name (77) (W)	CharacterString "Analog Input *n*"
Object_Type (79)	BACnetObjectType ENUMERATED: analog-input (0) analog-output (1) analog-value (2) binary-input (3) binary-output (4) binary-value (5) device (8) multi-state-input (13) multi-state-output (14) multi-state-value (19)
Present_Value (85) (W)	REAL (analog objects) ENUMERATED (binary objects) Unsigned (multi-state objets) (no index) (priority required when writing commandable objects) (input objects writeable only when out of service)
Status_Flags (111)	BACnetStatusFlags BIT STRING: fault(1), out-of-service(3)
Event_State (36)	BACnetEventState ENUMERATED: normal(0), fault(1)
Reliability (103)	BACnetReliability ENUMERATED: normal(0)

	Vendor specific: Modbus client/master, no response from slave (64) Modbus client/master, crc error (65) Modbus exception, illegal function code (66) Modbus exception, illegal data address (67) Modbus exception, illegal data value (68) Modbus exception, code+65, rarely used (69..79) Local device, configuration property fault (80) Faulty Modbus packet(81) BACnet IP client, device timeout (82) BACnet IP client, error returned by server (83)
Out_Of_Service (81) (W)	BOOLEAN
COV_Increment (22) (W)	REAL (analog objects only)
Priority_Array (87)	BACnetPriorityArray (commandable objects only) SEQUENCE SIZE (16) OF BACnetPriorityValue REAL (each element, analog output objects) ENUMERATED (each element, binary output objects) Unsigned (each element, multi-state output objects)
Relinquish_Default (104) (W)	REAL (analog objects) ENUMERATED (binary objects) Unsigned (multi-state objets)
Polarity (84)	BACnetPolarity (binary objects only) ENUMERATED: normal(0)
Number_Of_States (74)	Unsigned (multi-state objects only)
Units (117)	BACnetEngineeringUnits (analog objects only)

C.2 Device Object Properties

The following properties are found in the Device object of the BB3-7101/MX-71.

Property	Encoding
Object_Identifier (75)	BACnetObjectIdentifier
Object_Name (77)	CharacterString
Object_Type (79)	BACnetObjectType ENUMERATED: device (8)
System_Status (112)	BACnetDeviceStatus
Vendor_Name (121)	CharacterString

Vendor_Identifier (120)	Unsigned16 (should always return 208)
Model_Name (70)	CharacterString
Fimrware_Revision (44)	CharacterString
Application_Software_Version (12)	CharacterString
Protocol_Version (98)	Unsigned
Protocol_Revision (139)	Unsigned
Protocol_Services_Supported (97)	BACnetServicesSupported
Protocol_Object_Types_Supported (96)	BACnetObjectTypesSupported
Object_List (76)	BACnetARRAY[N] of BACnetObjectIdentifier
Max_APDU_Length_Accepted (62)	Unsigned
Segmentation_Supported (107)	BACnetSegmentation
APDU_Timeout (11)	Unsigned
Number_Of_APDU_Retries (73)	Unsigned
Device_Address_Binding (30)	List of BACnetAddressBinding
Database_Revision (155)	Unsigned

Appendix D BACnet Codes

D.1 BACnet Object Property Codes

BACnet property type codes may be found in your copy of the BACnet protocol specification, ANSI/ASHRAE Standard 135. That document is copyrighted, but the C enumeration shown below for reference is taken from open source code available under GPL at http://sourceforge.net, and provides essentially the same information (copyrighted by Steve Karg, licensed under GPL as noted at http://sourceforge.net).

```
typedef enum {
    PROP_ACKED_TRANSITIONS = 0,
    PROP_ACK_REQUIRED = 1,
    PROP_ACTION = 2,
    PROP_ACTION_TEXT = 3,
    PROP_ACTIVE_TEXT = 4,
    PROP_ACTIVE_VT_SESSIONS = 5,
    PROP_ALARM_VALUE = 6,
    PROP_ALARM_VALUES = 7,
    PROP_ALL = 8,
    PROP_ALL_WRITES_SUCCESSFUL = 9,
    PROP_APDU_SEGMENT_TIMEOUT = 10,
    PROP_APDU_TIMEOUT = 11,
    PROP_APPLICATION_SOFTWARE_VERSION = 12,
    PROP_ARCHIVE = 13,
    PROP_BIAS = 14,
    PROP_CHANGE_OF_STATE_COUNT = 15,
    PROP_CHANGE_OF_STATE_TIME = 16,
    PROP_NOTIFICATION_CLASS = 17,
    PROP_BLANK_1 = 18,
    PROP_CONTROLLED_VARIABLE_REFERENCE = 19,
    PROP_CONTROLLED_VARIABLE_UNITS = 20,
    PROP_CONTROLLED_VARIABLE_VALUE = 21,
    PROP_COV_INCREMENT = 22,
    PROP_DATE_LIST = 23,
    PROP_DAYLIGHT_SAVINGS_STATUS = 24,
    PROP_DEADBAND = 25,
    PROP_DERIVATIVE_CONSTANT = 26,
    PROP_DERIVATIVE_CONSTANT_UNITS = 27,
    PROP_DESCRIPTION = 28,
    PROP_DESCRIPTION_OF_HALT = 29,
    PROP_DEVICE_ADDRESS_BINDING = 30,
    PROP_DEVICE_TYPE = 31,
    PROP_EFFECTIVE_PERIOD = 32,
    PROP_ELAPSED_ACTIVE_TIME = 33,
    PROP_ERROR_LIMIT = 34,
    PROP_EVENT_ENABLE = 35,
    PROP_EVENT_STATE = 36,
    PROP_EVENT_TYPE = 37,
    PROP_EXCEPTION_SCHEDULE = 38,
    PROP_FAULT_VALUES = 39,
    PROP_FEEDBACK_VALUE = 40,
    PROP_FILE_ACCESS_METHOD = 41,
    PROP_FILE_SIZE = 42,
    PROP_FILE_TYPE = 43,
    PROP_FIRMWARE_REVISION = 44,
    PROP_HIGH_LIMIT = 45,
    PROP_INACTIVE_TEXT = 46,
    PROP_IN_PROCESS = 47,
    PROP_INSTANCE_OF = 48,
    PROP_INTEGRAL_CONSTANT = 49,
    PROP_INTEGRAL_CONSTANT_UNITS = 50,
    PROP_ISSUE_CONFIRMED_NOTIFICATIONS = 51,
    PROP_LIMIT_ENABLE = 52,
    PROP_LIST_OF_GROUP_MEMBERS = 53,
    PROP_LIST_OF_OBJECT_PROPERTY_REFERENCES = 54,
    PROP_LIST_OF_SESSION_KEYS = 55,
    PROP_LOCAL_DATE = 56,
    PROP_LOCAL_TIME = 57,
    PROP_LOCATION = 58,
    PROP_LOW_LIMIT = 59,
    PROP_MANIPULATED_VARIABLE_REFERENCE = 60,
    PROP_MAXIMUM_OUTPUT = 61,
    PROP_MAX_APDU_LENGTH_ACCEPTED = 62,
    PROP_MAX_INFO_FRAMES = 63,
    PROP_MAX_MASTER = 64,
    PROP_MAX_PRES_VALUE = 65,
    PROP_MINIMUM_OFF_TIME = 66,
    PROP_MINIMUM_ON_TIME = 67,
    PROP_MINIMUM_OUTPUT = 68,
    PROP_MIN_PRES_VALUE = 69,
    PROP_MODEL_NAME = 70,
    PROP_MODIFICATION_DATE = 71,
    PROP_NOTIFY_TYPE = 72,
    PROP_NUMBER_OF_APDU_RETRIES = 73,
    PROP_NUMBER_OF_STATES = 74,
    PROP_OBJECT_IDENTIFIER = 75,
    PROP_OBJECT_LIST = 76,
    PROP_OBJECT_NAME = 77,
    PROP_OBJECT_PROPERTY_REFERENCE = 78,
    PROP_OBJECT_TYPE = 79,
    PROP_OPTIONAL = 80,
    PROP_OUT_OF_SERVICE = 81,
    PROP_OUTPUT_UNITS = 82,
    PROP_EVENT_PARAMETERS = 83,
    PROP_POLARITY = 84,
    PROP_PRESENT_VALUE = 85,
    PROP_PRIORITY = 86,
    PROP_PRIORITY_ARRAY = 87,
    PROP_PRIORITY_FOR_WRITING = 88,
    PROP_PROCESS_IDENTIFIER = 89,
    PROP_PROGRAM_CHANGE = 90,
    PROP_PROGRAM_LOCATION = 91,
    PROP_PROGRAM_STATE = 92,
    PROP_PROPORTIONAL_CONSTANT = 93,
    PROP_PROPORTIONAL_CONSTANT_UNITS = 94,
    PROP_PROTOCOL_CONFORMANCE_CLASS = 95,       /* deleted in version 1
revision 2 */
    PROP_PROTOCOL_OBJECT_TYPES_SUPPORTED = 96,
    PROP_PROTOCOL_SERVICES_SUPPORTED = 97,
    PROP_PROTOCOL_VERSION = 98,
    PROP_READ_ONLY = 99,
    PROP_REASON_FOR_HALT = 100,
    PROP_RECIPIENT = 101,
    PROP_RECIPIENT_LIST = 102,
    PROP_RELIABILITY = 103,
    PROP_RELINQUISH_DEFAULT = 104,
    PROP_REQUIRED = 105,
    PROP_RESOLUTION = 106,
    PROP_SEGMENTATION_SUPPORTED = 107,
    PROP_SETPOINT = 108,
    PROP_SETPOINT_REFERENCE = 109,
    PROP_STATE_TEXT = 110,
    PROP_STATUS_FLAGS = 111,
    PROP_SYSTEM_STATUS = 112,
    PROP_TIME_DELAY = 113,
    PROP_TIME_OF_ACTIVE_TIME_RESET = 114,
    PROP_TIME_OF_STATE_COUNT_RESET = 115,
    PROP_TIME_SYNCHRONIZATION_RECIPIENTS = 116,
    PROP_UNITS = 117,
    PROP_UPDATE_INTERVAL = 118,
    PROP_UTC_OFFSET = 119,
    PROP_VENDOR_IDENTIFIER = 120,
    PROP_VENDOR_NAME = 121,
    PROP_VT_CLASSES_SUPPORTED = 122,
    PROP_WEEKLY_SCHEDULE = 123,
    PROP_ATTEMPTED_SAMPLES = 124,
    PROP_AVERAGE_VALUE = 125,
    PROP_BUFFER_SIZE = 126,
    PROP_CLIENT_COV_INCREMENT = 127,
    PROP_COV_RESUBSCRIPTION_INTERVAL = 128,
    PROP_CURRENT_NOTIFY_TIME = 129,
    PROP_EVENT_TIME_STAMPS = 130,
    PROP_LOG_BUFFER = 131,
    PROP_LOG_DEVICE_OBJECT = 132,
    /* The enable property is renamed from log-enable in
       Addendum b to ANSI/ASHRAE 135-2004(135b-2) */
    PROP_ENABLE = 133,
    PROP_LOG_INTERVAL = 134,
    PROP_MAXIMUM_VALUE = 135,
    PROP_MINIMUM_VALUE = 136,
    PROP_NOTIFICATION_THRESHOLD = 137,
    PROP_PREVIOUS_NOTIFY_TIME = 138,
    PROP_PROTOCOL_REVISION = 139,
    PROP_RECORDS_SINCE_NOTIFICATION = 140,
    PROP_RECORD_COUNT = 141,
    PROP_START_TIME = 142,
    PROP_STOP_TIME = 143,
    PROP_STOP_WHEN_FULL = 144,
    PROP_TOTAL_RECORD_COUNT = 145,
    PROP_VALID_SAMPLES = 146,
    PROP_WINDOW_INTERVAL = 147,
    PROP_WINDOW_SAMPLES = 148,
    PROP_MAXIMUM_VALUE_TIMESTAMP = 149,
    PROP_MINIMUM_VALUE_TIMESTAMP = 150,
    PROP_VARIANCE_VALUE = 151,
    PROP_ACTIVE_COV_SUBSCRIPTIONS = 152,
    PROP_BACKUP_FAILURE_TIMEOUT = 153,
    PROP_CONFIGURATION_FILES = 154,
    PROP_DATABASE_REVISION = 155,
    PROP_DIRECT_READING = 156,
    PROP_LAST_RESTORE_TIME = 157,
    PROP_MAINTENANCE_REQUIRED = 158,
    PROP_MEMBER_OF = 159,
    PROP_MODE = 160,
    PROP_OPERATION_EXPECTED = 161,
    PROP_SETTING = 162,
    PROP_SILENCED = 163,
    PROP_TRACKING_VALUE = 164,
    PROP_ZONE_MEMBERS = 165,
    PROP_LIFE_SAFETY_ALARM_VALUES = 166,
    PROP_MAX_SEGMENTS_ACCEPTED = 167,
    PROP_PROFILE_NAME = 168,
    PROP_AUTO_SLAVE_DISCOVERY = 169,
    PROP_MANUAL_SLAVE_ADDRESS_BINDING = 170,
    PROP_SLAVE_ADDRESS_BINDING = 171,
    PROP_SLAVE_PROXY_ENABLE = 172,
    PROP_LAST_NOTIFY_TIME = 173,
    PROP_SCHEDULE_DEFAULT = 174,
    PROP_ACCEPTED_MODES = 175,
    PROP_ADJUST_VALUE = 176,
    PROP_COUNT = 177,
    PROP_COUNT_BEFORE_CHANGE = 178,
    PROP_COUNT_CHANGE_TIME = 179,
    PROP_COV_PERIOD = 180,
    PROP_INPUT_REFERENCE = 181,
    PROP_LIMIT_MONITORING_INTERVAL = 182,
    PROP_LOGGING_DEVICE = 183,
    PROP_LOGGING_RECORD = 184,
    PROP_PRESCALE = 185,
    PROP_PULSE_RATE = 186,
    PROP_SCALE = 187,
    PROP_SCALE_FACTOR = 188,
    PROP_UPDATE_TIME = 189,
    PROP_VALUE_BEFORE_CHANGE = 190,
    PROP_VALUE_SET = 191,
    PROP_VALUE_CHANGE_TIME = 192,
    /* enumerations 193-206 are new */
    PROP_ALIGN_INTERVALS = 193,
    PROP_GROUP_MEMBER_NAMES = 194,
    PROP_INTERVAL_OFFSET = 195,
    PROP_LAST_RESTART_REASON = 196,
    PROP_LOGGING_TYPE = 197,
    PROP_MEMBER_STATUS_FLAGS = 198,
    PROP_NOTIFICATION_PERIOD = 199,
    PROP_PREVIOUS_NOTIFY_RECORD = 200,
    PROP_REQUESTED_UPDATE_INTERVAL = 201,
    PROP_RESTART_NOTIFICATION_RECIPIENTS = 202,
    PROP_TIME_OF_DEVICE_RESTART = 203,
    PROP_TIME_SYNCHRONIZATION_INTERVAL = 204,
    PROP_TRIGGER = 205,
    PROP_UTC_TIME_SYNCHRONIZATION_RECIPIENTS = 206,
    /* enumerations 207-211 are used in Addendum d to ANSI/ASHRAE
135-2004 */
    PROP_NODE_SUBTYPE = 207,
    PROP_NODE_TYPE = 208,
    PROP_STRUCTURED_OBJECT_LIST = 209,
    PROP_SUBORDINATE_ANNOTATIONS = 210,
    PROP_SUBORDINATE_LIST = 211,
    /* enumerations 212-225 are used in Addendum e to ANSI/ASHRAE
135-2004 */
    PROP_ACTUAL_SHED_LEVEL = 212,
    PROP_DUTY_WINDOW = 213,
    PROP_EXPECTED_SHED_LEVEL = 214,
    PROP_FULL_DUTY_BASELINE = 215,
    /* enumerations 216-217 are used in Addendum i to ANSI/ASHRAE
135-2004 */
    PROP_BLINK_PRIORITY_THRESHOLD = 216,
    PROP_BLINK_TIME = 217,
    /* enumerations 212-225 are used in Addendum e to ANSI/ASHRAE
135-2004 */
    PROP_REQUESTED_SHED_LEVEL = 218,
    PROP_SHED_DURATION = 219,
    PROP_SHED_LEVEL_DESCRIPTIONS = 220,
    PROP_SHED_LEVELS = 221,
    PROP_STATE_DESCRIPTION = 222,
    /* enumerations 223-225 are used in Addendum i to ANSI/ASHRAE
135-2004 */
    PROP_FADE_TIME = 223,
    PROP_LIGHTING_COMMAND = 224,
    PROP_LIGHTING_COMMAND_PRIORITY = 225,
    /* enumerations 226-235 are used in Addendum f to ANSI/ASHRAE
135-2004 */
    PROP_DOOR_ALARM_STATE = 226,
    PROP_DOOR_EXTENDED_PULSE_TIME = 227,
    PROP_DOOR_MEMBERS = 228,
    PROP_DOOR_OPEN_TOO_LONG_TIME = 229,
    PROP_DOOR_PULSE_TIME = 230,
    PROP_DOOR_STATUS = 231,
    PROP_DOOR_UNLOCK_DELAY_TIME = 232,
    PROP_LOCK_STATUS = 233,
    PROP_MASKED_ALARM_VALUES = 234,
    PROP_SECURED_STATUS = 235,
    /* enumerations 236-243 are used in Addendum i to ANSI/ASHRAE
135-2004 */
    PROP_OFF_DELAY = 236,
    PROP_ON_DELAY = 237,
    PROP_POWER = 238,
    PROP_POWER_ON_VALUE = 239,
    PROP_PROGRESS_VALUE = 240,
    PROP_RAMP_RATE = 241,
    PROP_STEP_INCREMENT = 242,
    PROP_SYSTEM_FAILURE_VALUE = 243,
    /* enumerations 244-311 are used in Addendum j to ANSI/ASHRAE
135-2004 */
    PROP_ABSENTEE_LIMIT = 244,
    PROP_ACCESS_ALARM_EVENTS = 245,
    PROP_ACCESS_DOORS = 246,
    PROP_ACCESS_EVENT = 247,
    PROP_ACCESS_EVENT_AUTHENTICATION_FACTOR = 248,
    PROP_ACCESS_EVENT_CREDENTIAL = 249,
    PROP_ACCESS_EVENT_TIME = 250,
    PROP_ACCESS_RULES = 251,
    PROP_ACCESS_RULES_ENABLE = 252,
    PROP_ACCESS_TRANSACTION_EVENTS = 253,
    PROP_ACCOMPANIED = 254,
    PROP_ACTIVATION_TIME = 255,
    PROP_ACTIVE_AUTHENTICATION_POLICY = 256,
    PROP_ASSIGNED_ACCESS_RIGHTS = 257,
    PROP_AUTHENTICATION_FACTOR_INPUT_LIST = 258,
    PROP_AUTHENTICATION_FACTORS = 259,
    PROP_AUTHENTICATION_POLICY_LIST = 260,
    PROP_AUTHENTICATION_POLICY_NAMES = 261,
    PROP_AUTHORIZATION_MODE = 262,
    PROP_BELONGS_TO = 263,
    PROP_CREDENTIAL_DISABLE = 264,
    PROP_CREDENTIAL_STATUS = 265,
    PROP_CREDENTIALS = 266,
    PROP_CREDENTIALS_IN_ZONE = 267,
    PROP_DAYS_REMAINING = 268,
    PROP_ENTRY_POINTS = 269,
    PROP_EXIT_POINTS = 270,
    PROP_EXPIRY_TIME = 271,
    PROP_EXTENDED_TIME_ENABLE = 272,
    PROP_FAILED_ATTEMPT_EVENTS = 273,
    PROP_FAILED_ATTEMPTS = 274,
    PROP_FAILED_ATTEMPTS_TIME = 275,
    PROP_FORMAT_CLASS_SUPPORTED = 276,
    PROP_FORMAT_TYPE = 277,
    PROP_LAST_ACCESS_EVENT = 278,
    PROP_LAST_ACCESS_POINT = 279,
    PROP_LAST_CREDENTIAL_ADDED = 280,
    PROP_LAST_CREDENTIAL_ADDED_TIME = 281,
    PROP_LAST_CREDENTIAL_REMOVED = 282,
    PROP_LAST_CREDENTIAL_REMOVED_TIME = 283,
    PROP_LAST_USE_TIME = 284,
    PROP_LOCKDOWN = 285,
    PROP_LOCKDOWN_RELINQUISH_TIME = 286,
    PROP_MASTER_EXEMPTION = 287,
    PROP_MAX_FAILED_ATTEMPTS = 288,
    PROP_MEMBERS = 289,
    PROP_MASTER_POINT = 290,
    PROP_NUMBER_OF_AUTHENTICATION_POLICIES = 291,
    PROP_OCCUPANCY_COUNT = 293,
    PROP_OCCUPANCY_COUNT_ENABLE = 294,
    PROP_OCCUPANCY_COUNT_EXEMPTION = 295,
    PROP_OCCUPANCY_LOWER_THRESHOLD = 296,
    PROP_OCCUPANCY_LOWER_THRESHOLD_ENFORCED = 297,
    PROP_OCCUPANCY_STATE = 298,
    PROP_OCCUPANCY_UPPER_LIMIT = 299,
    PROP_OCCUPANCY_UPPER_LIMIT_ENFORCED = 300,
    PROP_PASSBACK_EXEMPTION = 301,
    PROP_PASSBACK_MODE = 302,
    PROP_PASSBACK_TIMEOUT = 303,
    PROP_POSITIVE_ACCESS_RULES = 304,
    PROP_READ_STATUS = 305,
    PROP_REASON_FOR_DISABLE = 306,
    PROP_THREAT_AUTHORITY = 307,
    PROP_THREAT_LEVEL = 308,
    PROP_TRACE_FLAG = 309,
    PROP_TRANSACTION_NOTIFICATION_CLASS = 310,
    PROP_USER_EXTERNAL_IDENTIFIER = 311,
    /* enumerations 312-313 are used in Addendum k to ANSI/ASHRAE
135-2004 */
    PROP_CHARACTER_SET = 312,
    PROP_STRICT_CHARACTER_MODE = 313,
    /* enumerations 312-313 are used in Addendum k to ANSI/ASHRAE
135-2004 */
    PROP_BACKUP_AND_RESTORE_STATE = 314,
    PROP_BACKUP_PREPARATION_TIME = 315,
    PROP_RESTORE_PREPARATION_TIME = 316,
    /* enumerations 317-323 are used in Addendum j to ANSI/ASHRAE
135-2004 */
    PROP_USER_INFORMATION_REFERENCE = 317,
    PROP_USER_NAME = 318,
    PROP_USER_TYPE = 319,
    PROP_USES_REMAINING = 320,
    PROP_VENDOR_FORMAT_IDENTIFIER = 321,
    PROP_ZONE_FROM = 322,
    PROP_ZONE_TO = 323,
    /* enumerations 324-325 are used in Addendum i to ANSI/ASHRAE
```

```
135-2004 */
    PROP_BINARY_ACTIVE_VALUE = 324,
    PROP_BINARY_INACTIVE_VALUE = 325
        /* The special property identifiers all, optional, and required
*/
        /* are reserved for use in the ReadPropertyConditional and */
        /* ReadPropertyMultiple services or services not defined in this
standard. */
        /* Enumerated values 0-511 are reserved for definition by
ASHRAE.  */
        /* Enumerated values 512-4194303 may be used by others subject to
the  */
        /* procedures and constraints described in Clause 23.  */
} BACNET_PROPERTY_ID;
```

D.2 BACnet Engineering Units Codes

BACnet engineering units codes may be found in your copy of the BACnet protocol specification, ANSI/ASHRAE Standard 135. That document is copyrighted, but the C enumeration shown below for reference is taken from open source code available under GPL at http://sourceforge.net, and provides essentially the same information (copyrighted by Steve Karg, licensed under GPL as noted at http://sourceforge.net).

```
typedef enum {
    /* Acceleration */
    UNITS_METERS_PER_SECOND_PER_SECOND = 166,
    /* Area */
    UNITS_SQUARE_METERS = 0,
    UNITS_SQUARE_CENTIMETERS = 116,
    UNITS_SQUARE_FEET = 1,
    UNITS_SQUARE_INCHES = 115,
    /* Currency */
    UNITS_CURRENCY1 = 105,
    UNITS_CURRENCY2 = 106,
    UNITS_CURRENCY3 = 107,
    UNITS_CURRENCY4 = 108,
    UNITS_CURRENCY5 = 109,
    UNITS_CURRENCY6 = 110,
    UNITS_CURRENCY7 = 111,
    UNITS_CURRENCY8 = 112,
    UNITS_CURRENCY9 = 113,
    UNITS_CURRENCY10 = 114,
    /* Electrical */
    UNITS_MILLIAMPERES = 2,
    UNITS_AMPERES = 3,
    UNITS_AMPERES_PER_METER = 167,
    UNITS_AMPERES_PER_SQUARE_METER = 168,
    UNITS_AMPERE_SQUARE_METERS = 169,
    UNITS_FARADS = 170,
    UNITS_HENRYS = 171,
    UNITS_OHMS = 4,
    UNITS_OHM_METERS = 172,
    UNITS_MILLIOHMS = 145,
    UNITS_KILOHMS = 122,
    UNITS_MEGOHMS = 123,
    UNITS_SIEMENS = 173, /* 1 mho equals 1 siemens */
    UNITS_SIEMENS_PER_METER = 174,
    UNITS_TESLAS = 175,
    UNITS_VOLTS = 5,
    UNITS_MILLIVOLTS = 124,
    UNITS_KILOVOLTS = 6,
    UNITS_MEGAVOLTS = 7,
    UNITS_VOLT_AMPERES = 8,
    UNITS_KILOVOLT_AMPERES = 9,
    UNITS_MEGAVOLT_AMPERES = 10,
    UNITS_VOLT_AMPERES_REACTIVE = 11,
    UNITS_KILOVOLT_AMPERES_REACTIVE = 12,
    UNITS_MEGAVOLT_AMPERES_REACTIVE = 13,
    UNITS_VOLTS_PER_DEGREE_KELVIN = 176,
    UNITS_VOLTS_PER_METER = 177,
    UNITS_DEGREES_PHASE = 14,
    UNITS_POWER_FACTOR = 15,
    UNITS_WEBERS = 178,
    /* Energy */
    UNITS_JOULES = 16,
    UNITS_KILOJOULES = 17,
    UNITS_KILOJOULES_PER_KILOGRAM = 125,
    UNITS_MEGAJOULES = 126,
    UNITS_WATT_HOURS = 18,
    UNITS_KILOWATT_HOURS = 19,
    UNITS_MEGAWATT_HOURS = 146,
    UNITS_BTUS = 20,
    UNITS_KILO_BTUS = 147,
    UNITS_MEGA_BTUS = 148,
    UNITS_THERMS = 21,
    UNITS_TON_HOURS = 22,
    /* Enthalpy */
    UNITS_JOULES_PER_KILOGRAM_DRY_AIR = 23,
    UNITS_KILOJOULES_PER_KILOGRAM_DRY_AIR = 149,
    UNITS_MEGAJOULES_PER_KILOGRAM_DRY_AIR = 150,
    UNITS_BTUS_PER_POUND_DRY_AIR = 24,
    UNITS_BTUS_PER_POUND = 117,
    /* Entropy */
    UNITS_JOULES_PER_DEGREE_KELVIN = 127,
    UNITS_KILOJOULES_PER_DEGREE_KELVIN = 151,
    UNITS_MEGAJOULES_PER_DEGREE_KELVIN = 152,
    UNITS_JOULES_PER_KILOGRAM_DEGREE_KELVIN = 128,
    /* Force */
    UNITS_NEWTON = 153,
    /* Frequency */
    UNITS_CYCLES_PER_HOUR = 25,
    UNITS_CYCLES_PER_MINUTE = 26,
    UNITS_HERTZ = 27,
    UNITS_KILOHERTZ = 129,
    UNITS_MEGAHERTZ = 130,
    UNITS_PER_HOUR = 131,
    /* Humidity */
    UNITS_GRAMS_OF_WATER_PER_KILOGRAM_DRY_AIR = 28,
    UNITS_PERCENT_RELATIVE_HUMIDITY = 29,
    /* Length */
    UNITS_MILLIMETERS = 30,
    UNITS_CENTIMETERS = 118,
    UNITS_METERS = 31,
    UNITS_INCHES = 32,
    UNITS_FEET = 33,
    /* Light */
    UNITS_CANDELAS = 179,
    UNITS_CANDELAS_PER_SQUARE_METER = 180,
    UNITS_WATTS_PER_SQUARE_FOOT = 34,
    UNITS_WATTS_PER_SQUARE_METER = 35,
    UNITS_LUMENS = 36,
    UNITS_LUXES = 37,
    UNITS_FOOT_CANDLES = 38,
    /* Mass */
    UNITS_KILOGRAMS = 39,
    UNITS_POUNDS_MASS = 40,
    UNITS_TONS = 41,
    /* Mass Flow */
    UNITS_GRAMS_PER_SECOND = 154,
    UNITS_GRAMS_PER_MINUTE = 155,
    UNITS_KILOGRAMS_PER_SECOND = 42,
    UNITS_KILOGRAMS_PER_MINUTE = 43,
    UNITS_KILOGRAMS_PER_HOUR = 44,
    UNITS_POUNDS_MASS_PER_SECOND = 119,
    UNITS_POUNDS_MASS_PER_MINUTE = 45,
    UNITS_POUNDS_MASS_PER_HOUR = 46,
    UNITS_TONS_PER_HOUR = 156,
    /* Power */
    UNITS_MILLIWATTS = 132,
    UNITS_WATTS = 47,
    UNITS_KILOWATTS = 48,
    UNITS_MEGAWATTS = 49,
    UNITS_BTUS_PER_HOUR = 50,
    UNITS_KILO_BTUS_PER_HOUR = 157,
    UNITS_HORSEPOWER = 51,
    UNITS_TONS_REFRIGERATION = 52,
    /* Pressure */
    UNITS_PASCALS = 53,
    UNITS_HECTOPASCALS = 133,
    UNITS_KILOPASCALS = 54,
    UNITS_MILLIBARS = 134,
    UNITS_BARS = 55,
    UNITS_POUNDS_FORCE_PER_SQUARE_INCH = 56,
    UNITS_CENTIMETERS_OF_WATER = 57,
    UNITS_INCHES_OF_WATER = 58,
    UNITS_MILLIMETERS_OF_MERCURY = 59,
    UNITS_CENTIMETERS_OF_MERCURY = 60,
    UNITS_INCHES_OF_MERCURY = 61,
    /* Temperature */
    UNITS_DEGREES_CELSIUS = 62,
    UNITS_DEGREES_KELVIN = 63,
    UNITS_DEGREES_KELVIN_PER_HOUR = 181,
    UNITS_DEGREES_KELVIN_PER_MINUTE = 182,
    UNITS_DEGREES_FAHRENHEIT = 64,
    UNITS_DEGREE_DAYS_CELSIUS = 65,
    UNITS_DEGREE_DAYS_FAHRENHEIT = 66,
    UNITS_DELTA_DEGREES_FAHRENHEIT = 120,
    UNITS_DELTA_DEGREES_KELVIN = 121,
    /* Time */
    UNITS_YEARS = 67,
    UNITS_MONTHS = 68,
    UNITS_WEEKS = 69,
    UNITS_DAYS = 70,
    UNITS_HOURS = 71,
    UNITS_MINUTES = 72,
    UNITS_SECONDS = 73,
    UNITS_HUNDREDTHS_SECONDS = 158,
    UNITS_MILLISECONDS = 159,
    /* Torque */
    UNITS_NEWTON_METERS = 160,
    /* Velocity */
    UNITS_MILLIMETERS_PER_SECOND = 161,
    UNITS_MILLIMETERS_PER_MINUTE = 162,
    UNITS_METERS_PER_SECOND = 74,
    UNITS_METERS_PER_MINUTE = 163,
    UNITS_METERS_PER_HOUR = 164,
    UNITS_KILOMETERS_PER_HOUR = 75,
    UNITS_FEET_PER_SECOND = 76,
    UNITS_FEET_PER_MINUTE = 77,
    UNITS_MILES_PER_HOUR = 78,
    /* Volume */
    UNITS_CUBIC_FEET = 79,
    UNITS_CUBIC_METERS = 80,
    UNITS_IMPERIAL_GALLONS = 81,
    UNITS_LITERS = 82,
    UNITS_US_GALLONS = 83,
    /* Volumetric Flow */
    UNITS_CUBIC_FEET_PER_SECOND = 142,
    UNITS_CUBIC_FEET_PER_MINUTE = 84,
    UNITS_CUBIC_METERS_PER_SECOND = 85,
    UNITS_CUBIC_METERS_PER_MINUTE = 165,
    UNITS_CUBIC_METERS_PER_HOUR = 135,
    UNITS_IMPERIAL_GALLONS_PER_MINUTE = 86,
    UNITS_LITERS_PER_SECOND = 87,
    UNITS_LITERS_PER_MINUTE = 88,
    UNITS_LITERS_PER_HOUR = 136,
    UNITS_US_GALLONS_PER_MINUTE = 89,
    /* Other */
    UNITS_DEGREES_ANGULAR = 90,
    UNITS_DEGREES_CELSIUS_PER_HOUR = 91,
    UNITS_DEGREES_CELSIUS_PER_MINUTE = 92,
    UNITS_DEGREES_FAHRENHEIT_PER_HOUR = 93,
    UNITS_DEGREES_FAHRENHEIT_PER_MINUTE = 94,
    UNITS_JOULE_SECONDS = 183,
    UNITS_KILOGRAMS_PER_CUBIC_METER = 186,
    UNITS_KW_HOURS_PER_SQUARE_METER = 137,
    UNITS_KW_HOURS_PER_SQUARE_FOOT = 138,
    UNITS_MEGAJOULES_PER_SQUARE_METER = 139,
    UNITS_MEGAJOULES_PER_SQUARE_FOOT = 140,
    UNITS_NO_UNITS = 95,
    UNITS_NEWTON_SECONDS = 187,
    UNITS_NEWTONS_PER_METER = 188,
    UNITS_PARTS_PER_MILLION = 96,
    UNITS_PARTS_PER_BILLION = 97,
    UNITS_PERCENT = 98,
    UNITS_PERCENT_OBSCURATION_PER_FOOT = 143,
    UNITS_PERCENT_OBSCURATION_PER_METER = 144,
    UNITS_PERCENT_PER_SECOND = 99,
    UNITS_PER_MINUTE = 100,
    UNITS_PER_SECOND = 101,
    UNITS_PSI_PER_DEGREE_FAHRENHEIT = 102,
    UNITS_RADIANS = 103,
    UNITS_RADIANS_PER_SECOND = 184,
    UNITS_REVOLUTIONS_PER_MINUTE = 104,
    UNITS_SQUARE_METERS_PER_NEWTON = 185,
    UNITS_WATTS_PER_METER_PER_DEGREE_KELVIN = 189,
    UNITS_WATTS_PER_SQUARE_METER_DEGREE_KELVIN = 141,
       ; /* Enumerated values 0-255 are reserved for definition by
ASHRAE. */
        /* Enumerated values 256-65535 may be used by others subject to
*/
        /* the procedures and constraints described in Clause 23. */
        /* The last enumeration used in this version is 189. */
    MAX_UNITS = 190
} BACNET_ENGINEERING_UNITS;
```

Appendix E Modbus Reference Information

E.1 Function Codes, Error Codes, and More

Modbus Register Types

The types of registers referenced in Modbus devices include the following:
- Coil (Discrete Output)
- Discrete Input
- Input Register
- Holding Register

Whether a particular device includes all of these register types is up to the manufacturer. It is very common to find all I/O mapped to holding registers only. Coils are 1-bit registers, are used to control discrete outputs, and may be read or written. Discrete Inputs are 1-bit registers used as inputs, and may only be read. Input registers are 16-bit registers used for input, and may only be read. Holding registers are the most universal 16-bit register, may be read or written, and may be used for a variety of things including inputs, outputs, configuration data, or any requirement for "holding" data.

Modbus Function Codes

Modbus protocol defines several function codes for accessing Modbus registers. There are four different data blocks defined by Modbus, and the addresses or register numbers in each of those overlap. Therefore, a complete definition of where to find a piece of data requires both the address (or register number) and function code (or register type).

The function codes most commonly recognized by Modbus devices are indicated in the table below. This is only a subset of the codes available - several of the codes have special applications that most often do not apply.

Function Code	Register Type
1	Read Coil
2	Read Discrete Input
3	Read Holding Registers
4	Read Input Registers
5	Write Single Coil
6	Write Single Holding Register

15	Write Multiple Coils
16	Write Multiple Holding Registers

Modbus Exception (error) Codes

When a Modbus slave recognizes a packet, but determines that there is an error in the request, it will return an exception code reply instead of a data reply. The exception reply consists of the slave address or unit number, a copy of the function code with the high bit set, and an exception code. If the function code was 3, for example, the function code in the exception reply will be 0x83. The exception codes will be one of the following:

1	Illegal Function	The function code received in the query is not recognized by the slave or is not allowed by the slave.
2	Illegal Data Address	The data address (register number) received in the query is not an allowed address for the slave, i.e., the register does not exist. If multiple registers were requested, at least one was not permitted.
3	Illegal Data Value	The value contained in the query's data field is not acceptable to the slave.
4	Slave Device Failure	An unrecoverable error occurred.
6	Slave Device Busy	The slave is engaged in processing a long-duration command. The master should try again later.
10 (hex 0A)	Gateway Path Unavailable	Gateway could not establish communication with target device.
11 (hex 0B)	Gateway Target Device Failed to Respond	Specialized use in conjunction with gateways, indicates no response was received from the target device.
17 (hex 11)	Gateway Target Device Failed to Respond	No response from slave, request timed out.

Modicon convention notation for Modbus registers

Modbus was originally developed by Gould-Modicon, which is presently Schneider Electric. The notation originally used by Modicon is still often used today, even though considered obsolete by present Modbus standards. The advantage in using the Modicon notation is that two pieces of information are included in a single number: (a) The register type; (b) The register number. A register number offset defines the type.

The types of registers referenced in Modbus devices, and supported by Babel Buster gateways, include the following:
- Coil (Discrete Output)
- Discrete Input
- Input Register
- Holding Register

Valid address ranges as originally defined for Modbus were 0 to 9999 for each of the above register types. Valid ranges allowed in the current specification are 0 to 65,535. The address range applies to each type of register, and one needs to look at the function code in the Modbus message packet to determine what register type is being referenced. The Modicon convention uses the first digit of a register reference to identify the register type.

Register types and reference ranges recognized by Babel Buster gateways are as follows:

0x = Coil = 00001-09999
1x = Discrete Input = 10001-19999
3x = Input Register = 30001-39999
4x = Holding Register = 40001-49999

Translating references to addresses, reference 40001 selects the holding register at address 0000, most often referred to as holding register number 1. The reference 40001 will appear in documentation using Modicon notation, but Babel Buster gateways require specifying "holding register" and entering that register number as just "1".

On occasion, it was necessary to access more than 10,000 of a register type using Modicon notation. Based on the original convention, there is another defacto standard that looks very similar. Additional register types and reference ranges recognized by Babel Buster gateways are as follows:

0x = Coil = 000001-065535
1x = Discrete Input = 100001-165535
3x = Input Register = 300001-365535
4x = Holding Register = 400001-465535

If registers are 16-bits, how does one read Floating Point or 32-bit data?

Modbus protocol defines a holding register as 16 bits wide; however, there is a widely used defacto standard for reading and writing data wider than 16 bits. The most common are IEEE 754 floating point, and 32-bit integer. The convention may also be extended to double precision floating point and 64-bit integer data.

The wide data simply consists of two consecutive "registers" treated as a single wide register. Floating point in 32-bit IEEE 754 standard, and 32-bit integer data, are widely used. Although the convention of register pairs is widely recognized, agreement on whether the high order or low order register should come first is not standardized. For this reason, many devices, including all Control Solutions gateways, support register "swapping". This means you simply check the "swapped" option (aka "High reg first" in some devices) if the other device treats wide data in the opposite order relative to Control Solutions default order.

Control Solutions Modbus products all default to placing the high order register first, or in the lower numbered register. This is known as "big endian", and is consistent with Modbus protocol which is by definition big endian.

What does notation like 40001:7 mean?

This is a commonly used notation for referencing individual bits in a register. This particular example, 40001:7, references (Modicon) register 40001, bit 7. Bits are generally numbered starting at bit 0, which is the least significant or right most bit in the field of 16 bits found in a Modbus register.

How do I read individual bits in a register?

Documentation tends to be slightly different for every Modbus device. But if your device packs multiple bits into a single holding register, the documentation will note up to 16 different items found at the same register number or address. The bits may be identified with “Bn” or “Dn” or just “bit n”. Most of the time, the least significant bit will be called bit 0 and the most significant will be bit 15. It is possible you could find reference to bit 1 through bit 16, in which case just subtract one from the number to reference the table below.

You cannot read just one bit from a holding register. There is no way to do that - Modbus protocol simply does not provide that function. You must read all 16 bits, and then test the individual bit you are interested in for true or false (1 or 0). Babel Buster gateways provide an automatic way of doing that by including a "mask" in each register map or rule. Each time the register is read, the mask will be logically AND-ed with the data from the register, and the result will be right justified to yield a 1 or 0 based on whether the selected bit was 1 or 0. Babel Buster gateways provide optimization when successive read maps or rules are selecting different bits from the same register. The Modbus register will be read from the slave once, and the 16-bit data will be shared with successive maps or rules, with each map or rule selecting its bit of interest.

The bit mask shown in the expanded form of the Babel Buster RTU read map is a 4 digit hexadecimal (16 bit) value used to mask out one or more bits in a register. The selected bits will be right justified, so a single bit regardless of where positioned in the source register will be stored locally as 0 or 1. The hex bit mask values would be as follows:

B0/D0/bit 0 mask = 0001
B1/D1/bit 1 mask = 0002
B2/D2/bit 2 mask = 0004
B3/D3/bit 3 mask = 0008
B4/D4/bit 4 mask = 0010
B5/D5/bit 5 mask = 0020
B6/D6/bit 6 mask = 0040
B7/D7/bit 7 mask = 0080
B8/D8/bit 8 mask = 0100
B9/D9/bit 9 mask = 0200
B10/D10/bit 10 mask = 0400
B11/D11/bit 11 mask = 0800
B12/D12/bit 12 mask = 1000
B13/D13/bit 13 mask = 2000
B14/D14/bit 14 mask = 4000

B15/D15/bit 15 mask = 8000

Some Modbus devices also back two 8-bit values into a single 16-bit register. The two values will typically be documented as "high byte" and "low byte" or simply have "H" and "L" indicated. If you run into this scenario, the masking for bytes is as follows:

High byte mask = FF00
Low byte mask = 00FF

When the mask value in a Babel Buster gateway is more than just one bit, the mask is still logicalle AND-ed with the data from the Modbus slave, and the entire resulting value is right justified to produce an integer value of less than the original bit width of the original register.

There have been a few instances of documenting packed bits in a 32-bit register. Although Modbus protocol is strictly 16-bit registers, some implementations force you to read pairs of registers. If your device documents 32 packed bits, then you would insert 0000 in front of each mask above, and the remainder of the list would be as follows:

B16/D16/bit 16 mask = 00010000
B17/D17/bit 17 mask = 00020000
B18/D18/bit 18 mask = 00040000
B19/D19/bit 19 mask = 00080000
B20/D20/bit 20 mask = 00100000
B21/D21/bit 21 mask = 00200000
B22/D22/bit 22 mask = 00400000
B23/D23/bit 23 mask = 00800000
B24/D24/bit 24 mask = 01000000
B25/D25/bit 25 mask = 02000000
B26/D26/bit 26 mask = 04000000
B27/D27/bit 27 mask = 08000000
B28/D28/bit 28 mask = 10000000
B29/D29/bit 29 mask = 20000000
B30/D30/bit 30 mask = 40000000
B31/D31/bit 31 mask = 80000000

Deciphering Modbus Documentation

Documentation for Modbus is not well standardized. Actually there is a standard, but not well followed when it comes to documentation. You will have to do one or more of the following to decipher which register a manufacturer is really referring to:

a) Look for the register description, such as holding register, coil, etc. If the documentation says #1, and tells you they are holding registers, then you have holding register #1. You also have user friendly documentation.

b) Look at the numbers themselves. If you see the first register on the list having a number 40001, that really tells you register #1, and it is a holding register. This form of notation is often referred to as the old Modicon convention.

c) Look for a definition of function codes to be used. If you see a register #1, along with notation telling you to use function codes 3 and 16, that also tells you it is holding register #1.

IMPORTANT: Register 1 is address 0. Read on…

d) Do the numbers in your documentation refer to the register number or address? Register #1 is address zero. If it is not clear whether your documentation refers to register or address, and you are not getting the expected result, try plus or minus one for register number. All Control Solutions products refer to register numbers in configuration software or web pages. However, some manufacturers document their devices showing address, not register numbers. When you have addresses, you must add one when entering that register into configuration software from Control Solutions.

Can I put 2 gateways on the same Modbus network?

You can not have more than one Master on a Modbus RTU (RS-485) network. Therefore, if the gateway is to be configured as the Master, you can only have 1 gateway. You cannot use multiple gateways to read more points from the same Modbus slave device.

Multiple gateways configured as slaves can reside on the same Modbus RS-485 network.

If you are using RS-232 devices, you can have only two devices total, regardless of how they are configured. RS-232 is not multi-drop.

How many devices can I have on a Modbus RTU network?

Logically you can address over 250 devices; however, the RS-485 transceivers are not capable of physically driving that many devices. Modbus protocol states that the limit is 32 devices, and most RS-485 transceivers will agree with this. Only if all devices on the network have low load transceivers can you have more than 32 devices.

Babel Buster® 3
MODEL BB3-7101-SP
MODEL MX-71-SP
NETWORK GATEWAY

CONTROL SOLUTIONS MINNESOTA

Appendix F Using Wireshark for Trouble Shooting

F.1 Hardware Requirements

There are no particular hardware requirements regarding the PC you run Wireshark on. Basically anything running any version of Windows can run Wireshark. There are also Linux and Mac versions.

The "hardware requirement" that is of most concern is the means of connecting to the network. We typically just connect everything Ethernet to a switch and don't worry about it. However, switches are really unmanaged routers, and they filter traffic. Therefore, your PC will not see traffic passing back and forth between two other devices that are not the PC. In order to see that network traffic using Wireshark, you need to come up with the right kind of network connection.

If your PC itself is one end of the network conversation you wish to capture, for example when running the Network Discovery Tool, then Wireshark will capture all network traffic to and from the PC however connected. It is when your PC wants to simply "eavesdrop" that you run into problems with the network switch.

A while back, 10BaseT hubs were common. A 10BaseT hub is not as smart as a switch and does not filter traffic. If you have an old 10BaseT hub collecting dust somewhere, you now have a new use for it. It will let Wireshark see all traffic from the PC that goes between any other devices connected to that 10BaseT hub. Beware of devices that call themselves "hubs" but support 100BaseT connections. These are switches.

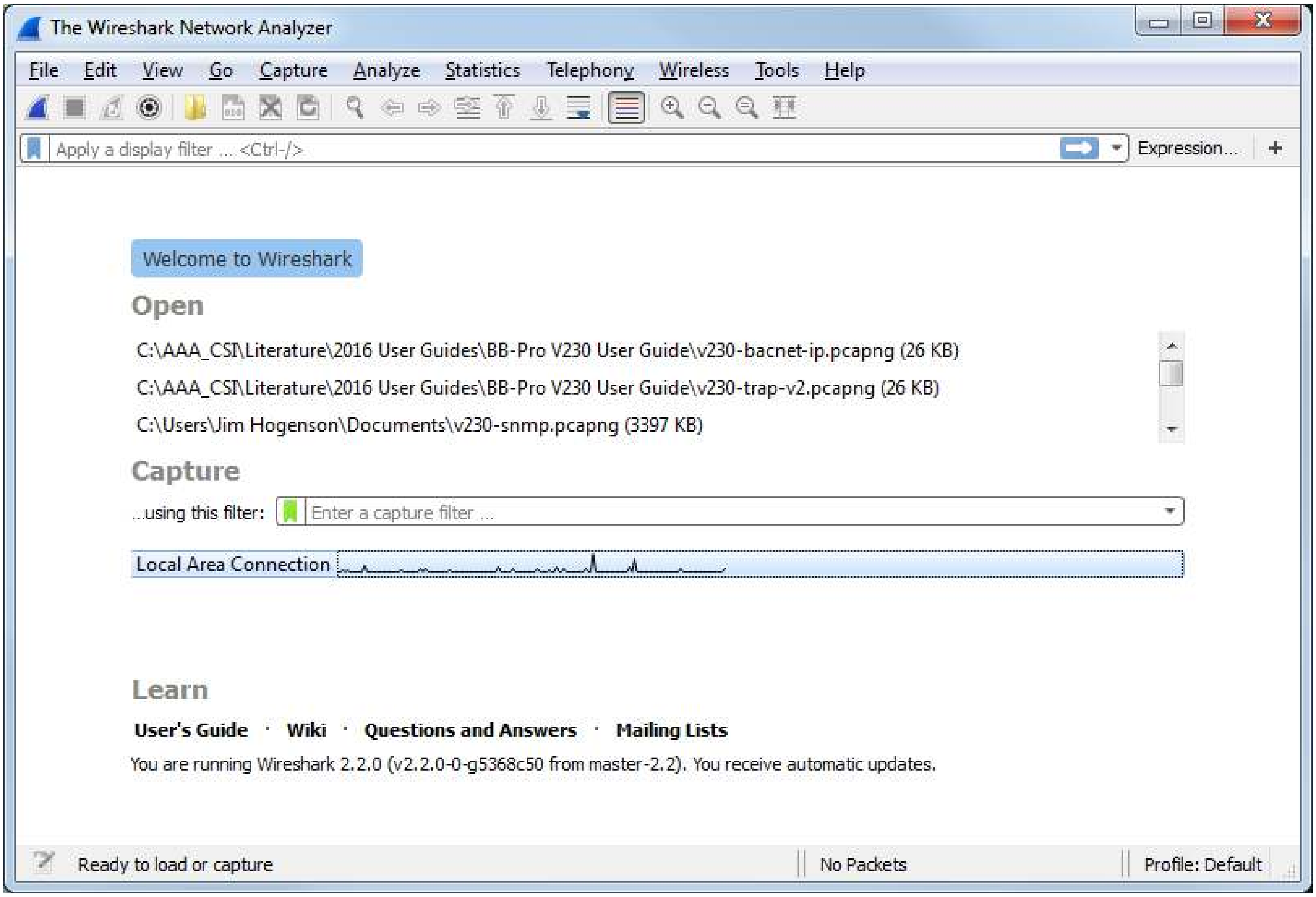
Since manufacturers of hubs decided nobody should have a use for them anymore, they are generally out of production. Finding a 10BaseT hub for sale is not easy (try eBay). But there are other alternatives.

One means of monitoring network traffic is to get a managed switch that supports "port mirroring". One such device we have tested is the TP-LINK model TL-SG105E. Setting it up requires utility software (provided with the switch) and takes a little effort to get configured. But once configured, it works well without any further monkeying around. And it is inexpensive.

The other means of monitoring traffic is with the use of a device made specifically for use with Wireshark. The "SharkTap" provides two connections for the network pass-through, and a third "tap" connection where you connect your PC running Wireshark. There is no configuration required. It is the simplest way to monitor network traffic, and it is a current production item available on Amazon (as of 2020).

F.2 Example of Using Wireshark

Using Wireshark is fairly easy. Get a copy at www.wireshark.org and install it. Once installed, running it is straight forward. As of version 2.2.0 of Wireshark, the startup screen looks like the following. Double click on Local Area Connection to start capturing network traffic on your PC's Ethernet port. If you have multiple network connections, they will all be listed. Be sure to select the one that represents your Ethernet connection, typically "Local Area Connection".


The Wireshark Network Analyzer
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Expression...
+
Welcome to Wireshark
Open

C:\AAA_CSI\Literature\2016 User Guides\BB-Pro V230 User Guide\v230-bacnet-ip.pcapng (26 KB)
C:\AAA_CSI\Literature\2016 User Guides\BB-Pro V230 User Guide\v230-trap-v2.pcapng (26 KB)
C:\Users\Jim Hogenson\Documents\v230-snmp.pcapng (3397 KB)

Capture
...using this filter:
Enter a capture filter ...
Local Area Connection
Learn
User's Guide · Wiki · Questions and Answers · Mailing Lists

You are running Wireshark 2.2.0 (v2.2.0-0-g5368c50 from master-2.2). You receive automatic updates.

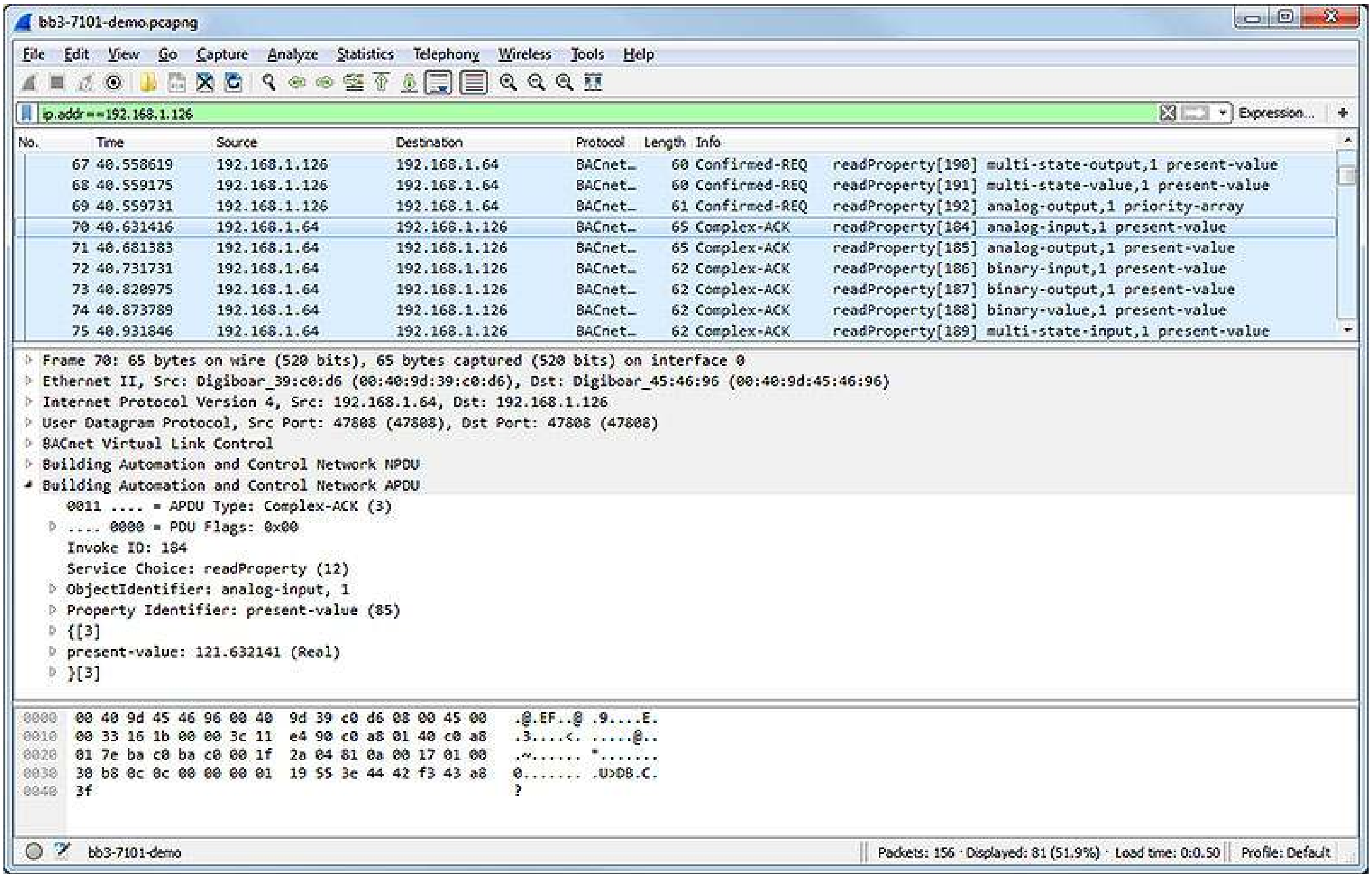
Ready to load or capture | No Packets | Profile: Default

The screen will look something like the example below once Wireshark starts collecting data. Click the red icon in the toolbar to stop capturing traffic. Control Solutions technical support will often ask for a copy of the Wireshark data when a network issue seems evident. You can save a copy of all of the network traffic captured under the File menu, and you will generally save it to a .pcap or .pcapng file. A Wireshark log with .pcap extension can be posted directly as an attachment in support tickets while .pcapng needs to be zipped first.

The screen shot below shows Wireshark capturing BACnet IP traffic between the BB3-7101/MX-71 and another BACnet device. If you click on a packet, the details of that packet will be displayed in the lower part of the screen. You can expand the tree view to see further detail.

A lot of times you will see a lot of network traffic that is not of interest to you. You can filter network traffic to only display traffic to/from the device you are interested in. Do this by entering "ip.addr==192.168.1.126" in the Filter window as illustrated below. (Substitute your own device's IP address.)

The example illustrated here is a Complex-ACK, or in other words, reply to a Read Property request.



ip.addr==192.168.1.126

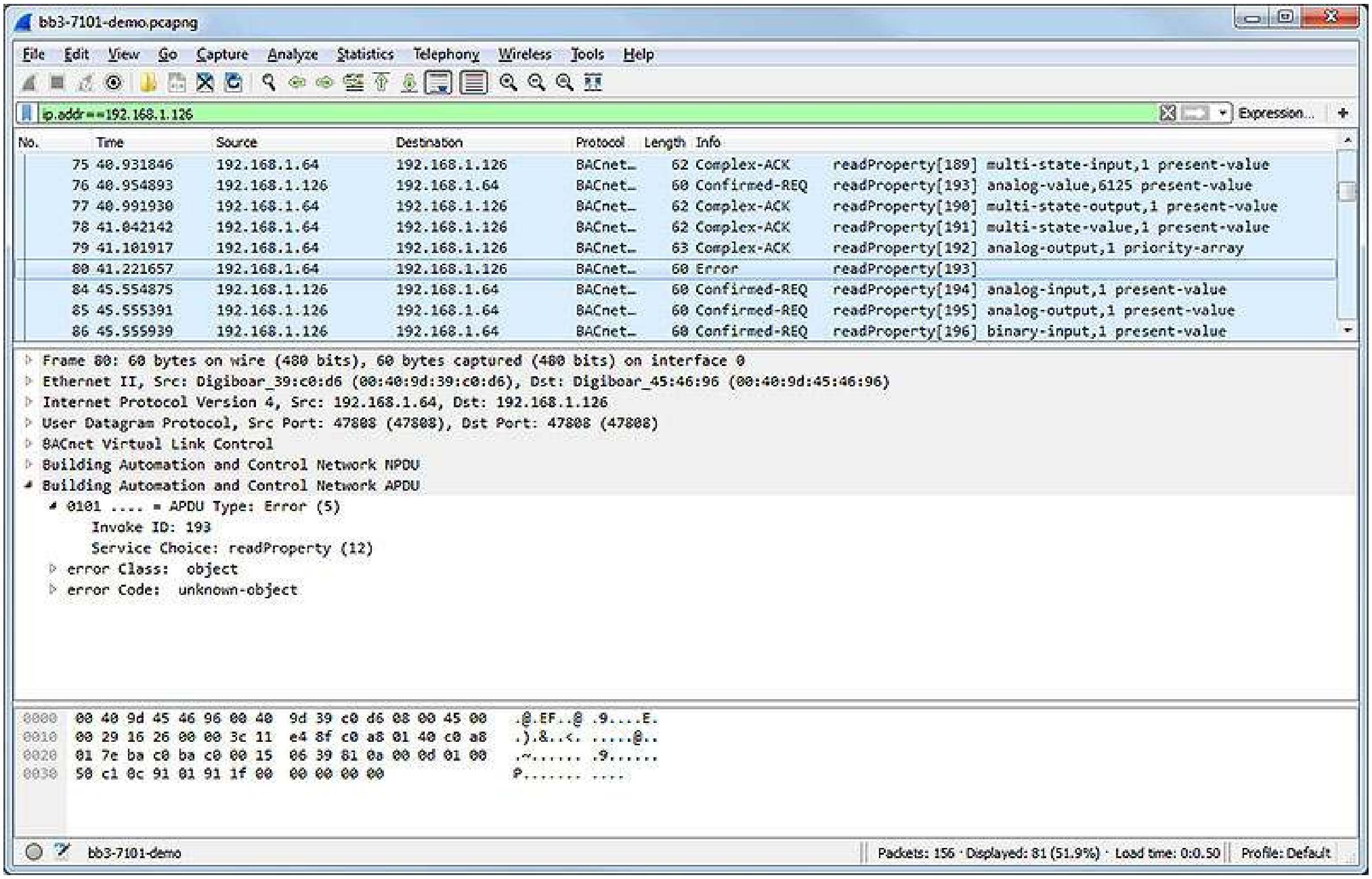
```
No.     Time         Source          Destination     Protocol  Length Info
     67 40.558619    192.168.1.126   192.168.1.64    BACnet…   60 Confirmed-REQ   readProperty[190] multi-state-output,1 present-value
     68 40.559175    192.168.1.126   192.168.1.64    BACnet…   60 Confirmed-REQ   readProperty[191] multi-state-value,1 present-value
     69 40.559731    192.168.1.126   192.168.1.64    BACnet…   61 Confirmed-REQ   readProperty[192] analog-output,1 priority-array
     70 40.631416    192.168.1.64    192.168.1.126   BACnet…   65 Complex-ACK     readProperty[184] analog-input,1 present-value
     71 40.681383    192.168.1.64    192.168.1.126   BACnet…   65 Complex-ACK     readProperty[185] analog-output,1 present-value
     72 40.731731    192.168.1.64    192.168.1.126   BACnet…   62 Complex-ACK     readProperty[186] binary-input,1 present-value
     73 40.820975    192.168.1.64    192.168.1.126   BACnet…   62 Complex-ACK     readProperty[187] binary-output,1 present-value
     74 40.873789    192.168.1.64    192.168.1.126   BACnet…   62 Complex-ACK     readProperty[188] binary-value,1 present-value
     75 40.931846    192.168.1.64    192.168.1.126   BACnet…   62 Complex-ACK     readProperty[189] multi-state-input,1 present-value
```

```
▷ Frame 70: 65 bytes on wire (520 bits), 65 bytes captured (520 bits) on interface 0
▷ Ethernet II, Src: Digiboar_39:c0:d6 (00:40:9d:39:c0:d6), Dst: Digiboar_45:46:96 (00:40:9d:45:46:96)
▷ Internet Protocol Version 4, Src: 192.168.1.64, Dst: 192.168.1.126
▷ User Datagram Protocol, Src Port: 47808 (47808), Dst Port: 47808 (47808)
▷ BACnet Virtual Link Control
▷ Building Automation and Control Network NPDU
◢ Building Automation and Control Network APDU
    0011 .... = APDU Type: Complex-ACK (3)
  ▷ .... 0000 = PDU Flags: 0x00
    Invoke ID: 184
    Service Choice: readProperty (12)
  ▷ ObjectIdentifier: analog-input, 1
  ▷ Property Identifier: present-value (85)
  ▷ {[3]
  ▷ present-value: 121.632141 (Real)
  ▷ }[3]
```

```
0000  00 40 9d 45 46 96 00 40  9d 39 c0 d6 08 00 45 00   .@.EF..@ .9....E.
0010  00 33 16 1b 00 00 3c 11  e4 90 c0 a8 01 40 c0 a8   .3....<. .....@..
0020  01 7e ba c0 ba c0 00 1f  2a 04 81 0a 00 17 01 00   .~...... *.......
0030  30 b8 0c 0c 00 00 00 01  19 55 3e 44 42 f3 43 a8   0....... .U>DB.C.
0040  3f                                                 ?
```

bb3-7101-demo   Packets: 156 · Displayed: 81 (51.9%) · Load time: 0:0.50   Profile: Default

This next example shows what an error reply will look like. This error resulted in the example screen shot in Section 5.4 of this User Guide.


bb3-7101-demo.pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Expression...

```
ip.addr==192.168.1.126
```

```
No.   Time        Source          Destination     Protocol  Length Info
   75 40.931846   192.168.1.64    192.168.1.126   BACnet…   62 Complex-ACK    readProperty[189] multi-state-input,1 present-value
   76 40.954893   192.168.1.126   192.168.1.64    BACnet…   60 Confirmed-REQ  readProperty[193] analog-value,6125 present-value
   77 40.991930   192.168.1.64    192.168.1.126   BACnet…   62 Complex-ACK    readProperty[190] multi-state-output,1 present-value
   78 41.042142   192.168.1.64    192.168.1.126   BACnet…   62 Complex-ACK    readProperty[191] multi-state-value,1 present-value
   79 41.101917   192.168.1.64    192.168.1.126   BACnet…   63 Complex-ACK    readProperty[192] analog-output,1 priority-array
   80 41.221657   192.168.1.64    192.168.1.126   BACnet…   60 Error          readProperty[193]
   84 45.554875   192.168.1.126   192.168.1.64    BACnet…   60 Confirmed-REQ  readProperty[194] analog-input,1 present-value
   85 45.555391   192.168.1.126   192.168.1.64    BACnet…   60 Confirmed-REQ  readProperty[195] analog-output,1 present-value
   86 45.555939   192.168.1.126   192.168.1.64    BACnet…   60 Confirmed-REQ  readProperty[196] binary-input,1 present-value
```

```
▷ Frame 80: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
▷ Ethernet II, Src: Digiboar_39:c0:d6 (00:40:9d:39:c0:d6), Dst: Digiboar_45:46:96 (00:40:9d:45:46:96)
▷ Internet Protocol Version 4, Src: 192.168.1.64, Dst: 192.168.1.126
▷ User Datagram Protocol, Src Port: 47808 (47808), Dst Port: 47808 (47808)
▷ BACnet Virtual Link Control
▷ Building Automation and Control Network NPDU
◢ Building Automation and Control Network APDU
   ◢ 0101 .... = APDU Type: Error (5)
        Invoke ID: 193
        Service Choice: readProperty (12)
   ▷ error Class:  object
   ▷ error Code:  unknown-object
```

```
0000  00 40 9d 45 46 96 00 40  9d 39 c0 d6 08 00 45 00   .@.EF..@ .9....E.
0010  00 29 16 26 00 00 3c 11  e4 8f c0 a8 01 40 c0 a8   .).&..<. .....@..
0020  01 7e ba c0 ba c0 00 15  06 39 81 0a 00 0d 01 00   .~...... .9......
0030  50 c1 0c 91 01 91 1f 00  00 00 00 00              P....... ....
```

```
bb3-7101-demo    Packets: 156 · Displayed: 81 (51.9%) · Load time: 0:0.50    Profile: Default
```

Babel Buster® 3
MODEL BB3-7101-SP
MODEL MX-71-SP
NETWORK GATEWAY

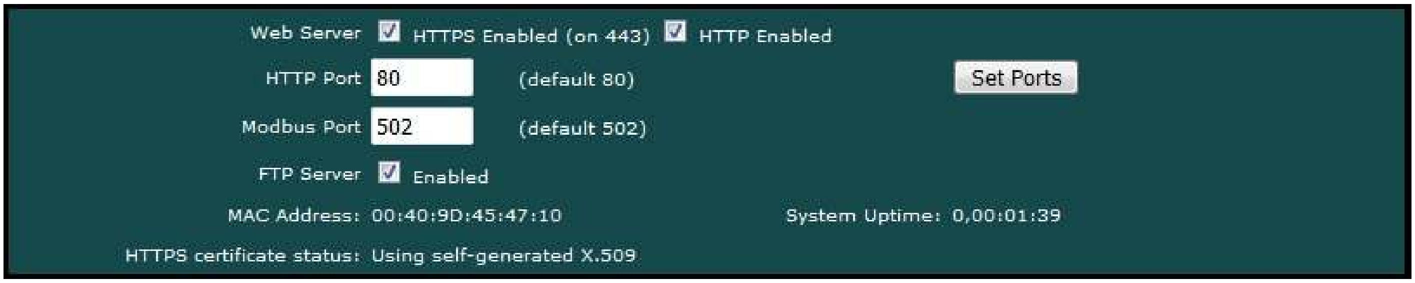
CONTROL SOLUTIONS MINNESOTA

Appendix G SSL Certificates for Secure Web (HTTPS)

The secure web server (HTTPS) requires SSL certificates in order to establish secure connections. The HTTPS certificates are only required if HTTPS is enabled on the Network configuration page in the Babel Buster BB3-7101/MX-71.

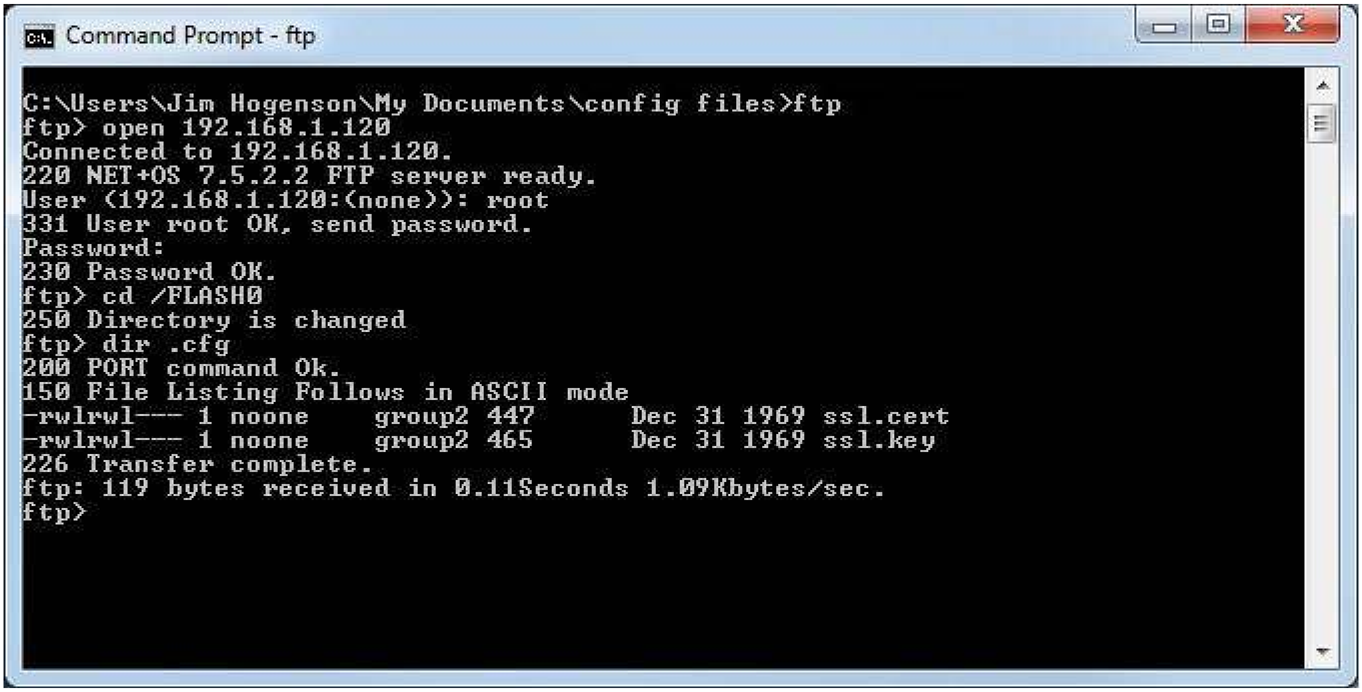
G.1 X.509 Auto-Certificate Generation

The Babel Buster BB3-7101/MX-71 Gateway will automatically generate X.509 certificates if no external certificates are found or could not be loaded correctly. These will be generated one time and saved in the Flash file system for subsequent reuse. When the self-generated X.509 certificates are in use, this will be indicated at the bottom of the Network configuration page.



Web Server ☑ HTTPS Enabled (on 443) ☑ HTTP Enabled
HTTP Port 80 (default 80)
Modbus Port 502 (default 502)
FTP Server ☑ Enabled
MAC Address: 00:40:9D:45:47:10 System Uptime: 0,00:01:39
HTTPS certificate status: Using self-generated X.509

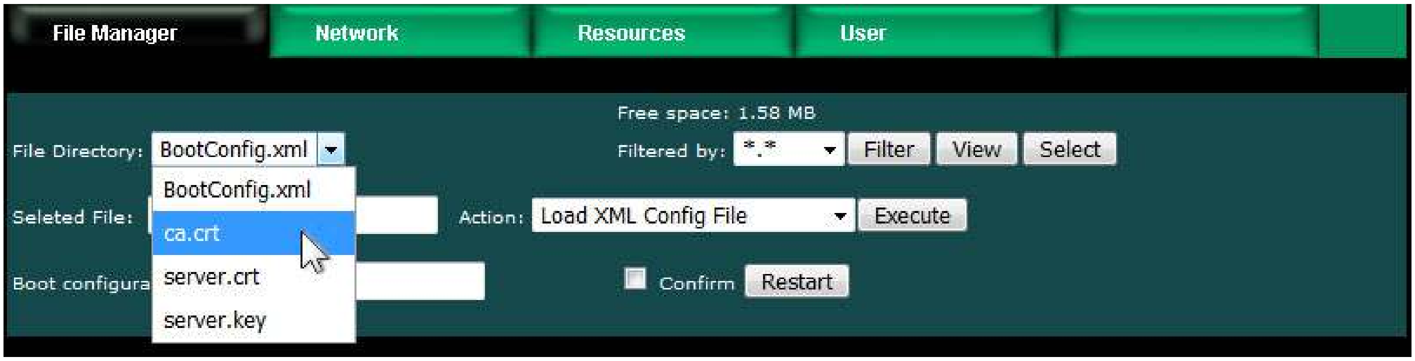
If there is a need to delete the self-generated certificates, you can do so by logging in via FTP. Change directory to /FLASH0, then to .cfg. The two certificate files that were self-generated are ssl.cert and ssl.key.



```
C:\Users\Jim Hogenson\My Documents\config files>ftp
ftp> open 192.168.1.120
Connected to 192.168.1.120.
220 NET+OS 7.5.2.2 FTP server ready.
User (192.168.1.120:(none)): root
331 User root OK, send password.
Password:
230 Password OK.
ftp> cd /FLASH0
250 Directory is changed
ftp> dir .cfg
200 PORT command Ok.
150 File Listing Follows in ASCII mode
-rwlrwl--- 1 noone    group2 447      Dec 31 1969 ssl.cert
-rwlrwl--- 1 noone    group2 465      Dec 31 1969 ssl.key
226 Transfer complete.
ftp: 119 bytes received in 0.11Seconds 1.09Kbytes/sec.
ftp>
```

G.2 External Certificates

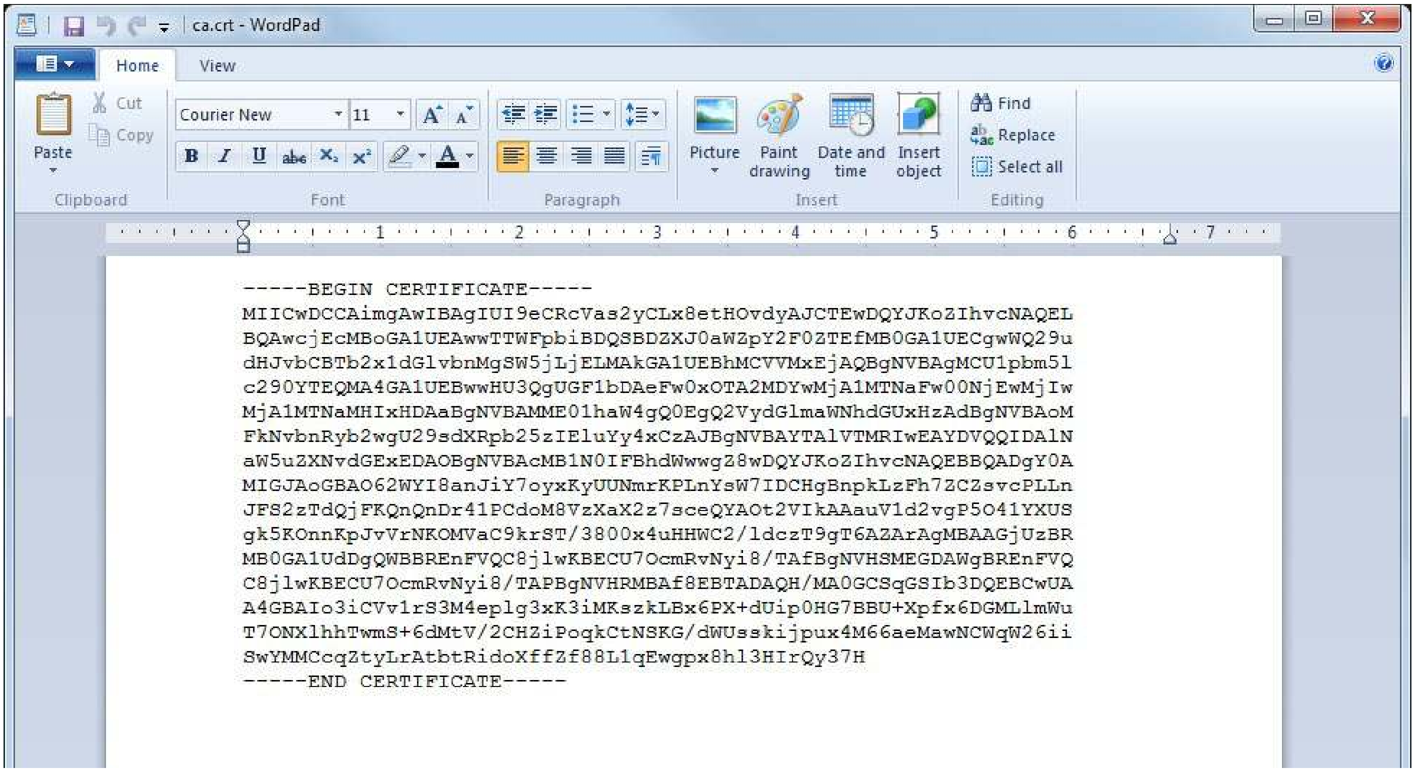
There are three certificates that you must generate and upload to use SSL certificates other than the self-generated X.509 certificates.



The required certificates are as follows, and must use exactly these names.

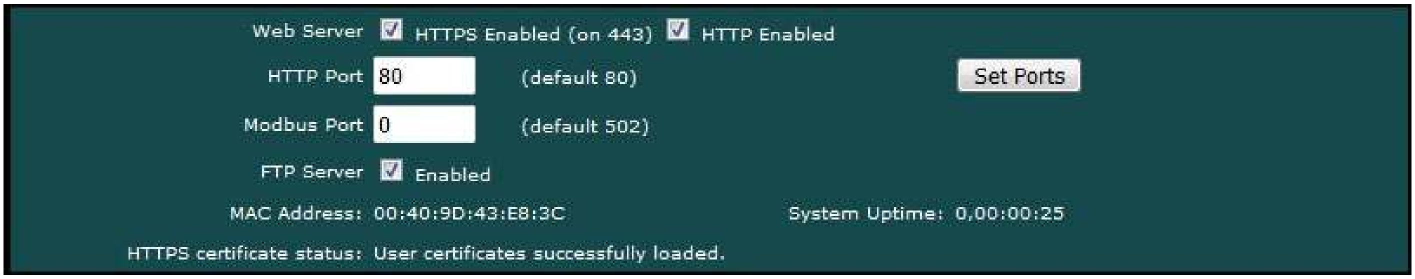
ca.crt	CA Root certificate in PEM format
server.crt	Server certificate in PEM format
server.key	Server private key in PEM format

The content of each certificate file will look something like the screen shot below. If you require external certificates for your secure web server, the requirement was likely imposed by your IT department. They should be able to provide the necessary certificates for you. For globally accessed use, the Root CA would come from somebody like GoDaddy or DigiCert (formerly Symantec).



```
-----BEGIN CERTIFICATE-----
MIICwDCCAimgAwIBAgIUI9eCRcVas2yCLx8etHOvdyAJCTEwDQYJKoZIhvcNAQEL
BQAwcjEcMBoGA1UEAwwTTWFpbiBDQSBDZXJ0aWZpY2F0ZTEfMB0GA1UECgwWQ29u
dHJvbCBTb2x1dGlvbnMgSW5jLjELMAkGA1UEBhMCVVMxEjAQBgNVBAgMCU1pbm5l
c290YTEQMA4GA1UEBwwHU3QgUGF1bDAeFw0xOTA2MDYwMjA1MTNaFw00NjEwMjIw
MjA1MTNaMHIxHDAaBgNVBAMME01haW4gQ0EgQ2VydGlmaWNhdGUxHzAdBgNVBAoM
FkNvbnRyb2wgU29sdXRpb25zIEluYy4xCzAJBgNVBAYTAlVTMRIwEAYDVQQIDAlN
aW5uZXNvdGExEDAOBgNVBAcMB1N0IFBhdWwwgZ8wDQYJKoZIhvcNAQEBBQADgY0A
MIGJAoGBAO62WYI8anJiY7oyxKyUUNmrKPLnYsW7IDCHgBnpkLzFh7ZCZsvcPLLn
JFS2zTdQjFKQnQnDr41PCdoM8VzXaX2z7sceQYAOt2VIkAAauV1d2vgP5O41YXUS
gk5KOnnKpJvVrNKOMVaC9krST/3800x4uHHWC2/ldczT9gT6AZArAgMBAAGjUzBR
MB0GA1UdDgQWBBREnFVQC8jlwKBECU7OcmRvNyi8/TAfBgNVHSMEGDAWgBREnFVQ
C8jlwKBECU7OcmRvNyi8/TAPBgNVHRMBAf8EBTADAQH/MA0GCSqGSIb3DQEBCwUA
A4GBAIo3iCVv1rS3M4eplg3xK3iMKszkLBx6PX+dUip0HG7BBU+Xpfx6DGMLlmWu
T7ONXlhhTwmS+6dMtV/2CHZiPoqkCtNSKG/dWUsskijpux4M66aeMawNCWqW26ii
SwYMMCcqZtyLrAtbtRidoXffZf88L1qEwgpx8hl3HIrQy37H
-----END CERTIFICATE-----
```

If external certificates were loaded successfully, that will be indicated at the bottom of the Network configuration page.



Web Server ☑ HTTPS Enabled (on 443) ☑ HTTP Enabled

HTTP Port 80 (default 80) Set Ports

Modbus Port 0 (default 502)

FTP Server ☑ Enabled

MAC Address: 00:40:9D:43:E8:3C System Uptime: 0,00:00:25

HTTPS certificate status: User certificates successfully loaded.

G.3 Certificate Generation Script (Linux)

The art and science of generating SSL certificates is beyond the scope of this document. An example SSL certificate generation script is provided here as a reference.

The following script, run on a Linux system with OpenSSL installed, will generate the three required SSL certificate files. It will generate a number of intermediate files as well - you don't need to upload them. Replace references to Control Solutions in this script with your own company name.

```
#!/bin/bash
echo hello
# This will create some self signed certs, using one master CA.
#
# these can be the webserver DNS name, or an IP address, however you
access
# the resource, this needs to match.
```

```
if [ -z "$1"] || [ -z "$2"]; then
echo 'Usage: gen.sh <server-name> <client-name>'
echo ' <server-name> and <client-name> can be IP addresses'
echo ' or DNS names.'
exit 1
fi
SNAME=$1
CNAME=$2
#
# Bits for strength, 1024, 2048, 4096, etc.. (suggest 2k or 4k for web
servers)
BITS=1024
#
# HASH - Options are sha256, sha512, sha1, md5
HASH="sha256"
SN=`date +%Y%m%d%H%M%S`
################
# below is the entry for the CRL
# Do not use http://www.csimn.com/crl.pem for production keys and
certificates
# cat <<EOF >> extensions.cnf
# [ extensions_section ]
# crlDistributionPoints = URI:http://www.csimn.com/crl.pem
#
# basicConstraints = CA:FALSE
# keyUsage = nonRepudiation, digitalSignature, keyEncipherment
# subjectAltName = DNS:${SNAME},IP:${SNAME}
# EOF
#################################################################
#################################################################
# first, lets generate some private keys...
openssl genrsa -out server.key ${BITS}
openssl genrsa -out client.key ${BITS}
# ok, and now the MAIN CA
openssl req -x509 -${HASH} -nodes -days 10000 -newkey rsa:${BITS} -keyout
ca.key -out ca.crt -subj "/CN=Main CA Certificate/O=Control Solutions
Inc./C=US/ST=Minnesota/L=St Paul"
######
#
# Create a CSR for both server and client
# Replace these values with one appropriate for your organization
openssl req -out server.csr -key server.key -new -subj "/CN=${SNAME}
/O=Control Solutions Inc./C=US/ST=Minnesota/L=St Paul"
openssl req -out client.csr -key client.key -new -subj "/CN=${CNAME}
/O=Control Solutions Inc./C=US/ST=Minnesota/L=St Paul"
#
#
######
# Sign the keys with the CA
openssl x509 -req -days 3650 -in server.csr -CA ca.crt -CAkey ca.key
-set_serial ${SN}01 -out server.crt -${HASH}
openssl x509 -req -days 3650 -in client.csr -CA ca.crt -CAkey ca.key
-set_serial ${SN}02 -out client.crt -${HASH}
```

```
# Create a windows file to import the client keys if needed in this
format
openssl pkcs12 -export -clcerts -in client.crt -inkey client.key -out
client.p12
# Create the client keys as a complete pem file if needed in this format
openssl pkcs12 -in client.p12 -out client-full.pem -clcerts
# mv -f server.key svrkey.pem
# mv -f server.crt svrcert.pem
# mv -f client.key clntkey.pem
# mv -f client.crt clntcert.pem
# cp -f ca.crt cacert.pem
####
# cleanup
# rm -f client.csr server.csr
#DLS 20160420
echo '***********************************************************'
echo '* WARNING: Do not use this script to generate production *'
echo '* keys and certificates. This script is for *'
echo '* demonstration purposes only. *'
echo '***********************************************************'
```

Babel Buster® 3
MODEL BB3-7101-SP
MODEL MX-71-SP
NETWORK GATEWAY

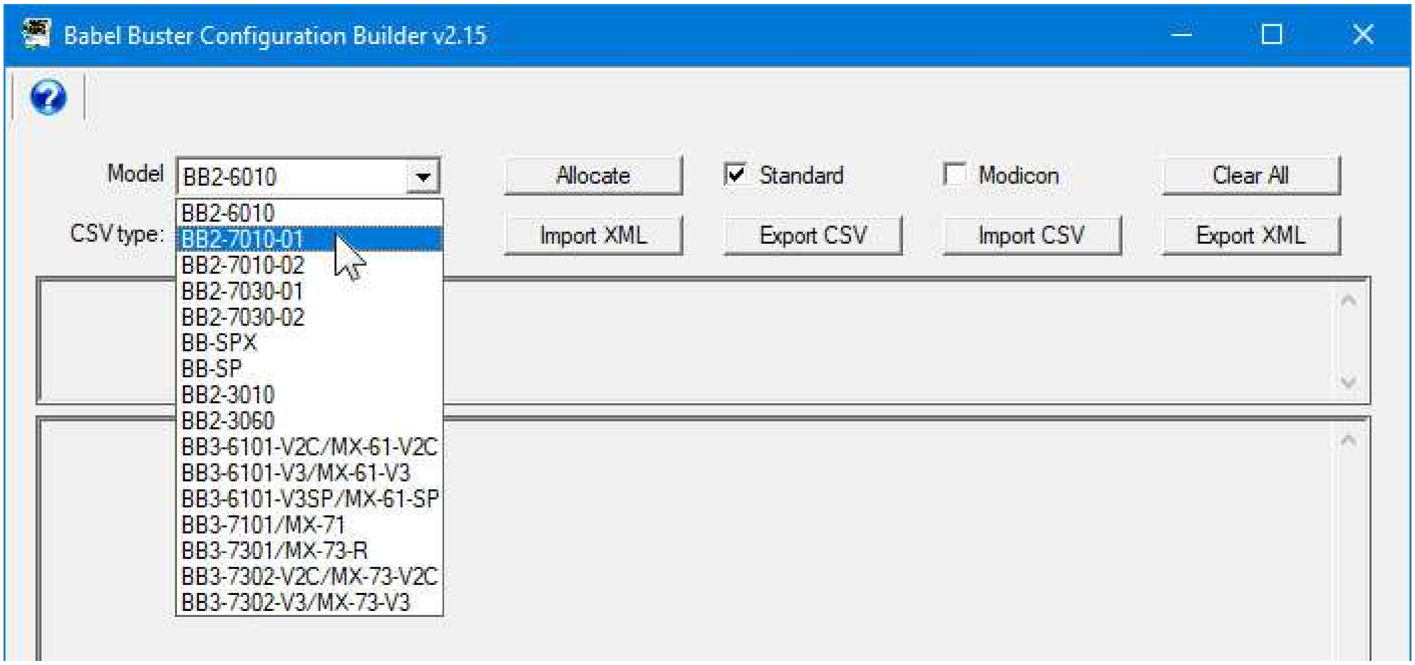
CONTROL SOLUTIONS MINNESOTA

Appendix H Converting Older XML Files

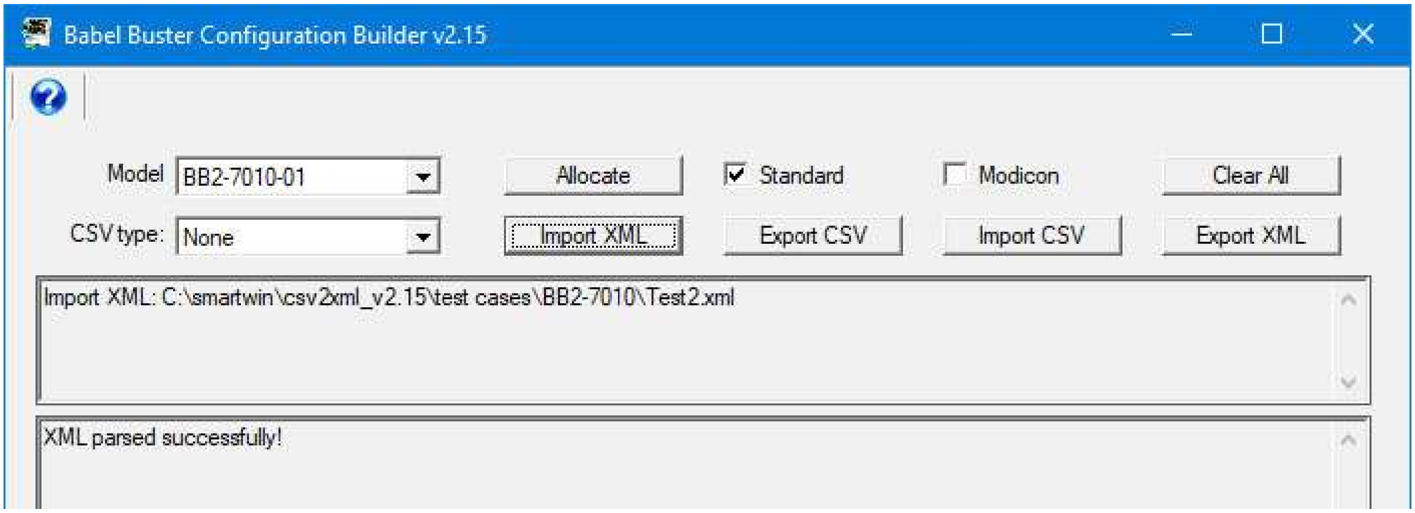
H.1 BB2-7010 to BB3-7101/MX-71 Conversion

You cannot load a BB2-7010 configuration XML file into the BB3-7101/MX-71, but you can easily convert it into a BB3-7101/MX-71 XML file using the Babel Buster Configuration Builder tool available under the Support/Tools link at csimn.com.

Converting the file is simple. Start by selecting the BB2-7010 model.



Click the Import XML button. The Windows file dialog window will open. Select your XML file from wherever you saved it on your PC. Upon opening the XML file, it will read the file into the configuration builder tool.



Model BB2-7010-01

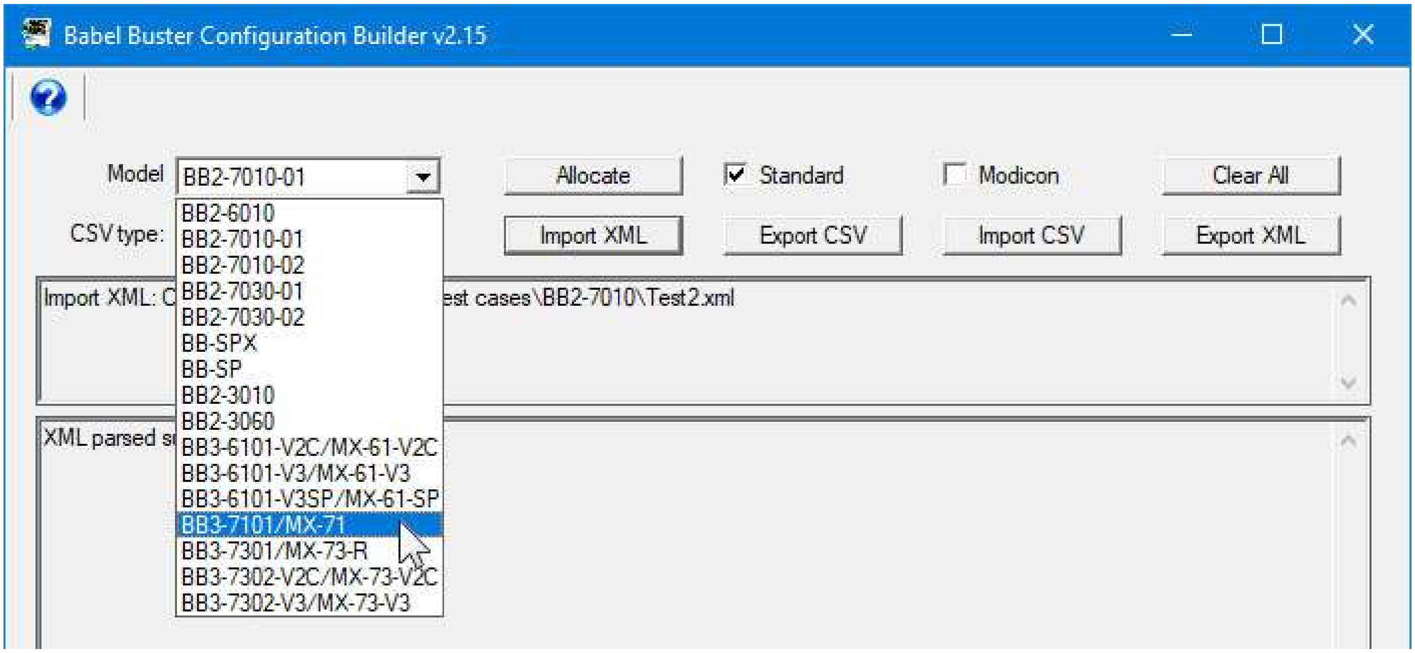
CSV type: None

☑ Standard ☐ Modicon

Import XML: C:\smartwin\csv2xml_v2.15\test cases\BB2-7010\Test2.xml

XML parsed successfully!

Now go back to the model list and switch to BB3-7101/MX-71.



Model BB2-7010-01

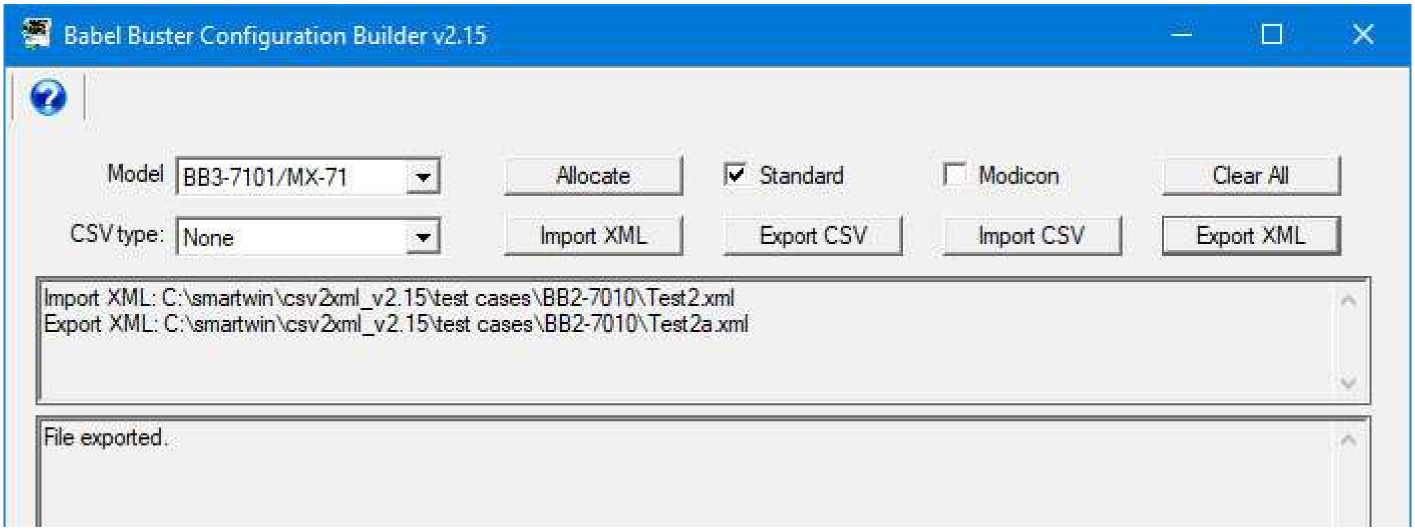
CSV type:

☑ Standard ☐ Modicon

Import XML: C ... est cases\BB2-7010\Test2.xml

XML parsed s

Click the Export XML button. The Windows file dialog window will open again, this time wanting you to provide a name for the new XML file you are about to create. Choose a different name than the one you imported so that you don't lose your original file. Once the export is complete, you may now upload this new file to your BB3-7101/MX-71 gateway.



Model BB3-7101/MX-71

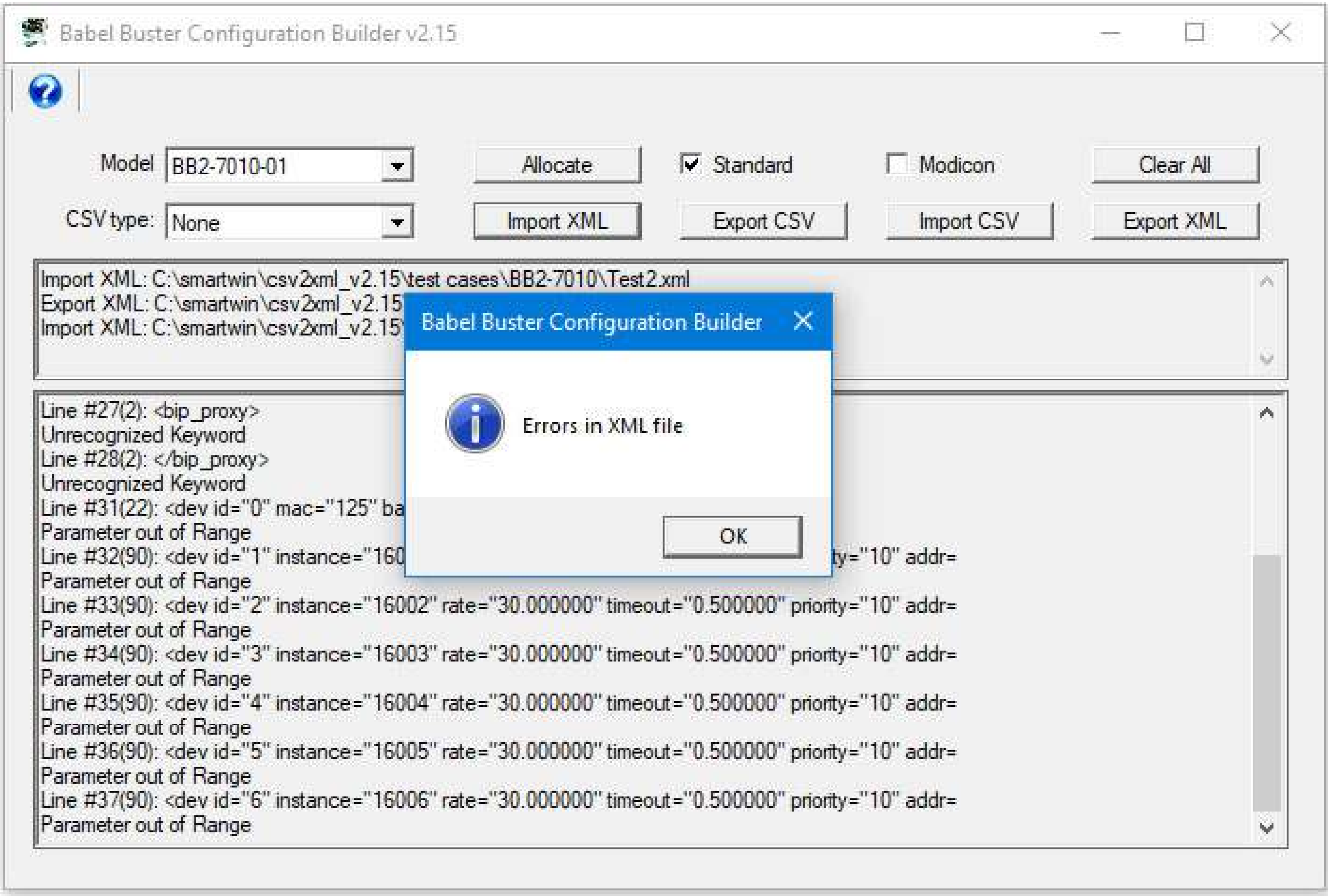
CSV type: None

☑ Standard ☐ Modicon

```
Import XML: C:\smartwin\csv2xml_v2.15\test cases\BB2-7010\Test2.xml
Export XML: C:\smartwin\csv2xml_v2.15\test cases\BB2-7010\Test2a.xml
```

```
File exported.
```

If things to terribly wrong, your screen may look something like the one below instead of the good examples above. If you are importing a BACnet related XML file and see "Invalid object...", the first thing to check is to click the Allocate button and be sure you do have objects allocated that are sufficient for the file being imported.


Babel Buster Configuration Builder v2.15
Allocate
Clear All
Import XML
Export CSV
Import CSV
Export XML
Babel Buster Configuration Builder
OK

Model BB2-7010-01

CSV type: None

☑ Standard ☐ Modicon

```
Import XML: C:\smartwin\csv2xml_v2.15\test cases\BB2-7010\Test2.xml
Export XML: C:\smartwin\csv2xml_v2.15
Import XML: C:\smartwin\csv2xml_v2.15
```

Errors in XML file

```
Line #27(2): <bip_proxy>
Unrecognized Keyword
Line #28(2): </bip_proxy>
Unrecognized Keyword
Line #31(22): <dev id="0" mac="125" ba
Parameter out of Range
Line #32(90): <dev id="1" instance="160                                    ty="10" addr=
Parameter out of Range
Line #33(90): <dev id="2" instance="16002" rate="30.000000" timeout="0.500000" priority="10" addr=
Parameter out of Range
Line #34(90): <dev id="3" instance="16003" rate="30.000000" timeout="0.500000" priority="10" addr=
Parameter out of Range
Line #35(90): <dev id="4" instance="16004" rate="30.000000" timeout="0.500000" priority="10" addr=
Parameter out of Range
Line #36(90): <dev id="5" instance="16005" rate="30.000000" timeout="0.500000" priority="10" addr=
Parameter out of Range
Line #37(90): <dev id="6" instance="16006" rate="30.000000" timeout="0.500000" priority="10" addr=
Parameter out of Range
```

H.2 CSV File Export

The BB3-7101/MX-71 will not export CSV files directly. To create CSV files from an existing BB3-7101/MX-71 configuration, download a copy of the file to your PC, and then use the Babel Buster Configuration Builder to export CSV files as needed. To save a copy of the XML configuration file to your PC, select the file and click the View button on the File Manager page in the BB3-7101/MX-71. Your browser will now display the XML file. **DO NOT** do a text copy/paste to try to create an XML file - doing so will result in an invalid file format that cannot be loaded again. You must use the browser's "save as" or "save page" function. The browser should default to wanting to save a file with a .xml suffix. If correctly saved on your PC, you should be able to double click on the saved file and it will result in opening the file automatically in your browser. It was saved correctly if the browser does not give any error messages when displaying the XML (which should now look exactly as it did when you first clicked the View button).

The Babel Buster Configuration Builder will export more types of CSV files than the BB3-7101/MX-71 itself will import. The most frequently used CSV files - namely read/write maps for Modbus RTU, Modbus TCP, and BACnet - can be imported directly into the BB3-7101/MX-71. You do not generally have to import any Objects CSV file to start using objects - the objects are created automatically when the Resources page is updated. The only time you might be concerned about importing objects is if you have a long list of unique object names. In this case, import the CSV file into the configuration builder tool, and export as XML and then upload the XML file instead.

Importing device lists directly into the BB3-7101/MX-71 is also not supported. You generally only need to enter things like IP address once, and any one IP address can only be used one time. The effort to enter the IP addresses into the web UI of the BB3-7101/MX-71 is no greater than the effort to enter them into a spread sheet and then save and import. In fact, when it comes to setting up device lists, the CSV import would usually end up being more work. If you do have a device list in the form of a CSV file, import that into the configuration builder tool, export as XML (along with any other CSV files you import as part of the configuration), and then upload the resulting XML file to the BB3-7101/MX-71.