



# User Guide

## Babel Buster 3

**Model BB3-6101-V3SP**  
**Model MX-61-SP**

**Modbus-SNMP Gateway  
 with SNMPv3  
 and Proprietary Protocol  
 Script Basic Programming  
 Rev. 1.2 – June 2021**

© 2021 Control Solutions, Inc.

## User Guide Contents

### [1 Introduction](#)

- 1.1 How to Use This Guide
- 1.2 Important Safety Notice
- 1.3 Warranty

### [2 Connecting Gateway for the First Time](#)

- 2.1 Where to Start
- 2.2 Overview of Model BB3-6101/MX-61
  - 2.2.1 Application of the BB3-6101/MX-61
  - 2.2.2 Object Server Model for a Gateway
- 2.3 What is New in Model BB3-6101/MX-61
- 2.4 Connectors and Indicators
- 2.5 Opening the Web User Interface

### [3 System Configuration and Resources](#)

- 3.1 Using the File Manager
  - 3.1.1 Load, Save, Create XML Configuration File
  - 3.1.2 Select Startup Configuration
  - 3.1.3 Delete a File
  - 3.1.4 Import CSV File
  - 3.1.5 Clear Configuration
- 3.2 Configuration Files and Restoring Default Settings
- 3.3 Network Configuration
  - 3.3.1 IPv4, IPv6 Settings
  - 3.3.2 NTP Time Server Settings
  - 3.3.3 Port Settings
- 3.4 Resource Allocation
- 3.5 User Login Passwords

### [4 Configuring Local Registers](#)

- 4.1 Creating Local Registers
- 4.2 Special Features of Local Registers
- 4.3 Local Register Calculate Rules
- 4.4 Local Register Copy Rules
- 4.5 Device Status Reporting

### [5 Programming with Script Basic](#)

- 5.1 Creating a Program
- 5.2 Testing the Program
- 5.3 Setting the Program to Auto-Run on Startup
- 5.4 Serial Port Functions
- 5.5 Example: Data Logger Capture
  - 5.5.1 Program Code - Capture
- 5.6 Example: Querying Serial Device
  - 5.6.1 Program Code - Query
  - 5.6.2 Program Code - Reply

### [6 Configuring Gateway as a Modbus TCP Client](#)

- 6.1 Modbus TCP Device Configuration
- 6.2 Modbus TCP Client Read Maps
- 6.3 Modbus TCP Client Write Maps
- 6.4 Modbus TCP Client Data Displayed by Server
- 6.5 Modbus TCP Errors

### [7 Configuring Gateway as a Modbus TCP Server](#)

- 7.1 Modbus TCP Device Configuration

- 7.2 Modbus TCP Server Register Map
- 7.3 Modbus TCP Server Diagnostic
- 7.4 Modbus TCP Server Register Remapping

## [8 Configuring Gateway as an SNMP Server](#)

- 8.1 Creating Local SNMP MIB
- 8.2 Supported Data Formats, RFC 6340
- 8.3 SNMPv3 Users, Authentication, Privacy
- 8.4 Agent ID
- 8.5 SNMPv3 Engine Info
- 8.6 SNMPv2 Community
- 8.7 Testing the SNMP Agent

## [9 Configuring SNMP Trap Sender](#)

- 9.1 SNMP Trap Destinations
- 9.2 SNMP Trap Triggers
- 9.3 SNMP Trap Summary
- 9.4 Testing the SNMP Trap Sender

## [Appendix A Hardware Details](#)

- A.1 Wiring
- A.2 Front Panel LED Indicators
- A.3 RS-485 Line Termination and Bias
- A.4 Soft Configuration Reset
- A.5 Discovering Lost IP Address
- A.6 Forced Hard Configuration Reset
- A.7 Firmware Update Notes

## [Appendix B Modbus CSV Import Files](#)

- B.1 Modbus TCP Client Read/Write Maps
- B.2 Register Types
- B.3 Register Formats

## [Appendix C Modbus Reference Information](#)

- C.1 Function Codes, Error Codes, and More

## [Appendix D Trouble Shooting](#)

- D.1 Modbus TCP Trouble Shooting
- D.2 SNMP Trouble Shooting
- D.3 Wireshark Hardware Requirements
- D.4 Example of Using Wireshark

## [Appendix E SSL Certificates for Secure Web \(HTTPS\)](#)

- E.1 X.509 Auto-Certificate Generation
- E.2 External Certificates
- E.3 Certificate Generation Script (Linux)

## [Appendix F Converting Older XML Files](#)

- F.1 BB2-6010/SPX to BB3-6101/MX-61 Conversion
- F.2 CSV File Export



# 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 BB3-6101-V3SP/MX-61-SP is used to share data between proprietary serial devices, Modbus TCP devices, and SNMP. 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 BB3-6101.

You may need to obtain information such as a "register map" from the vendor of the Modbus device you wish to connect, if applicable. 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 SNMP network.

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 BB3-6101; however, the screens in the MX-61 are identical with the only exception being model number indicated at the top of the page.

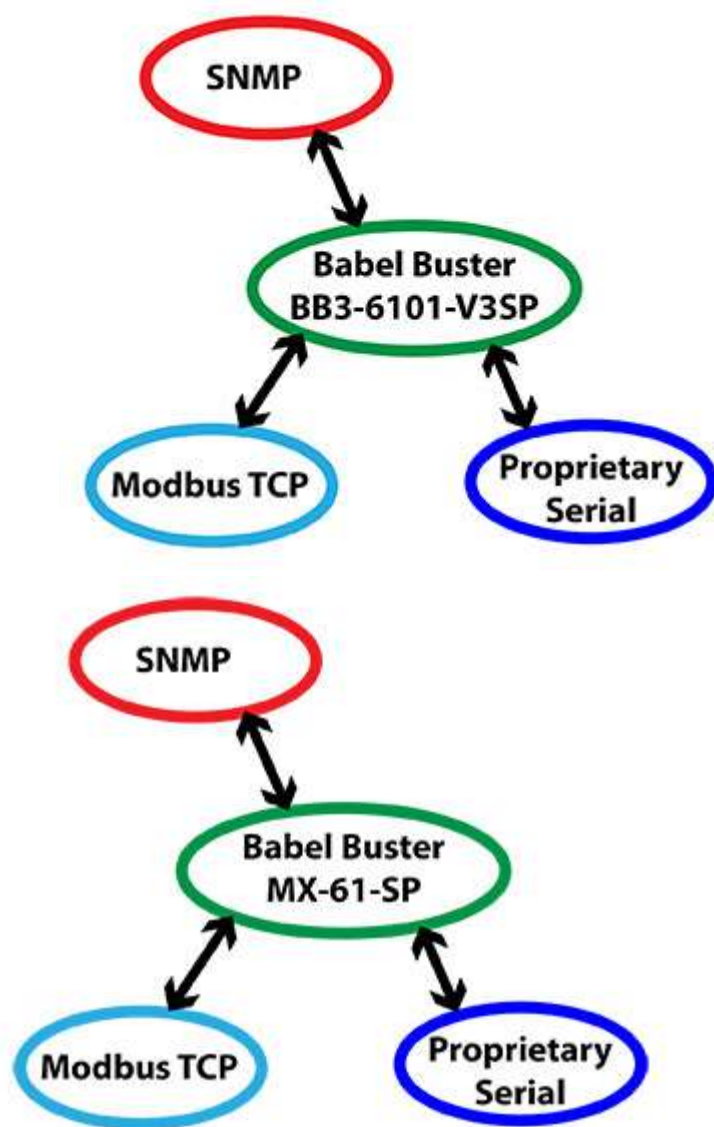
### 2.2 Overview of Model BB3-6101/MX-61

#### 2.2.1 Application of the BB3-6101/MX-61

The Babel Buster BB3-6101/MX-61 is a gateway used to interface between SNMP and Modbus, and proprietary serial devices. The gateway may be used as an SNMP Agent or Modbus TCP client and server. Beyond that, the serial port will function according to however you program it using the built-in Script Basic.

The BB3-6101/MX-61 may be thought of as a data server with multiple network ports that have access to the internal database. The BB3-6101/MX-61 is not a router - SNMP cannot interact directly with Modbus devices or vice versa. Both SNMP and Modbus have access to the internal data objects which are updated according to rules you define. Your Script Basic program also has access to the internal data objects, referencing them as "registers".

The BB3-6101/MX-61 can be thought of as a collection of sensors and actuators. The means by which the sensors and actuators interface to hardware will vary by application. In the case of the BB3-6101/MX-61, that hardware consists of your proprietary serial device, or one or more Modbus devices which the BB3-6101/MX-61 is set up to automatically interact with, without any constraints on the SNMP side.



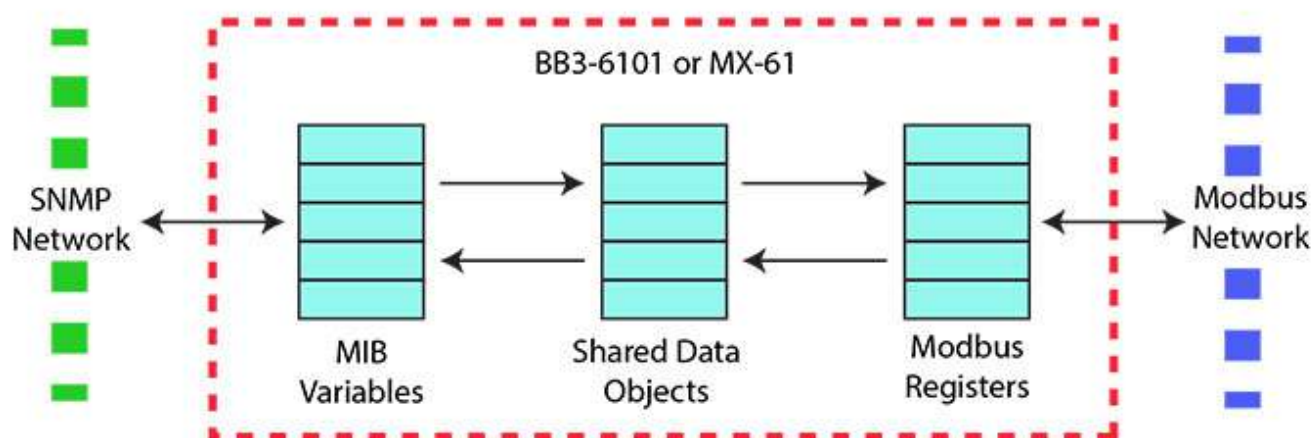
The intended application for the BB3-6101-V3SP is interfacing a proprietary serial device to the SNMP management system. The BB3-6101/MX-61 will interact with the serial device according to however you program it, and store data provided by your program in local registers you assign. The SNMP manager may then use the standard SNMP GET to access the content of those data points.

### 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 BB3-6101/MX-61 will have it functioning as a Modbus master (client).

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.

NOTE: The above diagram and discussion are general descriptions relevant to the more common Modbus application of the BB3-6101/MX-61. Where proprietary serial protocols are involved, they would either replace the "Modbus Network" in the diagram above, or become a third network if both proprietary serial and Modbus TCP (plus SNMP) are all used at the same time.

## 2.3 What is New in Model BB3-6101/MX-61

The BB3-6101 is a significant enhancement over its predecessor, the BB2-6010. The MX-61 is the equivalent enhancement over its predecessor, the SPX. The hardware includes a faster processor and hardware encryption engine for efficient rendering of

secure web pages and for support of encryption as needed for SNMPv3. The software includes numerous enhancements.

- SNMPv3 support
- IPv6 support
- Secure (HTTPS) web server
- Faster processor
- Higher point count, up to 1,000 MIB variables
- User defined register map for Modbus
- Greater flexibility in assigning local register data types
- Support for reading character strings from Modbus
- CSV import of register maps for client/master configuration
- Menu options to clear part or all of configuration

Control Solutions has benchmark tested a configuration in which the BB3-6101 MIB size was 1,500 variables and external SNMP manager tools could successfully walk the entire MIB in a very short time.

**IMPORTANT:** Configuration files from older gateways are not directly usable in the BB3-6101/MX-61, but the Babel Buster Configuration Builder program can be used to convert a BB2-6010/SPX configuration into CSV files that may be imported into the BB3-6101/MX-61.

## 2.4 Connectors and Indicators

Follow these steps to make the initial connection to the BB3-6101/MX-61.

(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".



(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.

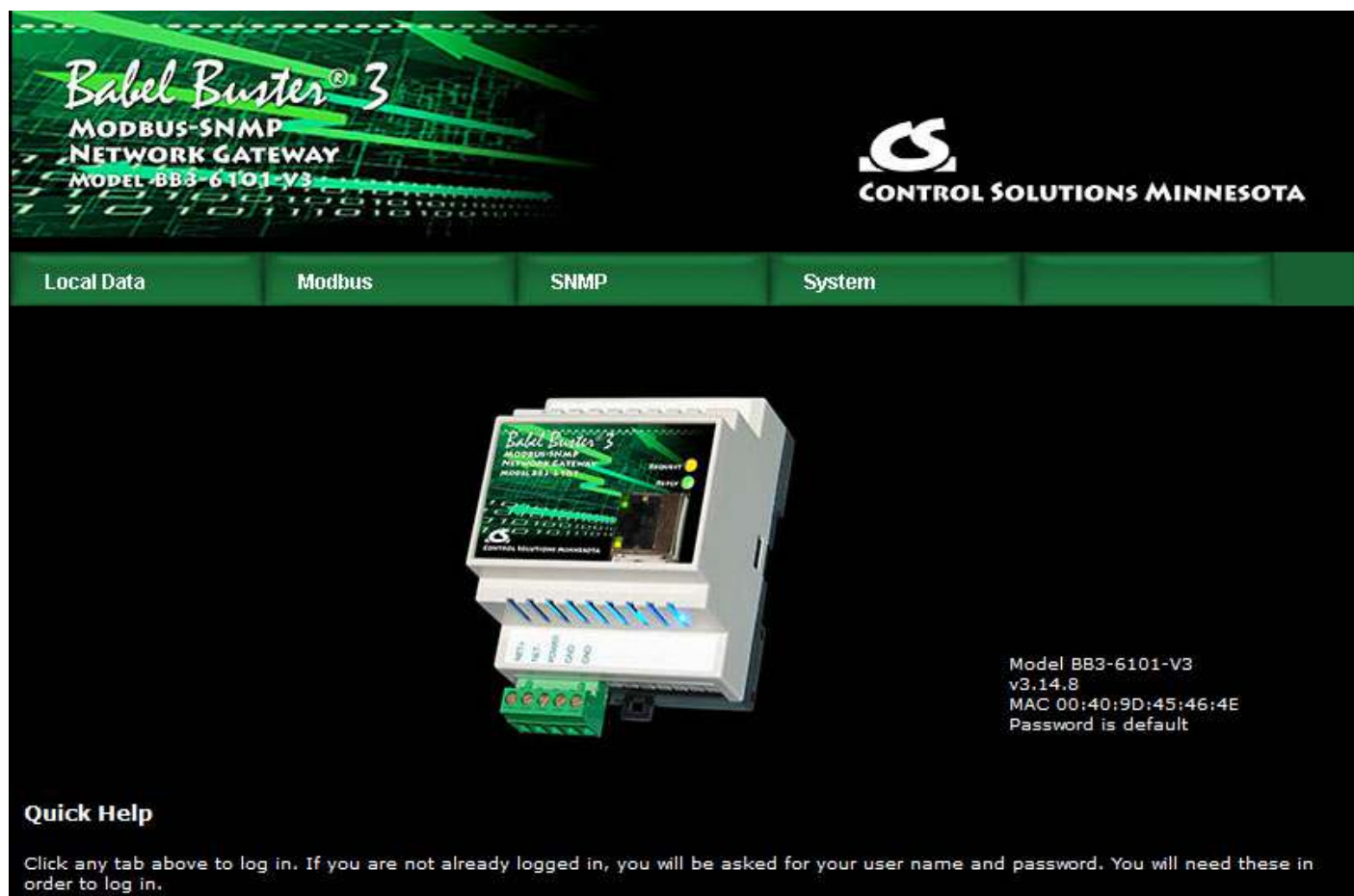
Refer to Appendix A for additional detail pertaining to connections and indicators as well as optional internal jumper settings.

## **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. 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.





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 BB3-6101/MX-61 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.

**Babel Buster® 3**  
MODBUS-SNMP  
NETWORK GATEWAY  
MODEL BB3-6101-V3

**CONTROL SOLUTIONS MINNESOTA**

Local Data   Modbus   SNMP   System

System Setup

File Manager   **Network**   Resources   User(s)

**IPv4 Settings**   ☒ Automatic   ☐ Static

IPv4 Static IP Address: 192.168.1.125   IPv4 Configured IP Address: 192.168.1.125   Apply

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:fe45:464e

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 3 for more about the File Manager.

Local Data   Modbus   SNMP   System

System Setup

**File Manager**   Network   Resources   User(s)

Free space: 0.96 MB

File Directory: BootConfig.xml   Filtered by: \*.xml   Filter   View   Select

Selected File: BootConfig.xml   Action: Save XML Config File   Execute

Boot configuration: BootConfig.xml   Confirm   Restart





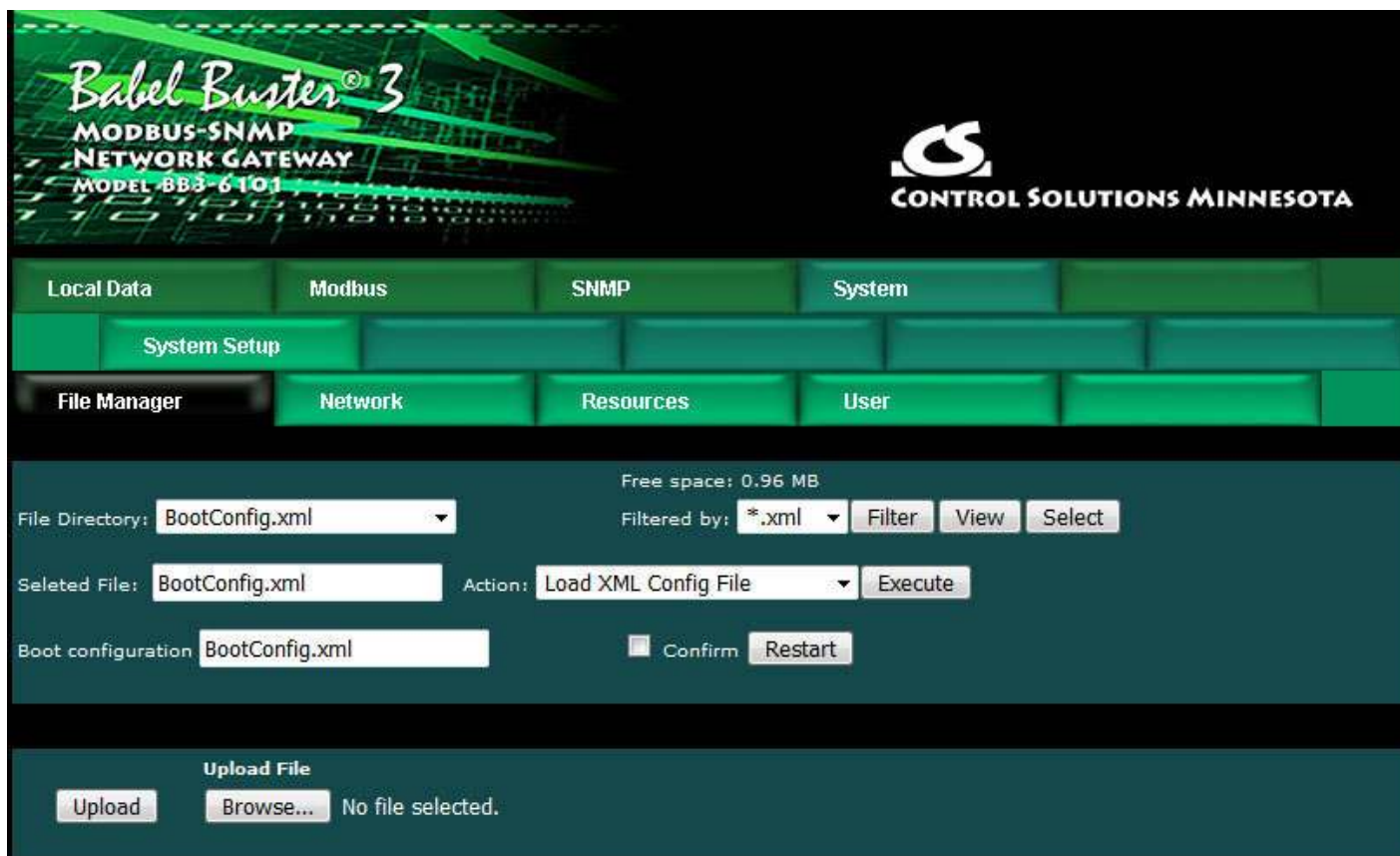
## 3. System Configuration and Resources

### 3.1 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 client configuration here.



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.

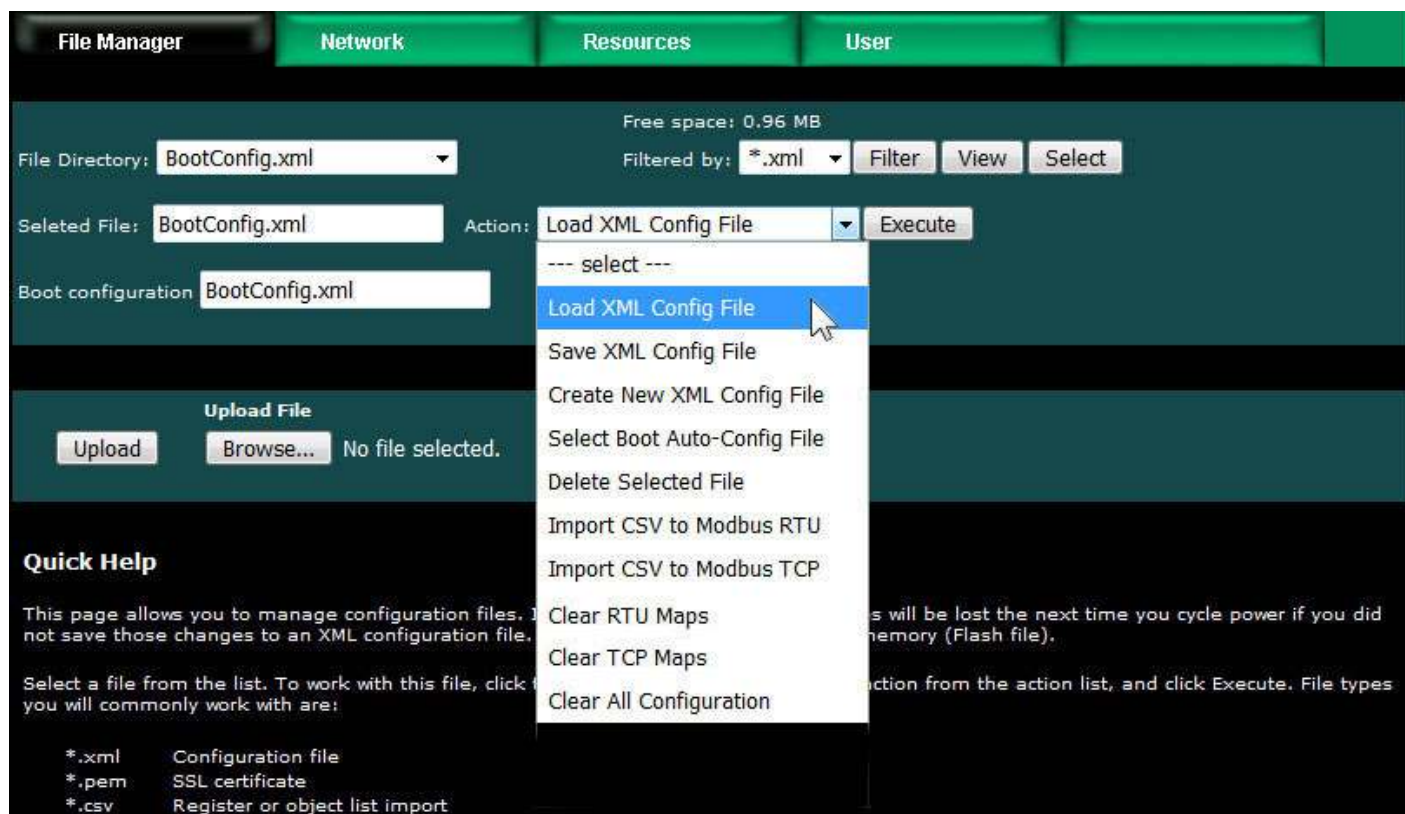


File type filters are as follows:

- \*.xml XML configuration files
- \*.pem SSL certificates
- \*.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.



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.

### 3.1.1 Load, Save, Create XML Configuration File

**IMPORTANT:** Configuration files from older gateways are not directly usable in the BB3-6101/MX-61, but the Babel Buster Configuration Builder program can be used to convert a BB2-6010/SPX configuration into CSV files that may be imported into the BB3-6101/MX-61.

**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 registers 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.

### 3.1.2 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 devices.

### 3.1.3 Delete a File

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

### 3.1.4 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.

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.

### 3.1.5 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 All Configuration:** Execute this action to completely wipe out all configuration. This includes all Modbus maps and devices, all SNMP rules and devices, and all local registers. 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 simply 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.

## 3.2 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.

### **3.3 Network Configuration**

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



### 3.3.1 IPv4, IPv6 Settings

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.

### 3.3.2 NTP Time Server Settings

The Babel Buster maintains time and date via SNTP services.

Primary NTP Server	<input type="text" value="132.163.96.2"/>	Secondary NTP Server	<input type="text" value="129.6.15.30"/>
Daylight Time Start Rule	<input type="text" value="3.2.0/02:00:00"/>	Daylight Time End Rule	<input type="text" value="11.1.0/02:00:00"/>
Standard GMT Offset	<input type="text" value="-360"/> Minutes	Daylight GMT Offset	<input type="text" value="-300"/> Minutes
NTP Refresh Period	<input type="text" value="300"/> Minutes		
Current Local Time <b>2020-07-02 11:05:23</b> <input type="button" value="Refresh"/>			

NTP setup: Enter a primary and secondary IP address of NTP servers, such as those found at [www.nist.gov](http://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.

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 ( $1 \leq m \leq 12$ )
- n indicates which week of the month ( $1 \leq n \leq 5$ ). 5 = the last week in the month.
- d indicates what day of the week ( $0 \leq d \leq 6$ ). 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 gateway functionality has no use for time and date. The only time you have a need for valid time and date is when using secure connections. If you are using a secure web connection and having trouble connecting, be sure NTP is set up here. Also, if you are using SNMPv3 and are getting errors related to "timeliness", check NTP here.

### 3.3.3 Port Settings



The screenshot shows a configuration window with a dark teal background. At the top, there are two checked checkboxes: "Web Server" and "HTTPS Enabled (on 443)". Below these are input fields for "HTTP Port" (set to 80) and "Modbus Port" (set to 502), each with a "(default)" label. A "Set Ports" button is to the right of the HTTP Port field. Below the Modbus Port field is an input field for "MIB Offset" (set to 0). Further down is a checked checkbox for "FTP Server" with the label "Enabled". At the bottom left, the "MAC Address" is displayed as "00:40:9D:45:45:EA". At the bottom right, the "System Uptime" is shown as "0,01:31:11". A status line at the very bottom reads "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 E) 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.

The MIB offset lets you effectively move the entire MIB. This is required if more than one Babel Buster are going to be used on the same network but their configuration is different. Most SNMP managers do not know how to deal with MIBs that have the same set of OIDs but which mean different things in different devices using the "same" MIB. If the offset is changed, you will need to Reload SNMP to cause it to take effect.

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 port numbers or enabling/disabling features requires restarting the Babel Buster before they will take effect.

### 3.4 Resource Allocation

Historically, Control Solutions gateways had a fixed set of 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.

Local Data	Modbus	SNMP	System
System Setup			
File Manager	Network	Resources	User(s)

☐ Confirm

Resource	Current	Pending
Local Modbus Register Pool Size	400	<input type="text" value="400"/>
Data Calculate Rule Count	100	<input type="text" value="100"/>
Data Copy Rule Count	100	<input type="text" value="100"/>
MIB Variable Count, Integer 32-bit	200	<input type="text" value="200"/>
MIB Variable Count, Unsigned 64-bit	20	<input type="text" value="20"/>
MIB Variable Count, Float 32-bit	20	<input type="text" value="20"/>
MIB Variable Count, Float 64-bit	20	<input type="text" value="20"/>
MIB Variable Count, Char String	20	<input type="text" value="20"/>
Number of SNMP Trap Sender Devices	5	<input type="text" value="5"/>
Number of SNMP Trap Send Rules	50	<input type="text" value="50"/>
Number of Modbus TCP Devices	10	<input type="text" value="10"/>
Number of Modbus TCP Client Read Maps	100	<input type="text" value="100"/>
Number of Modbus TCP Client Write Maps	20	<input type="text" value="20"/>
Number of Modbus RTU Read Maps	200	<input type="text" value="200"/>
Number of Modbus RTU Write Maps	50	<input type="text" value="50"/>
Number of Modbus TCP Server Connections	20	<input type="text" value="20"/>
Estimated Memory Utilization	25.14%	25.14%

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%.

### 3.5 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 BB3-6101/MX-61 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 BB3-6101/MX-61 (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.

Local Data	Modbus	SNMP	System		
System Setup					
File Manager	Network	Resources	User		
<input type="button" value="Change"/>					
User Name	Password	Privilege Level	IP Filter	Confirm	Change
		Restricted ▼	0.0.0.0	<input type="checkbox"/>	<input type="checkbox"/>
		Restricted ▼	0.0.0.0	<input type="checkbox"/>	<input type="checkbox"/>
		Restricted ▼	0.0.0.0	<input type="checkbox"/>	<input type="checkbox"/>
		Restricted ▼	0.0.0.0	<input type="checkbox"/>	<input type="checkbox"/>
		Restricted ▼	0.0.0.0	<input type="checkbox"/>	<input type="checkbox"/>
root	●●●●●●●●	Unrestricted	0.0.0.0	<input type="checkbox"/>	<input type="checkbox"/>
root confirm					



## 4. Configuring Local Registers

### 4.1 Creating Local Registers

Babel Buster gateways originally came with a predefined set of local registers that were accessible externally as Modbus registers. Newer models take a completely different approach to defining registers. When you first take the Babel Buster BB3-6101/MX-61 out of the box, it has no registers. You get to create registers as you need them, and several data formats are supported, from 16-bit to 64-bit, both integer and floating point.

Your first visit to the Local Registers page will be the uneventful display illustrated below.

The screenshot shows the 'Local Registers' page in the Babel Buster 3 web interface. The header includes the 'Babel Buster® 3' logo and 'MODBUS-SNMP NETWORK GATEWAY MODEL BB3-6101'. Below the header is a navigation bar with tabs: 'Local Data', 'Modbus', 'SNMP', 'System', and 'Data'. The 'Local Registers' tab is selected. Below the navigation bar are buttons for 'Calculate', 'Copy', and 'Report'. A status bar indicates 'Showing registers from 1' with 'Update', '< Prev', and 'Next >' buttons. The main table has columns: 'Local Register #', 'Register Name', 'Set', 'Register Data', and 'Register Format'. The first row shows '00001' in the 'Local Register #' column, '---' in 'Register Name', a checkbox in 'Set', '---' in 'Register Data', and '---' in 'Register Format'.

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.



Local Register #	Register Name	Set	Register Data	Register Format
00001	---	<input type="checkbox"/>	---	---

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, there are restrictions on changing data format because this impacts how many Modbus registers are actually used. You can change unsigned 16-bit to signed 16-bit, for example, but cannot change 16-bit to 32-bit. 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 in all the various rules and maps the existing numbers needed to get changed. Therefore, such changes are prohibited.

Select the data format for the new registers to be created. The designations Signed and Unsigned refer to integers. Float refers to 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).

**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, character strings, 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.



The screenshot shows the 'Local Registers' configuration window. At the top, there are tabs for 'Local Data', 'Modbus', 'SNMP', and 'System'. Below these are buttons for 'Data', 'Local Registers', 'Calculate', 'Copy', and 'Report'. The 'Local Registers' tab is selected. The interface includes a 'Register #' field with the value '1', a 'Links: -----' field, and 'Update', '< Prev', and 'Next >' buttons. A dropdown menu for 'Register data format' is open, showing options: None, Signed 16-bit, Unsigned 16-bit, Signed 32-bit, Unsigned 32-bit, Signed 64-bit, Unsigned 64-bit, Single Float (highlighted), Double Float, Character String, Mod10 2-reg, Mod10 3-reg, and Mod10 4-reg. Other fields include 'Size: 0', 'Register name: Data Value 1', 'Least significant data', 'Display as hexadecimal', 'power-up', 'If not updated by remote source within 0 seconds', 'First register number to add this many: 1', 'Add New', 'Register # 1', and 'Delete'.

In addition to selecting data format for creating new registers, provide a temporary register name. You can change this later, but it is usually helpful to start with something for a name, and it is suggested that the name ends with a number. The reason why is illustrated shortly.

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.

When first creating registers, you do not need to enter any of the default information on the last line. (You can, but don't have to.) The size only applies when creating character string variables (illustrated later, below).

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.

We have now created 10 new floating point registers. The important thing to observe here is the register numbers in the first column. Remember that Modbus protocol assigns register numbers (or addresses) in 16-bit increments. Since single precision floating point registers are 32-bit registers, each floating point value occupies 2 spots in the Modbus register map. The local register number assignments reflect this fact.

Note that the name given in the screen above was "Data Value 1" but our series of 10 new registers came up with 10 sequential names. If the last thing to appear in the name given when assigning new registers is a number, this value will be incremented by one in the name of each successive new register.

Local Register #	Register Name	Set	Register Data	Register Format
00001	Data Value 1	<input type="checkbox"/>	0.000000	Single Float
00003	Data Value 2	<input type="checkbox"/>	0.000000	Single Float
00005	Data Value 3	<input type="checkbox"/>	0.000000	Single Float
00007	Data Value 4	<input type="checkbox"/>	0.000000	Single Float
00009	Data Value 5	<input type="checkbox"/>	0.000000	Single Float
00011	Data Value 6	<input type="checkbox"/>	0.000000	Single Float
00013	Data Value 7	<input type="checkbox"/>	0.000000	Single Float
00015	Data Value 8	<input type="checkbox"/>	0.000000	Single Float
00017	Data Value 9	<input type="checkbox"/>	0.000000	Single Float
00019	Data Value 10	<input type="checkbox"/>	0.000000	Single Float

Now let's proceed to add some character strings. Click on the last register number assigned thus far to open the register detail page.

Local Registers		Calculate	Copy	Report	
Showing registers from 1					Update < Prev Next >
Local Register #	Register Name	Set	Register Data	Register Format	
00001	Data Value 1	<input type="checkbox"/>	0.000000	Single Float	
00003	Data Value 2	<input type="checkbox"/>	0.000000	Single Float	
00005	Data Value 3	<input type="checkbox"/>	0.000000	Single Float	
00007	Data Value 4	<input type="checkbox"/>	0.000000	Single Float	
00009	Data Value 5	<input type="checkbox"/>	0.000000	Single Float	
00011	Data Value 6	<input type="checkbox"/>	0.000000	Single Float	
00013	Data Value 7	<input type="checkbox"/>	0.000000	Single Float	
00015	Data Value 8	<input type="checkbox"/>	0.000000	Single Float	
00017	Data Value 9	<input type="checkbox"/>	0.000000	Single Float	
00019	Data Value 10	<input type="checkbox"/>	0.000000	Single Float	

Select Character String for data format. This is the one time a size must be specified. In the example below, we are going to allocate 40-character strings. This means that each "register" will actually be a series of 20 Modbus registers (two ASCII characters per register). The character string processing assumes that any display device used in conjunction with these registers will display the high order byte on the left and low order byte on the right in any given 2-character portion of the display screen. This is consistent with display devices that have been tested with Babel Buster.

Local Registers		Calculate	Copy	Report	
Register #	19	Links: -----		Update	< Prev Next >
Register data format:	Character String	Size:	40	Register name	Char String 1
Least significant data should be in first register:	<input type="checkbox"/>	Display as hexadecimal:	<input type="checkbox"/>		
Apply this default value:	0.000000	At power-up	<input type="checkbox"/>	If not updated by remote source within	0 seconds.
First register number to add:	21	Add this many:	4	Add New	Register # 19 Delete



After clicking Add New, we now see our character string registers in our register list (click on the Local Registers tab to get back to the register list). Note that the register numbers increment by 20 to accommodate our 40-character strings.



Local Data	Modbus	SNMP	System		
Data					
Local Registers	Calculate	Copy	Report		
Showing registers from 1 <input type="text"/> <input type="button" value="Update"/> <input type="button" value=" &lt; Prev"/> <input type="button" value=" Next &gt;"/>					
Local Register #	Register Name	Set	Register Data	Register Format	
<u>00001</u>	Data Value 1	<input type="checkbox"/>	0.000000	Single Float	
<u>00003</u>	Data Value 2	<input type="checkbox"/>	0.000000	Single Float	
<u>00005</u>	Data Value 3	<input type="checkbox"/>	0.000000	Single Float	
<u>00007</u>	Data Value 4	<input type="checkbox"/>	0.000000	Single Float	
<u>00009</u>	Data Value 5	<input type="checkbox"/>	0.000000	Single Float	
<u>00011</u>	Data Value 6	<input type="checkbox"/>	0.000000	Single Float	
<u>00013</u>	Data Value 7	<input type="checkbox"/>	0.000000	Single Float	
<u>00015</u>	Data Value 8	<input type="checkbox"/>	0.000000	Single Float	
<u>00017</u>	Data Value 9	<input type="checkbox"/>	0.000000	Single Float	
<u>00019</u>	Data Value 10	<input type="checkbox"/>	0.000000	Single Float	
<u>00021</u>	Char String 1	<input type="checkbox"/>		Char String[40]	
<u>00041</u>	Char String 2	<input type="checkbox"/>		Char String[40]	
<u>00061</u>	Char String 3	<input type="checkbox"/>		Char String[40]	
<u>00081</u>	Char String 4	<input type="checkbox"/>		Char String[40]	

## 4.2 Special Features of Local Registers

There are a couple of features that you may go back and change at any time after registers are created.

Local Data	Modbus	SNMP	System		
Data					
Local Registers	Calculate	Copy	Report		

Register #  Links: -----

Register data format:  Size:  Register name

Least significant data should be in first register: ☐ Display as hexadecimal: ☒

Apply this default value:  ☐ At power-up ☒ If not updated by remote source within  seconds.

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

You may select hexadecimal display of data. This only applies to display on the local web pages, and does not change what Modbus sees externally (Modbus only sees a collection of bits, and that fact never changes). Hexadecimal display of a floating point value is probably meaningless, but hexadecimal display of registers containing a set of status bits packed into a single register is often easier to interpret when displayed as hexadecimal.

You have the option of applying a default value under two conditions: (1) Automatically at power-up; (2) Any time this register is not updated by some remote source within the timeout given (this assumes the Babel Buster gateway is acting as a slave or server).

If this register is being used to hold a value provided by some other device acting as master, and you want a way of externally detecting when that device has failed to communicate, you can cause a "flag" value to appear in this register when a new value has not been received within the given timeout period.

Applying a default at power-up is potentially useful if you want this gateway to always write a certain value to some other Modbus slave any time the system wakes up.

The other use for default at power-up is when you need to use a constant value in a calculate rule. Set aside a register whose only purpose is to hold this constant for later use.

### **4.3 Local Register Calculate Rules**

The Babel Buster BB3-6101 or MX-61 includes the ability to do simple calculations based on simple template rules. Select the operation, one or two operands as applicable, and a register to place the result in. Operations like "multiply" will use registers A "and" B. Operations like "sum" can add up the contents of a series of registers by selecting "thru" instead of "and".

Local Data		Modbus		SNMP		System	
Data							
Local Registers		Calculate		Copy		Report	
Showing 1 to 1 of 1				Update		< Prev Next >	
Rule #	Perform Operation	Using Register #	And/Thru Using	This Reg#/Value	Place Result in Register #		
1	none	0	and	0	0		
# Rules Enabled: 1						Insert Delete	

**Quick Help**

This simple set of calculations provides a means of programming for very simple operations such as logically ANDing or ORing a couple of registers to be sent out to another register. Select the operation to be performed from the drop list. All but logical NOT operations require two register numbers as applicable.

Some operations require ranges of registers. Select "and" or "thru" to select two registers or multiple registers in a range. Average, sum, and logic operations of multiple registers.

Delete will remove a rule from the list. Insert will insert a new rule before the rule number shown, and is used for placing rules between existing rules. It is 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.

Error checking will prevent the entry of an invalid register number in the rule. However, if a previously valid rule is invalidated by that register being later deleted, the invalid rule will prevent other updates. Because error trapping happens sooner than the insert/delete process, the delete will not remove the invalid rule. To avoid this problem, enter zero for register number, update, then delete (or enter a new valid register number if you intend to keep the rule).

Selecting "comp = N" 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 show "none" as the operation. 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 register 1 by register 3 and places the result in register 5.

Local Registers		Calculate		Copy		Report	
Showing 1 to 2 of 2				Update		< Prev Next >	
Rule #	Perform Operation	Using Register #	And/Thru Using	This Reg#/Value	Place Result in Register #		
1	multiply	1	and	3	5		
2	none	0	and	0	0		
# Rules Enabled: 2						Insert Delete	

If registers 1 and 3 contain the values shown below, then the result shown in register 5 would be expected.

Local Register #	Register Name	Set	Register Data	Register Format
00001	Data Value 1	<input type="checkbox"/>	100.000000	Single Float
00003	Data Value 2	<input type="checkbox"/>	0.250000	Single Float
00005	Data Value 3	<input type="checkbox"/>	25.000000	Single Float

Constants may be introduced into the calculation by reserving a register to hold that

constant, and then configuring it to apply a default value at power-up. This default value should be the constant you wish to include in a calculation.

The other option is to use the set operation, but the value is limited to unsigned integer with the set operation. The example illustrated below will set register 1 to a value of 12345.

Rule #	Perform Operation	Using Register #	And/Thru Using	This Reg#/Value	Place Result in Register #	
1	set	1	using	12345	1	
2	none	0	and	0	0	

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

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

Operations available on one register:

logic NOT	Generate bit-wise negation of register
test = 0	Set result to 'true' if register is zero
test < 0	Set result to 'true' if register is less than zero
test > 0	Set result to 'true' if register is greater than zero



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

set	Set register to given value (unsigned 32-bit integer)
skip = N	Skip next operation if register is equal to given value
skip < N	Skip next operation if register is less than given value
skip > N	Skip next operation if register is greater than given value
comp = N	Compare, set result 'true' if register is equal to given value
comp < N	Compare, set result 'true' if register is less than given value
comp > N	Compare, set result 'true' if register is greater than given value
pack	Perform Pack operation (see text)
fill	Perform Fill operation (see text)
unpack	Perform Unpack operation (see text)

Operations "using" a given value will have an unsigned integer value in the "This Reg#/Value" column rather than a register 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(s) if true. The true value will be the maximum unsigned 16-bit or 32-bit integer value if the result register is integer. If displayed as unsigned hexadecimal, it will be FFFF or FFFFFFFF. Displayed as signed 16-bit integer, "true" will be 32767. If the result register is floating point, then "true" will just be 1.0. The purpose of using FFFF for unsigned integer true is so that the result is useful as a bitmask.

Pack and fill are used for packing multiple local registers into a single register 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 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 register 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 register (after first clearing the bits in the affected position of the result register).

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

Local Registers		Calculate	Copy	Report		
Showing 1 to 8 of 8						Update < Prev Next >
Rule #	Perform Operation	Using Register #	And/Thru Using	This Reg#/Value	Place Result in Register #	
1	pack	2	using	Fh	1	
2	pack	3	using	F0h	1	
3	pack	4	using	F00h	1	
4	fill	1	using	3000h	1	
5	pack	13	using	FFh	11	
6	pack	15	using	FF0000h	11	
7	fill	11	using	80000000h	11	
8	none	0	and	0	0	
# Rules Enabled: 8						Insert Delete

The example below shows the content of result registers 1 and 11 using the above calculate rules and the local register values shown. The contents of registers 2, 3, and 4 are packed into register 1. The contents of registers 13 and 15 are packed into 11 (all 32-bit register pairs).

Local Registers		Calculate	Copy	Report		
Showing registers from 1						Update < Prev Next >
Local Register #	Register Name	Set	Register Data			Register Format
00001	Register 1	<input type="checkbox"/>		3432		Unsigned 16-bit
00002	Register 2	<input type="checkbox"/>		2		Unsigned 16-bit
00003	Register 3	<input type="checkbox"/>		3		Unsigned 16-bit
00004	Register 4	<input type="checkbox"/>		4		Unsigned 16-bit
00005	Register 5	<input type="checkbox"/>		0		Unsigned 16-bit
00006	Register 6	<input type="checkbox"/>		0		Unsigned 16-bit
00007	Register 7	<input type="checkbox"/>		0		Unsigned 16-bit
00008	Register 8	<input type="checkbox"/>		0		Unsigned 16-bit
00009	Register 9	<input type="checkbox"/>		0		Unsigned 16-bit
00010	Register 10	<input type="checkbox"/>		0		Unsigned 16-bit
00011	Register 11	<input type="checkbox"/>		807F003F		Unsigned 32-bit
00013	Register 12	<input type="checkbox"/>		63		Unsigned 32-bit
00015	Register 13	<input type="checkbox"/>		127		Unsigned 32-bit
00017	Register 14	<input type="checkbox"/>		0		Unsigned 32-bit
00019	Register 15	<input type="checkbox"/>		0		Unsigned 32-bit

This process can be reversed using the "unpack" operation. The following calculate rules exactly reverse the above packing operations (discarding fill in this case).

Local Registers		Calculate		Copy		Report	
Showing 1 to 6 of 6							
<input type="button" value="Update"/> <input type="button" value=" &lt; Prev"/> <input type="button" value="Next &gt;"/>							
Rule #	Perform Operation	Using Register #	And/Thru Using	This Reg#/Value	Place Result in Register #		
1	unpack ▼	1	using ▼	Fh	2		
2	unpack ▼	1	using ▼	F0h	3		
3	unpack ▼	1	using ▼	F00h	4		
4	unpack ▼	11	using ▼	FFh	13		
5	unpack ▼	11	using ▼	FF0000h	15		
6	none ▼	0	and ▼	0	0		
# Rules Enabled: 6							
<input type="button" value="Insert"/> <input type="button" value="Delete"/>							

Register 1 is unpacked into registers 2, 3 and 4. Register 11 is unpacked into registers 13 and 15 (all 32-bit register pairs).

Local Registers

Calculate

Copy

Report

Showing registers from 1

Update

< Prev

Next >

Local Register #	Register Name	Set	Register Data	Register Format
00001	Register 1	<input type="checkbox"/>	3432	Unsigned 16-bit
00002	Register 2	<input type="checkbox"/>	2	Unsigned 16-bit
00003	Register 3	<input type="checkbox"/>	3	Unsigned 16-bit
00004	Register 4	<input type="checkbox"/>	4	Unsigned 16-bit
00005	Register 5	<input type="checkbox"/>	0	Unsigned 16-bit
00006	Register 6	<input type="checkbox"/>	0	Unsigned 16-bit
00007	Register 7	<input type="checkbox"/>	0	Unsigned 16-bit
00008	Register 8	<input type="checkbox"/>	0	Unsigned 16-bit
00009	Register 9	<input type="checkbox"/>	0	Unsigned 16-bit
00010	Register 10	<input type="checkbox"/>	0	Unsigned 16-bit
00011	Register 11	<input type="checkbox"/>	807F003F	Unsigned 32-bit
00013	Register 12	<input type="checkbox"/>	63	Unsigned 32-bit
00015	Register 13	<input type="checkbox"/>	127	Unsigned 32-bit
00017	Register 14	<input type="checkbox"/>	0	Unsigned 32-bit
00019	Register 15	<input type="checkbox"/>	0	Unsigned 32-bit

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

Local Registers		Calculate	Copy	Report		
Showing 1 to 8 of 8						
<div>Update</div> <div>&lt; Prev</div> <div>Next &gt;</div>						
Rule #	Perform Operation	Using Register #	And/Thru Using	This Reg#/Value	Place Result in Register #	
1	comp = N	1	using	10	2	
2	comp < N	1	using	10	3	
3	comp > N	1	using	10	4	
4	set	5	using	202	5	
5	skip = N	6	using	0	6	
6	set	7	using	0	7	
7	set	8	using	88	8	
8	none	0	and	0	0	
<div># Rules Enabled: 8</div> <div>Insert</div> <div>Delete</div>						

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

Local Data		Modbus	SNMP	System		
Data						
Local Registers		Calculate	Copy	Report		
Showing registers from 1						
<div>Update</div> <div>&lt; Prev</div> <div>Next &gt;</div>						
Local Register #	Register Name	Set	Register Data	Register Format		
00001	Register 1	<input type="checkbox"/>	10	Signed 16-bit		
00002	Register 2	<input type="checkbox"/>	32767	Signed 16-bit		
00003	Register 3	<input type="checkbox"/>	0	Signed 16-bit		
00004	Register 4	<input type="checkbox"/>	0	Signed 16-bit		
00005	Register 5	<input type="checkbox"/>	202	Signed 16-bit		
00006	Register 6	<input type="checkbox"/>	0	Signed 16-bit		
00007	Register 7	<input type="checkbox"/>	0	Signed 16-bit		
00008	Register 8	<input type="checkbox"/>	88	Signed 16-bit		
00009	Register 9	<input type="checkbox"/>	0	Signed 16-bit		

#### 4.4 Local Register Copy Rules

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



**Babel Buster® 3**  
MODBUS-SNMP  
NETWORK GATEWAY  
MODEL BB3-6101

**CONTROL SOLUTIONS MINNESOTA**

Local Data | Modbus | IoT Cloud | System

Data

Local Registers | Calculate | **Copy** | Report

Showing 1 to 2 of 2

Rule #	Source Register #	Destination Register #
1	1	3
2	0	0

# Rules Enabled: 2

Update < Prev Next > Insert Delete

The above rule would cause the following data copy to occur:

Local Registers

Calculate

Copy

Report

Showing registers from 1

Update

< Prev

Next >

Local Register #	Register Name	Set	Register Data	Register Format
00001	Data Value 1	<input type="checkbox"/>	1234.000000	Single Float
00003	Data Value 2	<input type="checkbox"/>	1234.000000	Single Float
00005	Data Value 3	<input type="checkbox"/>	0.000000	Single Float

Here, however, is a much more interesting use of the copy rule:

Local Registers

Calculate

Copy

Report

Showing 1 to 2 of 2

Update

< Prev

Next >

Rule #	Source Register #	Destination Register #
1	1	21
2	0	0

# Rules Enabled: 2

Insert

Delete

The above rule would cause the following copy to happen. Note that "copy" also means data reformatting. Therefore, if you need to convert a number to a character string (or vice versa), simply copy it from one register to the other. In this example, our copy rule is converting floating point to an ASCII string ready to be sent to a display device.



Local Registers		Calculate	Copy	Report	
Showing registers from 1				Update	< Prev Next >
Local Register #	Register Name	Set	Register Data	Register Format	
00001	Data Value 1	<input type="checkbox"/>	45.980000	Single Float	
00003	Data Value 2	<input type="checkbox"/>	0.000000	Single Float	
00021	Char String 1	<input type="checkbox"/>	45.980000	Char String[40]	
00041	Char String 2	<input type="checkbox"/>		Char String[40]	

String conversion is not the only conversion you can do. If you need to convert floating point to integer, or vice versa, the copy rule will also do that. Note, however, that if you need to read an integer from a remote Modbus slave but want the result stored locally as floating point, you can do that conversion as part of the read map and do not need a separate copy rule to accomplish that conversion.

## 4.5 Device Status Reporting

The Babel Buster BB3-6101/MX-61 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 register 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 BB3-6101/MX-61 also includes the ability to report device errors to an assigned status register rather than rely on default values. This reporting is configured on the Report page.

Local Data	Modbus	SNMP	System
Data			

Local Registers	Calculate	Copy	Report
-----------------	-----------	------	--------

Showing 1 to 5 of 5

Update < Prev Next >

Report Status of	Device or Unit #	To This Register	With This Delay (Sec.)	Delete
Modbus RTU Master	1	10	20	<input type="checkbox"/>
Modbus RTU Master	2	11	20	<input type="checkbox"/>
Modbus RTU Master	3	12	20	<input type="checkbox"/>
Modbus TCP Client	1	16	20	<input type="checkbox"/>
Modbus TCP Client	2	17	20	<input type="checkbox"/>

Modbus TCP Client
2
17
20
Add



This optional list allows reporting device errors as register 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 a register 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 register 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.



## 5. Programming with Script Basic

### 5.1 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.



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 compiled help file available for Script Basic.



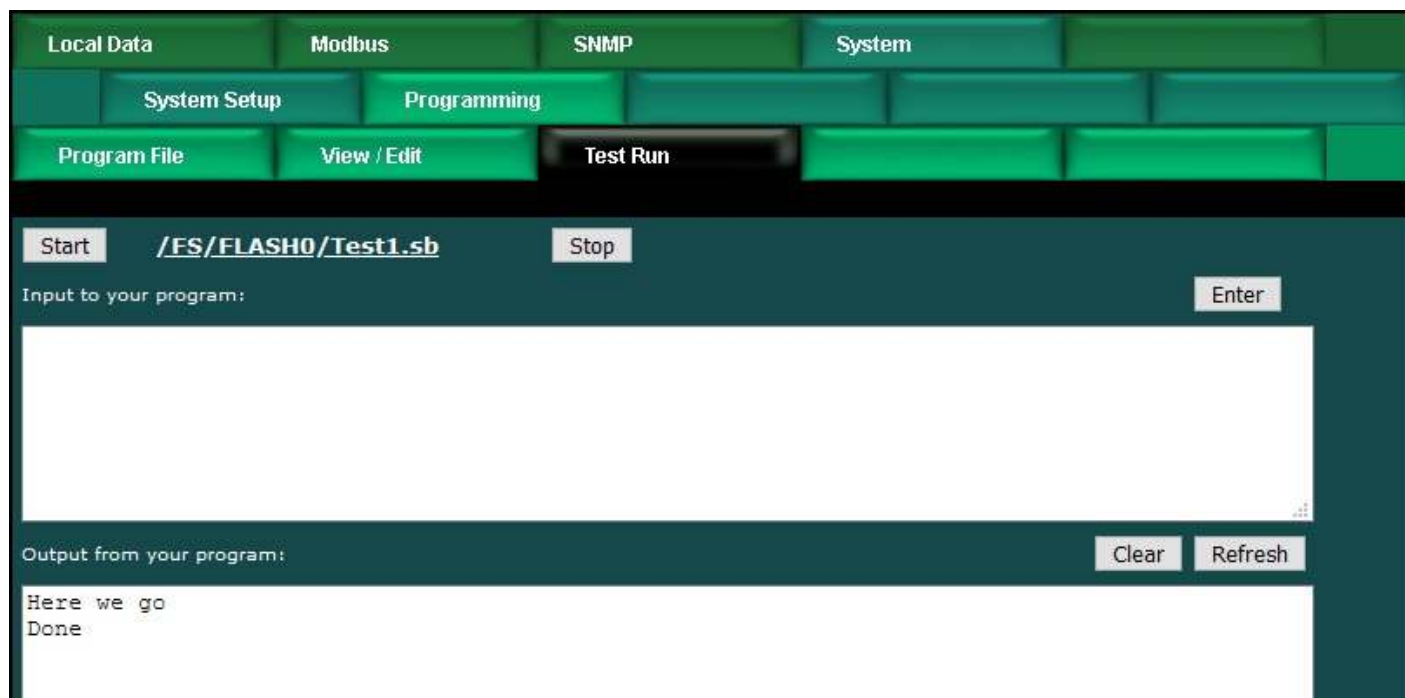
## 5.2 Testing the Program

The virtual terminal 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 Virtual Terminal 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. 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.

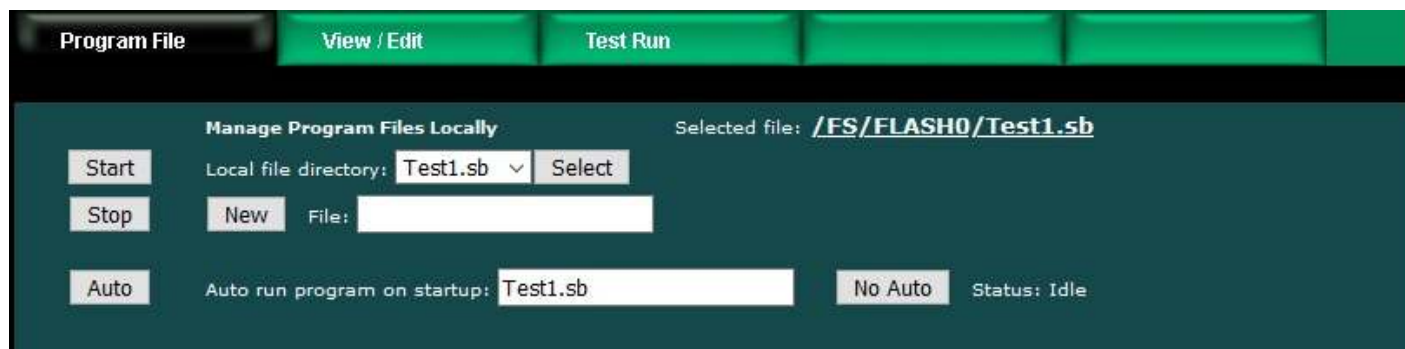
An abbreviated screen shot of the Virtual Terminal is illustrated below. In this example, output from the above test program is illustrated.



### 5.3 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.



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.

### 5.4 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.

## 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 example illustrates capturing one character at a time and outputting 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 terminates in this simple example.



## 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 communication 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.

## LED Control:

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 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 Modbus registers (or more correctly, data objects accessible as Modbus registers). 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
10001	Request LED Flashes Green
10002	Request LED Flashes Yellow



10003	Reply LED Flashes Green
10004	Reply LED Flashes Red
10005	Request LED Green Static On/Off
10006	Request LED Yellow Static On/Off
10007	Reply LED Green Static On/Off
10008	Reply LED Red Static On/Off

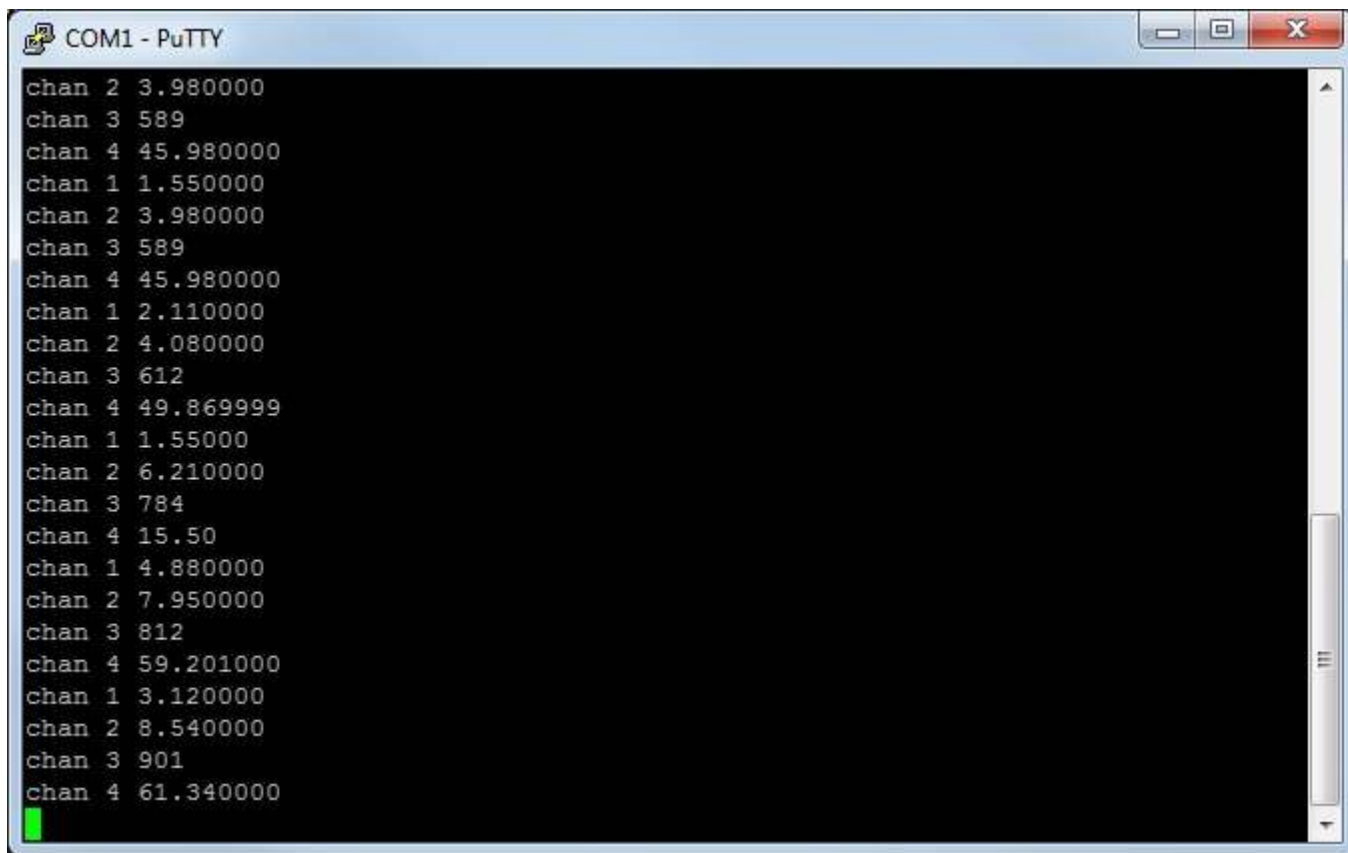
For example, to flash the Reply LED Red, you would use:

```
setreg 10004,1
```

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.

## 5.5 Example: Data Logger Capture

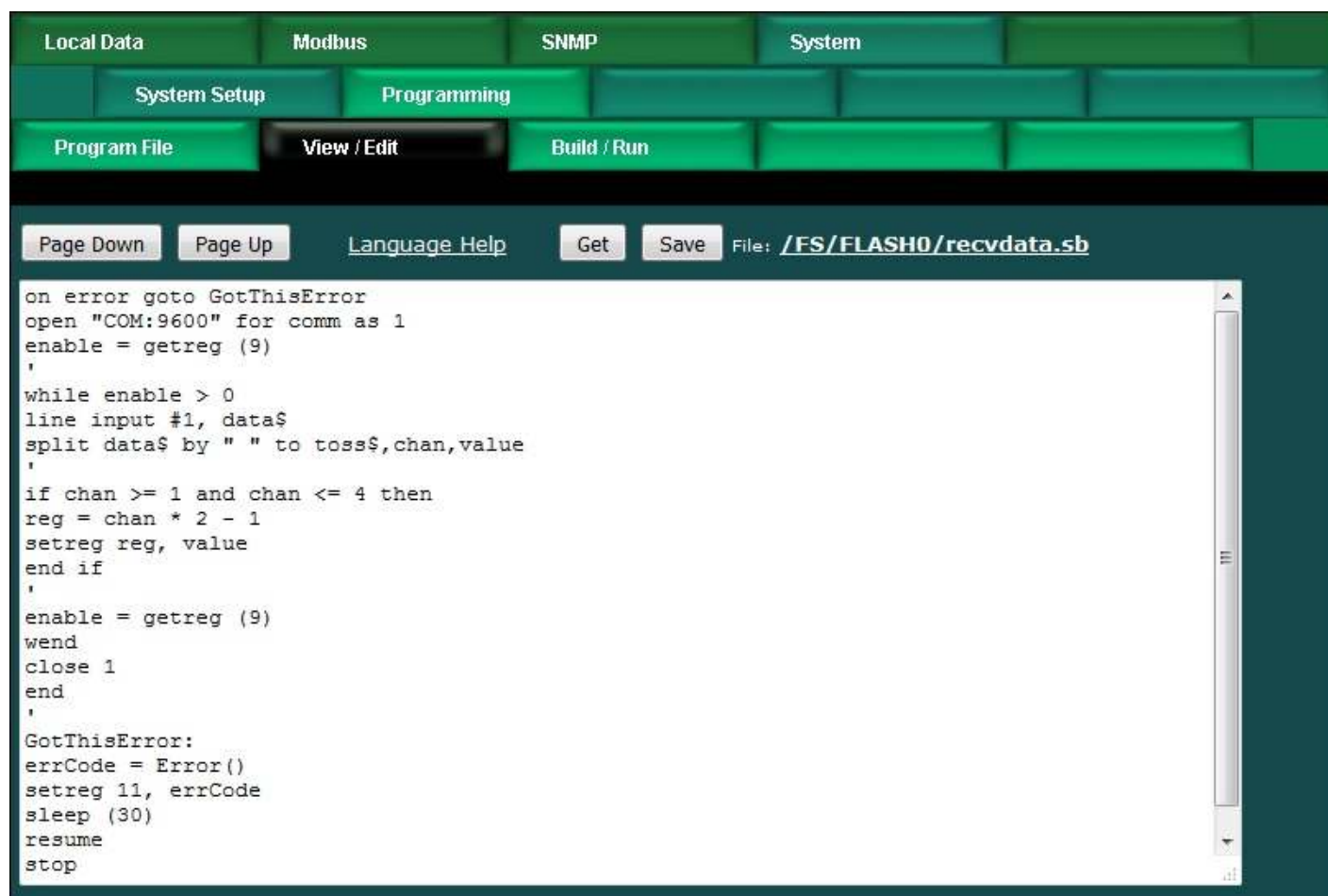
Our first example will capture data from a data logger type device that periodically sends strings of data. In this case, it is simply a channel number and that channel's data value. This device is a 4-channel device, and its output is illustrated here by simply connecting it to a PC (via RS485 to RS232 adapter) and running PuTTY to see its output.



```
COM1 - PuTTY
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 registers 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 "data". Then, based on channel number, we calculate a register number. This is needed because we are going to store the data in floating point registers which occupy 2 Modbus registers worth of address space each.



The local register values are going to continue changing as new data is received. The screen shot below illustrates the most recently received data in this example.

Local Registers		Calculate	Copy		
Showing registers from <input type="text" value="1"/>				<input type="button" value="Update"/>	<input type="button" value=" &lt; Prev"/> <input type="button" value="Next &gt;"/>
Local Register #	Register Name	Set	Register Data	Register Format	
<u>00001</u>	Data Value 1	<input type="checkbox"/>	3.120000	Single Float	
<u>00003</u>	Data Value 2	<input type="checkbox"/>	8.540000	Single Float	
<u>00005</u>	Data Value 3	<input type="checkbox"/>	901.000000	Single Float	
<u>00007</u>	Data Value 4	<input type="checkbox"/>	61.340000	Single Float	
<u>00009</u>	Enable Logger	<input type="checkbox"/>	1.000000	Single Float	
<u>00011</u>	Data Value 6	<input type="checkbox"/>	0.000000	Single Float	

We are going to make these data values available via our own internal MIB so that other SNMP devices can do an SNMP Get to retrieve these numbers. To do this, we configure the local MIB to reference the registers we are using. Note that while our

local registers are floating point, we are going to provide the results as scaled integers. This is the most universally compatible and interoperable means of sharing SNMP data.

This example provides the data as variables in our local MIB. You could also set the Babel Buster to send traps based on these numbers exceeding certain thresholds. This would be done as discussed in the Trap Sender portion of this user guide.

The screenshot shows the Babel Buster software interface with the 'Local MIB' tab selected. The interface includes a top navigation bar with 'Local Data', 'Modbus', 'SNMP', and 'System'. Below this is a sub-navigation bar with 'Local MIB', 'Client Setup', 'Client Data', 'Trap Sender', and 'Trap Receiver'. A row of data type buttons includes 'Integer 32-bit', 'Unsigned 64-bit', 'Float 32-Bit', 'Float 64-Bit', and 'Char String'. The main area displays a table of 5 maps, showing '1' to 5 of 5. The table has columns for Map #, Local SNMP OID, Local Register #, Scale Factor, Local Value, and Local Name. At the bottom, there are buttons for 'Reload SNMP', 'Map # 1', 'Remove', and 'Insert Before'.

Map #	Local SNMP OID	Local Register #	Scale Factor	Local Value	Local Name
1	1.3.6.1.4.1.3815.1.6.1.1.1.2.1	1	x100	3.120000	Data Value 1
2	1.3.6.1.4.1.3815.1.6.1.1.1.2.2	3	x100	8.540000	Data Value 2
3	1.3.6.1.4.1.3815.1.6.1.1.1.2.3	5	x100	901.000000	Data Value 3
4	1.3.6.1.4.1.3815.1.6.1.1.1.2.4	7	x100	61.340000	Data Value 4
5	1.3.6.1.4.1.3815.1.6.1.1.1.2.5	0	x1	---	---

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.

### 5.5.1 Program Code - Capture

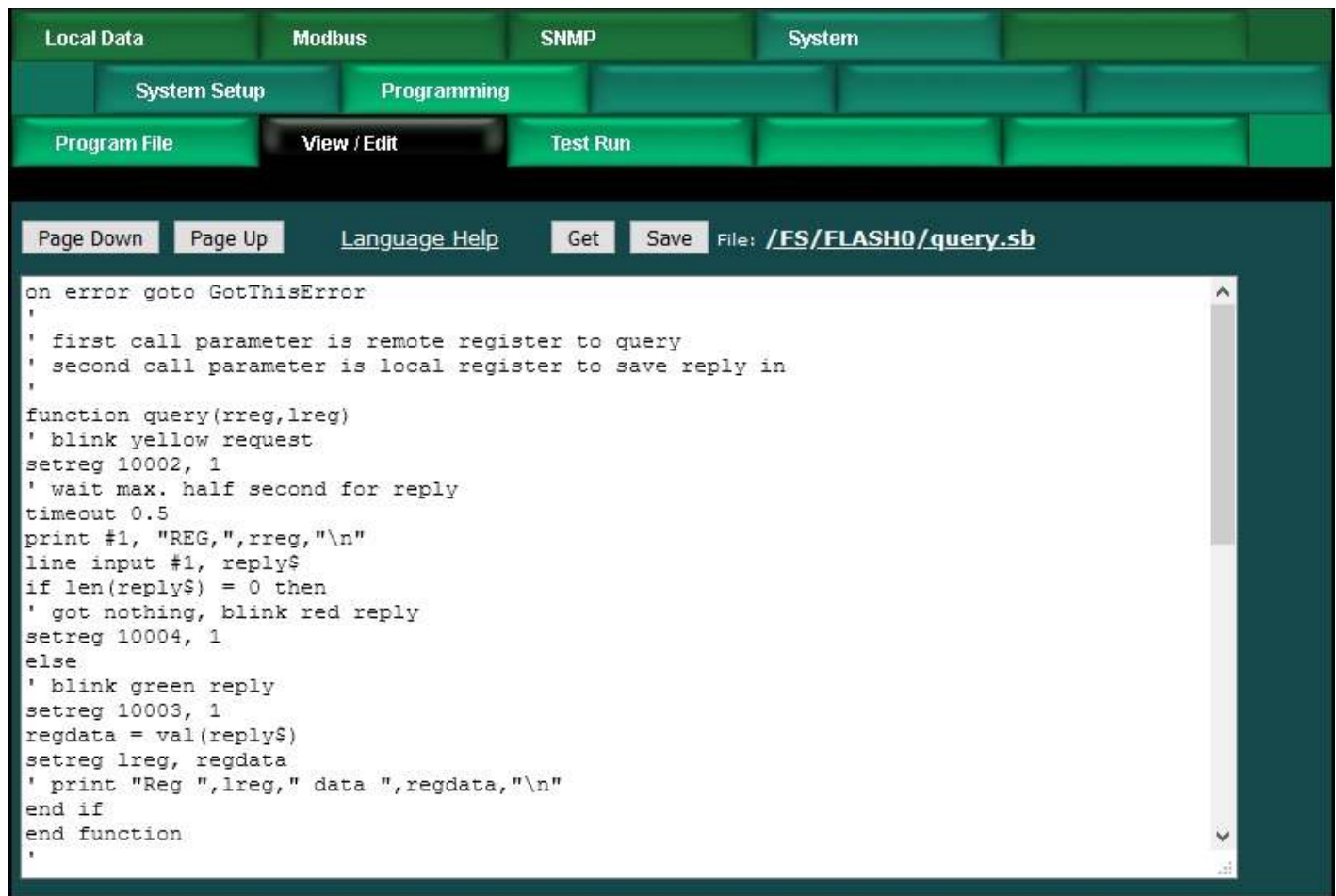
Here is the source code for the example program in this section.

```
on error goto GotThisError
open "COM:9600" for comm as 1
enable = getreg (9)
'
while enable > 0
line input #1, data$
split data$ by " " to toss$,chan,value
'
if chan >= 1 and chan <= 4 then
reg = chan * 2 - 1
setreg reg, value
end if
'
enable = getreg (9)
wend
close 1
end
'
GotThisError:
```

```
errCode = Error()  
setreg 11, errCode  
sleep (30)  
resume  
stop
```

## 5.6 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 a register number to request the data value for register X in the remote device. We will receive simply a number that we then put in one of our local registers.



To illustrate this simple query program, a second BB3-6101/MX-61 was programmed to be the repsonding side of this interaction. The communication between the two was monitored using PuTTY.

```

COM1 - 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
REG, 1
55
REG, 2
88
REG, 10
201
QUIT, 0

```

The local register values following the last set of queries illustrated above is shown in the following screen shot.

**Babel Buster® 3**  
MODBUS-SNMP  
NETWORK GATEWAY  
MODEL BB3-6101-V3SP

**CONTROL SOLUTIONS MINNESOTA**

Local Data    Modbus    SNMP    System

Data

Local Registers    Calculate    Copy

Showing registers from 1    Update    < Prev    Next >

Local Register #	Register Name	Set	Register Data	Register Format
00001	Register 1	<input type="checkbox"/>	55	Signed 16-bit
00002	Register 2	<input type="checkbox"/>	88	Signed 16-bit
00003	Register 3	<input type="checkbox"/>	201	Signed 16-bit
00004	Register 4	<input type="checkbox"/>	0	Signed 16-bit

### 5.6.1 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 to query
' second call parameter is local register to save reply in
'
function query(rreg,lreg)
' blink yellow request
setreg 10002, 1
' wait max. half second for reply
timeout 0.5
print #1, "REG,",rreg,"\n"
line input #1, reply$
if len(reply$) = 0 then
' got nothing, blink red reply
setreg 10004, 1
else
' blink green reply
setreg 10003, 1
regdata = val(reply$)
setreg lreg, regdata
' print "Reg ",lreg," data ",regdata,"\n"
end if
end function
'
' query program
'
open "COM:9600" for comm as 1
print "Here we go\n"
enable = 1
do while enable > 0
' expand this list to query whatever you wish
query (1, 1)
query (2, 2)
query (10, 3)
sleep (1)
' continue until register 20 is zero
enable = getreg (20)
loop
print #1, "QUIT,0","\n"
close #1
print "Done\n"
end
'
GotThisError:
errCode = Error()
print "Error: ",errcode,"\n"
setreg 10003, 1
resume
stop
```

### 5.6.2 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 10003, 1
resume
stop
```



## 6. Configuring Gateway as a Modbus TCP Client

The Babel Buster BB3-6101/MX-61 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 register when received. Data written to a remote device is taken from a local register when sent. The local registers are the same collection of registers that are accessible to other Modbus TCP clients as a collection of holding registers. These local registers are also accessible to SNMP.

The following section details the process of setting up read and write maps via the web interface. Once familiar with map content, you have the option of importing bulk configurations as a CSV file. The import function is found on the File Manager page, and details about the content and format of the CSV file are found in Appendix B.

### 6.1 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. 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.

**Babel Buster® 3**  
MODBUS-SNMP  
NETWORK GATEWAY  
MODEL BB3-6101

**CONTROL SOLUTIONS MINNESOTA**

Local Data | Modbus | SNMP | System | Serial Port | TCP Setup | TCP Data | Server Remap

Devices | Client Read Map | Client Write Map

Device # 1 [Update] [< Prev] [Next >]

Local Name: Device 1

Use: ☐ Static IPv4 ☒ Static IPv6 ☐ Domain Lookup

IP Address: 192.168.1.134

Domain Name:

Port: 502

☐ 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: 5.0 Seconds

Connection Status: 0 [Clear]

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. (Control Solutions BB2-6010-GW or any other model with the -GW suffix would provide this type of TCP to RTU routing.) Unit numbers are entered on each individual read or write 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.

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.

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.

The static IP address for the Modbus TCP device can be either IPv4 or IPv6. Along with selecting the desired IP version, be sure to enter the applicable IP address format. In addition, if IPv6 is used, be sure IPv6 is enabled on the Network configuration page.

The screenshot shows the configuration page for Device # 2. The 'Devices' tab is selected. The 'Client Read Map' and 'Client Write Map' tabs are also visible. The 'Device #' field is set to 2. The 'Local Name' is 'Device 2'. The 'Use' section has three radio buttons: 'Static IPv4', 'Static IPv6' (which is selected), and 'Domain Lookup'. The 'IP Address' field contains the IPv6 address 'fe80::c28:cf9b:b7f8:639e'. The 'Port' is set to 502. The 'Domain Name' field is empty. The 'Unit (optional)' is set to 1. There is a checkbox for 'Use FC 5/6 instead of 15/16' which is unchecked. The 'Default Poll Period' is set to 5.0 seconds. The 'Connection Status' is 0, and there is a 'Clear' button next to it. Navigation buttons 'Update', '< Prev', and 'Next >' are at the top right.

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. If found, the IP address provided by DNS will be displayed here (but not saved in the XML configuration file).

The screenshot shows the configuration page for Device # 3. The 'Devices' tab is selected. The 'Client Read Map' and 'Client Write Map' tabs are also visible. The 'Device #' field is set to 3. The 'Local Name' is 'Device 3'. The 'Use' section has three radio buttons: 'Static IPv4', 'Static IPv6', and 'Domain Lookup' (which is selected). The 'IP Address' field contains the IPv4 address '50.97.153.22'. The 'Port' is set to 502. The 'Domain Name' field contains 'myModbusDevice.net'. The 'Unit (optional)' is set to 1. There is a checkbox for 'Use FC 5/6 instead of 15/16' which is unchecked. The 'Default Poll Period' is set to 5.0 seconds. The 'Connection Status' is 0, and there is a 'Clear' button next to it. Navigation buttons 'Update', '< Prev', and 'Next >' are at the top right.

## 6.2 Modbus TCP Client Read Maps

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



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

Map number simply tells you 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.

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.

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

Read remote registers into local registers. This page creates a map entry that reads data from one or more remote Modbus/TCP servers for

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

**Client Read Map**

Showing 1 to 1 of 1

Update < Prev Next >

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

**Quick Help**

Read remote registers into local registers. This page creates a map entry that reads data from one or more remote Modbus/TCP servers for processing here. Click on map number to view more detail and insert/delete rules.

Rule number simply tells you which map entry to edit. To edit a specific map, enter the rule number in the "Showing" box, then click Update.

Rules entered on this page only affect local register data. To write data to remote devices, go to the Client Write Map to write data to those devices. The full parameter set is different for read and write maps.

An abbreviated version of a list of rules is shown on this page. Any of the parameters shown may be changed here and registered by clicking

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 register data conversion.)

**Client Read Map**

Showing 1 to 2 of 2

Update < Prev Next >

Map #	Remote Type	Remote Register Format	Remote Register #	Remote Device	Local Register #	Name
1	Holding Register	UINT-16	1	ModSim	1	Data Value 1
2	None	None	0	None	0	

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 Register is where data read from the remote Modbus TCP server will be stored locally in the Babel Buster BB3-6101/MX-61. If the local register data format does not match what you are reading from the Modbus device, the data will be converted automatically when it is read.

Devices		Client Read Map		Client Write Map								
Showing 1 to 2 of 2										Update	< Prev	Next >
Map #	Remote Type	Remote Register Format	Remote Register #	Remote Device	Local Register #	Name						
1	Holding Register	UINT-16	1	ModSim	1	Data Value 1						
2	None	None	0	None	0							

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

Devices		Client Read Map		Client Write Map								
Map # 1										Update	< Prev	Next >
Read <span>Holding Register</span> as <span>Unsigned 16-bit</span> Size: <span>0</span> From register # <span>1</span> at <span>ModSim</span> unit # <span>2</span> With low register first if checked: <input type="checkbox"/> Apply bit mask if applicable: <span>00000000</span> then apply scale: <span>1.800000</span> and offset: <span>32.000000</span> Save in local register # <span>1</span> named <span>Data Value 1</span> Repeat this process every <span>5.0</span> seconds. Apply this default value: <span>0.000000</span> after <span>0</span> read failure(s). <input type="checkbox"/> Enable this map only when index register <span>0</span> is set to a value of <span>0</span>												
# Client Read Maps Enabled: 3										Insert	Delete	

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 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 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 register 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 register number is provided and the Enable box is checked, then this read map will only be executed when the index register (local register) contains the value given. This allows multiple read maps to supply data to the same local register based on the value of the index register. It also allows reading to simply be suspended if a single read map supplies data to the local register. 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 register 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 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.

Local Register #	Register Name	Set	Register Data	Register Format
00001	Data Value 1	<input type="checkbox"/>	77.000000	Single Float
00003	Data Value 2	<input type="checkbox"/>	0.000000	Single Float
00005	Data Value 3	<input type="checkbox"/>	0.000000	Single Float

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 multiplied by scale, then added to offset, and then stored in the local register. 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 register 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, it is sometimes necessary to split that register into multiple local registers. 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 5.2 works exactly the same for TCP. In that example, four consecutive read maps reference the same remote register, each selecting a different bit. The first map is selecting bit 0, the second selecting bit 1, and so on.

### 6.3 Modbus TCP Client Write Maps

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

Devices		Client Read Map		Client Write Map			
Showing 1 to 2 of 2				Update		< Prev Next >	
Map #	Local Register #	Remote Type	Remote Register Format	Remote Register #	Remote Device	Name	
1	3	Holding Register	UINT-16	2	ModSim	Data Value 2	
2	0	None	None	0	None		

The data direction is reversed but the same selections are still made. Select the local register 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.



Devices Client Read Map Client Write Map

Map # 1 Update < Prev Next >

Read local register # 3 named Data Value 2

Write remote register ☐ when local register changes by > 0.000000 or ☐ when 0.0 seconds have elapsed with no change.

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

Apply scale: 0.000000 and offset: 0.000000 Then if applicable, apply bit mask: 00000000 and bit fill: 00000000

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

To register # 2 at ModSim unit # 0 With low register first if checked: ☐

Repeat this process ☒ at least ☐ no more than every 5.0 seconds.

☐ Enable this map only when index register 0 is set to a value of 0

# Client Write Maps Enabled: 2 Insert Delete

The local register 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 register 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 register 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 registers 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 register 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 minimum 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.

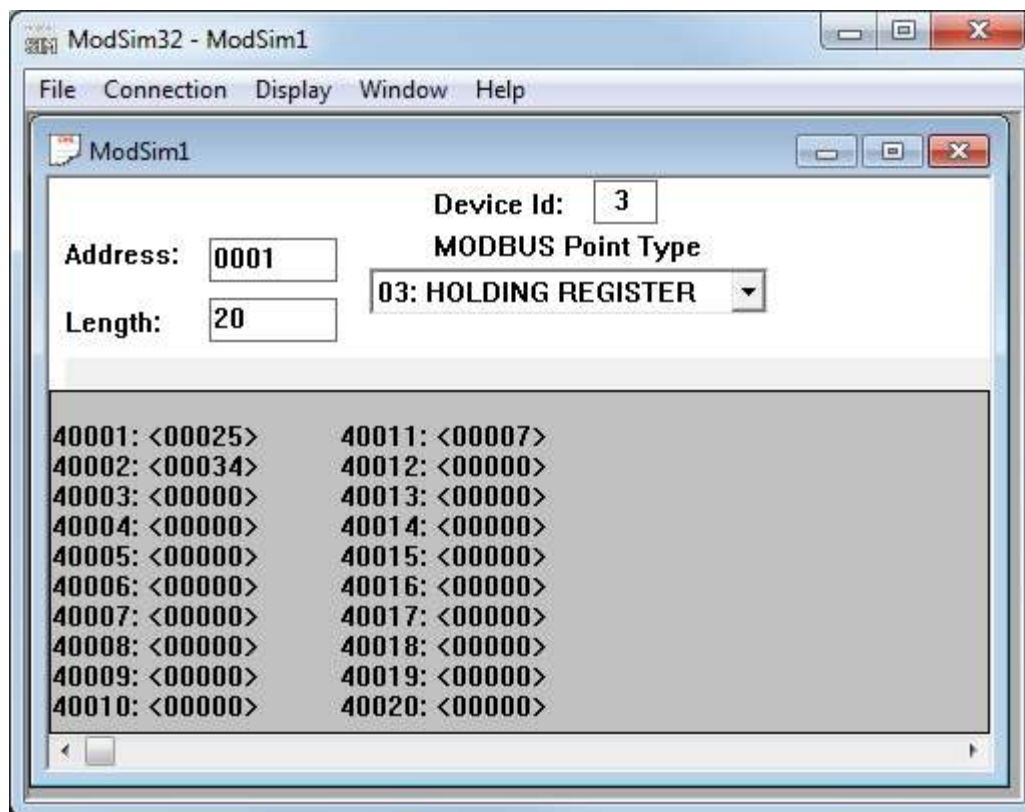
You have the option of making this Write Map conditional. If an index register number is provided and the Enable box is checked, then this write map will only be executed when the index register (local register) contains the value given. This allows multiple write maps to supply data to the same remote register based on the value of the local index register. 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 or to define any map presently having zero for a source register 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.

Local Registers		Calculate	Copy		
Showing registers from 1					Update < Prev Next >
Local Register #	Register Name	Set	Register Data	Register Format	
00001	Data Value 1	<input type="checkbox"/>	77.000000	Single Float	
00003	Data Value 2	<input type="checkbox"/>	33.500000	Single Float	
00005	Data Value 3	<input type="checkbox"/>	0.000000	Single Float	

If the local register and remote register are not the same format, then data is converted automatically when written. In the example above, the Write Map is writing register 3, 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.



## 6.4 Modbus TCP Client Data Displayed by Server

The TCP Registers page shows a list of local registers 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.



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 Registers		Error Counts	Errors: Read Maps	Errors: Write Maps			
TCP Device #1		Showing 1 to 2 of 2		Update	< Prev	Next >	
Dir.	Reg. Type	Remote Reg. #	Register Name	Local Reg. #	Update	Register Data	Time since Last update
From	Holding Reg	00001	Data Value 1	00001	<input type="checkbox"/>	77.000000	1.970
To	Holding Reg	00002	Data Value 2	00003	<input type="checkbox"/>	33.500000	1.730
TCP Device # 1		< Prev Dev		Next Dev >			

## 6.5 Modbus TCP Errors

The Error Counts page shows a tabulation, by TCP device, of all errors observed. In the example below, we can see that TCP device #1 is apparently not configured correctly as it is getting as many exception errors as messages sent.

 					
Local Data		Modbus	SNMP	System	
Serial Port		TCP Setup		TCP Data	Server Remap
TCP Registers		Error Counts	Errors: Read Maps	Errors: Write Maps	
					Update
Device	Reset	Total Messages	No Responses	Exceptions	
1	<input type="checkbox"/>	1146	0	1146	
2	<input type="checkbox"/>	0	0	0	
3	<input type="checkbox"/>	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.

TCP Registers

Error Counts

Errors: Read Maps

Errors: Write Maps

Update

Map #	Register Name	Error Description	Exception Code
1	Data Value 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.



TCP Registers		Error Counts	Errors: Read Maps	Errors: Write Maps	
Update					
Device	Reset	Total Messages	No Responses	Exceptions	
1	<input type="checkbox"/>	0	4	0	
2	<input type="checkbox"/>	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.

TCP Registers

Error Counts

Errors: Read Maps

Errors: Write Maps

Update

Map #	Register Name	Error Description	Exception Code
1	Data Value 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.

Devices	Client Read Map	Client Write Map		
Device # 1	Update < Prev Next >			
Local Name <input type="text" value="Device 1"/>				
Use <input checked="" type="radio"/> Static IPv4 <input type="radio"/> Static IPv6 <input type="radio"/> Domain Lookup				
IP Address <input type="text" value="192.168.1.134"/>			Port: <input type="text" value="502"/>	
Domain Name <input type="text"/>				
Unit (optional) <input type="text" value="1"/>		<input type="checkbox"/> Use FC 5/6 instead of 15/16		
Default Poll Period <input type="text" value="5.0"/> Seconds			Connection Status <input type="text" value="118"/> <input type="button" value="Clear"/>	

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





## 7. Configuring Gateway as a Modbus TCP Server

### 7.1 Modbus TCP Device Configuration

There is really little to do to configure the Babel Buster BB3-6101/MX-61 to be a Modbus TCP server. The gateway needs an IP address and you have already set that via the Network page. The only other thing is to verify that the Modbus Port number is set to a non-zero number. Port 502 is the port set aside for standard Modbus TCP use and should be used unless you have a specific reason not to.

If you will not be using Modbus TCP and wish to disable it, enter zero for Modbus Port, and click Set Ports. Following the next restart, you will be unable to connect via Modbus TCP with port set to zero.

**IMPORTANT:** The Modbus port will be initially set to zero as shipped from the factory. You will need to change it to 502 and restart before connecting via Modbus TCP for the first time.

The screenshot shows the 'Network' tab of the Babel Buster configuration interface. It has a top navigation bar with 'Config File', 'Network', 'Resources', and 'User'. Below the navigation bar, there are three small square icons. The main area contains three input fields: 'HTTP Port' with the value '80' and '(default 80)', 'Modbus Port' with the value '502' and '(default 502)', and 'Telnet Port' with the value '0' and '(default 23)'. A 'Set Ports' button is located to the right of the input fields.

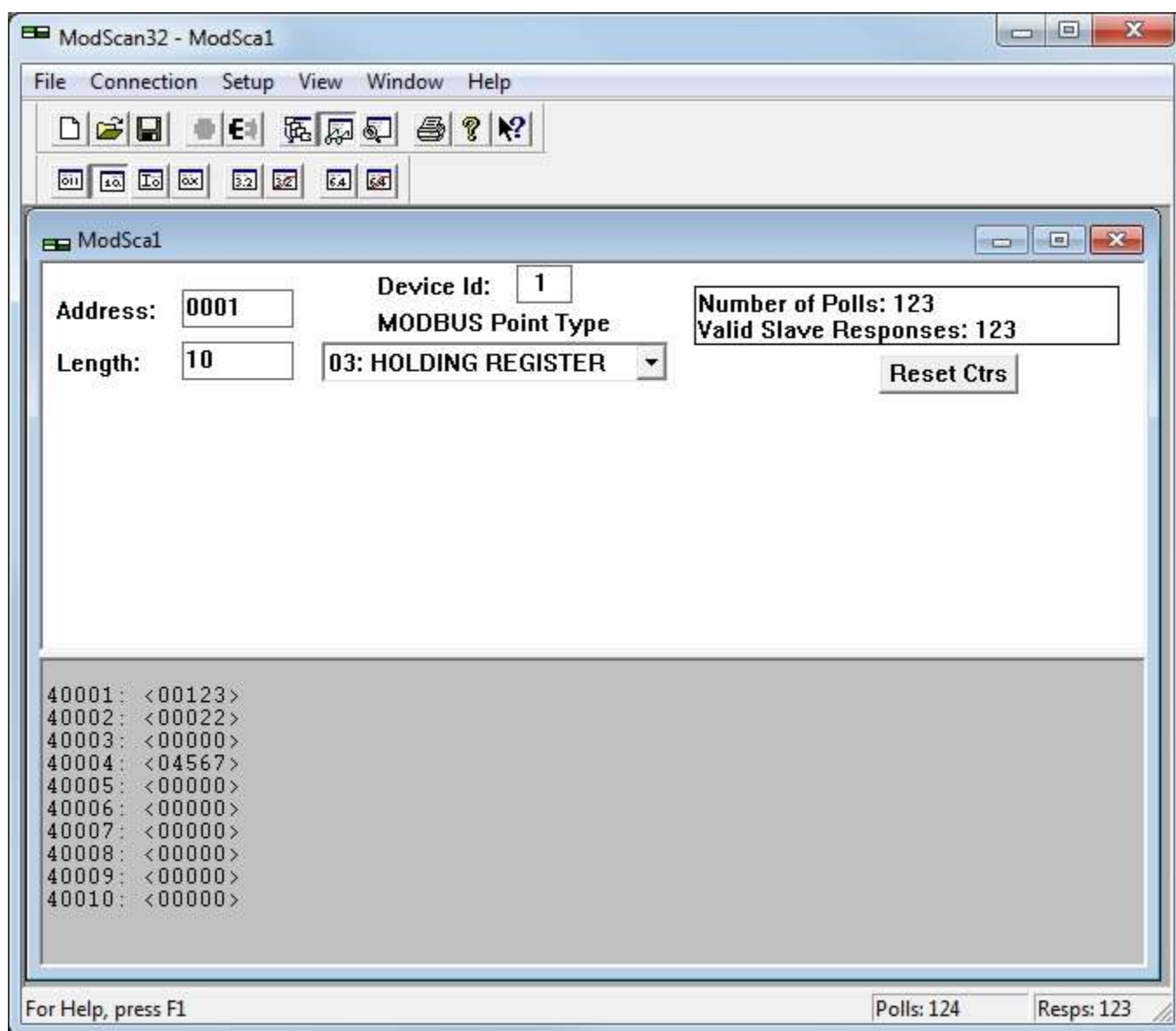
### 7.2 Modbus TCP Server Register Map

The local registers in the Babel Buster BB3-6101/MX-61 will be most often accessed as holding registers, but can also be accessed as input registers (for reading but input registers cannot be written to). If the local register is defined as 16-bit signed or unsigned, then it can also be accessed as a coil or discrete input (for reading). When accessed as a single bit Modbus register, the value read by the Modbus master will be 0, or 1 if the local register contains 1 or any other non-zero value. Of course the remote master can only write 0 or 1 to a coil. Note also that a local register defined as something bigger than 16-bit cannot be accessed as a coil or discrete input.

The register numbers that the remote Modbus TCP client should read or write are simply those shown in the first column on the Local Registers page.

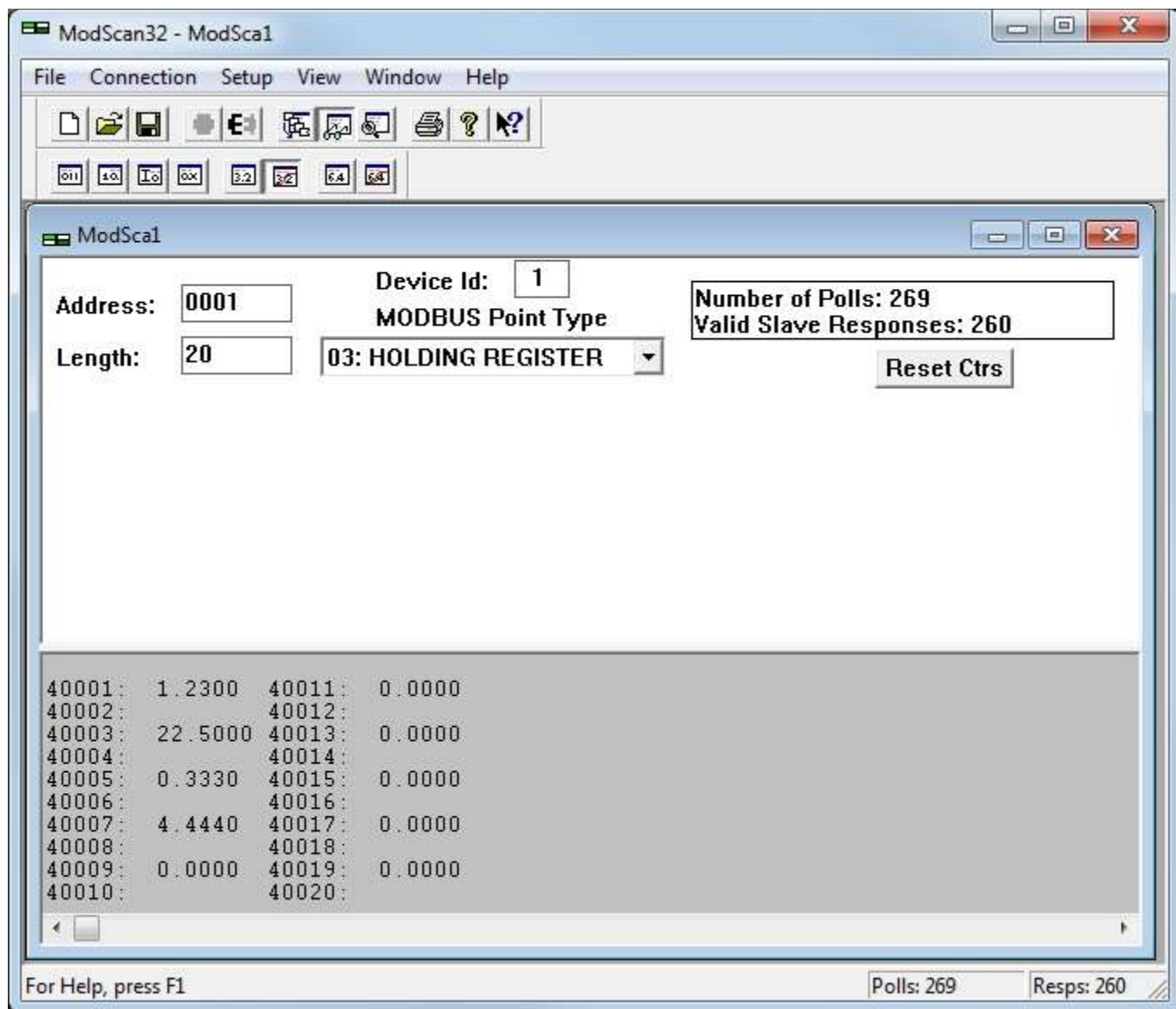
Local Data		Modbus	IoT Cloud	System
Data				
Local Registers		Calculate	Copy	
Showing registers from 1 <input type="text"/> <input type="button" value="Update"/> <input type="button" value="Prev"/> <input type="button" value="Next"/>				
Local Register #	Register Name	Set	Register Data	Register Format
<a href="#">00001</a>	Data Value 1	<input type="checkbox"/>	123	Unsigned 16-bit
<a href="#">00002</a>	Data Value 2	<input type="checkbox"/>	22	Unsigned 16-bit
<a href="#">00003</a>	Data Value 3	<input type="checkbox"/>	0	Unsigned 16-bit
<a href="#">00004</a>	Data Value 4	<input type="checkbox"/>	4567	Unsigned 16-bit
<a href="#">00005</a>	Data Value 5	<input type="checkbox"/>	0	Unsigned 16-bit
<a href="#">00006</a>	Data Value 6	<input type="checkbox"/>	0	Unsigned 16-bit

If the local registers are defined as unsigned 16-bit and contain the data illustrated above, then ModScan will read them as illustrated below.



Local Registers		Calculate	Copy		
Showing registers from 1					Update < Prev Next >
Local Register #	Register Name	Set	Register Data		Register Format
00001	Data Value 1	<input type="checkbox"/>	1.230000		Single Float
00003	Data Value 2	<input type="checkbox"/>	22.500000		Single Float
00005	Data Value 3	<input type="checkbox"/>	0.333000		Single Float
00007	Data Value 4	<input type="checkbox"/>	4.444000		Single Float
00009	Data Value 5	<input type="checkbox"/>	0.000000		Single Float

If the local registers are defined as floating point and contain the data illustrated above, then ModScan will read them as illustrated below. Note that registers defined as having data formats greater than 16 bits must be read on the proper boundaries. You are not allowed to read half of a floating point number or just part of any other larger data value. If you were to attempt to read register #2 in the this example, you would get an exception error. This also means you cannot read just 2 characters in the middle of a larger character string. When the register is defined as character string, you must access the entire character string as a collective set of registers (i.e. read/write multiple holding registers in a single request).



### 7.3 Modbus TCP Server Diagnostic

The Babel Buster brand new out of the box will have no registers configured. Although no registers are configured, there will be a single holding register accessible 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".

### 7.4 Modbus TCP Server Register Remapping

Remote Modbus clients (masters) will normally read the register numbers indicated on the Local Data page. In some cases, you may want to reassign those registers to be at arbitrary numbers all over the map in order to emulate some piece of existing equipment. When you have this requirement, this page is where you create your arbitrary register map.

Server remapping is automatically in effect any time maps are displayed on this page. If nothing is displayed here, then refer to the Local Data page to see what Modbus register numbers your client should read when this gateway is acting as a server. If anything is displayed here, then only these registers will be recognized by the server and any attempt to read registers not listed here will result in an exception being returned.

Showing 1 to 5 of 5

Update < Prev Next >

Map #	Remap this many registers	Starting at local register #	Mapping as	Starting at remote register #
1	5	1	Holding Register(s)	28901
2	5	6	Input Register(s)	28901
3	2	11	Discrete Input(s)	28901
4	2	13	Coil(s)	28901
5	0	0	None	0

☐ Confirm Clear All Insert Delete

To remap registers, start by entering a count of registers to remap. With a count greater than one, you can remap many registers in a row with just one line on the above table. Provide the starting local register number - this is the number displayed on the Local Data page. Select what type of Modbus register you want this to pretend to be. And finally, provide the starting register number that you want the remote Modbus client to see these registers as. Click Update to accept the line and make room for another.

The example above illustrates the following: Local registers 1 through 5 will be read as holding registers 28901 through 28905. Local registers 6 through 10 will be read as input registers 28901 through 28905. Local registers 11 and 12 will be read as discrete inputs 28901 and 28902. Local registers 13 and 14 will be read as coils 28901 and 28902. The holding registers and coils may also be written at these register numbers.

When this table is complete, be sure to go to the File Manager page and save your updated configuration.

Use Prev and Next to scroll through the table if it is larger than a few lines. Delete will remove the map number shown in the "Showing" 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 "none" for a register type (unused rule).

To completely erase this entire table, check the "Confirm" box and click Clear All.

Modbus holding registers default to being 16-bit integer. If your remote client expects to see floating point, then create the local register as floating point on the Local Data



page. The remote client will have access to whatever data types are used to create the local registers on the Local Data page.

When holding register (or input register) data is more than 16 bits, the register order matters, but is not standardized by Modbus protocol. The order in which multiple registers will be provided to the remote client is part of the definition of the local register created on the Local Data page. Click on the register number in the first column on that page to see how that register is defined.

Modbus sometimes "packs" multiple data items into a single holding register. You may "pack" registers such that a single register viewed remotely is constructed from the contents of multiple local registers. To accomplish this, refer to the pack and fill operations found on the Calculate page under Local Data.



## 8. Configuring Gateway as an SNMP Server

### 8.1 Creating Local SNMP MIB

The Babel Buster BB3-6101/MX-61 starts out with no variables in its MIB just like it starts out with no local registers. When you do create local registers, you then have a choice of where to make them show up in the MIB. There are five branches in the BB3 MIB, although only the Integer branch is guaranteed to be universally accessible to all other SNMP devices. It is up to you to select which branch of the MIB to place each local register in. You do not need to place all local registers in the MIB, only those that you want externally accessible via SNMP. You must also place local registers in the MIB in order to generate SNMP traps related to those local registers.

Map #	Local SNMP OID	Local Register #	Scale Factor	Local Value	Local Name
1	1.3.6.1.4.1.3815.1.6.1.1.1.2.1	0	x1	---	---

To add a local register to a MIB branch, simply enter the local register number at the next available OID which will always automatically be at the bottom of the list. When placing registers in the Integer branch, you also have the option of applying a scale factor. Scaled integer is the most universally recognized means of transmitting non-integer data.

Integer 32-bit    **Unsigned 64-bit**    Float 32-Bit    Float 64-Bit    Char String

Showing 1 to 1 of 1    Update    < Prev    Next >

Map #	Local SNMP OID	Local Register #	Scale Factor	Local Value	Local Name
1	1.3.6.1.4.1.3815.1.6.1.1.1.2.1	0	x1	---	---

Auto-fill 0 variables starting at index 1 with    Auto-Fill    ☐ Confirm    Clear

Reload SNMP    Map # 1    Remove    Insert Before

**Quick Help**

Rule number simply tells you where you're at on the list of the list. To advance directly to a specific map, enter the desired number in the "Rule" box, then click Update.

This page enables SNMP Get/Set to registers indicated on the list. The available local OID's are assigned automatically. You may select which local registers are mapped to these OID's.

Internal data is multiplied by the scale factor when read by your SNMP manager (client). Data written by your SNMP client is divided by the scale factor before being stored internally. If the register is an integer value, then scale factor will most often be 1 (no scaling). However, if the register is a floating point register, the scale factor becomes more important. This is significant because floating point representations do exist, they are not universally accepted. Therefore, the oldest and best known recommendation is to use integers via SNMP is to scale them and send them as integers, or send them as ASCII strings (supported by the Char String MIB).

Local Registers    Calculate    Copy

Showing registers from 1

Update    < Prev    Next >

Local Register #	Register Name	Set	Register Data	Register Format
00001	Data Value 1	<input type="checkbox"/>	5.190000	Single Float
00003	Data Value 2	<input type="checkbox"/>	0.000000	Single Float

In this example, we have selected a scale factor of x100. That means that our local value of 5.19 will be transmitted (in a Get response) as integer 519 and it is up to the recipient to know that this is scaled x100.

Integer 32-bit    **Unsigned 64-bit**    Float 32-Bit    Float 64-Bit    Char String

Showing 1 to 2 of 2    Update    < Prev    Next >

Map #	Local SNMP OID	Local Register #	Scale Factor	Local Value	Local Name
1	1.3.6.1.4.1.3815.1.6.1.1.1.2.1	1	x100	5.190000	Data Value 1
2	1.3.6.1.4.1.3815.1.6.1.1.1.2.2	0	x1	---	---

Auto-fill 0 variables starting at index 1 with local register 1    Auto-Fill    ☐ Confirm    Clear

Reload SNMP    Map # 1    Remove    Insert Before

After adding new members to the MIB, it is necessary to click Reload SNMP before they will become accessible to an external SNMP manager's Get request.

Integer 32-bit	Unsigned 64-bit	Float 32-Bit	Float 64-Bit	Char String
----------------	-----------------	--------------	--------------	-------------

Showing 1 to 2 of 2 Update < Prev Next >

Map #	Local SNMP OID	Local Register #	Local Value	Local Name
1	1.3.6.1.4.1.3815.1.6.1.3.1.1.2.1	3	82.255997	Data Value 2
2	1.3.6.1.4.1.3815.1.6.1.3.1.1.2.2	0	---	---

Auto-fill 0 variables starting at index 0 with local register 0 Auto-Fill ☐ Confirm Clear

Reload SNMP 1 Float32 vars loaded Map # 1 Remove Insert Before

Local registers added to the Float 32-Bit MIB branch will be provided in IEEE 754 format per RFC 6340.

Local Registers	Calculate	Copy
-----------------	-----------	------

Showing registers from 1 Update < Prev Next >

Local Register #	Register Name	Set	Register Data	Register Format
00001	Data Value 1	<input type="checkbox"/>	5.190000	Single Float
00003	Data Value 2	<input type="checkbox"/>	89.255997	Single Float
00005	Data Value 3	<input type="checkbox"/>	0.000000	Single Float

Integer 32-bit	Unsigned 64-bit	Float 32-Bit	Float 64-Bit	Char String
----------------	-----------------	--------------	--------------	-------------

Showing 1 to 2 of 2 Update < Prev Next >

Map #	Local SNMP OID	Local Register #	Local Value	Local Name
1	1.3.6.1.4.1.3815.1.6.1.5.1.1.2.1	21	Test String	Character String 1
2	1.3.6.1.4.1.3815.1.6.1.5.1.1.2.2	0	---	---

Auto-fill 0 variables starting at index 0 with local register 0 Auto-Fill ☐ Confirm Clear

Reload SNMP 1 Char vars loaded Map # 1 Remove Insert Before

Local registers defined as character strings may be added to the Char String branch of the MIB. Registers in this branch will be provided as an Octet String.



Local Registers		Calculate	Copy		
Showing registers from 1					Update < Prev Next >
Local Register #	Register Name	Set	Register Data		Register Format
00001	Data Value 1	<input type="checkbox"/>	5.190000		Single Float
00003	Data Value 2	<input type="checkbox"/>	89.255997		Single Float
00005	Data Value 3	<input type="checkbox"/>	0.000000		Single Float
00007	Data Value 4	<input type="checkbox"/>	0.000000		Single Float
00009	Data Value 5	<input type="checkbox"/>	0.000000		Single Float
00011	Data Value 6	<input type="checkbox"/>	0.000000		Single Float
00013	Data Value 7	<input type="checkbox"/>	0.000000		Single Float
00015	Data Value 8	<input type="checkbox"/>	0.000000		Single Float
00017	Data Value 9	<input type="checkbox"/>	0.000000		Single Float
00019	Data Value 10	<input type="checkbox"/>	0.000000		Single Float
00021	Char String 1	<input type="checkbox"/>	Test String		Char String[40]
00041	Char String 2	<input type="checkbox"/>			Char String[40]

The "Auto-fill" at the bottom of each MIB page can be used to very quickly create a long list of sequential MIB table entries. Enter the number of variables to fill, the starting MIB index, and the starting local register number. Then click the Auto-Fill button.

Integer 32-bit	Unsigned 64-bit	Float 32-Bit	Float 64-Bit	Char String	
Showing 1 to 1 of 1					
Update	< Prev	Next >			
Map #	Local SNMP OID	Local Register #	Scale Factor	Local Value	Local Name
1	1.3.6.1.4.1.3815.1.6.1.1.1.2.1	0	x1	---	---
Auto-fill 10 variables starting at index 1 with local register 1 <button>Auto-Fill</button> <input type="checkbox"/> Confirm <button>Clear</button>					
Reload SNMP <input type="text"/> Map # 1 <button>Remove</button> <button>Insert Before</button>					

The example above will create the list illustrated below, assuming the first ten local registers are all 16-bit integer registers.



Integer 32-bit		Unsigned 64-bit		Float 32-Bit		Float 64-Bit		Char String	
Showing 1 to 11 of 11									
<input type="button" value="Update"/> <input type="button" value=" &lt; Prev"/> <input type="button" value="Next &gt;"/>									
Map #	Local SNMP OID	Local Register #	Scale Factor	Local Value	Local Name				
1	1.3.6.1.4.1.3815.1.6.1.1.1.2.1	1	x1	0	Register 1				
2	1.3.6.1.4.1.3815.1.6.1.1.1.2.2	2	x1	0	Register 2				
3	1.3.6.1.4.1.3815.1.6.1.1.1.2.3	3	x1	0	Register 3				
4	1.3.6.1.4.1.3815.1.6.1.1.1.2.4	4	x1	0	Register 4				
5	1.3.6.1.4.1.3815.1.6.1.1.1.2.5	5	x1	0	Register 5				
6	1.3.6.1.4.1.3815.1.6.1.1.1.2.6	6	x1	0	Register 6				
7	1.3.6.1.4.1.3815.1.6.1.1.1.2.7	7	x1	0	Register 7				
8	1.3.6.1.4.1.3815.1.6.1.1.1.2.8	8	x1	0	Register 8				
9	1.3.6.1.4.1.3815.1.6.1.1.1.2.9	9	x1	0	Register 9				
10	1.3.6.1.4.1.3815.1.6.1.1.1.2.10	10	x1	0	Register 10				
11	1.3.6.1.4.1.3815.1.6.1.1.1.2.11	0	x1	---	---				

Auto-fill  variables starting at index  with local register 

☐ Confirm

The example illustrated below is a little more complex. The first ten local registers are floating point, which means they are treated as register pairs. Then the next couple of local "registers" are character strings each 40 characters in length, and each occupying 20 consecutive Modbus registers. Note that the "auto-fill" automatically accounts for data size, and chooses the next local register appropriately. In this example, although the registers are a mix of floating point and character string, they are assigned to the Integer 32-bit branch of the MIB and will be read via SNMP as Integer values. The numbers in the character string registers are ASCII string representations of numbers, but will be converted to integer. For example, MIB index 12, with the character string "5.678", will be returned as an integer value 6 in an SNMP Get (converted and rounded).

Integer 32-bit		Unsigned 64-bit		Float 32-Bit		Float 64-Bit		Char String	
Showing 1 to 13 of 13									
<input type="button" value="Update"/> <input type="button" value=" &lt; Prev"/> <input type="button" value="Next &gt;"/>									
Map #	Local SNMP OID	Local Register #	Scale Factor	Local Value	Local Name				
1	1.3.6.1.4.1.3815.1.6.1.1.1.2.1	1	x1	0.000000	Data Value 1				
2	1.3.6.1.4.1.3815.1.6.1.1.1.2.2	3	x10	0.000000	Data Value 2				
3	1.3.6.1.4.1.3815.1.6.1.1.1.2.3	5	x10	0.000000	Data Value 3				
4	1.3.6.1.4.1.3815.1.6.1.1.1.2.4	7	x1	0.000000	Data Value 4				
5	1.3.6.1.4.1.3815.1.6.1.1.1.2.5	9	x100	0.000000	Data Value 5				
6	1.3.6.1.4.1.3815.1.6.1.1.1.2.6	11	x1	0.000000	Data Value 6				
7	1.3.6.1.4.1.3815.1.6.1.1.1.2.7	13	x1	0.000000	Data Value 7				
8	1.3.6.1.4.1.3815.1.6.1.1.1.2.8	15	x0.1	0.000000	Data Value 8				
9	1.3.6.1.4.1.3815.1.6.1.1.1.2.9	17	x0.01	0.000000	Data Value 9				
10	1.3.6.1.4.1.3815.1.6.1.1.1.2.10	19	x1	0.000000	Data Value 10				
11	1.3.6.1.4.1.3815.1.6.1.1.1.2.11	21	x1	1.234	Character String 1				
12	1.3.6.1.4.1.3815.1.6.1.1.1.2.12	41	x1	5.678	Character String 2				
13	1.3.6.1.4.1.3815.1.6.1.1.1.2.13	0	x1	---	---				

Auto-fill  variables starting at index  with local register 

☐ Confirm

The Auto-Fill simply sets up the MIB mapping table. Before the variables can be accessed by SNMP, you need to click Reload SNMP. The Remove and Insert Before buttons also just set up the mapping table, and you need to click Reload SNMP to make the new assignments accessible.

To clear the mapping table displayed, click the Confirm box and then click the Clear button. This clears the mapping table, but as with Auto-Fill, no changes (from the perspective of SNMP as viewed from the outside) take place until you click Reload SNMP. The Clear button is mainly useful if you change your mind about the auto-fill you just did.

Due to SNMP's memory allocation scheme, you can always increase the size of the MIB by simply clicking Reload SNMP; however, to reduce the size of the MIB, you need to save your configuration on the File Manager page, and then restart the gateway. If you reduce the number of entries in the mapping table, the original MIB variables at the end of the original list will simply return zero or empty strings until the gateway is restarted.

## 8.2 Supported Data Formats, RFC 6340

SNMP does not have a universally accepted representation for floating point. The one universally known data type is INTEGER. A commonly recommended means of transmitting floating point data is either as a scaled integer or as an ASCII character string. There is an RFC 6340 for representation of floating point based on IEEE 754 encoding. The "Float 32-bit" and "Float 64-bit" data types in the Babel Buster refer to RFC 6340 encoding.

Specifically, the data types found in the Babel Buster MIB are encoded with ASN types as follows:

Integer 32-Bit	INTEGER	ASN_INTEGER
Unsigned 64-Bit	COUNTER64	(ASN_APPLICATION   6)
Float 32-bit	OCTET STRING	ASN_OCTET_STR (length 4)
Float 64-bit	OCTET STRING	ASN_OCTET_STR (length 8)
Char String	OCTET STRING	ASN_OCTET_STR (length variable)

### 8.3 SNMPv3 Users, Authentication, Privacy

SNMPv3 user authentication is established on this page. The only difference between SNMPv2 and SNMPv3 is the user authentication.

**Babel Buster<sup>®</sup> 3**  
MODBUS-SNMP  
NETWORK GATEWAY  
MODEL BB3-6101-V3

**CONTROL SOLUTIONS MINNESOTA**

Local Data   Modbus   **SNMP**   System

Local MIB   **Security**   Trap Sender

User(s)   Agent ID

Update

User Name	Auth Type	Authentication Passphrase	Priv Type	Privacy Passphrase	Delete
jimh	MD5	jimsauth	DES	jimspriv	<input checked="" type="checkbox"/>
	None		None		<input type="checkbox"/>
	None		None		<input type="checkbox"/>

Enter the user name. Select authentication type and provide an authentication passphrase (password) if applicable. Select privacy type and provide a privacy passphrase (password) if applicable.

You may have an insecure user with no authentication or privacy. You may have a user with authentication but no privacy. You may have users with both authentication and privacy (encryption). However, you may not have privacy without authentication.

You may use this gateway as an SNMPv2 agent. In this case, a default SNMPv3 user is automatically created for you, but will not appear on the above list.

Check Delete and then Update to remove the user from the list.

### 8.4 Agent ID

The name, location, and contact listed here may be retrieved by the remote SNMP

client under the SNMP MIB-2 System branch starting at 1.3.6.1.2.1.1.

The image shows a configuration interface for an SNMPv3 engine. At the top, there are fields for 'User(s)' and 'Agent ID'. Below these, there are three input fields: 'System Name' with the value 'Babel Buster BB3-6101', 'System Location' with 'St. Paul, Minnesota', and 'System Contact' with 'www.csimn.com'. An 'Update' button is to the right of the System Name field. Below these fields, there is a section for 'SNMPv3' configuration. It includes an 'Engine ID' field with a long hexadecimal value '80000EE702FE80000000000000002409DFFFE45464E', an 'Engine Boots' field with the value '7' and a 'Set New' button, and a 'Seconds Since Start' field with the value '36713'. Below this, there is a section for 'SNMPv1/v2c' configuration, which includes a checkbox for 'Allow SNMPv1/v2c' and a 'v1/v2c Community' field with the value 'private'.

Browsing the MIB-2 System variables would return the following given the above example. Some of the System variables are either fixed or generated automatically.

The image shows a screenshot of the 'iReasoning MIB Browser' application. The window has a menu bar with 'File', 'Edit', 'Operations', 'Tools', 'Bookmarks', and 'Help'. Below the menu bar, there is a toolbar with 'Address' (set to '192.168.1.125'), 'Advanced...', 'OID' (set to '.1.3.6.1.2.1.1.7.0'), 'Operations' (set to 'Get Next'), and a 'Go' button. On the left side, there is a 'SNMP MIBs' tree view showing a hierarchy of MIBs, with 'iso.org.dod.internet' expanded. The main area of the window is a 'Result Table' with the following data:

Name/OID	Value	Type	IP:Port
sysDescr.0	Babel Buster BB3-6101	OctetString	192.168.1.1...
sysObjectID.0	bb3-reg-v1	OID	192.168.1.1...
sysUpTime.0	13 minutes 6 seconds (78684)	TimeTicks	192.168.1.1...
sysContact.0	www.csimn.com	OctetString	192.168.1.1...
sysName.0	Babel Buster BB3-6101	OctetString	192.168.1.1...
sysLocation.0	St. Paul, Minnesota	OctetString	192.168.1.1...
sysServices.0	72	Integer	192.168.1.1...

## 8.5 SNMPv3 Engine Info

Engine ID currently in use is displayed. The default engine ID is created per RFC 3411 based on the IPv6 IP address (required since the device is operating in dual stack mode). If a static IPv6 address is configured, that will be used, otherwise the auto-configured IPv6 address (link-local) will be used. The auto-configured IPv6 address is derived from the device's MAC address.



User(s)	Agent ID				
System Name	Babel Buster 883-6101				Update
System Location	St. Paul, Minnesota				
System Contact	www.csinn.com				
SNMPv3					
Engine ID	80000EE702FE800000000000000002409DFFFE45464E				
Engine Boots	7	<input type="checkbox"/>	Set New		
Seconds Since Start	36713				
SNMPv1/v2c					
	<input type="checkbox"/> Allow SNMPv1/v2c				
v1/v2c Community	private				

Engine Boots is the number of times this device has booted up. You normally have no need to alter this count. However, if you are replacing an existing SNMPv3 device, you should set the boots count to whatever the count was in the device being replaced since part of SNMPv3 security is to see that engine boots is incrementally bigger than before if it changes at all. To change the boot count, enter the new count, check the Set New box, click Update and then (after the page finishes refreshing), restart this device.

Seconds since start is displayed for information only.

## 8.6 SNMPv2 Community

Check Allow if SNMP v1/v2c should be permitted to access the MIB in this device. The v1/v2c community must be used by the remote SNMP v1/v2c client to Get or Set this device. The name "public" is no longer accepted (unless you explicitly enter it here - not recommended). Changes to the allow/disallow status for v1/v2c will take effect upon the next restart.

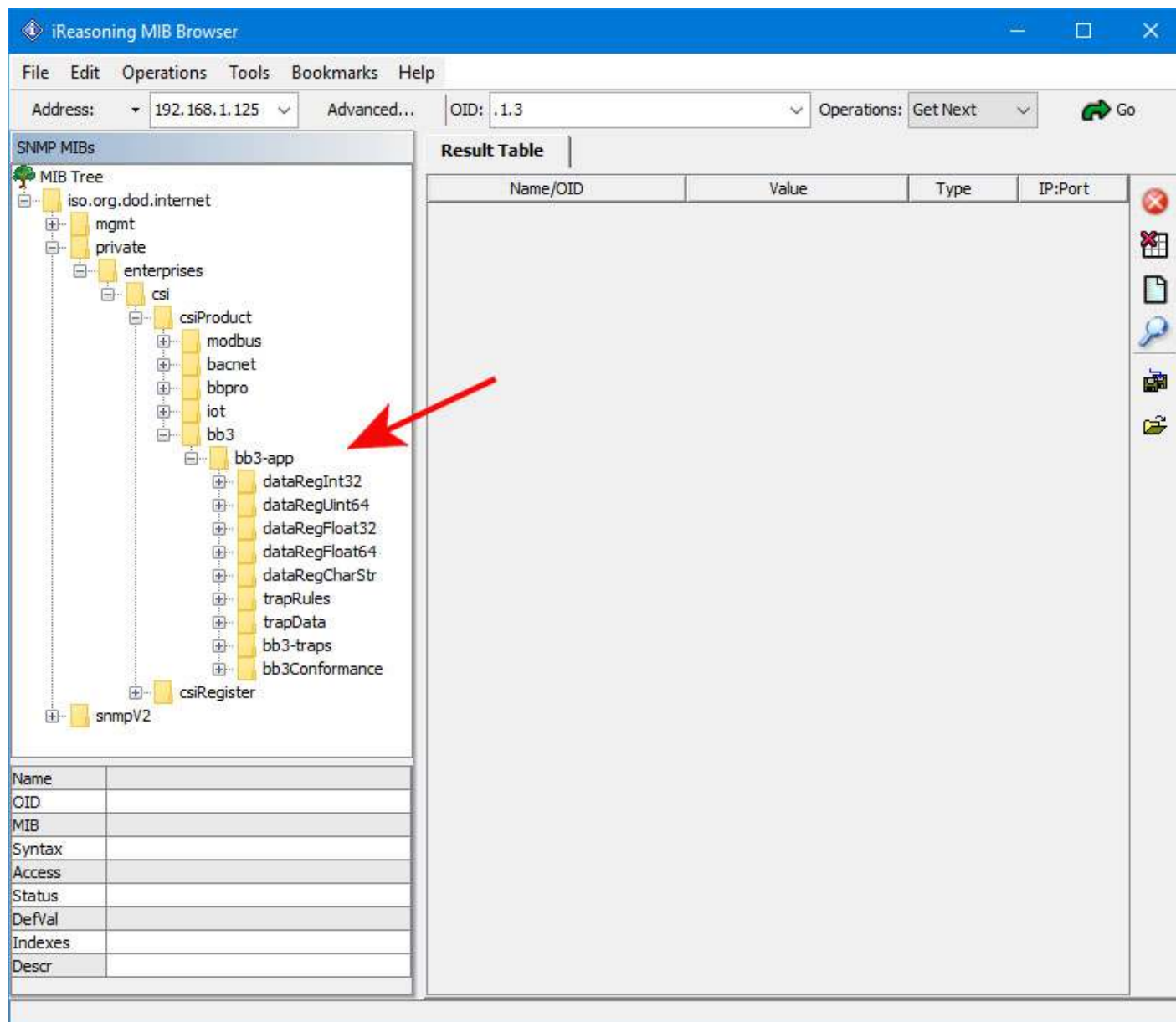


User(s)	Agent ID				
System Name	Babel Buster BB3-6101				Update
System Location	St. Paul, Minnesota				
System Contact	www.csimn.com				
SNMPv3					
Engine ID	80000EE702FE800000000000000002409DFFFE45464E				
Engine Boots	7	<input type="checkbox"/>	Set New		
Seconds Since Start	36713				
SNMPv1/v2c					
	<input type="checkbox"/>	Allow SNMPv1/v2c			
v1/v2c Community	private				

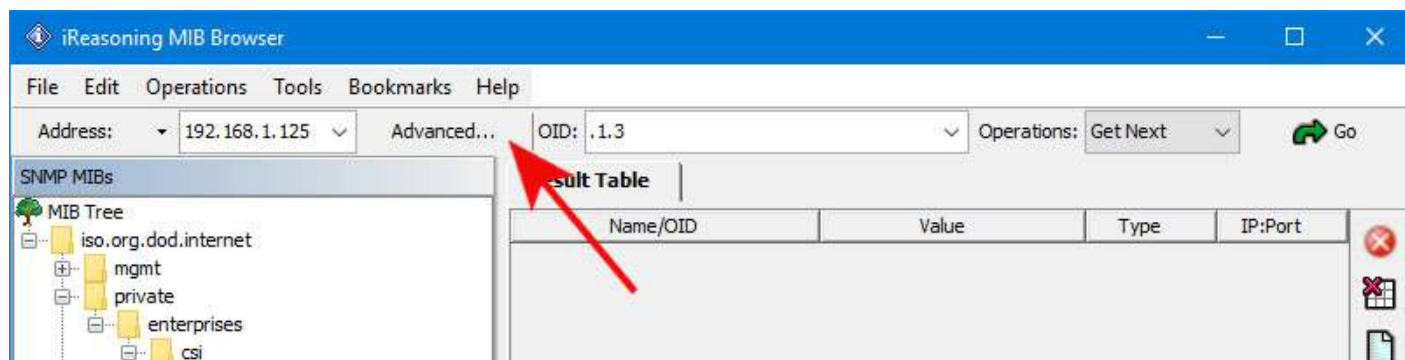
## 8.7 Testing the SNMP Agent

A variety of tools are available for browsing an SNMP MIB and receiving SNMP Traps. The tool used in the following examples is the iReasoning MIB Browser. Refer to the Tools section under Support at csimn.com for more information about SNMP tools.

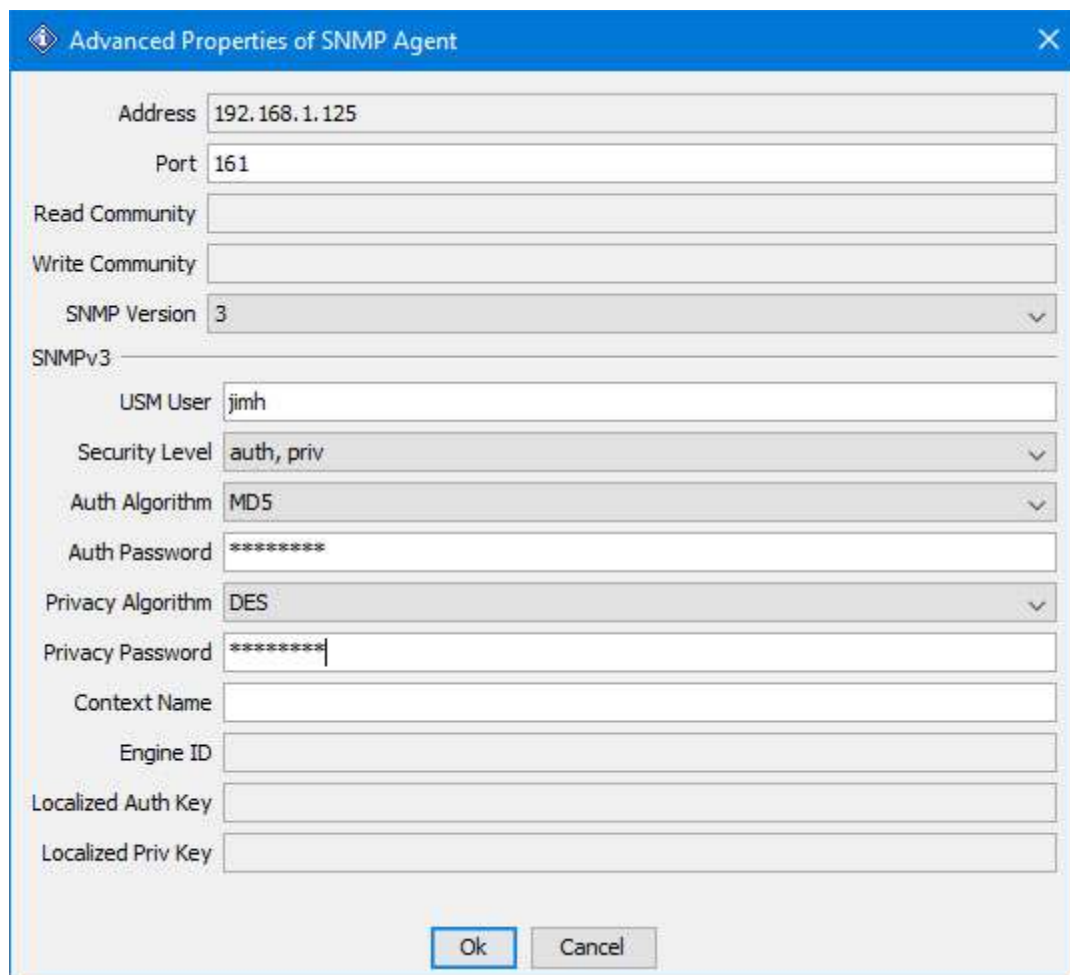
The MIB browser allows you to view the MIB variables in the Babel Buster. Before you can browse the MIB in a meaningful way, you need to load the MIB files that tell the browser what it needs to know about the BB3's MIB. There are one or more files that need to be loaded, and these are available under Documents and Links on the BB3-6101/MX-61 product page at csimn.com. Once you have loaded these files as illustrated below, you can view the tree structure of the MIB in the browser.



Enter the IP address of the Babel Buster BB3-6101/MX-61 in the Address window. In addition, click on Advanced... to set access parameters.



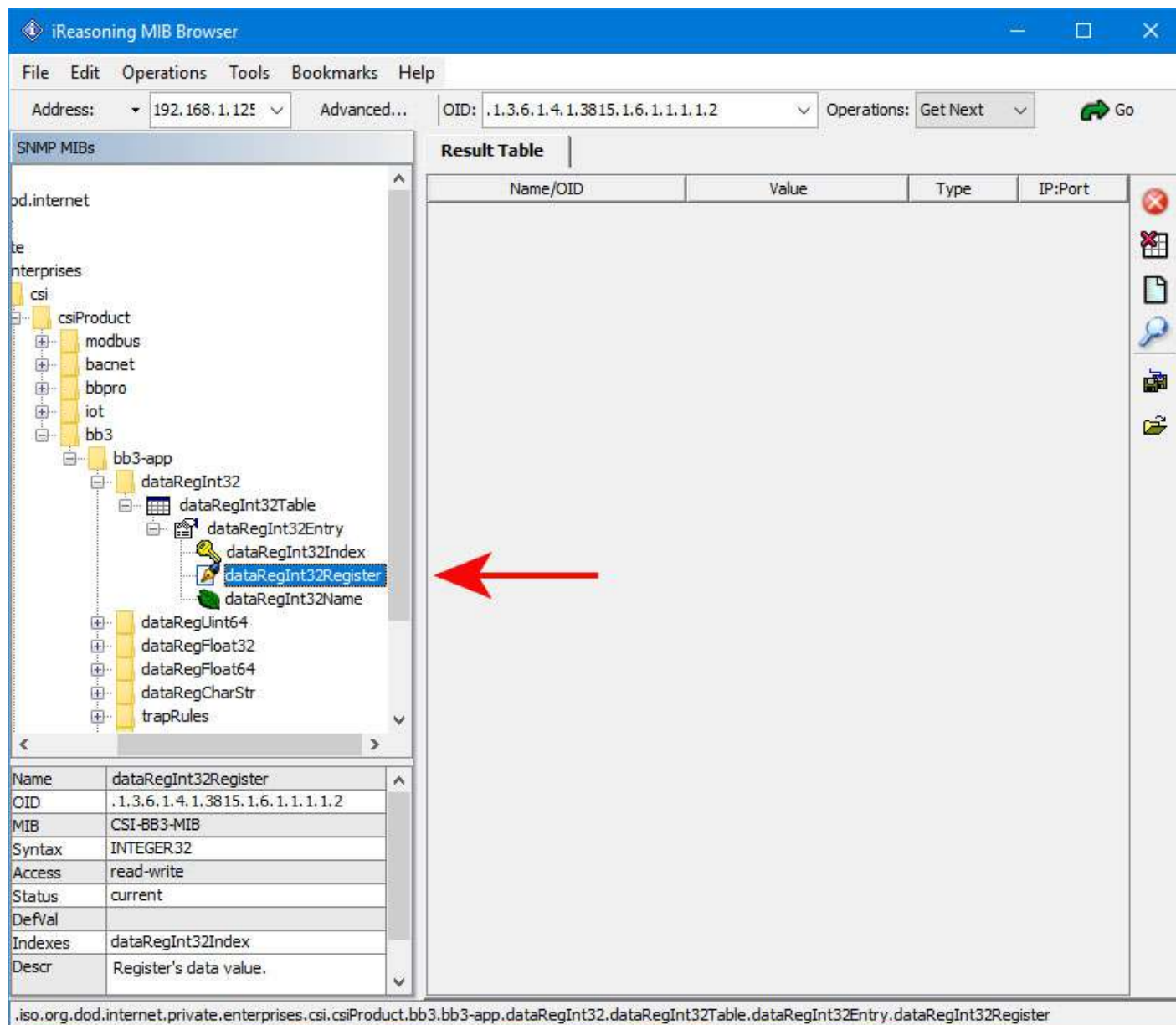
Click Advanced... to open the dialog that lets you enter the user name and password(s) required for accessing the MIB in the Babel Buster.



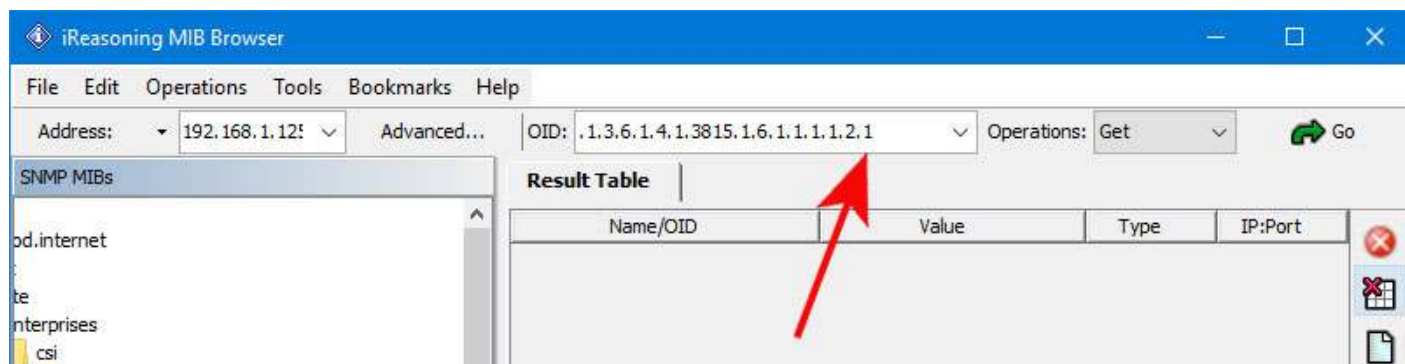
The image shows a Windows-style dialog box titled "Advanced Properties of SNMP Agent". It contains several input fields and dropdown menus for configuring an SNMP agent. The fields are arranged in a vertical list. At the bottom, there are "Ok" and "Cancel" buttons.

Address	192.168.1.125
Port	161
Read Community	
Write Community	
SNMP Version	3
SNMPv3	
USM User	jimh
Security Level	auth, priv
Auth Algorithm	MD5
Auth Password	*****
Privacy Algorithm	DES
Privacy Password	*****
Context Name	
Engine ID	
Localized Auth Key	
Localized Priv Key	

To read a MIB variable from the Babel Buster, start by selecting the register member of the data table. In this case, we are selecting the 32-Bit Integer branch of the MIB.

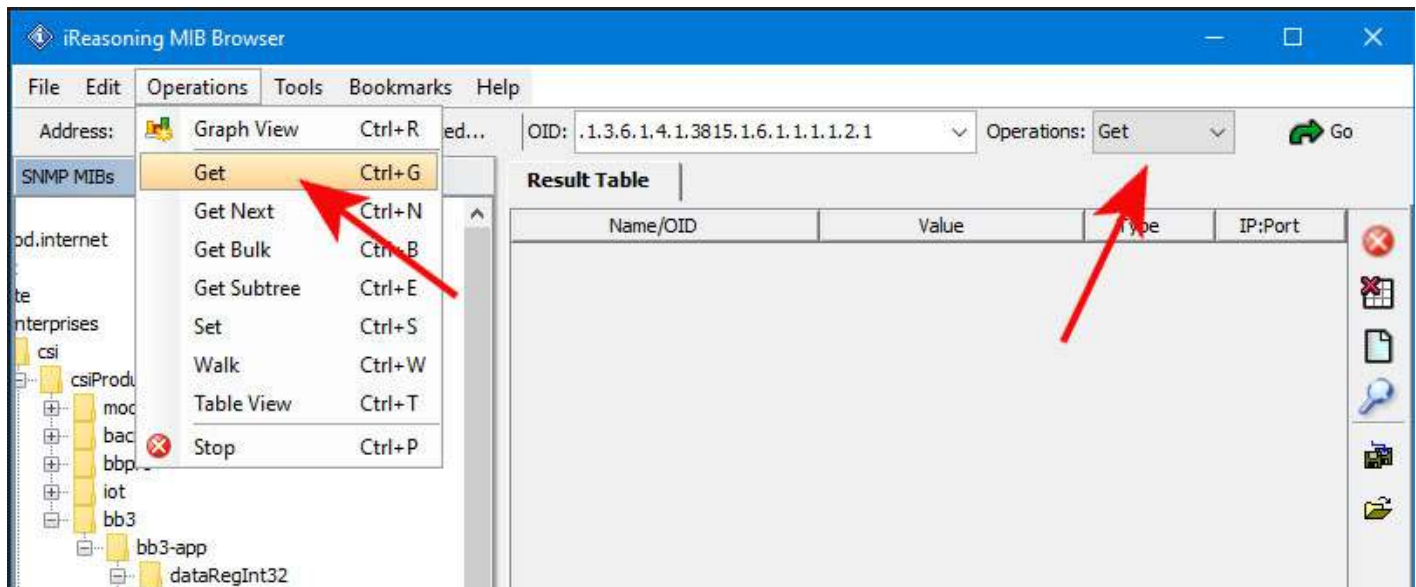


You will need to specify which row in the table you want to read. Do this by appending (by typing) a number to the OID that appeared in the OID window when you clicked on the table entry in the MIB tree view.

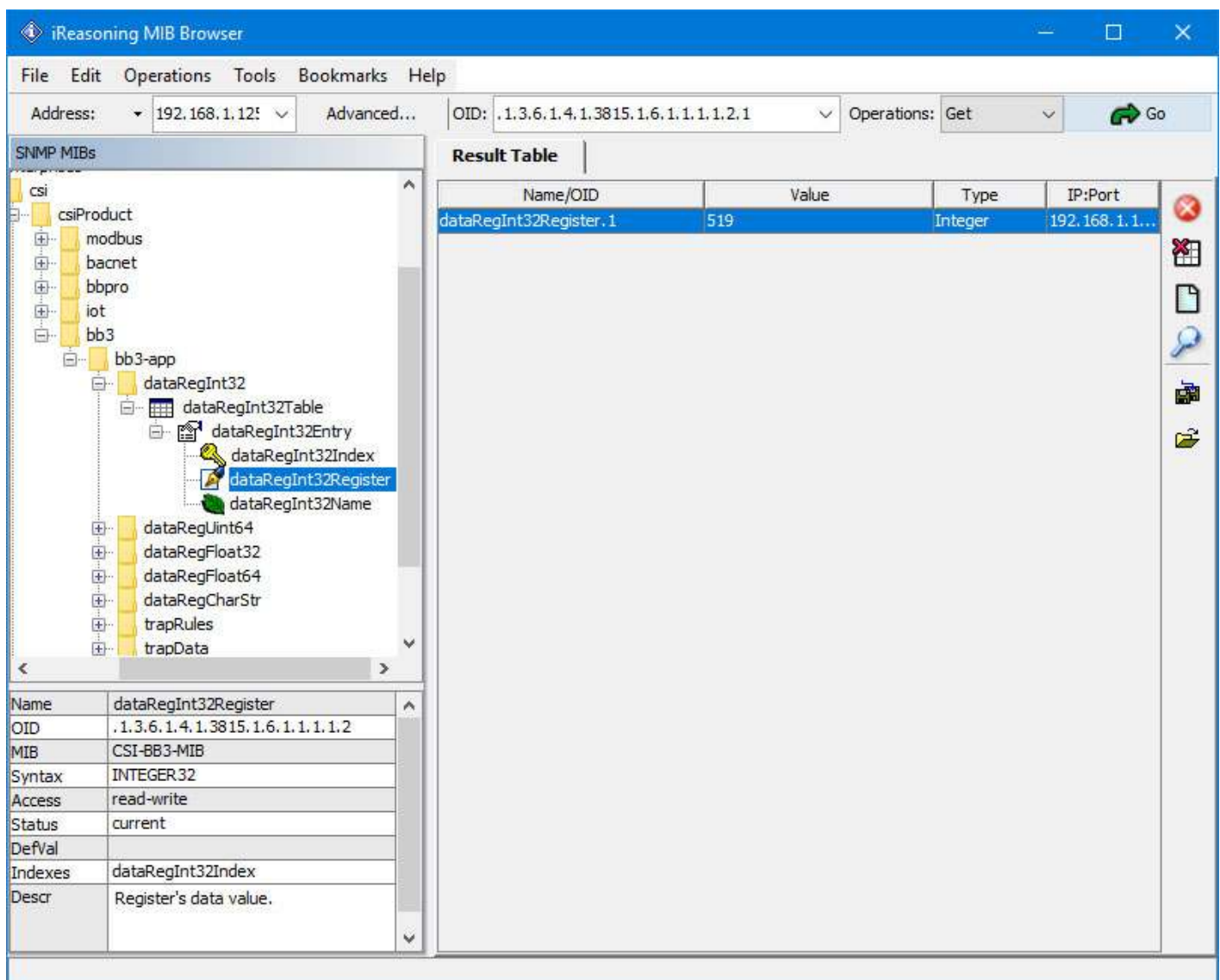


To cause the MIB browser to Get a value from the Babel Buster, select Get from the Operations menu. You can select Get from either menu, left or right, in the iReasoning browser. Once Get is selected on the right, you only need to click the Go button to

repeat the Get.



Upon successfully Getting a value, the display will appear as illustrated below.





If you attempt to Get a variable that does not exist, you will get the error message illustrated below. This will also happen if the variable has been added to the MIB but you forgot to click Reload SNMP after adding the local register to the MIB. This will also happen if you did not select the right variable from the MIB tree shown, or forgot to add the index to the end of the OID in the Object ID window.

The screenshot shows the iReasoning MIB Browser interface. The Address field is set to 192.168.1.12 and the OID is .1.3.6.1.4.1.3815.1.6.1.1.1.2.88. The Operations dropdown is set to Get. The Result Table displays the following data:

Name/OID	Value	Type	IP:Port
dataRegInt32Register.1	519	Integer	192.168.1.1...
dataRegInt32Register.88	No Such Instance	NoSuchInst...	192.168.1.1...

The MIB tree on the left shows the following structure:

- csi
  - csiProduct
    - modbus
    - bacnet
    - bbpro
    - iot
    - bb3
      - bb3-app
        - dataRegInt32
          - dataRegInt32Table
            - dataRegInt32Entry
              - dataRegInt32Index
              - dataRegInt32Register (selected)
              - dataRegInt32Name
      - dataRegUInt64
      - dataRegFloat32
      - dataRegFloat64
      - dataRegCharStr
      - trapRules
      - trapData

The Properties window at the bottom shows the following details for the selected object:

Property	Value
Name	dataRegInt32Register
OID	.1.3.6.1.4.1.3815.1.6.1.1.1.2
MIB	CSI-BB3-MIB
Syntax	INTEGER32
Access	read-write
Status	current
DefVal	
Indexes	dataRegInt32Index
Descr	Register's data value.

The status bar at the bottom displays the full OID: .iso.org.dod.internet.private.enterprises.csi.csiProduct.bb3.bb3-app.dataRegInt32.dataRegInt32Table.dataRegInt32Entry.dataRegInt32Register.88

Getting our floating point value will appear as follows. The bytes "42 A4 83 12" translate to the floating point version of 82.255997. You can use Google to locate conversion tools to go from IEEE 754 hexadecimal representation (as shown in the browser) to decimal.

The screenshot displays the iReasoning MIB Browser application. The top menu bar includes File, Edit, Operations, Tools, Bookmarks, and Help. Below the menu is a toolbar with fields for Address (192.168.1.12), Advanced..., OID (.1.3.6.1.4.1.3815.1.6.1.3.1.1.2.1), Operations (Get), and a Go button. The main window is divided into three sections:

- Left Panel (SNMP MIBs):** A tree view showing the hierarchy of MIBs. The selected path is: enterprises > csi > csiProduct > bb3 > bb3-app > dataRegFloat32 > dataRegFloat32Table > dataRegFloat32Entry > dataRegFloat32Index > dataRegFloat32Register. The entry is highlighted in blue.
- Right Panel (Result Table):** A table showing the result of the GET operation.
 

Name/OID	Value	Type	IP:Port
dataRegFloat32Register.1	0x42 A4 83 12	OctetString	192.168.1.1...
- Bottom Panel (Detailed View):** A table showing the details of the selected MIB entry.
 

Name	Value
Name	dataRegFloat32Register
OID	.1.3.6.1.4.1.3815.1.6.1.3.1.1.2
MIB	CSI-BB3-MIB
Syntax	Float32TC (OCTET STRING) (SIZE(4))
Access	read-write
Status	current
DefVal	
Indexes	dataRegFloat32Index
Descr	Register's data value.

The status bar at the bottom shows the full path of the selected entry: .iso.org.dod.internet.private.enterprises.csi.csiProduct.bb3.bb3-app.dataRegFloat32.dataRegFloat32Table.dataRegFloat32Entry.dataRegFloat32Register.1

The SNMP Get of our character string is illustrated below. Although encoded generically as an Octet String, the MIB browser is kind enough to display it as an ASCII string since it found the string to be all printable characters.

iReasoning MIB Browser

File Edit Operations Tools Bookmarks Help

Address: 192.168.1.12! Advanced... OID: .1.3.6.1.4.1.3815.1.6.1.5.1.1.2.1 Operations: Get Go

SNMP MIBs

enterprises

csi

csiProduct

modbus

bacnet

bbpro

iot

bb3

bb3-app

dataRegInt32

dataRegUInt64

dataRegFloat32

dataRegFloat64

dataRegCharStr

dataRegCharStrTable

dataRegCharStrEntry

dataRegCharStrIndex

dataRegCharStrRegister

dataRegCharStrName

trapRules

Result Table

Name/OID	Value	Type	IP:Port
dataRegCharStrRegister.1	Test String	OctetString	192.168.1.1...

Name dataRegCharStrRegister

OID .1.3.6.1.4.1.3815.1.6.1.5.1.1.2

MIB CSI-BB3-MIB

Syntax DisplayString (OCTET STRING) (S... ...)

Access read-write

Status current

DefVal

Indexes dataRegCharStrIndex

Descr Register's data value.

.iso.org.dod.internet.private.enterprises.csi.csiProduct.bb3.bb3-app.dataRegCharStr.dataRegCharStrTable.dataRegCharStrEntry.dataRegCharStrRegister.1



## 9. Configuring SNMP Trap Sender

### 9.1 SNMP Trap Destinations

The first step in sending traps is to tell Babel Buster where to send them. This is done on the Devices page of the Trap Sender. Traps or Notifications generated by this device will be sent to port 162 at the IP addresses provided here.

Traps may be sent as v1, v2c or v3. The same application varbinds will be included in either trap type, in addition to header and varbinds required for that trap type. (Note: We use the term "trap" generically here. The v2c/v3 "trap" is properly known as a Notification.)

To remove a destination, simply leave the IP address field blank. To add destinations, or change or remove existing entries, make the necessary changes here, then go to the File Manager page and save your configuration to the Boot configuration XML file. Then restart this device. The trap destinations are loaded one time prior to startup of the SNMP engine. To reload the trap destination table, you must restart the device.

The user name in v1 and v2c traps will be "public". The user name in v3 notifications will be the name entered above. There is only one SNMPv3 Notification user name for



all Notification destinations.

## 9.2 SNMP Trap Triggers

Create a Trap Trigger for each trap that should be sent.

A trap must reference a variable in the local MIB. Therefore, before you can create a trap trigger, you must assign the local register of interest to a spot in the local MIB. (See "Configuring Gateway as an SNMP Server".) Once the local register has a home in the MIB, select the branch and MIB index in the trap trigger rule. Do not enter local register number here. Enter the MIB table index from your local MIB configuration pages.

The register found in the MIB branch, at the table index given, is displayed for reference as "Looking up...". To change this register, go to that MIB branch and table index, and change the MIB definition, or select a different branch and/or table index.

Devices Trap Trigger Trap Summary

Trap # 1 Trigger rule presently tests FALSE Update < Prev Next >

Using MIB branch Integer 32-Bit and table index: 1 Looking up: 1: Data Value 1

Event is TRUE if the value is Greater than this value: 1000.000 this register: 0

Qualified by this hysteresis value: 0.00 this minimum On Time: 0:00:00 this minimum Off Time: 0:00:00

☒ Send if True, repeat 0 times. ☒ Send if False, repeat 0 times. Repeat Time: 0.0

Message if True Data Value 1 is High

Message if False Data Value 1 is Normal

# Trap Trigger Rules Enabled: 2 Insert Delete

Once the variable of interest is selected, define the threshold. This will be a test such as "greater than" or "less than", etc. You can provide a fixed value for the threshold, or you may reference a local register that will hold a threshold that may change from time to time. If register is zero, then the fixed value will be used. If a non-zero register number is given, then the fixed value is disregarded. The trap trigger will be "true" when the MIB variable meets the criteria given.

Qualifications are optional, and enabled only when values are nonzero. How hysteresis is applied depends on the comparison. For a test that becomes true if greater than, the test will not return to false until the local register is less than the test value by a margin of at least this hysteresis value. If a test becomes true if less than, it will not return to false until the local register is greater than the test value by a margin of at least this hysteresis value.

Special test types: "Deviates from" will test against the value given, and use Hysteresis as the margin of deviation. This is effectively a "greater than or less than" test for deviation from a setpoint. "Changes by" will become true each time the given variable changes by the value given, and Hysteresis has no effect on this test. If



"Changes by" references a value of zero, then this becomes a special test whereby the event is true any time something in the system updates the variable. A "changes by zero" should not be used when the variable is continuously read from a slave device since this will result in continuous traps.

You also have the option of specifying a minimum On and Off time. The "On" time means the rule must test true for this amount of time before the status will actually be set true and the trap will actually be sent. The "Off" time means the rule must test false for this amount of time before its status will actually be returned to false. Times are given in HH:MM:SS format (hours, minutes, seconds).

Check "Send if True" to send traps when the trap trigger rule meets all criteria for "true". Check "Send if False" if you would also like to send a trap when the trigger rule meets all criteria for "false".

Enter a repeat count if the trap should be repeated. If repeat count is zero, the trap will be sent one time. If repeat count is 1, the trap will be sent 2 times, and so on. The interval between traps will be the Repeat Time in seconds. Enter -1 for trap repeat count if the trap should simply repeat indefinitely at the Repeat Time interval. A repeat count of -1 for "Send if True" is acceptable. A repeat count of -1 for "Send if False" will be treated as no repeat since indefinite repeating of non-true events is ill-advised.

One of the varbinds in the trap message is an arbitrary ASCII character string, sent as an ASN Octet string. The "True" message will be sent when the trap event is true, and the "False" message will be sent when the trap event is false.

To delete the rule shown, click Delete. To insert a new rule before the rule displayed, click Insert. To add a rule to the end of the list, click Next when at the end of the list, enter new rule, and click Update. The number of rules enabled simply limits the scope of rule review so that you do not have to review a lot of unused rules. If the displayed rules are used up and you need more, increase the enabled number.

### 9.3 SNMP Trap Summary

The trap summary provides a page where you may see the present status of all of your trap trigger rules. Here we see the "true" status of the above trigger.

Devices		Trap Trigger		Trap Summary			
				Showing 1 to 2 of 2		Update	< Prev Next >
Trap #	Local MIB Variable	Local Register #	Test Result	Local Value	Local Name		
1	Integer.1	1	True	1105.000000	Data Value 1		
2	---	0	False	---	---		
1 Trap rule vars loaded				1 Trap data vars loaded			Reload SNMP

### 9.4 Testing the SNMP Trap Sender

A variety of tools are available for browsing an SNMP MIB and receiving SNMP Traps. The tool used in the following examples is the iReasoning MIB Browser. Refer to the

Tools section under Support at [csimn.com](http://csimn.com) for more information about SNMP tools.

The iReasoning MIB Browser allows you to browse the MIB, do table walks, etc., but of primary interest here, also allows you to test traps. When you open the iReasoning browser, under Tools, select Trap Receiver. The screen illustrated below shows the trap viewer after receiving the above trap from our Babel Buster.

The screenshot shows the iReasoning MIB Browser interface. The top menu bar includes File, Edit, Operations, Tools, Bookmarks, and Help. Below the menu, there are fields for Address (192.168.1.125), Advanced..., and OID (1.3.6.1.2.1.1.7.0). The Operations dropdown is set to Get Next, and a Go button is visible. On the left, the SNMP MIBs tree shows the MIB Tree with iso.org.dod.internet selected. The main area displays the Trap Receiver tab, which shows a table with columns: Description, Source, Time, and Severity. A single trap is listed with the following details:

Description	Source	Time	Severity
trapOID: .iso.org.dod.internet.private.enterprises.csi.csiProduct.bb3...	192.168.1.125	2020-08-19 09:13:40	

Below the table, the trap details are expanded, showing the following information:

- Source:** 192.168.1.125
- Timestamp:** 53 minutes 59 seconds
- SNMP Version:** 3
- Trap OID:** .iso.org.dod.internet.private.enterprises.csi.csiProduct.bb3.bb3-app.bb3-traps.trapRuleStateChange
- Variable Bindings:**
  - Name:** .iso.org.dod.internet.mgmt.mib-2.system.sysUpTime.0  
**Value:** [TimeTicks] 53 minutes 59 seconds (323961)
  - Name:** snmpTrapOID  
**Value:** [OID] trapRuleStateChange
  - Name:** .iso.org.dod.internet.private.enterprises.csi.csiProduct.bb3.bb3-app.trapData.trapDataTable.trapDataEntry.trapDataName.0  
**Value:** [OctetString] Data Value 1
  - Name:** .iso.org.dod.internet.private.enterprises.csi.csiProduct.bb3.bb3-app.trapData.trapDataTable.trapDataEntry.trapDataValue.0  
**Value:** [OctetString] 1105.000000

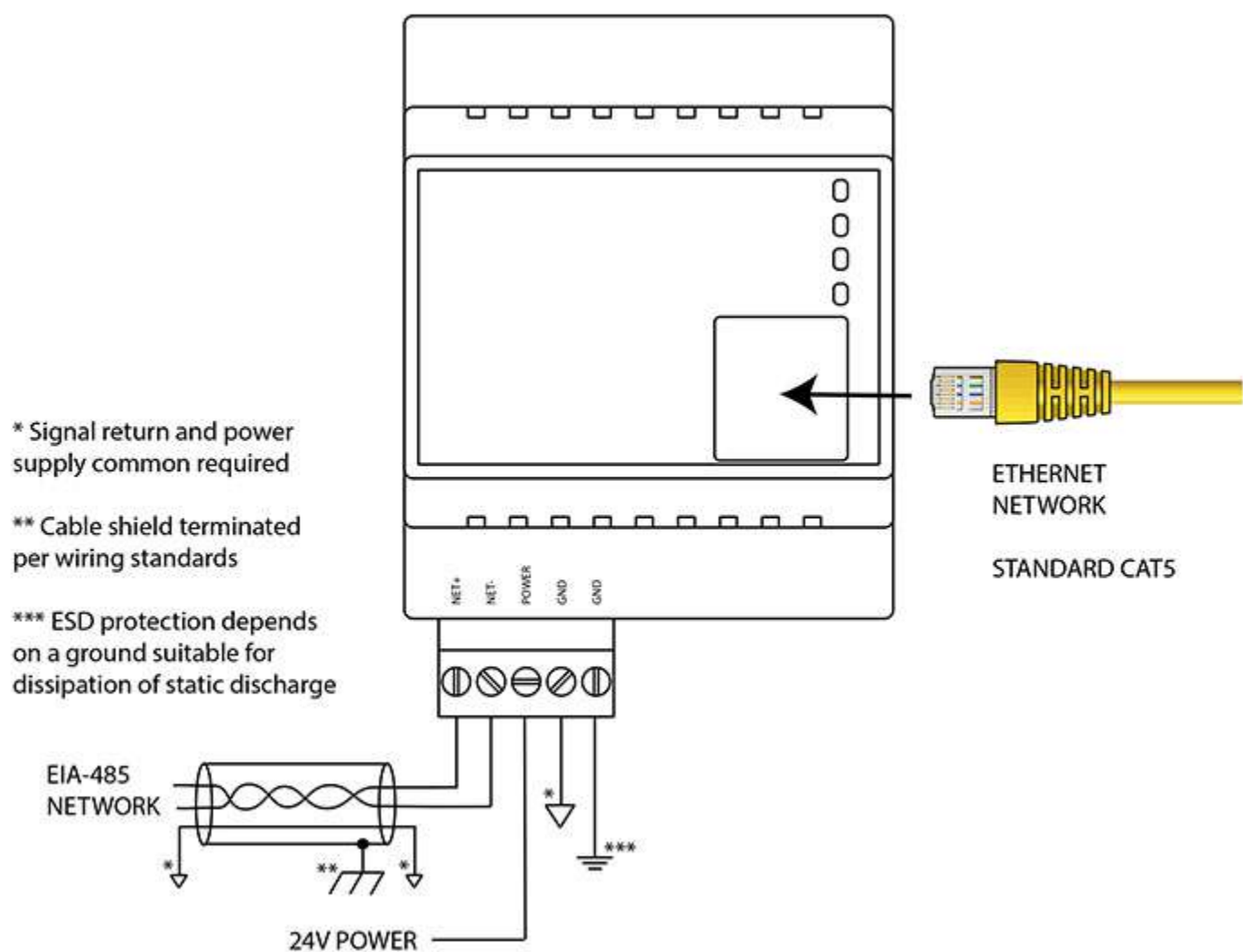
The bottom status bar shows the selected MIB: .iso.org.dod.internet.mgmt.mib-2.system.sysServices.0.



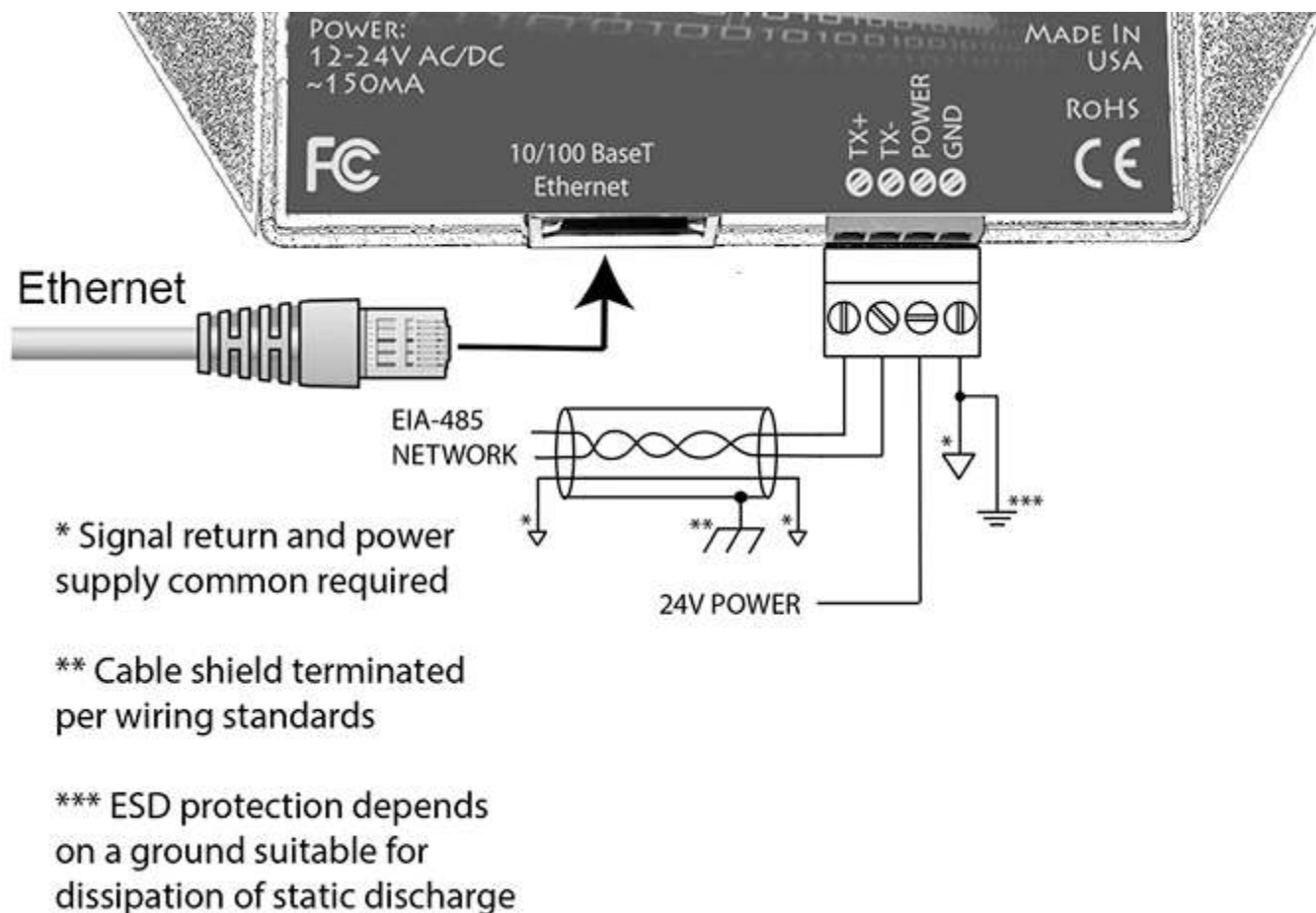
## Appendix A Hardware Details

### A.1 Wiring

Wiring for the Babel Buster BB3-6101 is illustrated below.



Wiring for the Babel Buster MX-61 is illustrated below.



Wire the gateway as illustrated. Follow all conventional standards for wiring of EIA-485 networks when connecting the 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.

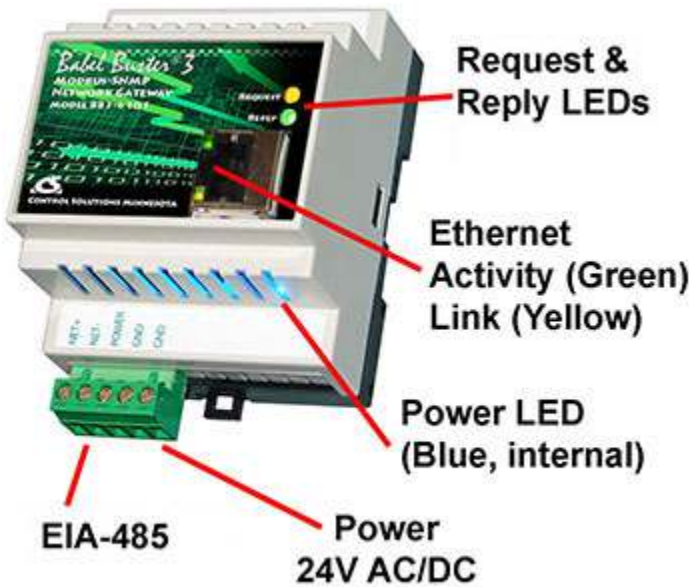
Note that in addition to connecting power supply common to a GND terminal, you must also connect a GND terminal to earth ground in order to ensure proper ESD protection.

**BB3-6101-232:** The standard BB3-6101 serial 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 BB3-6101-232, and RXD is data in to the gateway. Hardware handshake is not supported.

## A.2 Front Panel LED Indicators

### A.2.1 BB3-6101 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.



The more common use of the serial port on a BB3-6101 is Modbus RTU. The Request and Reply LEDs normally reflect Modbus RTU traffic, and the Ethernet activity LED will indicate network traffic in general. However, with the BB3-6101-V3SP, the serial port is used for proprietary custom protocols. In this instance, the LEDs are under control of the user provided Script Basic protocol interpretation program.

Babel Buster BB3-6101 LEDs indicate as follows (LEDs are bi-color):

REQUEST	Green or yellow as directed by the user's program.
REPLY	Green or red as directed by the user's program.
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-61 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-61 Request, Reply and Error LEDs reflect Modbus RTU traffic, and the Ethernet activity LED will indicate network traffic in general.

Babel Buster MX-61 LEDs indicate as follows (LEDs are each a single color):

Error (red)	<p>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.</p> <p>Operating as Modbus Slave, flashes red if an exception code is sent (meaning the received request resulted in an error).</p>
Request (yellow)	<p>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.</p>
Reply (green)	<p>Operating as Modbus Master, flashes green each time a good response is received.</p> <p>Operating as Modbus Slave, flashes green each time a good response is sent.</p>
Ethernet Activity	<p>Green LED is on solid during portions of the boot-up process, and then flashes briefly when Ethernet network traffic is detected.</p>

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.

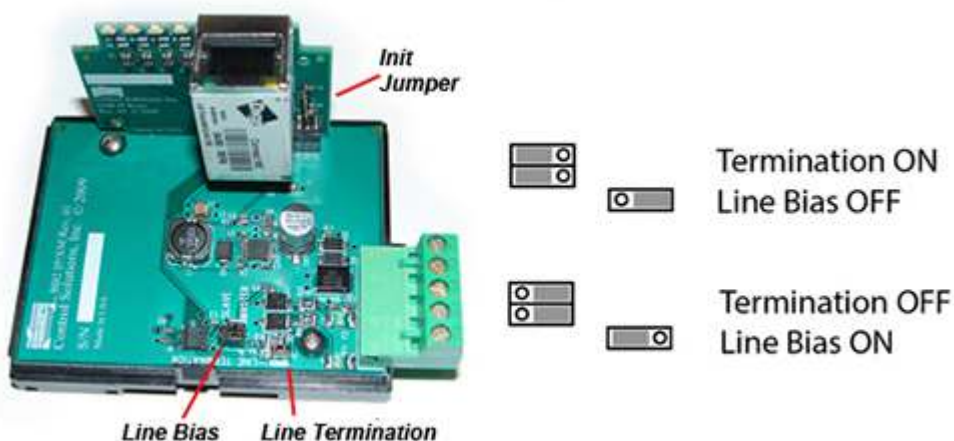
### A.3 RS-485 Line Termination & Bias

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.

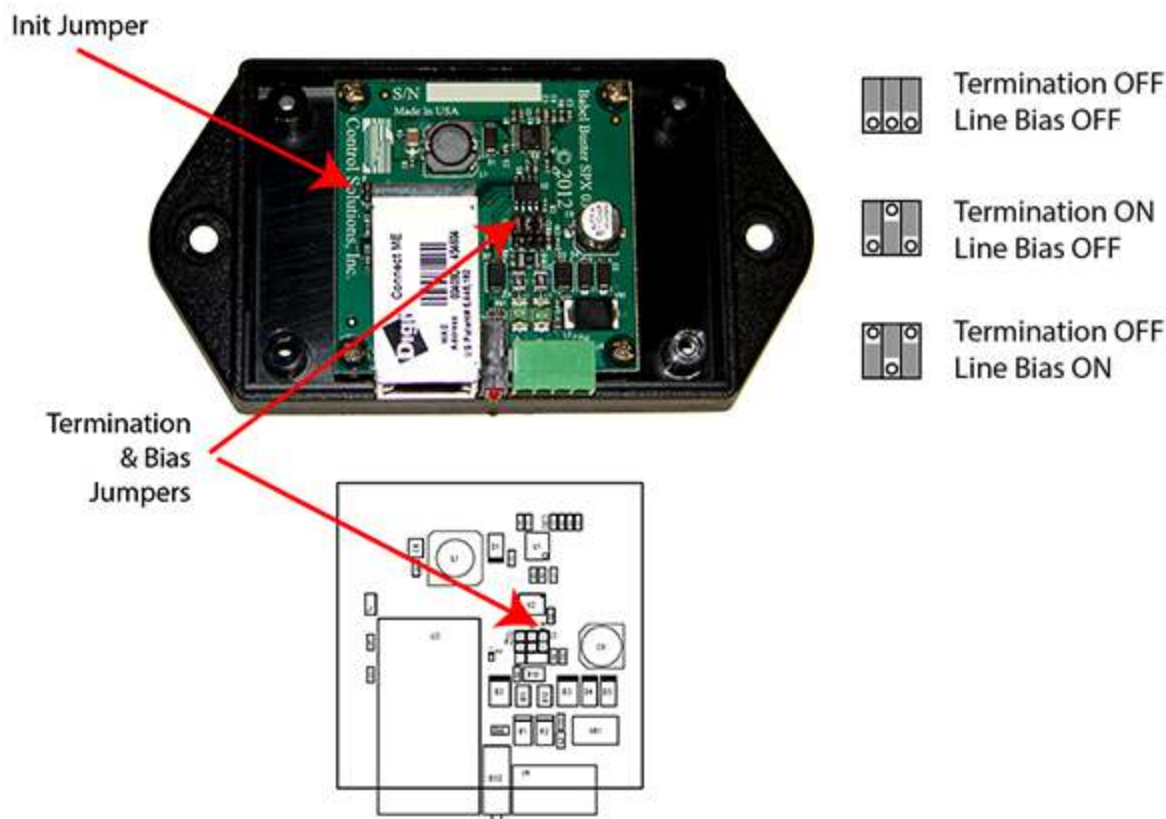
Enable line bias when needed. Line bias should only be enabled at one point on the network, and does not have to be the end point. Line bias holds the line in a known neutral state when no devices are transmitting. Without bias, the transition from offline to online by a transmitter can look like a false 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 BB3-6101:



Jumper locations for Babel Buster MX-61:



## 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 3.1.5. 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.

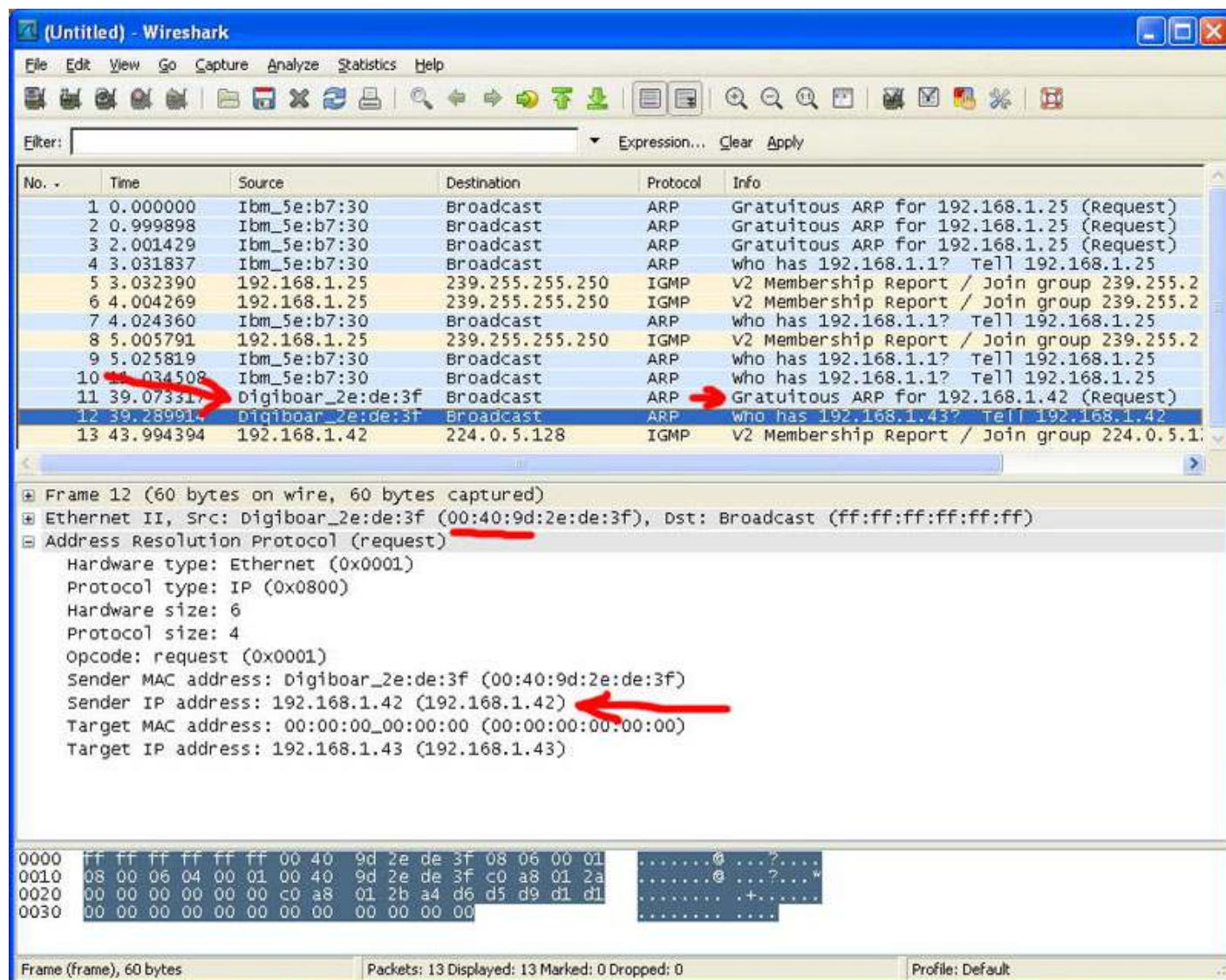
Upon power up, BB3-6101/MX-61 will ping its own IP address one or more times. This is part of its duplicate address resolution mechanism. If it finds another device with its own IP address, it will set its own IP address to a default pseudo-random address generally starting with 192.

Wait until you are certain BB3-6101/MX-61 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 BB3-6101/MX-61 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 BB3-6101/MX-61 is located at 192.168.1.42.



## 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 BB3-6101/MX-61 serves two purposes, and what it does depends on whether you apply the jumper before or after the BB3-6101/MX-61 boots up.

### Hard Configuration Reset:

Installing the jumper after bootup causes the BB3-6101/MX-61 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 BB3-6101/MX-61 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 BB3-6101/MX-61. If both LEDs on the RJ45 jack come on and remain on, remove the jumper and then power cycle the BB3-6101/MX-61.

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 repeat the allocation setup as described in Section 3.4.

### **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 BB3-6101/MX-61'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](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.





## 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 gotten 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 proceeds 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>.



## Appendix B Modbus CSV Import Files

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 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.

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 Reference B)
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 Reference C)
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 seconds are recognized).
LOCALREG	-	Specify a local Modbus register number where data read from a remote device will be placed, or where data written to a remote device will be taken from.

MASK	-	<p>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.</p> <p>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.</p>
DEFAULT	-	<p>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.</p> <p>When WRITING: The default value will be stored into the local object/register if POR is set to True, or if the amount of time specified by TIMEOUT is exceeded without an update to the local object by a remote client.</p>
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.

The minimum required header line for Modbus TCP must include RW, LOCALREG, TYPE, REG, FORMAT, and DEVNUM. All other columns are optional.

## B.2 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.3 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
FP	Floating Point, IEEE 754 32-bit	2
BIT	Bit (only used for Coil or Discrete Input registers)	-
MOD102	Mod-10, 2-register	2
MOD103	Mod-10, 3-register	3
MOD104	Mod-10, 4-register	4
S64	Signed 64-bit Integer	4





## Appendix C      Modbus Reference Information

### C.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 logically 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.



## Appendix D      Trouble Shooting

### D.1      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.

### D.2      SNMP Trouble Shooting

Assuming you have IP addresses configured correctly and the SNMP ports are open through any routers and firewalls between devices, the most common cause of not communicating is a mismatching community string for SNMPv2 or incorrect user credentials for SNMPv3. Without the correct credentials, most devices will simply ignore the request, making it look as if there is no connection when in fact there is nothing wrong with the connection.

Another common oversight is that when adding local registers to the local MIB, you need to click the Reload SNMP button at the bottom of the Local MIB pages to cause SNMP to reload its internal tables with the new configuration you just entered. The Local MIB pages are effectively a list of instructions for loading the SNMP MIB, but the MIB is not automatically rebuilt every time you add another line to the Local MIB pages. To reload the MIB according to the list of instructions you provided on the Local MIB web pages, you need to click Reload SNMP. (Note: The Reload SNMP button is found on multiple pages, but they all perform the same function and any of the Reload SNMP buttons will reload all branches of the MIB.)

Two of the most useful tools in trouble shooting SNMP are Wireshark and a MIB browser. An example follows.

### **D.3 Wireshark 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 MIB Browser, 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.

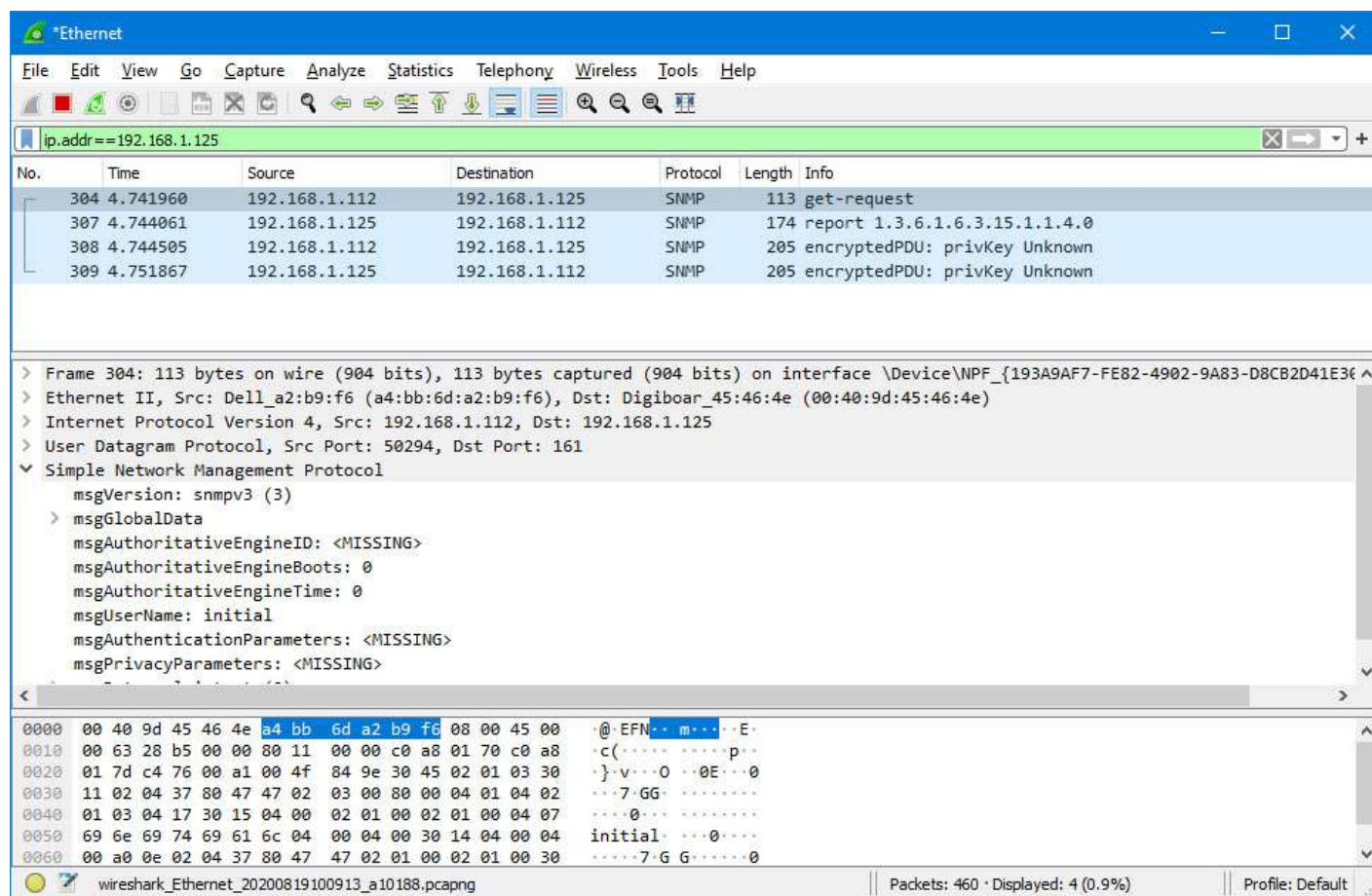
One means of monitoring network traffic is to get a managed switch that supports "port mirroring". One such device we have tested is the TP-LINK model TL-SG105E. Setting it up requires utility software (provided with the switch) and takes a little effort to get configured. But once configured, it works 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).

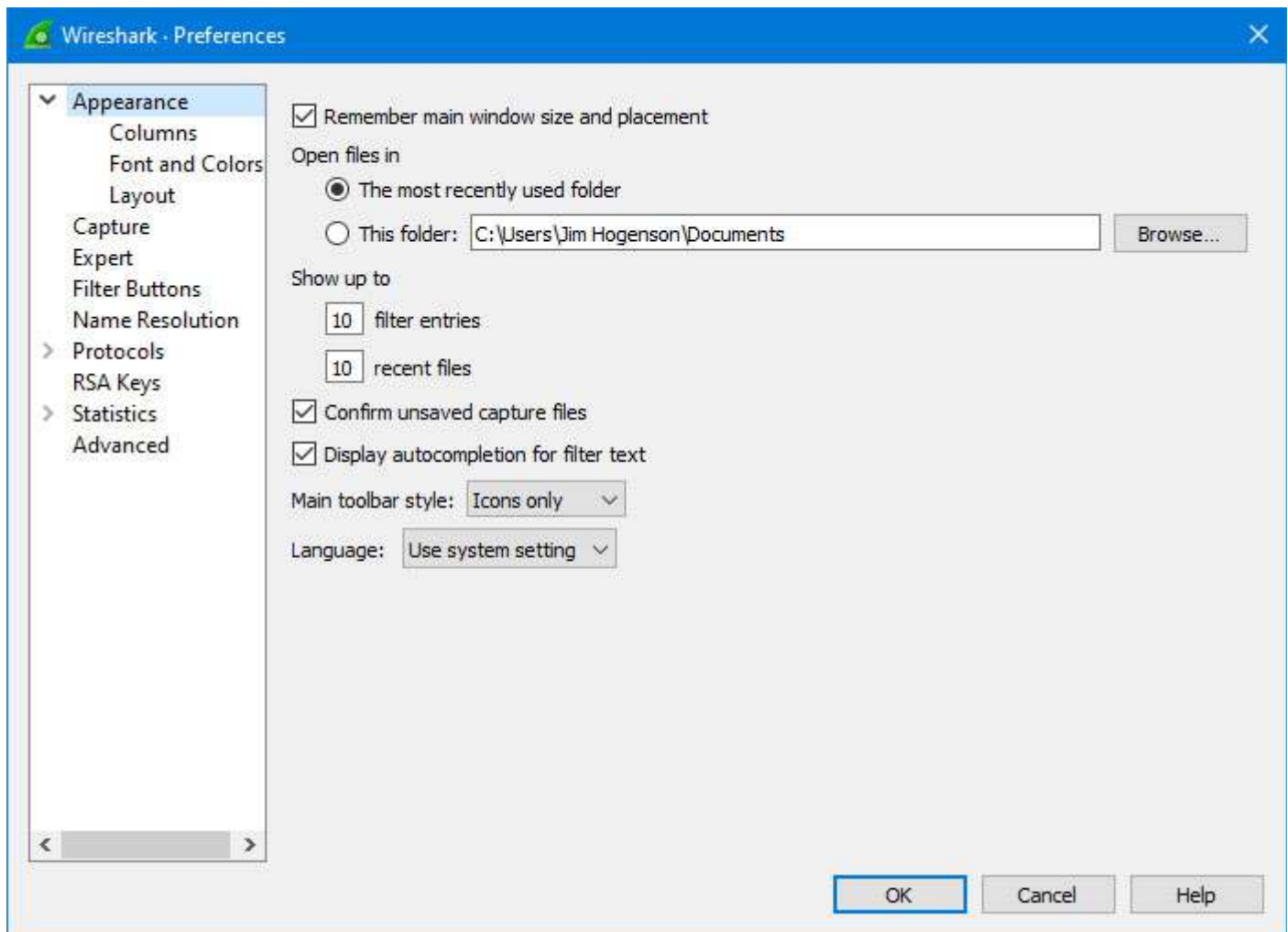


## D.4 Example of Using Wireshark

If you use Wireshark to capture a Get request in SNMPv3 with privacy configured, Wireshark won't be able to display anything meaningful until you provide Wireshark with some credentials.

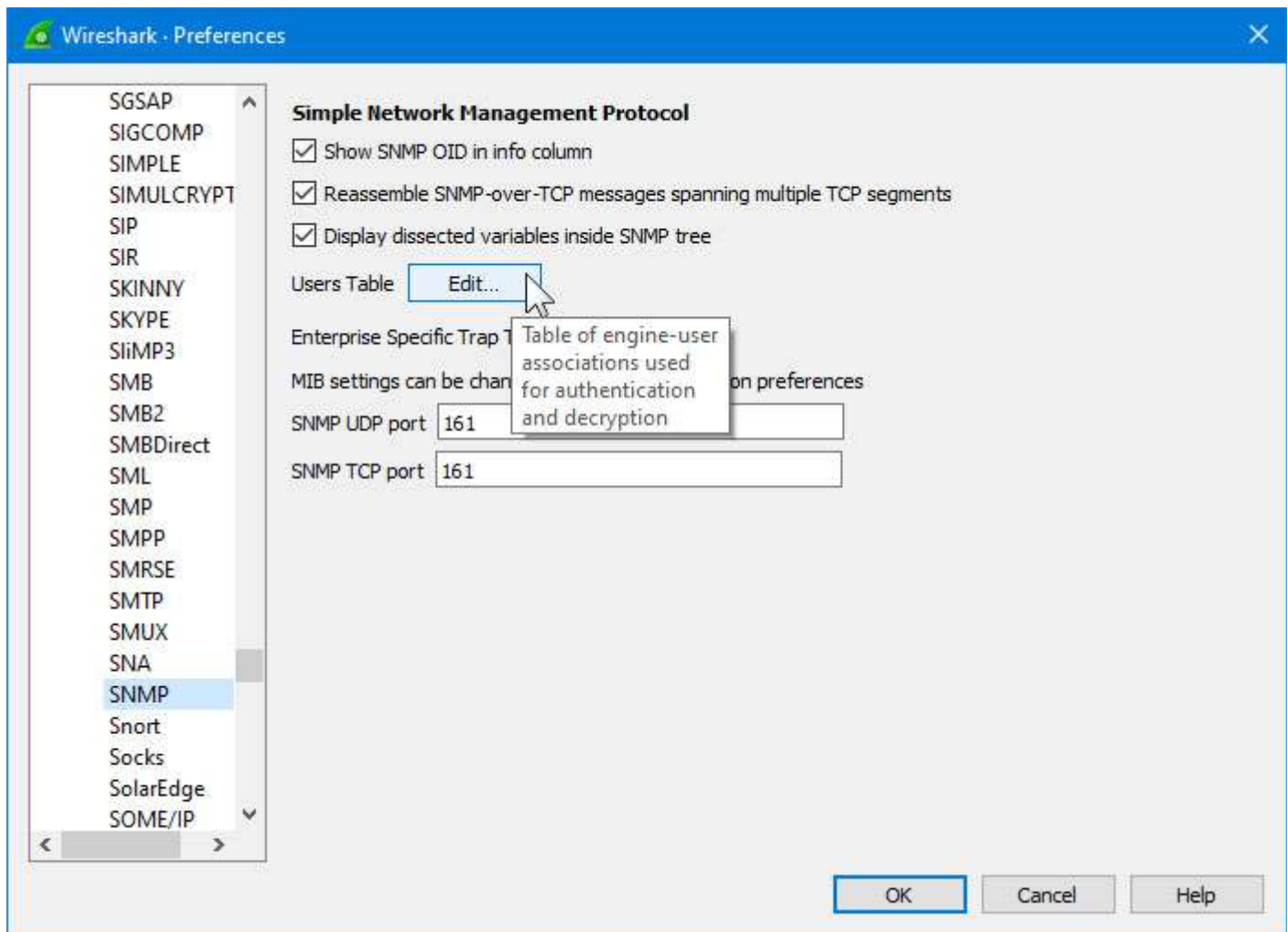


Under the Edit menu, select Preferences.

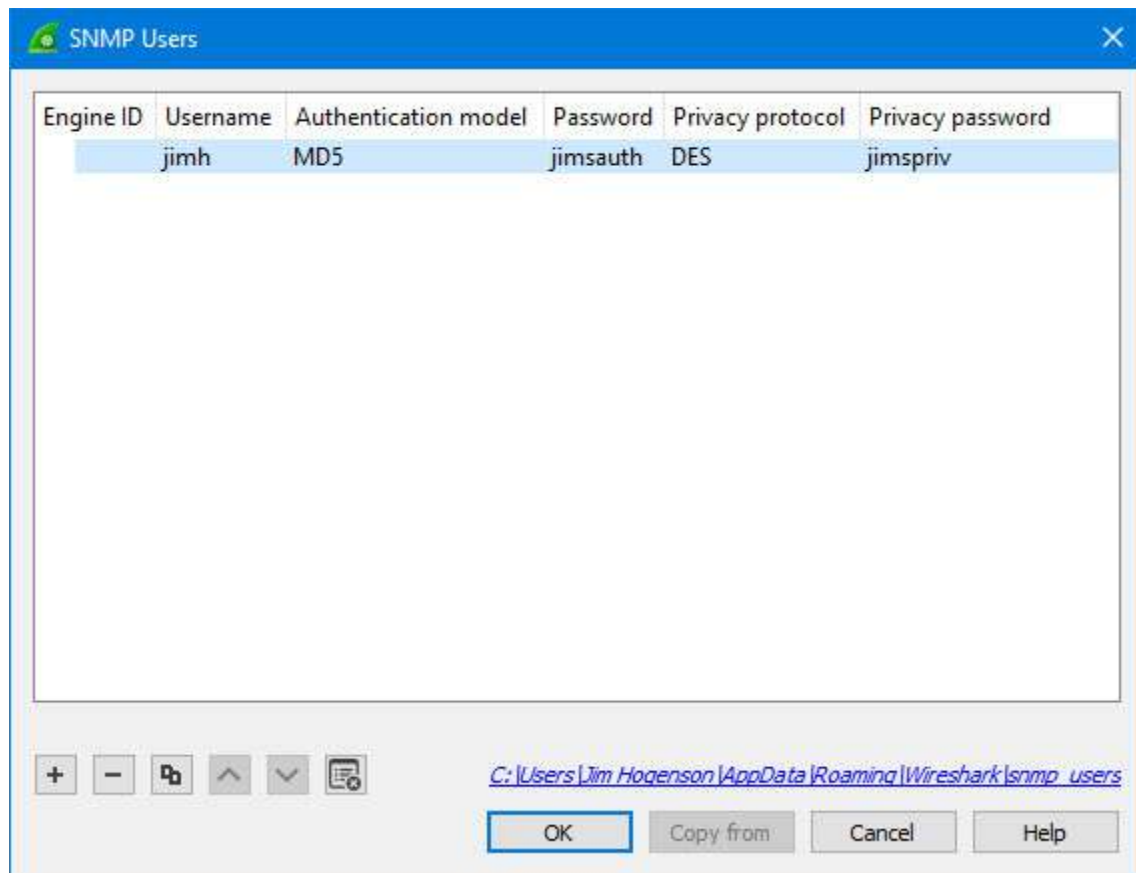


Expand the Protocols list and scroll down to SNMP. Select SNMP.





Next, click Edit (Users Table). Enter the user name and credentials being used in the Get request you wish to capture.



Once Wireshark has been provided with the user name and its associated credentials, Wireshark can show the full packet.

\*Ethernet

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

ip.addr==192.168.1.125

No.	Time	Source	Destination	Protocol	Length	Info
309	53.333473	192.168.1.125	192.168.1.112	SNMP	205	get-response 1.3.6.1.4.1.3815.1.6.1.1.1.2.1
3207	324.675586	192.168.1.112	192.168.1.125	SNMP	113	get-request
3208	324.677186	192.168.1.125	192.168.1.112	SNMP	174	report 1.3.6.1.6.3.15.1.1.4.0
3209	324.677860	192.168.1.112	192.168.1.125	SNMP	205	get-request 1.3.6.1.4.1.3815.1.6.1.1.1.2.1
3210	324.685085	192.168.1.125	192.168.1.112	SNMP	205	get-response 1.3.6.1.4.1.3815.1.6.1.1.1.2.1

> Frame 3210: 205 bytes on wire (1640 bits), 205 bytes captured (1640 bits) on interface \Device\NPF\_{193A9AF7-FE82-4902-9A83-D8CB2D41E3}

> Ethernet II, Src: Digiboar\_45:46:4e (00:40:9d:45:46:4e), Dst: Dell\_a2:b9:f6 (a4:bb:6d:a2:b9:f6)

> Internet Protocol Version 4, Src: 192.168.1.125, Dst: 192.168.1.112

> User Datagram Protocol, Src Port: 161, Dst Port: 57155

Simple Network Management Protocol

- msgVersion: snmpv3 (3)
- msgGlobalData
- msgAuthoritativeEngineID: 80000ee702fe800000000000002409dffe45464e
- msgAuthoritativeEngineBoots: 9
- msgAuthoritativeEngineTime: 6897
- msgUserName: jimh
- msgAuthenticationParameters: 4347f30b21c0b4b1e347fb11
- msgPrivacyParameters: 000000900004473
- msgData: encryptedPDU (1)
  - encryptedPDU: d4068f6f390d4106bd0e5cca5ca04707a494ae1e1c93c3c7...
    - Decrypted ScopedPDU: 3041041580000ee702fe80000000000002409dffe45464e
      - contextEngineID: 80000ee702fe800000000000002409dffe45464e
      - contextName:
      - data: get-response (2)
        - get-response
          - request-id: 931153739
          - error-status: noError (0)
          - error-index: 0
          - variable-bindings: 1 item
            - 1.3.6.1.4.1.3815.1.6.1.1.1.2.1: 110500
              - Object Name: 1.3.6.1.4.1.3815.1.6.1.1.1.2.1 (iso.3.6.1.4.1.3815.1.6.1.1.1.2.1)
              - Value (Integer32): 110500

[Response To: 3209]

[Time: 0.007225000 seconds]

0000 a4 bb 6d a2 b9 f6 00 40 9d 45 46 4e 08 00 45 00 ..m...@EFN..E-

0010 00 bf 00 12 00 00 40 11 f5 de c0 a8 01 7d c0 a8 .....@.....}

0020 01 70 00 a1 df 43 00 ab a1 08 30 81 a0 02 01 03 ..p..C...0....

0030 30 11 02 04 37 80 47 4b 02 03 00 80 00 04 01 03 0...7GK.....

Frame (205 bytes)    Decrypted ScopedPDU (72 bytes)

wireshark\_Ethernet\_20200819100913\_a10188.pcapng    Packets: 3463 · Displayed: 8 (0.2%)    Profile: Default



## Appendix E 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 BB3-6101/MX-61.

### E.1 X.509 Auto-Certificate Generation

The Babel Buster BB3-6101/MX-61 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.

The screenshot shows a configuration interface with a dark teal background. At the top, it says 'Web Server' followed by two checked checkboxes: 'HTTPS Enabled (on 443)' and 'HTTP Enabled'. Below these are three input fields: 'HTTP Port' with the value '80' and '(default 80)', 'Modbus Port' with the value '502' and '(default 502)', and 'MIB Offset' with the value '0'. To the right of these fields is a 'Set Ports' button. Below the input fields is a checked checkbox for 'FTP Server' with the label 'Enabled'. At the bottom left, it shows 'MAC Address: 00:40:9D:45:46:4E'. At the bottom right, it shows 'System Uptime: 0,01:23:30'. At the very bottom, it says 'HTTPS certificate status: Using self-generated X.509'.

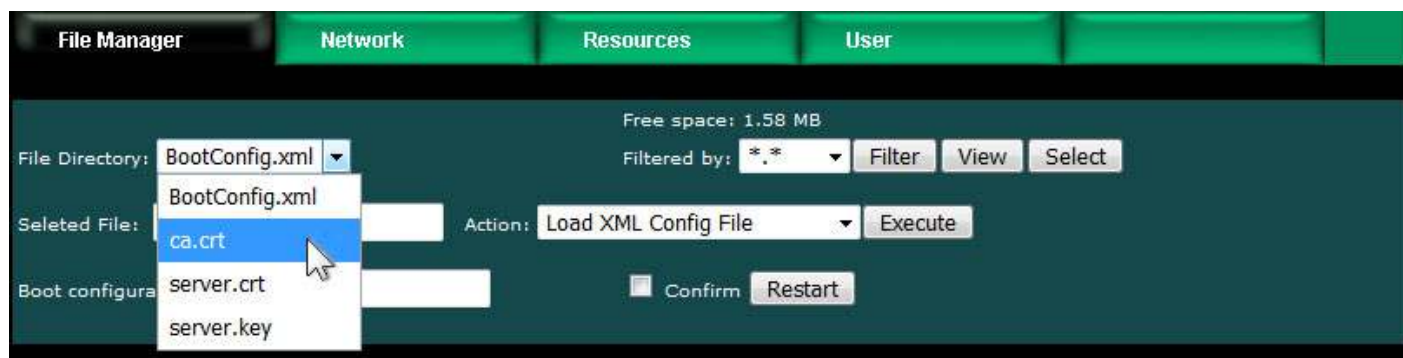
If there is a need to delete the self-generated certificates, you can do so by logging in via FTP. Change directory to /FLASH0, then to .cfg. The two certificate files that were self-generated are ssl.cert and ssl.key.



```
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>
```

## E.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.

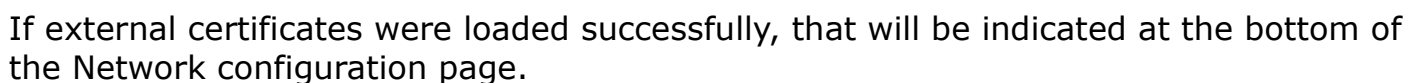


The required certificates are as follows, and must use exactly these names.

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).





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.

3 of 5

```

# 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 '*****'
```



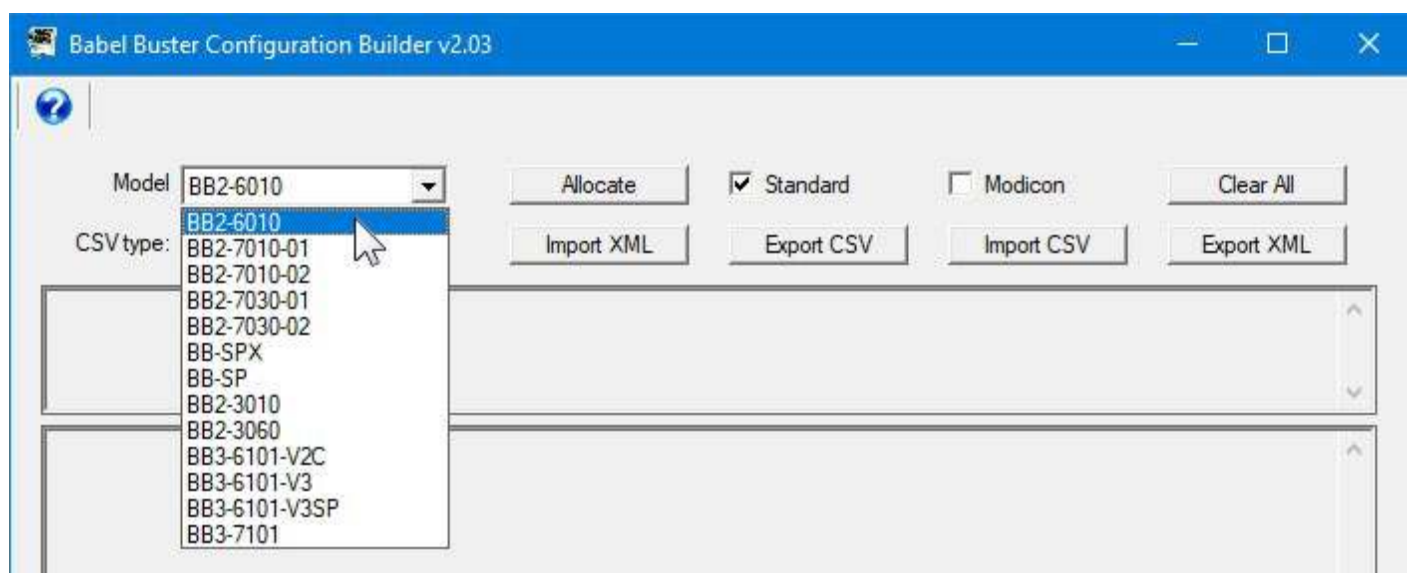
## Appendix F Converting Older XML Files

### F.1 BB2-6010/SPX to BB3-6101/MX-61 Conversion

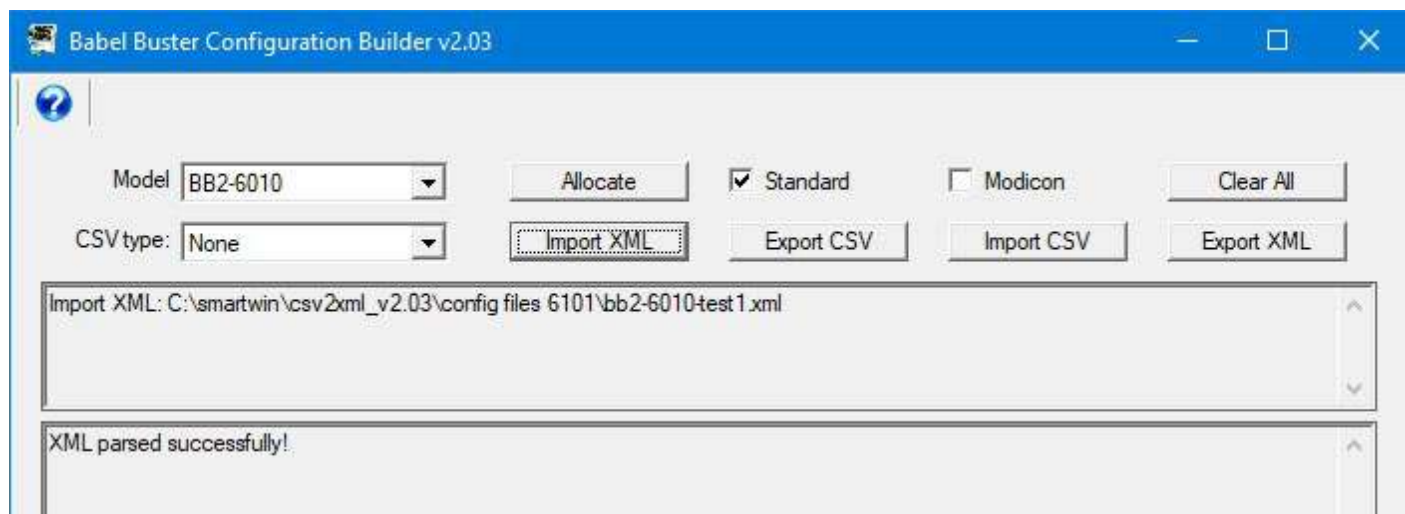
You cannot load a BB2-6010/SPX configuration XML file into the BB3-6101/MX-61, but you can easily convert it into a BB3-6101/MX-61 XML file using the Babel Buster Configuration Builder tool available under the Support/Tools link at [csimn.com](http://csimn.com).

In the examples below, SPX may be substituted for BB2-6010 and MX-61 may be substituted for BB3-6101. NOTE: An XML file exported for BB3-6101-V3SP will load correctly in an MX-61-SP.

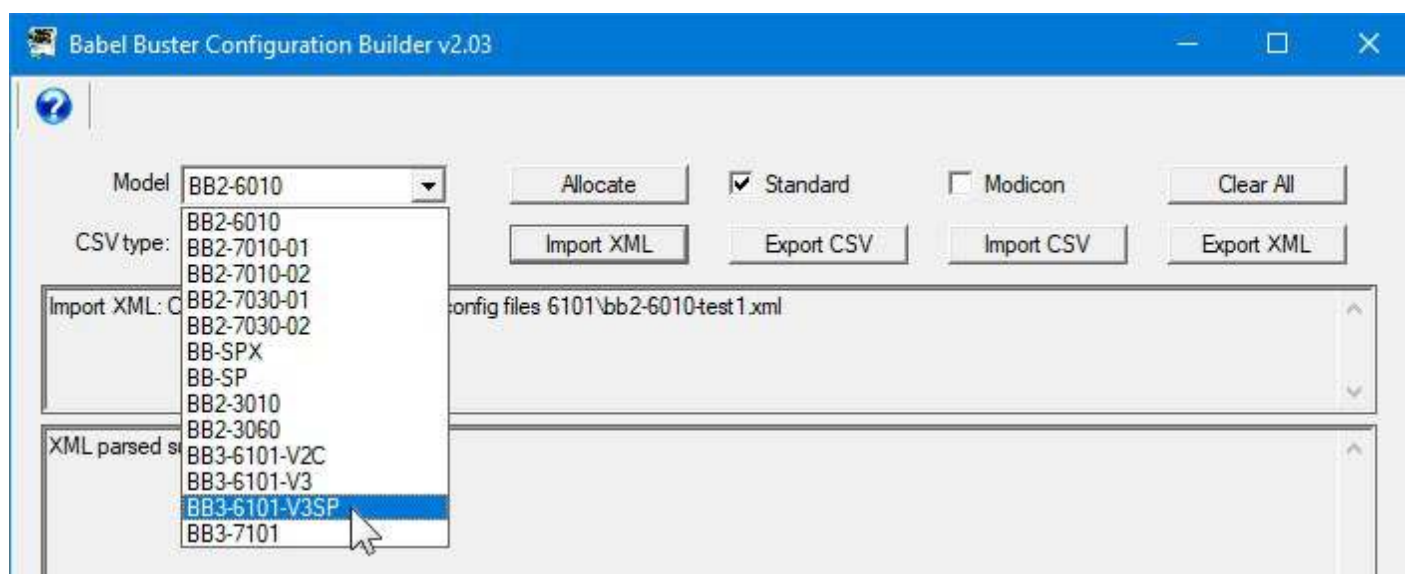
Converting the file is simple. Start by selecting the BB2-6010 model.



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.

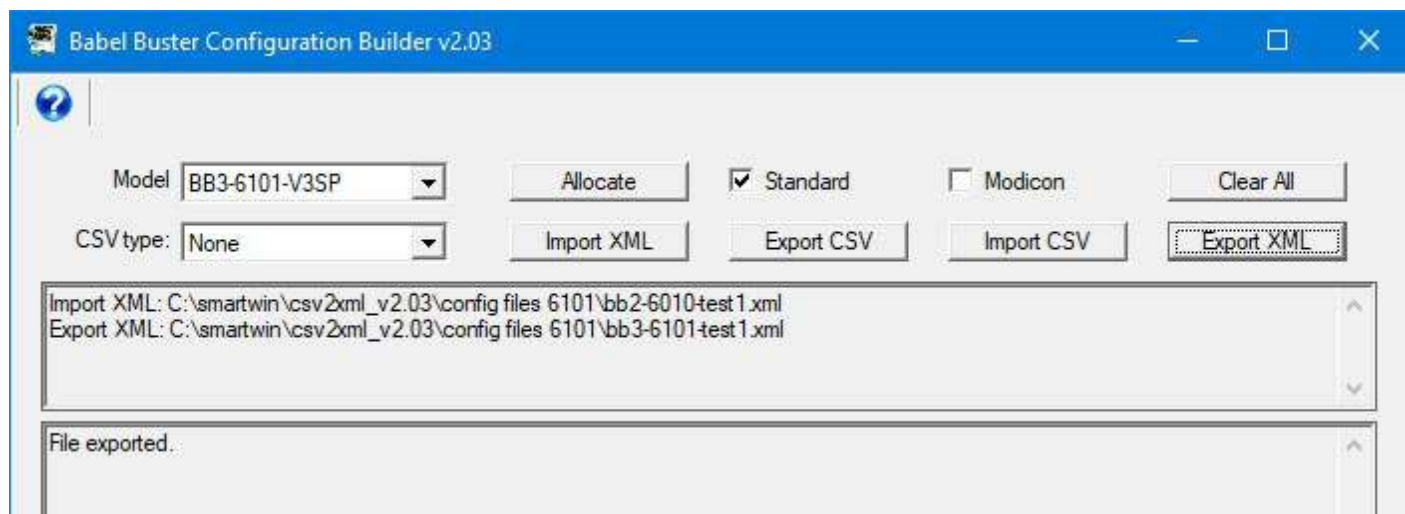


Now go back to the model list and switch to BB3-6101.



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 BB3-6101 gateway.





Note that there are two things not converted by the BB2-6010 to BB3-6101 conversion process: Resource allocation and SNMP trap rules.

Resource allocation is not generated in the Babel Buster 3 XML file because the older gateway did not have this feature. The resource section will be excluded from the XML file. This means that whatever resource allocation you previously had in effect in the BB3-6101 will remain as is. If your attempt to load the new XML file complains about invalid register number, check to see that it is not trying to reference a higher numbered local register than what you have allocated. Adjust your allocation if needed.

The other thing that cannot be converted is SNMP trap (notification) rules. This is because the trap structure in the Babel Buster 3 series gateway is too different to be able to make much use of the Babel Buster 2 trap rules. Rather than create a list of rules that cannot possibly be correct, the conversion process requires you to explicitly manipulate them. Note, however, that you can export the Babel Buster 2 trap rules as a CSV file, edit in bulk using a spreadsheet program, then import the CSV back into the configuration builder as Babel Buster 3 trap rules. Then proceed to generate a new XML file.

## F.2 CSV File Export

The BB3-6101 will not export CSV files directly. To create CSV files from an existing BB3-6101 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 BB3-6101. 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 BB3-6101 itself will import. The most frequently used CSV files - namely read/write maps for Modbus RTU and Modbus TCP - can be imported directly into the BB3-6101. Importing CSV lists of read/write maps assumes you have already created local registers.

Importing device lists directly into the BB3-6101 is 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 BB3-6101 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 BB3-6101.