

## User Guide

### Model VP6-1460-SNMP ValuPoint IoT Edge Server Remote Monitoring for SNMP

Rev. 1.0 – Sept. 2024

© 2024 Control Solutions, Inc.

### VP6-1460-SNMP User Guide Contents

- 1 [Introduction](#)
  - 1.1 How to Use This Guide
  - 1.2 Important Safety Notice
  - 1.3 Warranty
- 2 [Connecting the ValuPoint](#)
  - 2.1 Overview of ValuPoint Operation
  - 2.2 Where to Start
  - 2.3 Connectors and Indicators
  - 2.4 Web User Interface
  - 2.5 External Programming Tool
- 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 Configuring Physical I/O
  - 4.3 Special Features of Local Registers
  - 4.4 Local Register Calculate Rules
  - 4.5 Local Register Copy Rules
  - 4.6 Device Status Reporting
- 5 [Configuring ValuPoint as a Modbus RTU Master](#)
  - 5.1 Modbus RTU Device Configuration
  - 5.2 Modbus RTU Master Read Maps
  - 5.3 Modbus RTU Master Write Maps
  - 5.4 Modbus RTU Master Data Displayed by Slave
  - 5.5 Modbus RTU Errors
- 6 [Configuring ValuPoint 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 ValuPoint as a Modbus RTU Slave](#)
  - 7.1 Modbus RTU Device Configuration
  - 7.2 Modbus RTU Slave Register Map
  - 7.3 Modbus RTU Slave Diagnostic
- 8 [Configuring ValuPoint as a Modbus TCP Server](#)
  - 8.1 Modbus TCP Device Configuration
  - 8.2 Modbus TCP Server Register Map
  - 8.3 Modbus TCP Server Diagnostic

## [9 Configuring ValuPoint as an SNMP Server](#)

- 9.1 Creating Local SNMP MIB
- 9.2 Supported Data Formats, RFC 6340
- 9.3 SNMPv3 Users, Authentication, Privacy
- 9.4 Agent ID
- 9.5 SNMPv3 Engine Info
- 9.6 SNMPv2 Community
- 9.7 Testing the SNMP Agent

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

- 10.1 SNMP Trap Destinations
- 10.2 SNMP Trap Triggers
- 10.3 SNMP Trap Summary
- 10.4 Testing the SNMP Trap Sender

## [11 Configuring the Scheduler](#)

- 11.1 Weekly Schedule
- 11.2 On Demand Scheduled Events
- 11.3 Holidays
- 11.4 Astronomical Clock

## [12 User HTML](#)

- 12.1 Static HTML
- 12.2 Dynamic Data Tags in User HTML
- 12.3 Live JavaScript Gauges

## [Appendix A Hardware Details](#)

- A.1 Wiring, Physical I/O Connections
- A.2 Front Panel LED Indicators
- A.3 RS-485 Line Termination
- A.4 Soft Configuration Reset
- A.5 Discovering Lost IP Address
- A.6 Forced Hard Configuration Reset
- A.7 Firmware Update Notes
- A.8 Battery Replacement

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

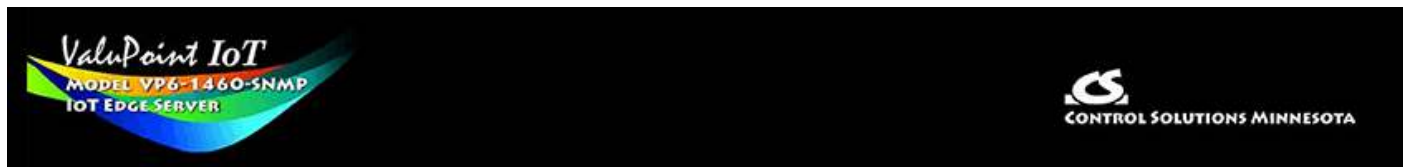
- B.1 Modbus RTU Master Read/Write Maps
- B.2 Modbus TCP Client Read/Write Maps
- B.3 Register Types
- B.4 Register Formats

## [Appendix C Trouble Shooting](#)

- C.1 Modbus RTU Trouble Shooting
- C.2 Modbus TCP Trouble Shooting
- C.3 Modbus Reference Information
- C.4 SNMP Trouble Shooting
- C.5 Wireshark Hardware Requirements
- C.6 Example of Using Wireshark

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

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



# 1. Introduction

## 1.1 How to Use This Guide

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

You should at least read Sections 1 and 2 to gain an understanding of how the ValuPoint functions. You can use Sections 3 through 12 as reference material to look up as needed. There is a "Quick Help" section at the bottom of each web page in the ValuPoint which is generally sufficient for quick reference in setting up the ValuPoint.

## 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.**

**CAUTION: The lithium battery contained in this device may explode if mistreated. DO NOT recharge, disassemble, or dispose of in fire.**

**No action is required of the user to activate the battery that backs up the real time clock. Important: Replace battery with BR1225A only. Use of another battery may present a risk of fire or explosion.**

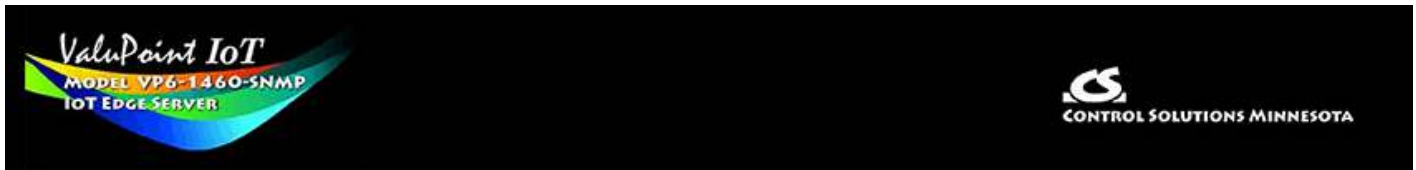
## 1.3 Warranty

**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 the ValuPoint

### 2.1 Overview of ValuPoint Operation

The ValuPoint VP6-1460-SNMP connects physical I/O (e.g. sensors) to SNMP. In addition, the VP6-1460-SNMP makes any Modbus device accessible to SNMP. You can use SNMP Get/Set to query I/O or Modbus points, or set up threshold rules to generate traps in response to physical I/O changes or Modbus changes. For example, you can have the ValuPoint send an SNMP Trap (Notification) when a contact closes.

#### Hardware Features of Model VP6-1460-SNMP

- 12 Analog/universal inputs, software selectable types
  - 0-10VDC, thermistor, discrete, dry contact, pulse
  - 0.1% reference, 12-bit resolution
  - Non-volatile totalizing count inputs (to 2Hz on all channels, to 1kHz on 4 channels)
- 2 Discrete outputs
  - Form A relay
  - 2A @ 120VAC
  - 2A @ 30VDC
- Battery backed real time clock/calendar
- Modbus TCP over Ethernet 10/100BaseT
- SNMP v1, v2c, v3
- Isolated RS-485 port, Modbus RTU at 1200 to 115200 baud
- Powered by 18-30VDC or 24VAC 50/60 Hz Class 2, 0.3A max.
- DIN rail mounting, 100mm H x 70mm W x 60mm D
- Operating temperature -40°C to +80°C; Humidity 5% to 90%
- FCC Class A, CE Mark
- Listed to UL 916 and (Canadian) C22.2 No. 205-M1983

### 2.2 Where to Start

- Start by connecting the ValuPoint as noted in the following section. Then set the IP address of your ValuPoint, and get familiar with the File Manager. You will find these covered in Section 3.
- Create some registers so you have a place to put data. Section 4 talks about this. Registers for the physical I/O points are created automatically. Configure the I/O as noted in section 4.2.
- Decide how you're going to talk to your Modbus device. Are you using Modbus TCP or RTU? Should the ValuPoint be master or slave? Based on how you answer these

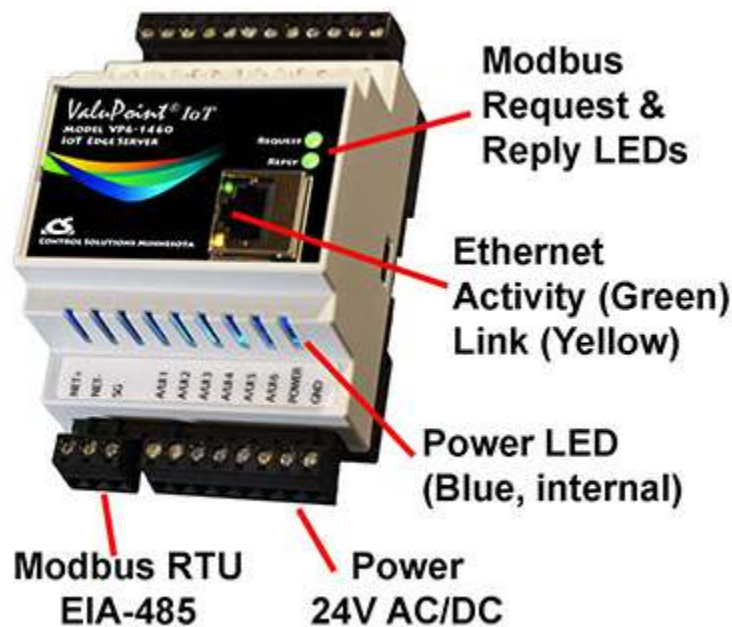
questions, you will choose from Sections 5 through 8.

- Assign your data points (local registers created as in Section 4) to the SNMP MIB as outlined in Section 9. Optionally configure traps as outlined in section 10.
- Are you interested in scheduling things that you want to happen? Take a look at the scheduler in Section 11.
- Last but not least is an advanced topic. If you want to create your own custom web pages to be served by the ValuPoint's internal server, that is covered in Section 12.
- Various details are covered in Appendix A through D. Be sure to look at the first 3 sections of Appendix A which cover hardware details you will need to be aware of. You can save the rest of the reference information for when you need it.

## 2.3 Connectors and Indicators

Follow these steps to make the initial connection to the ValuPoint VP6-1460.

- (a) Connect power. Apply +12 to +24VDC or 24VAC to the terminal marked "POWER", and common or ground the terminal marked "GND". (DC power is recommended.)



- (b) Connect a CAT5 cable between the RJ-45 jack on the ValuPoint, and your network switch or hub. You cannot connect directly to your PC unless you use a "crossover" cable.

- (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 about 20 seconds, 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.

## 2.4 Web User Interface

The default IP address as shipped is 10.0.0.101. Enter `http://10.0.0.101` in your browser's address window. Newer computers should be able to connect directly to that IP address. Older computers required that the PC be on the same subnet first, or that you add a route to your network configuration.

This generally works, but if this fails, you will need to temporarily change your computer's IP address to a fixed address that starts with 10.0.0. and ends with anything but 101.



**ValuPoint IoT**  
MODEL VP6-1460-SNMP  
IOT EDGE SERVER

**CONTROL SOLUTIONS MINNESOTA**

Local Data    Modbus    SNMP    System

Model VP6-1460-SNMP  
v6.01.9  
MAC 00:40:9D:DF:3A:5F

**Quick Help**

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

Open your browser, and enter `http://10.0.0.101/` in the address window. You should see a page with the "ValuPoint IoT VP6-1460" header shown above. From this point, you will find help on each page in the web site contained within the product.

When you click on any of the page tabs such as System, you will be asked for a user

name and password. The only default login as shipped is "root". The password is different for every ValuPoint shipped, and unique to your ValuPoint. Look for the root password document and/or label that was shipped with your device. If you have lost your root password, you will need to open a support ticket at <https://ticket.csimn.com> and provide the MAC address shown so that your original default password can be recovered. Or you can follow the procedure described in Appendix section A.6.

To change the IP address of the ValuPoint, 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.

The screenshot displays the Network configuration page in the ValuPoint web interface. The navigation menu at the top includes Local Data, Modbus, SNMP, System, System Setup, Scheduler, File Manager, Network (selected), Resources, User, and I/O Config. The main content area is divided into three sections:

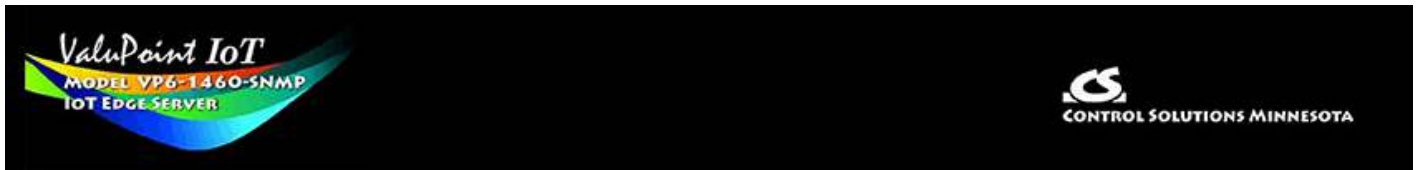
- IPv4 Settings:** Radio buttons for Automatic (selected) and Static. Fields include IPv4 Static IP Address (192.168.1.183), IPv4 Static Subnet Mask (255.255.255.0), and IPv4 Static Gateway (192.168.1.1). Corresponding fields for IPv4 Configured IP Address (192.168.1.183), IPv4 Subnet Mask (255.255.255.0), and IPv4 Gateway (192.168.1.1) are also present, along with an Apply button.
- IPv6 Settings:** Radio buttons for Disabled (selected), Automatic, and Static. Fields include IPv6 Link-Local IP Address (fe80::240:9dff:fedc:ddd), IPv6 Configured IP Address (---), IPv6 Static IP Address (---), IPv6 Prefix Length (64), and IPv6 Gateway Tunnel (::).
- DNS Settings:** Fields for Primary DNS (1.1.1.1) and Secondary DNS (8.8.8.8), with corresponding IPv6 addresses (::FFFF:1.1.1.1 and ::FFFF:8.8.8.8).

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 execute Save XML Config File on the File Manager page (System :: System Setup :: File Manager). Refer also to section 3.1.

## 2.5 External Programming Tool

The VP6-1460 has an available programming environment. The graphical programming environment is independent of the web user interface and is covered in a separate user guide supplement. The graphical programming environment is available for all VP6-14xx models and will function the same for both web and non-web versions of ValuPoint.





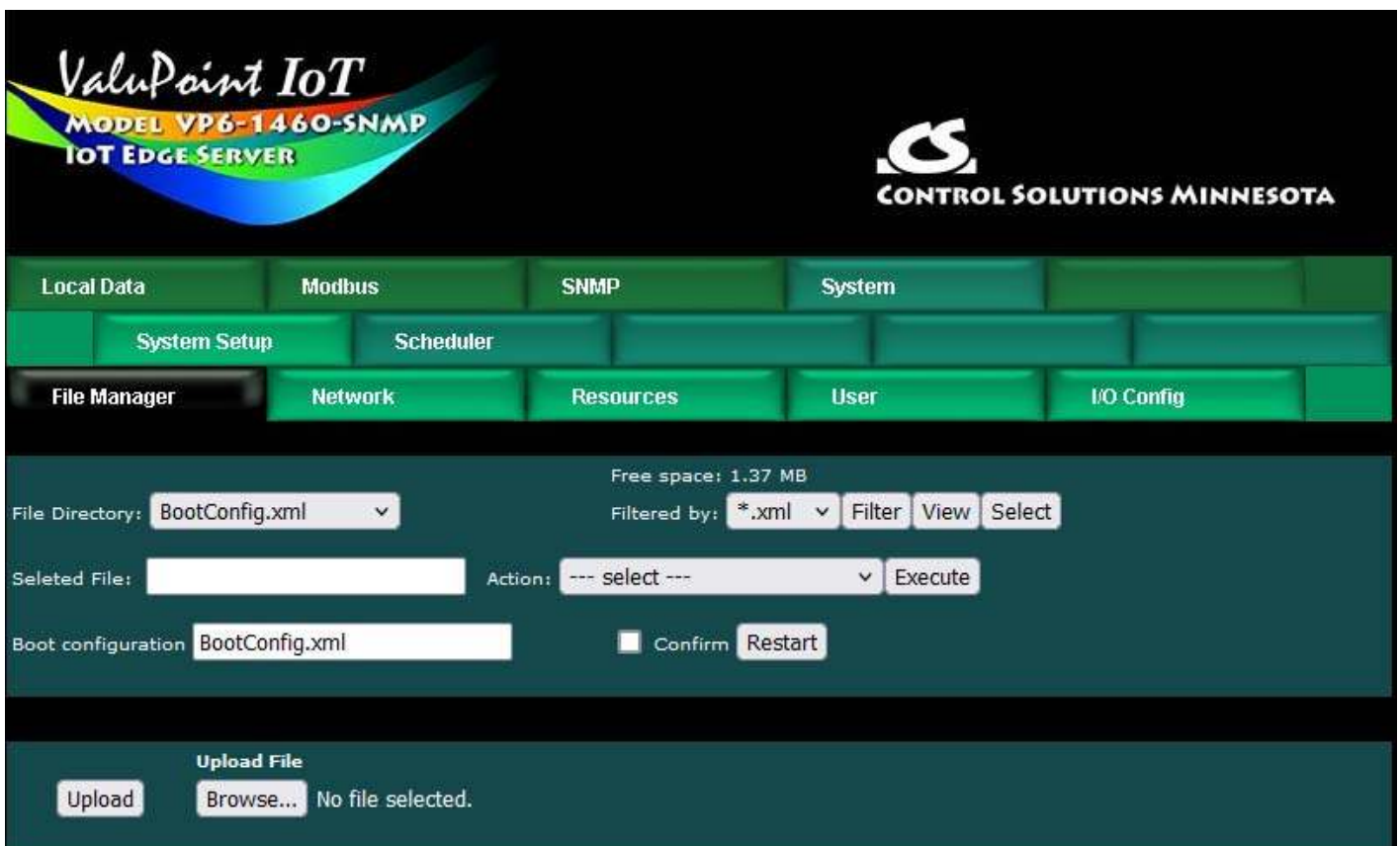
## 3. System Configuration and Resources

### 3.1 File Manager

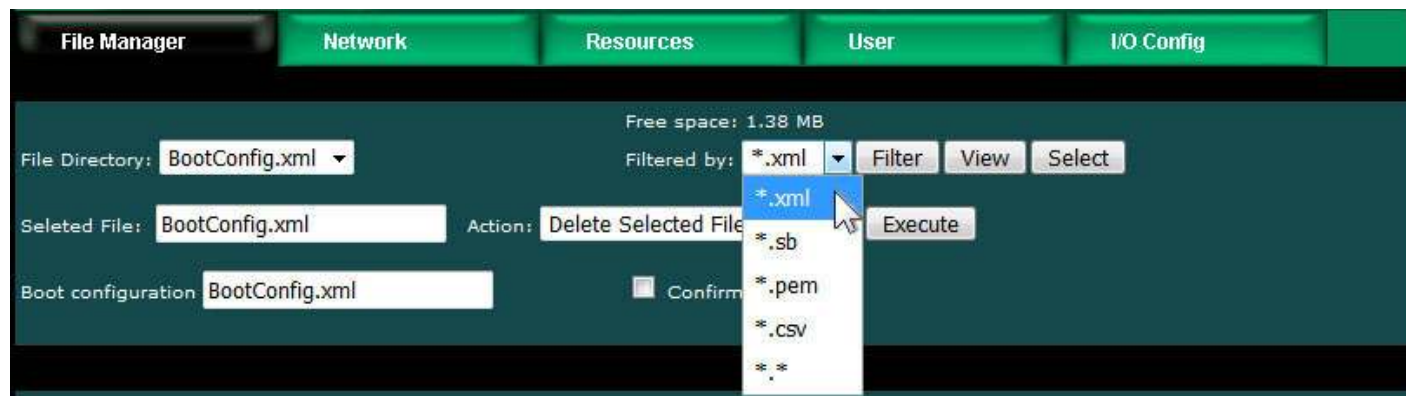
The File Manager page is probably one of the most important pages to know about. Among other things, this is where you tell the ValuPoint to save all of the changes you have made. The various "Update" buttons on the many pages in the web user interface only copy your configuration from your PC's browser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



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

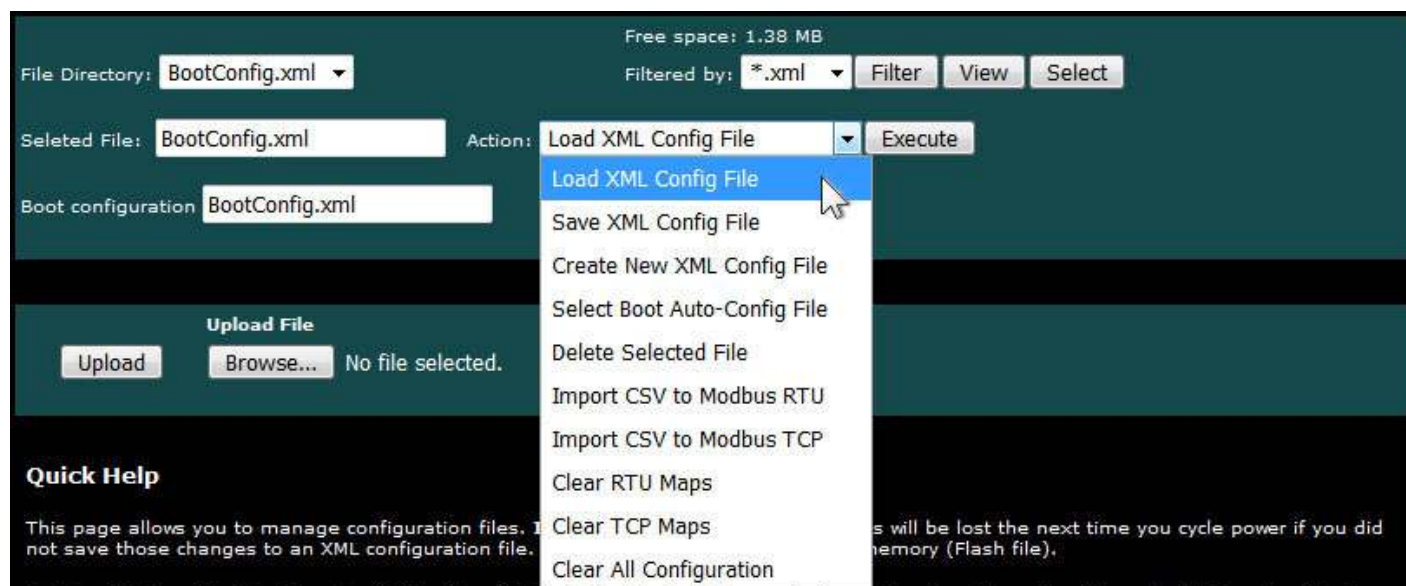


File type filters are as follows:

- \*.xml XML configuration files
- \*.sb Script Basic programs (not available with SNMP)
- \*.pem SSL certificates (for HTTPS)
- \*.csv CSV spreadsheet for Modbus register import
- \*.txt Text file (not used with SNMP)
- \*.\* 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 ValuPoint, 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 message about directory needing synchronizing, click the browser's "back" button again to return to this page and click Upload again. This gets the browser and HTTP server back in sync, and this requirement generally happens only once or twice following power-up.

**Restart:** To restart the ValuPoint, 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

**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 and select this action.

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 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 ValuPoint 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 ValuPoint'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 ValuPoint 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

**Delete Selected File:** Remove a file from the Flash file system by selecting it and executing this 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 IoT configuration, 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 ValuPoint. 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 ValuPoint 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 ValuPoint. Upon restart, and upon finding that the boot configuration name is BootConfig.xml, and it does not exist, the ValuPoint 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 ValuPoint. 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 ValuPoint 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 ValuPoints 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 ValuPoint, 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 ValuPoint's IP address as well as a few other important things.

The screenshot displays the Network configuration page with the following settings:

- IPv4 Settings:**
  - Mode:  Automatic,  Static
  - IPv4 Static IP Address: 192.168.1.188
  - IPv4 Static Subnet Mask: 255.255.255.0
  - IPv4 Static Gateway: 192.168.1.1
  - IPv4 Configured IP Address: 192.168.1.188
  - IPv4 Subnet Mask: 255.255.255.0
  - IPv4 Gateway: 192.168.1.1
- IPv6 Settings:**
  - Mode:  Disabled,  Automatic,  Static
  - IPv6 Link-Local IP Address: fe80::240:9dff:fedf:3a5f
  - IPv6 Configured IP Address: ---
  - IPv6 Static IP Address: ---
  - IPv6 Prefix Length: 64
  - IPv6 Gateway Tunnel: ::
- DNS Settings:**
  - Primary DNS: 8.8.8.8 (hex: ::FFFF:8.8.8.8)
  - Secondary DNS: 1.1.1.1 (hex: ::FFFF:1.1.1.1)

### 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 ValuPoint 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 ValuPoint maintains time and date via SNTP services or its internal

Real Time Clock/Calendar (RTC). The RTC can also be used as backup should SNTP be unavailable due to network disconnect.

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. You may need to restart the ValuPoint if NTP had never before been initialized.

Daylight savings time start/end rules consist of "date/time" where the date (m.n.d) indicates the day when summer time starts or ends, and time (hour:min:sec) is the current local time when summer time starts/ends. The date portion of the rule is formatted as follows:

m indicates the month ( $1 \leq m \leq 12$ )

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.

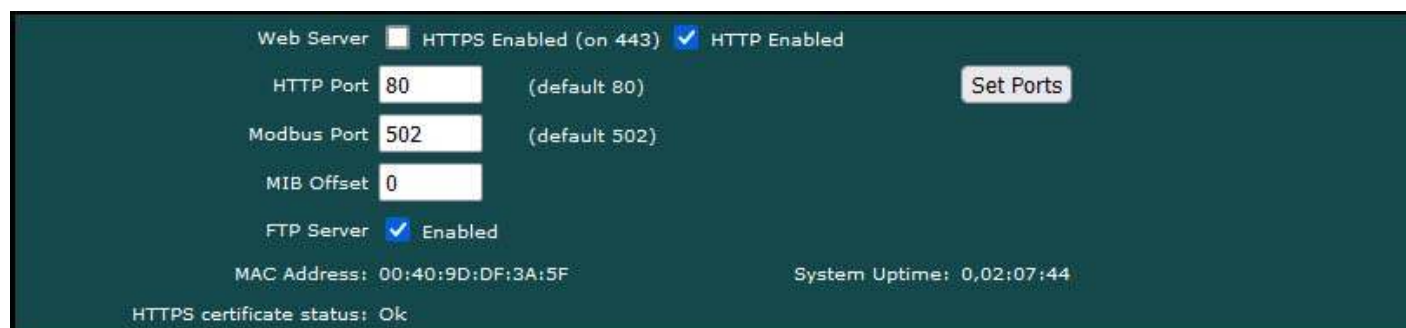
Latitude and longitude for the location of this device should be entered if you want to use the astronomical clock feature of the scheduler. Without latitude and longitude, the calculations for sunrise and sunset will be incorrect.

RTC (Real Time Clock) Setup: Check the "Use RTC" box, and enter the current date and time in the window below that box. Then click Set RTC. In order to use the scheduler without any SNTP, you do still need to enter Latitude and Longitude and click Set NTP. The RTC does not automatically adjust for daylight savings. If both NTP and RTC are enabled, then the RTC time/date will be updated from SNTP when available.

**CAUTION:** The lithium battery contained in this device may explode if mistreated. DO NOT recharge, disassemble, or dispose of in fire.

No action is required of the user to activate the battery that backs up the real time clock. Important: Replace battery with BR1225A only. Use of another battery may present a risk of fire or explosion.

### 3.3.3 Port Settings



Web Server  HTTPS Enabled (on 443)  HTTP Enabled

HTTP Port  (default 80)

Modbus Port  (default 502)

MIB Offset

FTP Server  Enabled

MAC Address: 00:40:9D:DF:3A:5F System Uptime: 0,02:07:44

HTTPS certificate status: Ok

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 D) 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 ValuPoint to make that new port take effect. Don't forget to append the port number to the ValuPoint's IP address when attempting to browse the web user interface if it has been moved 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 ValuPoint is 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 ValuPoint before they will take effect.

### 3.4 Resource Allocation

Historically, Control Solutions devices 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.

**Valupoint IoT**  
MODEL VP6-1460-SNMP  
IOT EDGE SERVER

**CONTROL SOLUTIONS MINNESOTA**

Local Data   Modbus   SNMP   System

System Setup   Scheduler

File Manager   Network   **Resources**   User   I/O Config

Check   Commit   Confirm   Restart

Resource	Current	Pending
Local Modbus Register Pool Size	400	400
Number of Data Calculate Rules	100	100
Number of Data Copy Rules	100	100
MIB Variable Count, Integer 32-bit	200	200
MIB Variable Count, Unsigned 64-bit	20	20
MIB Variable Count, Float 32-bit	20	20
MIB Variable Count, Float 64-bit	20	20
MIB Variable Count, Char String	20	20
Number of SNMP Trap Sender Devices	5	5
Number of SNMP Trap Send Rules	50	50
Number of Scheduler Weekly Events	50	50
Number of Scheduler On Demand Events	20	20
Number of Modbus RTU Read Maps	200	200
Number of Modbus RTU Write Maps	50	50
Number of Modbus TCP Devices	10	10
Number of Modbus TCP Client Read Maps	100	100
Number of Modbus TCP Client Write Maps	20	20
Number of Modbus TCP Server Connections	20	20
Estimated Memory Utilization	24.98%	24.98%
Persistent Data Status	Ok	Reset Persistent Data

The estimated memory utilization shown at the bottom gives you an indication of how close you are to running out of memory. You will not be allowed to commit a resource allocation greater than 100%.

The Reset Persistent Data button is used to clear (zero) stored persistent data. Changing the local register pool size will clear persistent data. To clear data without changing any allocations, use this button. Local registers have the option of being

configured as "persistent" which means the value contained in that register will be retained through restart or power cycle.

### 3.5 User Login Passwords

There is only one default login provided initially. That login is the username "root" and root's password is a unique password generated specifically for this particular ValuPoint. That unique password was provided for you in documentation included with the shipment. That unique password complies with California Consumer Privacy Act SB-327, which requires all Internet connected devices to have unique default passwords.

Once logged in as "root", you have the option of creating up to five additional logins.

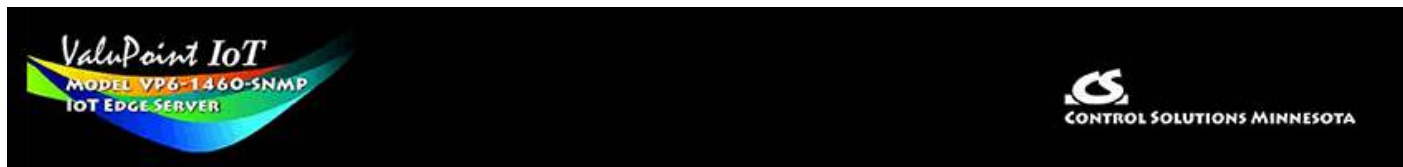
The privilege level Administrator lets that user see and change anything. The privilege level Maintenance allows the user to log in and see (and change) values in the local registers via the Local Registers page, but cannot access any other pages. The Restricted level will block access to everything except user defined web pages.

You also have the option of IP filtering. If set, then the user can only access ValuPoint's web pages from that IP address. Leave set to 0.0.0.0 to disable filtering.

Only the root user will see this version of the User page. Other users will only be able to change their own password. To add or change a user, enter the name and credentials, check Confirm Change, and click Change. To delete a user, clear the name field, check Confirm Change, and click the Change button.

The screenshot shows a web interface with a navigation menu at the top. The 'User' menu item is highlighted. Below the menu is a table with columns: User Name, Password, Privilege Level, IP Filter, and Confirm Change. A 'Change' button is located to the right of the table.

User Name	Password	Privilege Level	IP Filter	Confirm Change
system	•••••	Administrator ▼	0.0.0.0	<input type="checkbox"/>
		Restricted ▼	0.0.0.0	<input type="checkbox"/>
		Restricted ▼	0.0.0.0	<input type="checkbox"/>
		Restricted ▼	0.0.0.0	<input type="checkbox"/>
		Restricted ▼	0.0.0.0	<input type="checkbox"/>
root	•••••	Unrestricted	0.0.0.0	<input type="checkbox"/>
root confirm				



## 4. Configuring Local Registers

### 4.1 Creating Local Registers

Control Solutions devices 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 ValuPoint out of the box, it has no registers except for the initial set of registers assigned to the physical I/O points. You get to create remaining 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 as illustrated below.

Local Register #	Register Name	Set	Register Data	Register Format
00001	A/UI #1	<input type="checkbox"/>	0.000000	Single Float
00003	A/UI #2	<input type="checkbox"/>	0.005569	Single Float
00005	A/UI #3	<input type="checkbox"/>	0.000000	Single Float
00007	A/UI #4	<input type="checkbox"/>	0.005569	Single Float
00009	A/UI #5	<input type="checkbox"/>	0.000000	Single Float
00011	A/UI #6	<input type="checkbox"/>	0.005569	Single Float
00013	A/UI #7	<input type="checkbox"/>	0.000000	Single Float
00015	A/UI #8	<input type="checkbox"/>	0.005569	Single Float
00017	A/UI #9	<input type="checkbox"/>	0.000000	Single Float
00019	A/UI #10	<input type="checkbox"/>	0.002785	Single Float
00021	A/UI #11	<input type="checkbox"/>	0.008354	Single Float
00023	A/UI #12	<input type="checkbox"/>	0.002785	Single Float
00025	DO #1	<input type="checkbox"/>	0	Unsigned 16-bit
00026	DO #2	<input type="checkbox"/>	0	Unsigned 16-bit

You do not need to create any additional registers if you will only be using the I/O

points. To begin the process of creating additional registers, click on the last register number on the list. 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.

00023	A/UI #12	<input type="checkbox"/>	0.002785	Single Float
00025	DO #1	<input type="checkbox"/>	0	Unsigned 16-bit
00026	DO #2	<input type="checkbox"/>	0	Unsigned 16-bit

 Quick Help

Upon clicking a register number, the register detail will be displayed. This can be either detailed configuration of an existing register, or detail about new registers you are about to add. The only time you can select the data format is when adding new registers. Once the registers are created, the data format cannot be changed because this impacts how many Modbus registers are actually used. If you had a set of 16-bit registers defined, and wanted to change one of them to 32-bit, it would cause all of the remaining registers to be renumbered if such a change was allowed. While this may seem harmless at first, it becomes a huge mess trying to keep track of where in all the rules 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 ValuPoint as anything bigger than 16 bits, it is actually a pair (or series of 2 or more) of 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.

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. ValuPoint 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. The maximum possible register number will be the count indicated on the Resources page as Local Modbus Register Pool Size.

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
00025	DO #1	<input type="checkbox"/>	0	Unsigned 16-bit
00026	DO #2	<input type="checkbox"/>	0	Unsigned 16-bit
00027	Data Value 1	<input type="checkbox"/>	0.000000	Single Float
00029	Data Value 2	<input type="checkbox"/>	0.000000	Single Float
00031	Data Value 3	<input type="checkbox"/>	0.000000	Single Float
00033	Data Value 4	<input type="checkbox"/>	0.000000	Single Float
00035	Data Value 5	<input type="checkbox"/>	0.000000	Single Float
00037	Data Value 6	<input type="checkbox"/>	0.000000	Single Float
00039	Data Value 7	<input type="checkbox"/>	0.000000	Single Float
00041	Data Value 8	<input type="checkbox"/>	0.000000	Single Float
00043	Data Value 9	<input type="checkbox"/>	0.000000	Single Float
00045	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.

00041	Data Value 8	<input type="checkbox"/>	0.000000	Single Float
00043	Data Value 9	<input type="checkbox"/>	0.000000	Single Float
00045	Data Value 10	<input type="checkbox"/>	0.000000	Single Float

**Quick Help**

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 ValuPoint.

**Local Registers**   Calculate   Copy   Report

Register #    Links: -----   Update   < Prev   Next >

Register data format:    Size:    Register name

Least significant data should be in first register:    Display as hexadecimal:    Is Persistent:

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

First register number to add:    Add this many:    **Add New**   Register #    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 Registers**   Calculate   Copy   Report

Showing registers from    Update   < Prev   Next >

Local Register #	Register Name	Set	Register Data	Register Format
00027	Data Value 1	<input type="checkbox"/>	0.000000	Single Float
00029	Data Value 2	<input type="checkbox"/>	0.000000	Single Float
00031	Data Value 3	<input type="checkbox"/>	0.000000	Single Float
00033	Data Value 4	<input type="checkbox"/>	0.000000	Single Float
00035	Data Value 5	<input type="checkbox"/>	0.000000	Single Float
00037	Data Value 6	<input type="checkbox"/>	0.000000	Single Float
00039	Data Value 7	<input type="checkbox"/>	0.000000	Single Float
00041	Data Value 8	<input type="checkbox"/>	0.000000	Single Float
00043	Data Value 9	<input type="checkbox"/>	0.000000	Single Float
00045	Data Value 10	<input type="checkbox"/>	0.000000	Single Float
00047	Char String 1	<input checked="" type="checkbox"/>	Hello Modbus	Char String[40]
00067	Char String 2	<input checked="" type="checkbox"/>	String can be read by Modbus	Char String[40]
00087	Char String 3	<input checked="" type="checkbox"/>	Or written by Modbus	Char String[40]
00107	Char String 4	<input type="checkbox"/>		Char String[40]



You can set the character strings from the web page,. The strings can be read or written by Modbus as a series of registers with 2 characters per register.

## 4.2 Configuring Physical I/O

The first 26 registers in the ValuPoint are permanently assigned to the physical I/O points. Those register numbers are illustrated in the first screen shot in Section 4.1 above.

ValuPoint IoT  
MODEL VP6-1460-SNMP  
IOT EDGE SERVER

CONTROL SOLUTIONS MINNESOTA

Local Data Modbus SNMP System

System Setup Scheduler

File Manager Network Resources User I/O Config

Update

A/UI #	Hardware Type	Scale	Offset	Qualifier	Is Persistent
1	0-10V, 12-bit	0.00	0.00	0	<input type="checkbox"/>
2	0-10V, 12-bit	0.00	0.00	0	<input type="checkbox"/>
3	4-20mA, with resistor	6.250000	-25.00000	499	<input type="checkbox"/>
4	4-20mA, with resistor	6.250000	-25.00000	499	<input type="checkbox"/>
5	Pulse counter	0.00	0.00	0	<input checked="" type="checkbox"/>
6	Pulse counter	0.00	0.00	0	<input checked="" type="checkbox"/>
7	Pulse counter	0.00	0.00	0	<input checked="" type="checkbox"/>
8	Pulse counter	0.00	0.00	0	<input checked="" type="checkbox"/>
9	Dry contact, active closed	0.00	0.00	50	<input type="checkbox"/>
10	Dry contact, active closed	0.00	0.00	50	<input type="checkbox"/>
11	Dry contact, active closed	0.00	0.00	50	<input type="checkbox"/>
12	Dry contact, active closed	0.00	0.00	50	<input type="checkbox"/>

Coprocessor Status: Ok (v6.01.8)

The relay outputs do not require any configuration. The analog/universal inputs do require configuration on the I/O Config page. Select the desired input type and enter additional parameters as applicable.

File Manager		Network		Resources		User		I/O Config	
Update									
A/UI #	Hardware Type			Scale	Offset	Qualifier	Is Persistent		
1	0-10V, 12-bit			0.00	0.00	0	<input type="checkbox"/>		
2	Unconfigured			0.00	0.00	0	<input type="checkbox"/>		
3	0-10V, 12-bit			6.250000	-25.00000	499	<input type="checkbox"/>		
4	4-20mA, with resistor			6.250000	-25.00000	499	<input type="checkbox"/>		
	Discrete, active high						<input type="checkbox"/>		
	Discrete, active low						<input type="checkbox"/>		
5	Dry contact, active open			0.00	0.00	0	<input checked="" type="checkbox"/>		
6	Dry contact, active closed			0.00	0.00	0	<input checked="" type="checkbox"/>		
7	Pulse counter			0.00	0.00	0	<input checked="" type="checkbox"/>		
8	Resistance			0.00	0.00	0	<input checked="" type="checkbox"/>		
	Position pot, 1K-30K						<input checked="" type="checkbox"/>		
9	Thermistor, 10K type III, F			0.00	0.00	0	<input checked="" type="checkbox"/>		
10	Thermistor, 10K type II, F			0.00	0.00	50	<input type="checkbox"/>		
	Thermistor, 3K type II, F						<input type="checkbox"/>		
11	Thermistor, 20K type IV, F			0.00	0.00	50	<input type="checkbox"/>		
	Thermistor, 5K type II, F						<input type="checkbox"/>		
12	Thermistor, 10K type III, C			0.00	0.00	50	<input type="checkbox"/>		
	Thermistor, 10K type II, C						<input type="checkbox"/>		
	Thermistor, 3K type II, C						<input type="checkbox"/>		
	Thermistor, 20K type IV, C						<input type="checkbox"/>		
	Thermistor, 5K type II, C						<input type="checkbox"/>		

The A/UI inputs may function as any of the following:

- 0-10V, 12-bit resolution (reading is Volts)
- 4-20mA, with resistor (reading is mA)
- Discrete, active high
- Discrete, active low
- Dry contact, active open
- Dry contact, active closed
- Pulse counter
- Resistance (reading is ohms)
- Position pot, 1K-30K (reading is percentage)

- Thermistor, 10K type III, F (readings in degrees for all of following)
- Thermistor, 10K type II, F
- Thermistor, 3K type II, F
- Thermistor, 20K type IV, F
- Thermistor, 5K type II, F
- Thermistor, 10K type III, C
- Thermistor, 10K type II, C
- Thermistor, 3K type II, C
- Thermistor, 20K type IV, C
- Thermistor, 5K type II, C

Dedicated hardware is available for pulse counting on channels 5, 6, 7, and 8. The only limiting factor on maximum pulse rate on these inputs is the noise filtering on the inputs. The inputs have been verified to count at up to 1kHz provided the signal amplitude is sufficient. Pulse counting is supported on the remaining input channels, but the counting is done by software and therefore the rate is limited to about 2Hz.

The 4-20mA setting is used as a label since that is most common. However, the input is actually going to be 0-20mA, and the register value for that range will be 0..20. The scale and offset can be used to convert 4-20mA to a 0-100% value by using scale = 6.25 and offset = -25. A dropping resistor of 500 ohms will use the full 10 volt input range. A dropping resistor of 250 ohms can be used - it will simply provide less resolution.

Scale – Scaling applies the formula  $y=mx+b$ . When reading from a hardware input, the raw data as read is multiplied by the scale factor, then the offset is added to produce the resulting register value. NOTE: If no scale factor is given (zero is entered), no scaling will be done, as if scale=1 and offset=0.

Offset – The offset portion of the scaling as noted above.

Qualifier – Enter the configuration qualifier value, if applicable, for the selected configuration. Qualifiers are required only in the following modes:

*4-20mA mode:* The qualifier is the resistance in ohms of the dropping resistor used to convert the current to voltage. An external resistor must be provided, connected between the A/UI input and ground/common. The resistor needs to be 1/2 watt (2 watt to withstand 24V power), and is left external simply because miswiring the 4-20mA sensor can easily apply 24V power directly to the input and cause the dropping resistor to heat up and possibly fail. The external resistor is simple to replace, whereas an internal resistor on a circuit board would be more trouble to replace.

*Discrete and Dry Contact modes:* The qualifier is a threshold between 1% and 99% at which the input should trip from off to on or vice versa. The A/UI inputs are specified as 0-10V inputs. Therefore, since discrete inputs are sampled as analog values and compared to a threshold, the qualifier here is a percentage of 10V for the trip point. A value of 50% will mean a threshold of around 5V.

*Position Pot:* Position is simply a different interpretation of resistance measurement. The value resulting from position pot measurement will be a percentage from 0% to 100%, but this percentage will be a ratio based on the resistance value in ohms provided as the qualifier.

### 4.3 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.

The screenshot shows the 'Local Registers' configuration page in the ValuPoint IoT web interface. The interface includes a navigation menu with 'Local Data', 'Modbus', 'SNMP', and 'System'. Below this is a sub-menu with 'Data', 'Local Registers', 'Calculate', 'Copy', and 'Report'. The 'Local Registers' section is active, showing a table with one register: Register # 101, Links: -----, and an 'Update' button. Below the table, there are configuration options for the selected register: Register data format: Unsigned 32-bit, Size: 0, Register name: Unsigned Value 1. There are checkboxes for 'Least significant data should be in first register' (unchecked), 'Display as hexadecimal' (checked), and 'Is Persistent' (unchecked). There is also a field for 'Apply this default value: 0.000000' and a checkbox for 'At power-up' (checked) with a timeout of 45 seconds. At the bottom, there are fields for 'First register number to add: 127', 'Add this many: 1', and an 'Add New' button. On the right, there is a 'Register # 101' field and a 'Delete' button.

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 ValuPoint 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 server 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.

The "Is Persistent" option means this register will retain its most recent value through a power outage or restart, with time delay restrictions. ***The persistent selection does not take effect until the Reset Persistent Data button is clicked on the Resources page.***

The persistent data is stored in non-volatile memory which has a life expectancy of 4 million write cycles. However, if the data is written too frequently, the memory will still reach its end of life prematurely. Therefore, to extend the life of the EEPROM, persistent data is only updated once every 15 minutes. If the data has not been written in the past 15 minutes, new data will be written to EEPROM immediately. Otherwise writing to EEPROM will be delayed until the expiration of the timer for that register.

The most recently stored data is read from EEPROM and placed into the local register at startup.

#### 4.4 Local Register Calculate Rules

The ValuPoint 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".

These template rules can be useful for doing minor data manipulation or testing for purposes of enabling rules, or for generating SNMP traps.

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 an alternate to Script Basic programming for very simple operations such as logically ANDing or ORing a condition to produce a result to be sent out to another register. Select the operation to be performed from the drop list. All but logical NOT operations require two operands. Enter 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 require ranges of multiple registers.

Delete will remove a rule number shown in the "Showing" box. 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 (unused rules will be deleted).

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 error will prevent other updates. Because error trapping happens sooner than the insert/delete process, the delete will not remove the 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 a rule number effectively deletes the rule even though it will still appear in the list until deleted. Unused rules at the end of the list will

Here is an example of a template rule that multiplies register 27 by register 29 and places the result in register 31.

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	27	and	29	31
2	none	0	and	0	0

# Rules Enabled: 2

Insert Delete

If registers 27 and 28 contain the values shown below, then the result shown in register 31 would be expected.

Local Registers Calculate Copy Report

Showing registers from 27 Update < Prev Next >

Local Register #	Register Name	Set	Register Data	Register Format
00027	Data Value 1	<input type="checkbox"/>	100.000000	Single Float
00029	Data Value 2	<input type="checkbox"/>	0.250000	Single Float
00031	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 33 to a value of 123.

Rule #	Perform Operation	Using Register #	And/Thru Using	This Reg#/Value	Place Result in Register #
1	set	33	using	123	33
2	none	0	and	0	0

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

add	Add two registers
subtract	Subtract second register from first
multiply	Multiply two registers
divide	Divide first register by second
sum	Sum two or more registers
average	Average two or more registers
min	Find lowest value among two or more registers
max	Find highest value among two or more registers
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)
shift left	Shift content of register left by 'using' number of bits
shift right	Shift content of register right by 'using' number of bits

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 ValuPoint is functioning as a Modbus 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	28	using	Fh	27	
2	pack	29	using	F0h	27	
3	pack	30	using	F00h	27	
4	fill	27	using	3000h	27	
5	pack	37	using	FFh	35	
6	pack	39	using	FF0000h	35	
7	fill	35	using	80000000h	35	
8	none	0	and	0	0	
# Rules Enabled: 8						Insert Delete

The example below shows the content of result registers 27 and 35 using the above calculate rules and the local register values shown. The contents of registers 28, 29, and 30 are packed into register 27. The contents of registers 37 and 39 are packed into 35 (all 32-bit register pairs).

Local Registers		Calculate	Copy	Report		
Showing registers from 27						Update < Prev Next >
Local Register #	Register Name	Set	Register Data		Register Format	
00027	Single Register 1	<input type="checkbox"/>	3432		Unsigned 16-bit	
00028	Single Register 2	<input type="checkbox"/>	2		Unsigned 16-bit	
00029	Single Register 3	<input type="checkbox"/>	3		Unsigned 16-bit	
00030	Single Register 4	<input type="checkbox"/>	4		Unsigned 16-bit	
00031	Single Register 5	<input type="checkbox"/>	0		Unsigned 16-bit	
00032	Single Register 6	<input type="checkbox"/>	0		Unsigned 16-bit	
00033	Single Register 7	<input type="checkbox"/>	0		Unsigned 16-bit	
00034	Single Register 8	<input type="checkbox"/>	0		Unsigned 16-bit	
00035	Double Register 1	<input type="checkbox"/>	807F003F		Unsigned 32-bit	
00037	Double Register 2	<input type="checkbox"/>	63		Unsigned 32-bit	
00039	Double Register 3	<input type="checkbox"/>	127		Unsigned 32-bit	
00041	Double Register 4	<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						
Update < Prev Next >						
Rule #	Perform Operation	Using Register #	And/Thru Using	This Reg#/Value	Place Result in Register #	
1	unpack	27	using	Fh	28	
2	unpack	27	using	F0h	29	
3	unpack	27	using	F00h	30	
4	unpack	35	using	FFh	37	
5	unpack	35	using	FF0000h	39	
6	none	0	and	0	0	
# Rules Enabled: 6						
Insert Delete						

Register 27 is unpacked into registers 28, 29 and 30. Register 35 is unpacked into registers 37 and 39 (all 32-bit register pairs).

Local Registers		Calculate	Copy	Report		
Showing registers from 27						
Update < Prev Next >						
Local Register #	Register Name	Set	Register Data	Register Format		
00027	Single Register 1	<input type="checkbox"/>	3432	Unsigned 16-bit		
00028	Single Register 2	<input type="checkbox"/>	2	Unsigned 16-bit		
00029	Single Register 3	<input type="checkbox"/>	3	Unsigned 16-bit		
00030	Single Register 4	<input type="checkbox"/>	4	Unsigned 16-bit		
00031	Single Register 5	<input type="checkbox"/>	0	Unsigned 16-bit		
00032	Single Register 6	<input type="checkbox"/>	0	Unsigned 16-bit		
00033	Single Register 7	<input type="checkbox"/>	0	Unsigned 16-bit		
00034	Single Register 8	<input type="checkbox"/>	0	Unsigned 16-bit		
00035	Double Register 1	<input type="checkbox"/>	807F003F	Unsigned 32-bit		
00037	Double Register 2	<input type="checkbox"/>	63	Unsigned 32-bit		
00039	Double Register 3	<input type="checkbox"/>	127	Unsigned 32-bit		
00041	Double Register 4	<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 32 contains a zero. If register 32 is not equal to zero, then rule 6 will be executed.

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	comp = N	27	using	10	28	
2	comp < N	27	using	10	29	
3	comp > N	27	using	10	30	
4	set	30	using	202	31	
5	skip = N	32	using	0	32	
6	set	33	using	0	33	
7	set	34	using	88	34	
8	none	0	and	0	0	
# Rules Enabled: 8						Insert Delete

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

Local Registers		Calculate	Copy	Report		
Showing registers from 27						Update < Prev Next >
Local Register #	Register Name	Set	Register Data	Register Format		
00027	Single Register 1	<input type="checkbox"/>	10	Unsigned 16-bit		
00028	Single Register 2	<input type="checkbox"/>	65535	Unsigned 16-bit		
00029	Single Register 3	<input type="checkbox"/>	0	Unsigned 16-bit		
00030	Single Register 4	<input type="checkbox"/>	0	Unsigned 16-bit		
00031	Single Register 5	<input type="checkbox"/>	202	Unsigned 16-bit		
00032	Single Register 6	<input type="checkbox"/>	0	Unsigned 16-bit		
00033	Single Register 7	<input type="checkbox"/>	0	Unsigned 16-bit		
00034	Single Register 8	<input type="checkbox"/>	88	Unsigned 16-bit		
00035	Double Register 1	<input type="checkbox"/>	0	Unsigned 32-bit		

## 4.5 Local Register Copy Rules

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

ValuPoint IoT  
MODEL VP6-1460-SNMP  
IOT EDGE SERVER

CONTROL SOLUTIONS MINNESOTA

Local Data Modbus SNMP System

Data

Local Registers Calculate Copy Report

Showing 1 to 2 of 2 Update < Prev Next >

Rule #	Source Register #	Destination Register #
1	27	29
2	0	0

# Rules Enabled: 2 Insert Delete

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

Local Registers Calculate Copy Report

Showing registers from 27 Update < Prev Next >

Local Register #	Register Name	Set	Register Data	Register Format
00027	Data Value 1	<input type="checkbox"/>	123.000000	Single Float
00029	Data Value 2	<input type="checkbox"/>	123.000000	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	45	47
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 45 Update < Prev Next >

Local Register #	Register Name	Set	Register Data	Register Format
00045	Data Value 10	<input type="checkbox"/>	45.980000	Single Float
00047	Char String 1	<input type="checkbox"/>	45.980000	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.

## 4.6 Device Status Reporting

The ValuPoint read maps include the ability to set a default value upon 'n' read fails, meaning that if the ValuPoint 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 ValuPoint 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.

ValuPoint IOT  
MODEL VP6-1460-SNMP  
IOT EDGE SERVER

CONTROL SOLUTIONS MINNESOTA

Local Data Modbus SNMP System

Data

Local Registers Calculate Copy Report

Showing 1 to 3 of 3 Update < Prev Next >

Report Status of	Device or Unit #	To This Register	With This Delay (Sec.)	Delete
Modbus RTU Master	1	27	20	<input type="checkbox"/>
Modbus RTU Master	2	28	20	<input type="checkbox"/>
Modbus TCP Client	1	29	20	<input type="checkbox"/>

Modbus TCP Client 1 29 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.

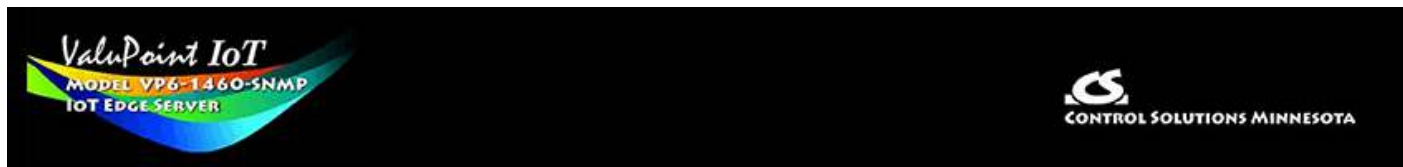
To remove a report from the list, check the box in the Delete column and then click Update. Click Prev or Next to scroll through the list.

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. Configuring ValuPoint as a Modbus RTU Master

The ValuPoint can be a Modbus RTU master or slave. As a master you can read Modbus data from, or write Modbus data to, other Modbus slaves. The ValuPoint 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 the SNMP MIB.

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.

### 5.1 Modbus RTU Device Configuration

Modbus device configuration for RTU really consists of just port configuration. When brand new out of the box, the RTU port defaults to RTU slave.

The screenshot shows the web interface for configuring the ValuPoint IoT device. The top navigation bar includes 'Local Data', 'Modbus', 'SNMP', and 'System'. Under 'Modbus', there are sub-tabs for 'RTU Setup', 'RTU Data', 'TCP Setup', and 'TCP Data'. The 'RTU Setup' tab is active, showing 'Local Device', 'RTU Read Map', and 'RTU Write Map' options. An 'Update' button is in the top right. The configuration area has two sections: 'I am the Master' (selected) and 'I am a Slave'. Under 'I am the Master', there are fields for 'Baud Rate' (19200), 'Parity' (None, 1 Stop Bit), 'Default Poll Rate' (0.000 Seconds), and 'Timeout' (0.000 Seconds). Under 'I am a Slave', there is a field for 'My Address or Unit #' (1) and a checkbox for 'Use FC 5/6 instead of 15/16 for unit numbers (slave addresses) starting at' (0).

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

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

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

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

The screenshot shows the 'RTU Setup' tab in a software interface. The interface has a dark green header with tabs for 'RTU Setup', 'RTU Data', 'TCP Setup', and 'TCP Data'. Below this is a sub-header with 'Local Device', 'RTU Read Map', and 'RTU Write Map'. An 'Update' button is in the top right. The main area is divided into two columns: 'I am the Master' (selected) and 'I am a Slave'. Under 'I am the Master', there are fields for 'Baud Rate' (19200), 'Parity' (None, 1 Stop Bit), 'Default Poll Rate' (5.000 Seconds), and 'Timeout' (0.500 Seconds). Under 'I am a Slave', there is a field for 'My Address or Unit #' (0). At the bottom, there is a checkbox 'Use FC 5/6 instead of 15/16 for unit numbers (slave addresses) starting at' followed by a field containing '0'.

## 5.2 Modbus RTU Master Read Maps

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



ValuPoint IoT  
MODEL VP6-1460-SNMP  
IOT EDGE SERVER

CONTROL SOLUTIONS MINNESOTA

Local Data Modbus SNMP System

RTU Setup RTU Data TCP Setup TCP Data

Local Device RTU Read Map RTU Write Map

Showing 1 to 1 of 1 Update < Prev Next >

Map #	Remote Type	Remote Register Format	Remote Register #	Remote Unit #	Local Register #	Name
1	None	None	0	0	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 RTU Write Map to write data to those devices. The full parameter set is different for read versus write.

Local Device RTU Read Map RTU Write Map

Showing 1 to 1 of 1 Update < Prev Next >

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

Quick

This page only applies if the local device is configured as a Master.

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

Local Device		RTU Read Map		RTU Write Map		
Showing 1 to 1 of 1						
Update < Prev Next >						
Map #	Remote Type	Remote Register Format	Remote Register #	Remote Unit #	Local Register #	Name
1	Holding Register	None	0	0	0	

**Quick Help**

This page only applies if the local device is configured as a Master.

Map number simply tells you where the data is stored in 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 text box, then click Update.

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

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

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.)

Local Device		RTU Read Map		RTU Write Map		
Showing 1 to 2 of 2						
Update < Prev Next >						
Map #	Remote Type	Remote Register Format	Remote Register #	Remote Unit #	Local Register #	Name
1	Holding Register	UINT-16	11	2	27	Data Value 1
2	None	None	0	0	0	

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

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

Local Device		RTU Read Map		RTU Write Map			
		Showing 1 to 2 of 2		Update		< Prev Next >	
Map #	Remote Type	Remote Register Format	Remote Register #	Remote Unit #	Local Register #	Name	
1	Holding Register	UINT-16	11	2	27	Data Value 1	
	None	None	0	0	0		

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

Local Device		RTU Read Map		RTU Write Map			
Map # 1		Update		< Prev		Next >	
Read	Holding Register	as	Unsigned 16-bit	Size:	0		
From register #	11	at Unit #	2	With low register first if checked:	<input type="checkbox"/>		
Apply bit mask if applicable:	0000	then apply scale:	0.000000	and offset:	0.000000		
Save in local register #	27	named	Data Value 1	Repeat this process every	5.0	seconds.	
Apply this default value:	0.000000	after	0	read failure(s).			
<input type="checkbox"/>	Enable this map only when index register	0	is set to a value of	0			
# RTU Read Maps Enabled:		2		Insert		Delete	

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

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

The poll rate ("Repeat this process...") determines how often the 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 RTU port.

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 slave if you know the slave 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.

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 slave 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).

Local Registers		Calculate	Copy	Report	
Showing registers from 27				Update	< Prev Next >
Local Register #	Register Name	Set	Register Data	Register Format	
00027	Data Value 1	<input type="checkbox"/>	77	Unsigned 16-bit	
00028	Data Value 2	<input type="checkbox"/>	0	Unsigned 16-bit	
00029	Data Value 3	<input type="checkbox"/>	0	Unsigned 16-bit	

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. ValuPoint 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.

Local Device	RTU Read Map	RTU Write Map		
Map # 1		Update < Prev Next >		
Read Holding Register as Unsigned 16-bit Size: 0				
From register # 11 at Unit # 2 With low register first if checked: <input type="checkbox"/>				
Apply bit mask if applicable: 0001 then apply scale: 0.000000 and offset: 0.000000				
Save in local register # 27 named Data Value 1 Repeat this process every 5.0 seconds.				
Apply this default value: 0.000000 after 0 read failure(s).				
<input type="checkbox"/> Enable this map only when index register 0 is set to a value of 0				
# RTU Read Maps Enabled: 5				Insert Delete

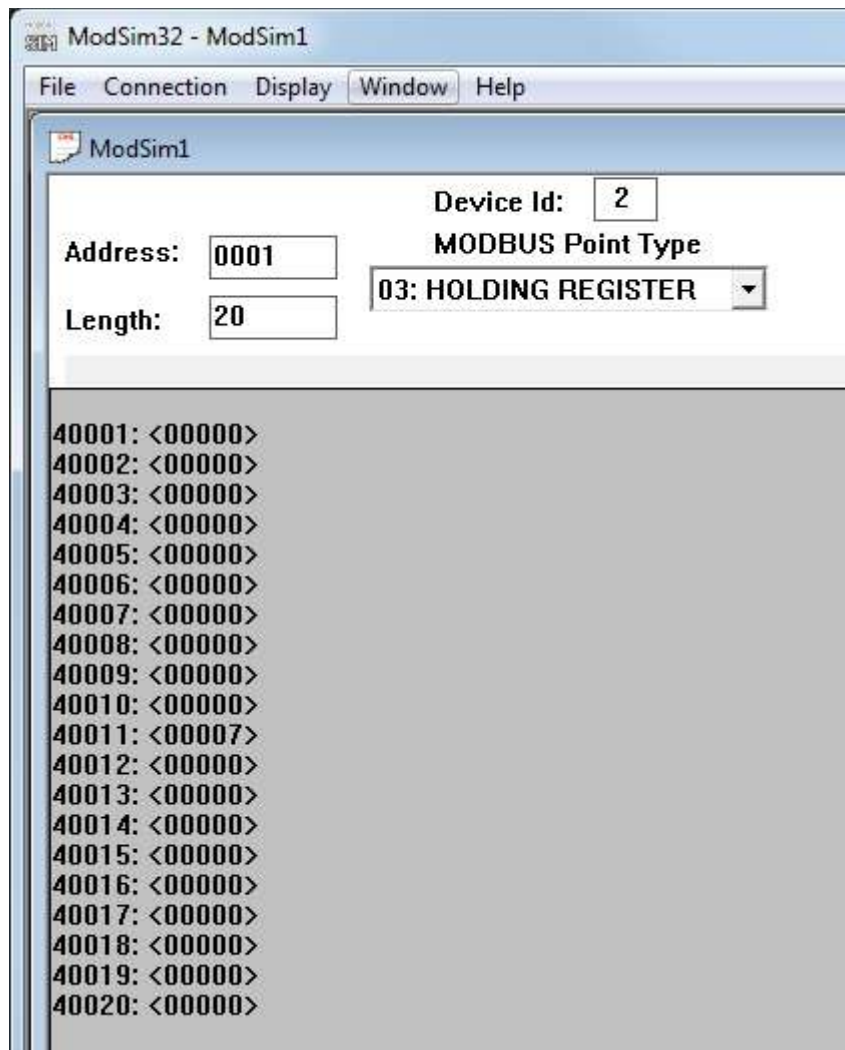
Reading bit 0 is illustrated above while reading bit 3 is illustrated below (bit 3 mask in binary would be 1000 which is hexadecimal 8 as entered in the configuration page). Refer to the Modbus Reference section in this user guide for a list of all possible mask values.

Local Device	RTU Read Map	RTU Write Map		
Map #	4		Update	< Prev Next >
Read	Holding Register	as	Unsigned 16-bit	Size: 0
From register #	11	at Unit #	1	With low register first if checked: <input type="checkbox"/>
Apply bit mask if applicable:	0008	then apply scale:	0.000000	and offset: 0.000000
Save in local register #	30	named	Data Value 4	Repeat this process every 5.0 seconds.
Apply this default value:	0.000000	after	0	read failure(s).
<input type="checkbox"/>	Enable this map only when index register	0	is set to a value of	0
# RTU Read Maps Enabled:	5		Insert	Delete

If the read maps referencing the same remote register are created in sequential contiguous order, the ValuPoint will optimize the RTU activity by reading the remote register once and then sharing the data with all of the read maps in the group. In the example illustrated here, 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.

Local Device	RTU Read Map	RTU Write Map				
		Showing	1	to 5 of 5		
			Update	< Prev Next >		
Map #	Remote Type	Remote Register Format	Remote Register #	Remote Unit #	Local Register #	Name
1	Holding Register	UINT-16	11	2	27	Data Value 1
2	Holding Register	UINT-16	11	2	28	Data Value 2
3	Holding Register	UINT-16	11	1	29	Data Value 3
4	Holding Register	UINT-16	11	1	30	Data Value 4
5	None	None	0	0	0	

To try out these read maps, we have set up ModSim with the remote register 11 containing a value of 7.



The holding register value of 7 translates into the following local register values when the bit mask option is used as illustrated.

Local Registers		Calculate	Copy	Report	
Showing registers from 27				Update	< Prev Next >
Local Register #	Register Name	Set	Register Data	Register Format	
00027	Data Value 1	<input type="checkbox"/>	1	Unsigned 16-bit	
00028	Data Value 2	<input type="checkbox"/>	1	Unsigned 16-bit	
00029	Data Value 3	<input type="checkbox"/>	1	Unsigned 16-bit	
00030	Data Value 4	<input type="checkbox"/>	0	Unsigned 16-bit	

### 5.3 Modbus RTU Master Write Maps

Getting the ValuPoint 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.

Local Device		RTU Read Map		RTU Write Map			
Showing 1 to 2 of 2							
Update < Prev Next >							
Map #	Local Register #	Remote Type	Remote Register Format	Remote Register #	Remote Unit #	Name	
1	28	Holding Register	UINT-16	12	1	Data Value 2	
2	0	None	None	0	0		

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 RTU slave. Select the register type, data format, and register number to be written to in that slave. Enter the slave address as Remote Unit. Click on the map number in the first column to access additional optional configuration parameters.

Local Device		RTU Read Map		RTU Write Map			
Map # 1							
Update < Prev Next >							
Read local register # 28 named Data Value 2							
Write remote register <input checked="" type="checkbox"/> when local register changes by > 0.000000 or <input type="checkbox"/> 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: 0000 and bit fill: 0000							
Write Holding Register as Unsigned 16-bit Size: 0 with blank padding if checked <input type="checkbox"/>							
To register # 12 at Unit # 1 With low register first if checked: <input type="checkbox"/>							
Repeat this process <input type="radio"/> at least <input checked="" type="radio"/> no more than every 0.0 seconds.							
<input type="checkbox"/> 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 slave periodically, or when it changes, or both. To send upon change (send on delta), check the first box and enter the amount by which the local 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, and remote unit (slave address). Data formats are the same as described above for Read Maps. Size is only specified for character strings. Use INT-16 or UINT-16 data format for coils - in this case format only affects local 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, ValuPoint lets you set that according to whatever the slave device requires. If the least significant data is found in the first (or lower numbered) register in your slave device, then check the box after "With low register first".

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

You have the option of making this Write Map conditional. If an index 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 slave if you know the slave is powered down.

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

## 5.4 Modbus RTU Master Data Displayed by Slave

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

The screenshot shows the 'RTU Registers' tab in a software interface. At the top, there are navigation tabs: 'RTU Setup', 'RTU Data', 'TCP Setup', and 'TCP Data'. Below these are sub-tabs: 'RTU Registers' (selected), 'Error Counts', 'Errors: Read Maps', and 'Errors: Write Maps'. The interface displays 'RTU Unit # (Slave Address): 1' and 'Showing 1 to 0 of 0'. There are 'Update', '< Prev', and 'Next >' buttons. A table with the following columns is shown: Dir., Reg. Type, Remote Reg. #, Register Name, Local Reg. #, Update, Register Data, and Time since Last update. The table contains one row with '---' in all cells. At the bottom, there is an 'RTU Unit #' field with '1' and '< Prev Unit' and 'Next Unit >' buttons.

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

The screenshot shows the 'RTU Registers' tab. The sub-tabs are the same as in the previous screenshot. The interface displays 'RTU Unit # (Slave Address): 1' and 'Showing 1 to 3 of 3'. There are 'Update', '< Prev', and 'Next >' buttons. A table with the following columns is shown: Dir., Reg. Type, Remote Reg. #, Register Name, Local Reg. #, Update, Register Data, and Time since Last update. The table contains three rows:
 

Dir.	Reg. Type	Remote Reg. #	Register Name	Local Reg. #	Update	Register Data	Time since Last update
From	Holding Reg	00011	<b>Data Value 3</b>	00029	<input type="checkbox"/>	1	540374.000
From	Holding Reg	00011	<b>Data Value 4</b>	00030	<input type="checkbox"/>	0	540374.000
To	Holding Reg	00012	<b>Data Value 2</b>	00028	<input type="checkbox"/>	1	540374.000

 At the bottom, there is an 'RTU Unit #' field with '1' and '< Prev Unit' and 'Next Unit >' buttons.

## 5.5 Modbus RTU Errors

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

**ValuPoint IOT**  
MODEL VP6-1460-SNMP  
IOT EDGE SERVER

**CONTROL SOLUTIONS MINNESOTA**

Local Data | Modbus | SNMP | System

RTU Setup | RTU Data | TCP Setup | TCP Data

RTU Registers | **Error Counts** | Errors: Read Maps | Errors: Write Maps

Showing devices from

Unit #	Reset	Total Messages	No Responses	CRC Errors	Exceptions
1	<input type="checkbox"/>	102007	0	0	0
2	<input type="checkbox"/>	683	457	0	226
3	<input type="checkbox"/>	0	0	0	0
4	<input type="checkbox"/>	0	0	0	0

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

RTU Registers | Error Counts | **Errors: Read Maps** | Errors: Write Maps

Map #	Register Name	Error Description	Exception Code
1	Data Value 1	Exception code returned by device	Illegal data address

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

RTU Registers | Error Counts | **Errors: Read Maps** | Errors: Write Maps

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

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

RTU Registers | **Error Counts** | Errors: Read Maps | Errors: Write Maps

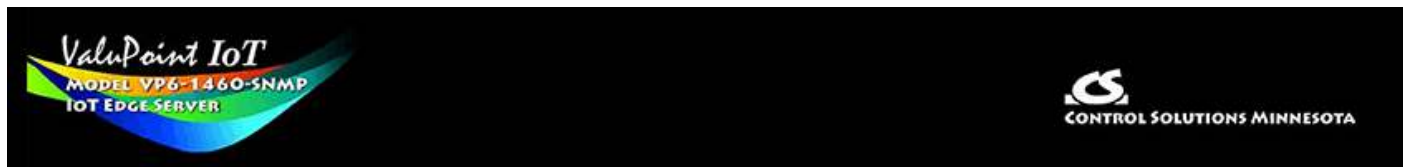
Showing devices from

Unit #	Reset	Total Messages	No Responses	CRC Errors	Exceptions
1	<input checked="" type="checkbox"/>	102007	0	0	0
2	<input checked="" type="checkbox"/>	1805	457	0	1328
3	<input type="checkbox"/>	0	0	0	0

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

incrementing again.

RTU Registers		Error Counts		Errors: Read Maps		Errors: Write Maps	
Unit #	Reset	Total Messages	No Responses	CRC Errors	Exceptions		
1	<input type="checkbox"/>	0	0	0	0		
2	<input type="checkbox"/>	3	0	0	0		
3	<input type="checkbox"/>	0	0	0	0		



## 6. Configuring ValuPoint as a Modbus TCP Client

The ValuPoint 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 ValuPoint 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 the SNMP MIB.

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.

ValuPoint IoT  
MODEL VP6-1460-SNMP  
IOT EDGE SERVER

CONTROL SOLUTIONS MINNESOTA

Local Data Modbus SNMP System

RTU Setup RTU Data TCP Setup TCP Data

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 Port: 502

Domain Name

Unit (optional) 1  Use FC 5/6 instead of 15/16

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.

If "Unit" on the devices page is left at zero, then unit #1 is used by default. Otherwise the number entered here becomes the default. However, each individual read and write map can have different unit numbers, as would be required for accessing multiple RTU devices via a gateway at one IP address. When a non-zero unit number is placed in the read or write map, it will override the default on the Devices page.

If coil and holding register writes must be done with write single function codes 5 and 6 rather than write multiple function codes 15 and 16, then check the "Use FC 5/6" box for this device. The ValuPoint will default to write multiple.

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 interface has a dark green header with tabs for "Devices", "Client Read Map", and "Client Write Map". The "Devices" tab is active. Below the header, there are navigation buttons: "Update", "< Prev", and "Next >". The main configuration area is a dark teal background with white text and input fields. The "Local Name" is "Device 2". Under "Use", "Static IPv6" is selected. The "IP Address" is "fe80::c28:cf9b:b7f8:639e". The "Port" is "502". The "Domain Name" field is empty. The "Unit (optional)" is "1", and the checkbox "Use FC 5/6 instead of 15/16" is unchecked. The "Default Poll Period" is "5.0" seconds. The "Connection Status" is "0", and there is a "Clear" button next to it.

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 interface is similar to the previous one. The "Local Name" is "Device 3". Under "Use", "Domain Lookup" is selected. The "IP Address" is "50.97.153.22". The "Port" is "502". The "Domain Name" is "myModbusDevice.net". The "Unit (optional)" is "1", and the checkbox "Use FC 5/6 instead of 15/16" is unchecked. The "Default Poll Period" is "5.0" seconds. The "Connection Status" is "0", and there is a "Clear" button next to it.

## 6.2 Modbus TCP Client Read Maps

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

ValuPoint IOT  
MODEL VP6-1460-SNMP  
IOT EDGE SERVER

CONTROL SOLUTIONS MINNESOTA

Local Data Modbus SNMP System

RTU Setup RTU Data TCP Setup TCP Data

Devices Client Read Map Client Write Map

Showing 1 to 1 of 1 Update < Prev Next >

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.

RTU Setup RTU Data TCP Setup TCP Data

Devices Client Read Map Client Write Map

Showing 1 to 1 of 1 Update < Prev Next >

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

Quick  
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 see more detail and insert/delete maps.

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



Devices		Client Read Map		Client Write 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. Click on map number to see more details.

Map number simply tells you where the data is stored. To enter a specific map, enter the desired number in the map number field.

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

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

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.)

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	27	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 ValuPoint. 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	27	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	Holding Register	as	Unsigned 16-bit	Size:	0		
From register #	1	at	ModSim	unit #	1	With low register first if checked: <input type="checkbox"/>	
Apply bit mask if applicable:	0000	then apply scale:	1.800000	and offset:	32.000000		
Save in local register #	27	named	Data Value 1	Repeat this process every	5.0	seconds.	
Apply this default value:	0.000000	after	0	read failure(s).			
<input type="checkbox"/>	Enable this map only when index register 0 is set to a value of 0						
# Client Read Maps Enabled:		2				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, ValuPoint 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 Registers		Calculate	Copy	Report	
Showing registers from				27	Update < Prev Next >
Local Register #	Register Name	Set	Register Data	Register Format	
00027	Data Value 1	<input type="checkbox"/>	77	Unsigned 16-bit	
00028	Data Value 2	<input type="checkbox"/>	0	Unsigned 16-bit	
00029	Data Value 3	<input type="checkbox"/>	0	Unsigned 16-bit	

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. ValuPoint 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 ValuPoint 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 ValuPoint 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	28	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 # 28 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: 0000 and bit fill: 0000

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

To register # 2 at ModSim unit # 1 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, ValuPoint 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.

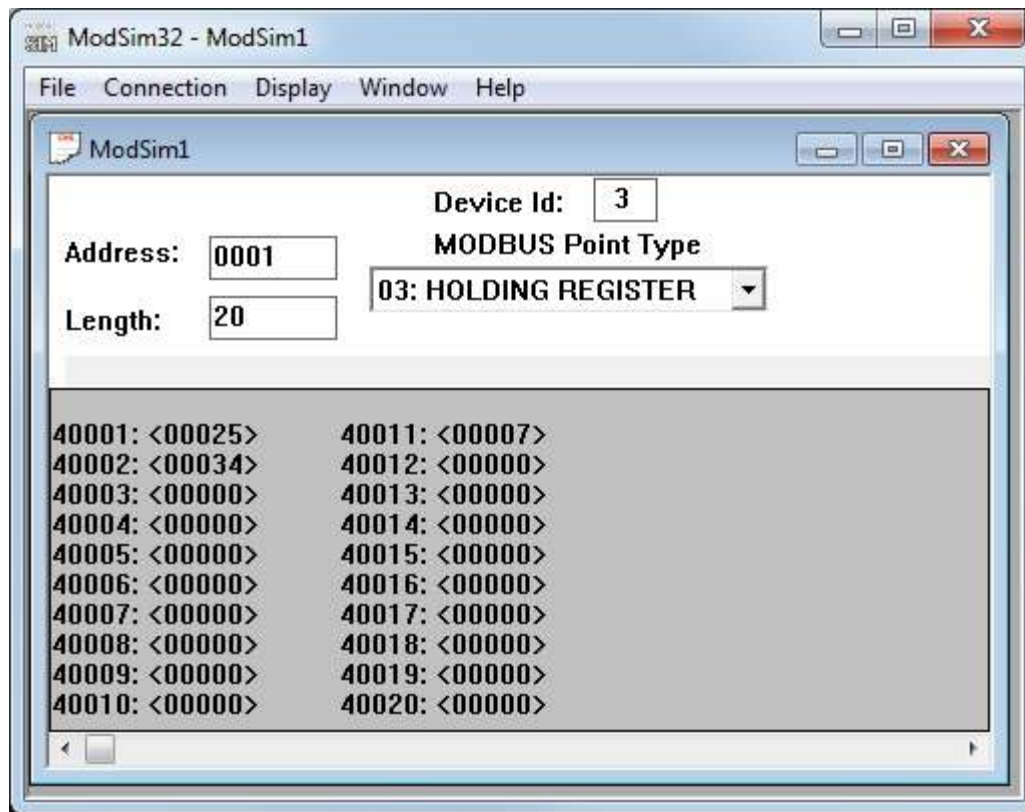
You have the option of making this Write Map conditional. If an index register number is provided and the Enable box is checked, then this write map will only be executed when the index register (local register) contains the value given. This allows multiple write maps to supply data to the same remote 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	Report	
Showing registers from				27	Update < Prev Next >
Local Register #	Register Name	Set	Register Data	Register Format	
00027	Data Value 1	<input type="checkbox"/>	77	Unsigned 16-bit	
00028	Data Value 2	<input type="checkbox"/>	34	Unsigned 16-bit	
00029	Data Value 3	<input type="checkbox"/>	0	Unsigned 16-bit	

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.

RTU Setup		RTU Data		TCP Setup		TCP Data	
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	00027	<input type="checkbox"/>	77	2.000
To	Holding Reg	00002	Data Value 2	00028	<input type="checkbox"/>	34	1.000
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	
RTU Setup		RTU Data		TCP Setup		TCP Data	
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 ValuPoint 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	
<input type="button" value="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	
<input type="button" value="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 # <input type="text" value="1"/>	<input type="button" value="Update"/> <input type="button" value=" &lt; Prev"/> <input type="button" value=" Next &gt;"/>			
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

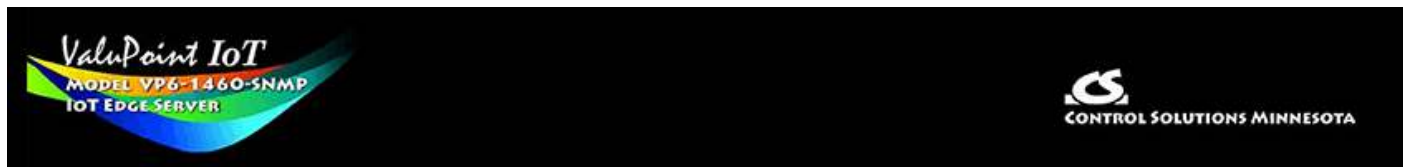
115 = Network interface not configured

116 = Connection timed out

118 = Host is unreachable

125 = Address not available

205 = DNS error



## 7. Configuring ValuPoint as a Modbus RTU Slave

### 7.1 Modbus RTU Device Configuration

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

The screenshot shows a web-based configuration interface for a Modbus RTU device. The interface has a dark green header with tabs for 'RTU Setup', 'RTU Data', 'TCP Setup', and 'TCP Data'. Below this is a sub-header with 'Local Device', 'RTU Read Map', and 'RTU Write Map'. An 'Update' button is in the top right. The main configuration area is divided into two sections: 'I am the Master' (disabled) and 'I am a Slave' (selected). Under 'I am the Master', there are fields for 'Default Poll Rate' (0.000) and 'Timeout' (0.000), both in seconds. Under 'I am a Slave', there is a field for 'My Address or Unit #' (1). At the bottom, there is a checkbox for 'Use FC 5/6 instead of 15/16 for unit numbers (slave addresses) starting at' followed by a field containing '0'. The 'Baud Rate' is set to 19200 and 'Parity' is set to 'None, 1 Stop Bit'.

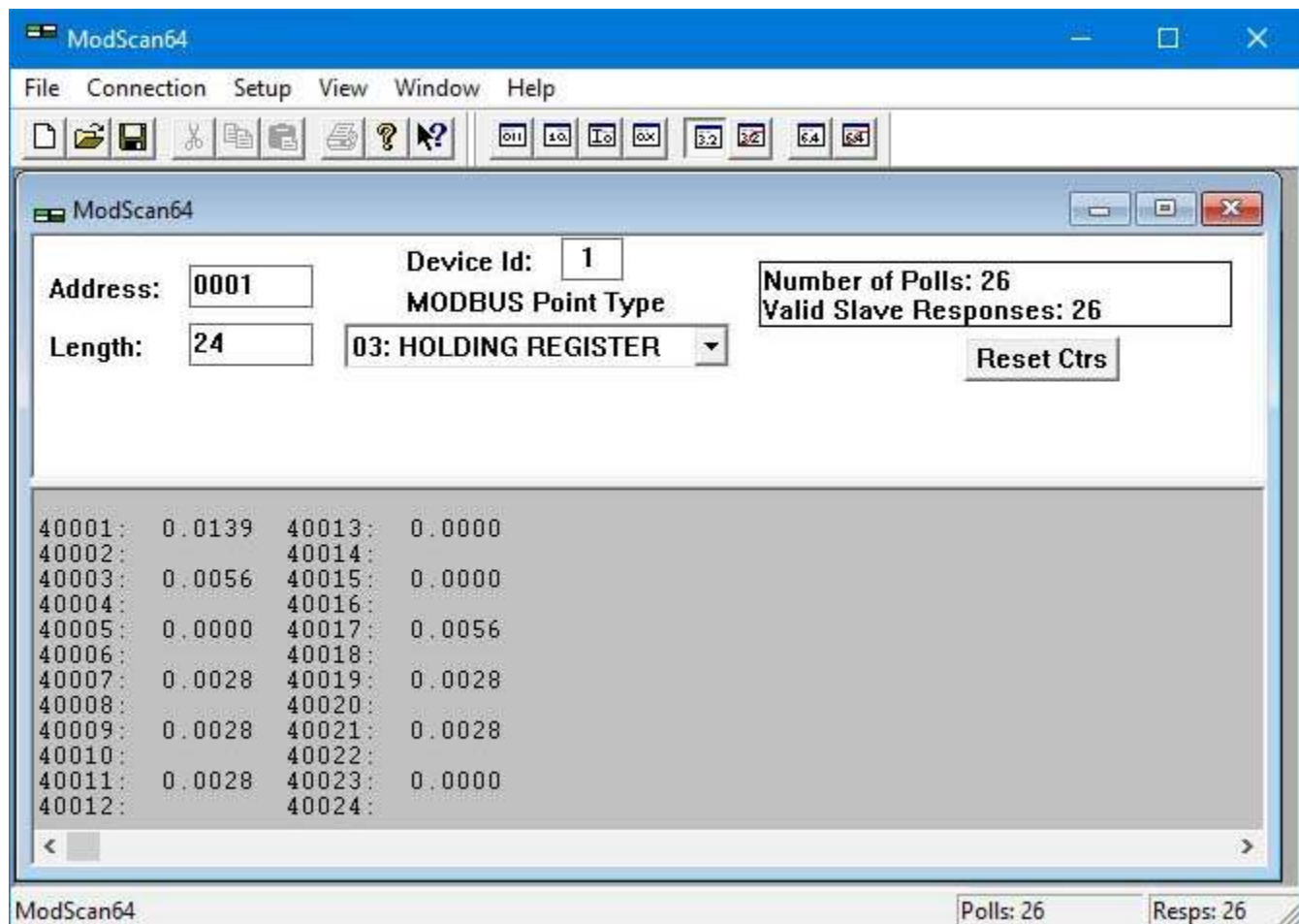
### 7.2 Modbus RTU Slave Register Map

The local registers in the ValuPoint 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.

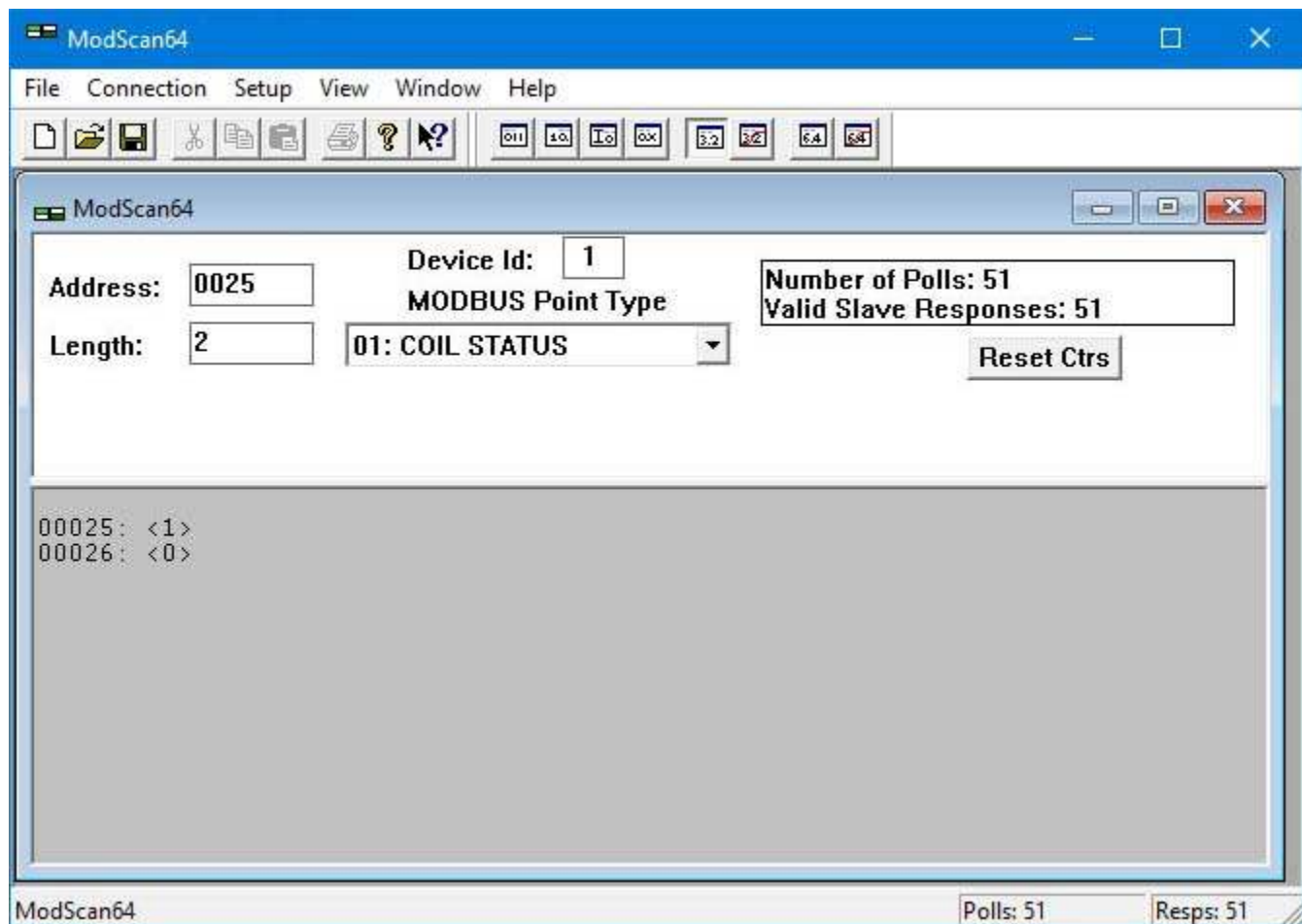
The register numbers that the remote Modbus RTU master should read or write are simply those shown in the first column on the Local Registers page.

Local Data		Modbus		SNMP		System	
Data							
Local Registers		Calculate		Copy		Report	
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		
00001	A/UI #1	<input type="checkbox"/>	0.027847		Single Float		
00003	A/UI #2	<input type="checkbox"/>	0.000000		Single Float		
00005	A/UI #3	<input type="checkbox"/>	0.002785		Single Float		
00007	A/UI #4	<input type="checkbox"/>	0.002785		Single Float		
00009	A/UI #5	<input type="checkbox"/>	0.000000		Single Float		
00011	A/UI #6	<input type="checkbox"/>	0.005569		Single Float		
00013	A/UI #7	<input type="checkbox"/>	0.002785		Single Float		
00015	A/UI #8	<input type="checkbox"/>	0.000000		Single Float		
00017	A/UI #9	<input type="checkbox"/>	0.002785		Single Float		
00019	A/UI #10	<input type="checkbox"/>	0.000000		Single Float		
00021	A/UI #11	<input type="checkbox"/>	0.000000		Single Float		
00023	A/UI #12	<input type="checkbox"/>	0.005569		Single Float		
00025	DO #1	<input type="checkbox"/>	1		Unsigned 16-bit		
00026	DO #2	<input type="checkbox"/>	0		Unsigned 16-bit		

The first 24 registers are defined as floating point register pairs and assigned to the 12 physical input points. Viewing the floating point data using ModScan is illustrated below. The default data format is "Most significant register first" when selecting floating point in ModScan.

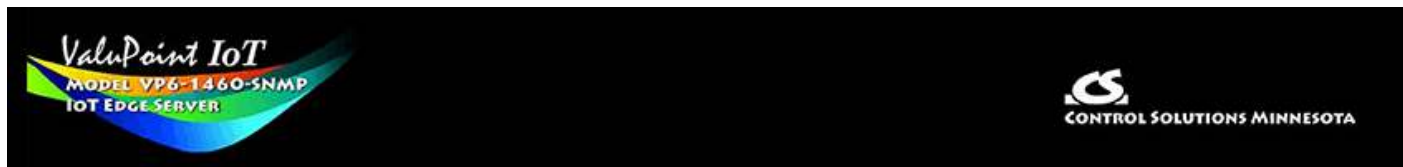


The two registers assigned to the relay outputs, registers 25 and 26, may be read and written as coils.



### 7.3 Modbus RTU Slave Diagnostic

The ValuPoint brand new out of the box will be configured as Modbus RTU slave, 9600 baud, slave address 1. Although no registers other than I/O points 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".



## 8. Configuring ValuPoint as a Modbus TCP Server

### 8.1 Modbus TCP Device Configuration

There is really little to do to configure the ValuPoint to be a Modbus TCP server. The ValuPoint 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.



### 8.2 Modbus TCP Server Register Map

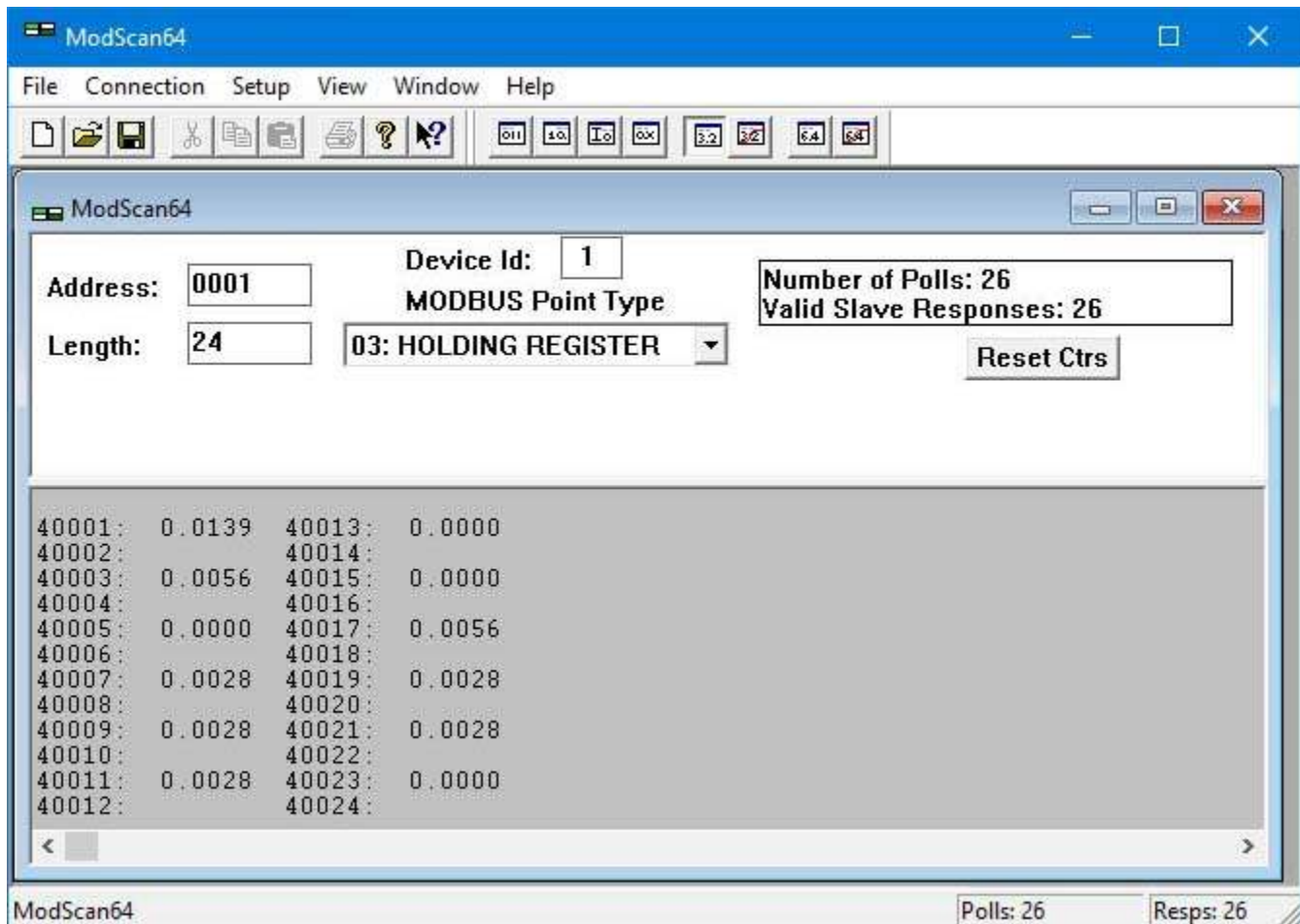
The local registers in the ValuPoint will be most often accessed as holding registers, but can also be accessed as input registers (for reading but input registers cannot be written to). If the local register is defined as 16-bit signed or unsigned, then it can also be accessed as a coil or discrete 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.

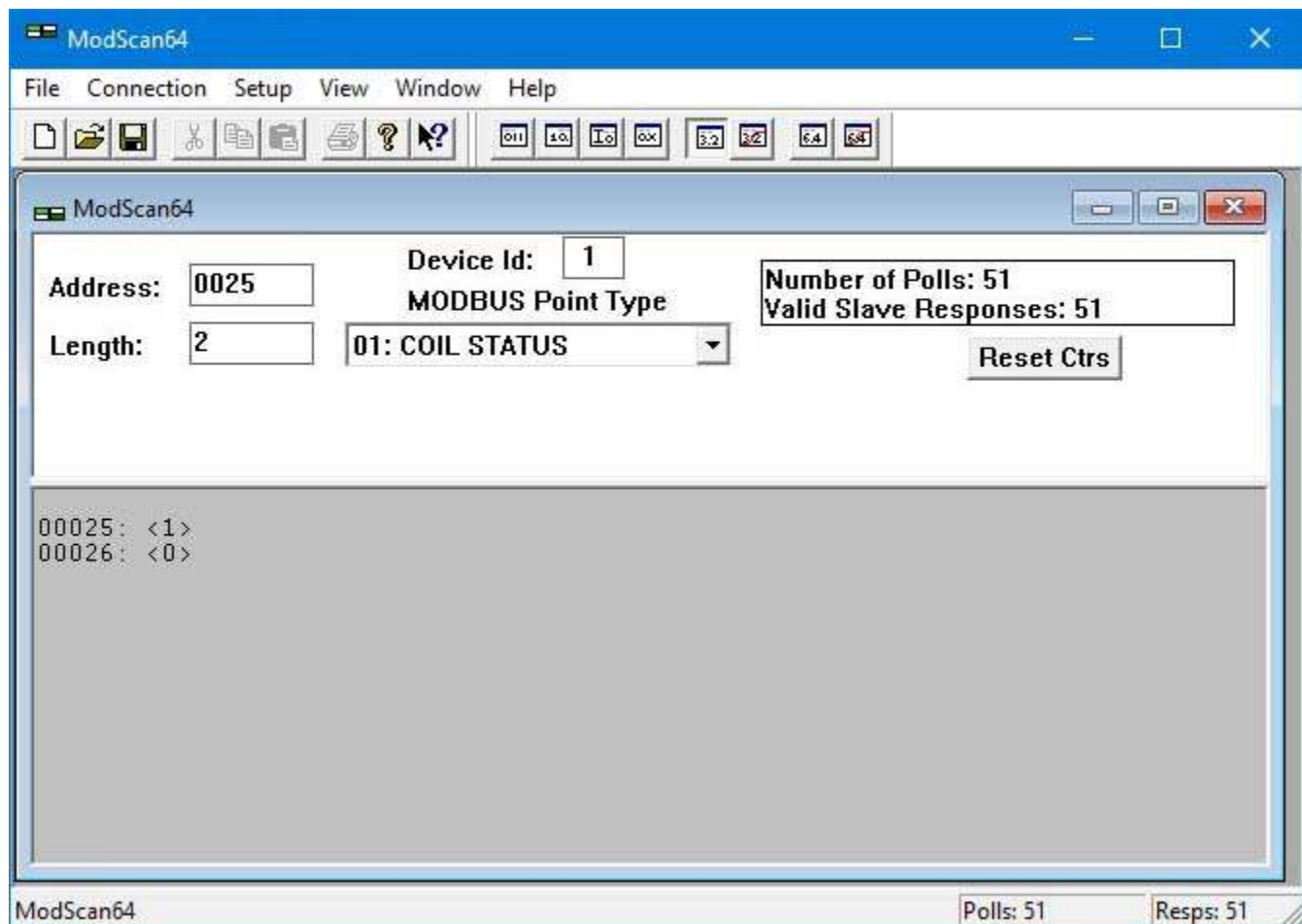
Local Data		Modbus		SNMP		System	
Data							
Local Registers		Calculate		Copy		Report	
Showing registers from <input type="text" value="1"/> <input type="button" value="Update"/> <input type="button" value=" &lt; Prev"/> <input type="button" value=" Next &gt;"/>							
Local Register #	Register Name	Set	Register Data			Register Format	
00001	A/UI #1	<input type="checkbox"/>	0.027847			Single Float	
00003	A/UI #2	<input type="checkbox"/>	0.000000			Single Float	
00005	A/UI #3	<input type="checkbox"/>	0.002785			Single Float	
00007	A/UI #4	<input type="checkbox"/>	0.002785			Single Float	
00009	A/UI #5	<input type="checkbox"/>	0.000000			Single Float	
00011	A/UI #6	<input type="checkbox"/>	0.005569			Single Float	
00013	A/UI #7	<input type="checkbox"/>	0.002785			Single Float	
00015	A/UI #8	<input type="checkbox"/>	0.000000			Single Float	
00017	A/UI #9	<input type="checkbox"/>	0.002785			Single Float	
00019	A/UI #10	<input type="checkbox"/>	0.000000			Single Float	
00021	A/UI #11	<input type="checkbox"/>	0.000000			Single Float	
00023	A/UI #12	<input type="checkbox"/>	0.005569			Single Float	
00025	DO #1	<input type="checkbox"/>	1			Unsigned 16-bit	
00026	DO #2	<input type="checkbox"/>	0			Unsigned 16-bit	

The first 24 registers are defined as floating point register pairs and assigned to the 12 physical input points. Viewing the floating point data using ModScan is illustrated below. The default data format is "Most significant register first" when selecting floating point in ModScan.



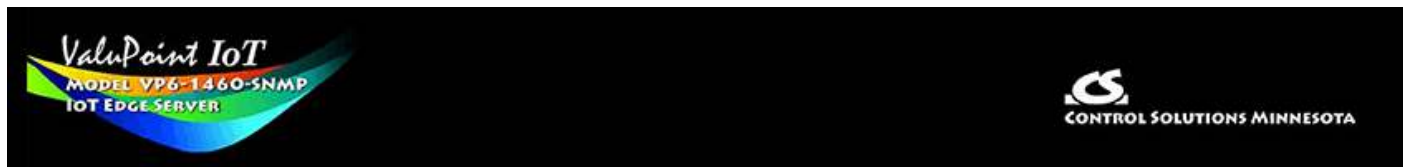
The two registers assigned to the relay outputs, registers 25 and 26, may be read and written as coils.





### 8.3 Modbus TCP Server Diagnostic

The ValuPoint brand new out of the box will have no registers configured other than I/O points. 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".



## 9. Configuring ValuPoint as an SNMP Server

### 9.1 Creating Local SNMP MIB

The ValuPoint VP6-1460-WNMP 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. The ValuPoint VP6-1460-SNMP uses the same MIB as the Babel Buster BB3-6101-V3 gateway. 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.

Showing 1 to 1 of 1

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 local register 1

Reload SNMP [ ] Map # 1 Remove Insert Before

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 "Showing" box, then click Update.

This page enables SNMP Get/Set to registers indicated on the page. 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. If real numbers via SNMP is to scale them and send them as ASCII strings (supported by the Char String MIB).

Local Registers    Calculate    Copy

Showing registers from 1 to 1 of 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 <input type="checkbox"/> 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 <input type="checkbox"/> 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	Auto-Fill	<input type="checkbox"/> Confirm	Clear
Reload SNMP					Map #	1	Remove	Insert Before	

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									
Update < Prev Next >									
Map #	Local SNMP OID	Local Register #	Scale Factor	Local Value	Local Name				
1	1.3.6.1.4.1.3815.1.6.1.1.1.1.2.1	1	x1	0	Register 1				
2	1.3.6.1.4.1.3815.1.6.1.1.1.1.2.2	2	x1	0	Register 2				
3	1.3.6.1.4.1.3815.1.6.1.1.1.1.2.3	3	x1	0	Register 3				
4	1.3.6.1.4.1.3815.1.6.1.1.1.1.2.4	4	x1	0	Register 4				
5	1.3.6.1.4.1.3815.1.6.1.1.1.1.2.5	5	x1	0	Register 5				
6	1.3.6.1.4.1.3815.1.6.1.1.1.1.2.6	6	x1	0	Register 6				
7	1.3.6.1.4.1.3815.1.6.1.1.1.1.2.7	7	x1	0	Register 7				
8	1.3.6.1.4.1.3815.1.6.1.1.1.1.2.8	8	x1	0	Register 8				
9	1.3.6.1.4.1.3815.1.6.1.1.1.1.2.9	9	x1	0	Register 9				
10	1.3.6.1.4.1.3815.1.6.1.1.1.1.2.10	10	x1	0	Register 10				
11	1.3.6.1.4.1.3815.1.6.1.1.1.1.2.11	0	x1	---	---				

Auto-fill 10 variables starting at index 1 with local register 1 Auto-Fill  Confirm Clear

Reload SNMP  Map # 1 Remove Insert Before

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									
Update < Prev Next >									
Map #	Local SNMP OID	Local Register #	Scale Factor	Local Value	Local Name				
1	1.3.6.1.4.1.3815.1.6.1.1.1.1.2.1	1	x1	0.000000	Data Value 1				
2	1.3.6.1.4.1.3815.1.6.1.1.1.1.2.2	3	x10	0.000000	Data Value 2				
3	1.3.6.1.4.1.3815.1.6.1.1.1.1.2.3	5	x10	0.000000	Data Value 3				
4	1.3.6.1.4.1.3815.1.6.1.1.1.1.2.4	7	x1	0.000000	Data Value 4				
5	1.3.6.1.4.1.3815.1.6.1.1.1.1.2.5	9	x100	0.000000	Data Value 5				
6	1.3.6.1.4.1.3815.1.6.1.1.1.1.2.6	11	x1	0.000000	Data Value 6				
7	1.3.6.1.4.1.3815.1.6.1.1.1.1.2.7	13	x1	0.000000	Data Value 7				
8	1.3.6.1.4.1.3815.1.6.1.1.1.1.2.8	15	x0.1	0.000000	Data Value 8				
9	1.3.6.1.4.1.3815.1.6.1.1.1.1.2.9	17	x0.01	0.000000	Data Value 9				
10	1.3.6.1.4.1.3815.1.6.1.1.1.1.2.10	19	x1	0.000000	Data Value 10				
11	1.3.6.1.4.1.3815.1.6.1.1.1.1.2.11	21	x1	1,234	Character String 1				
12	1.3.6.1.4.1.3815.1.6.1.1.1.1.2.12	41	x1	5,678	Character String 2				
13	1.3.6.1.4.1.3815.1.6.1.1.1.1.2.13	0	x1	---	---				

Auto-fill 12 variables starting at index 1 with local register 1  Auto-Fill  Confirm

12 Int32 vars loaded Map # 1

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 ValuPoint. 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 ValuPoint is restarted.

## 9.2 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 ValuPoint refer to RFC 6340 encoding.

Specifically, the data types found in the ValuPoint 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)

### 9.3 SNMPv3 Users, Authentication, Privacy

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

The screenshot shows the ValuPoint IoT IOT EDGE SERVER web interface. The navigation menu includes 'Local Data', 'Modbus', 'SNMP', and 'System'. Under 'SNMP', there are sub-menus for 'Local MIB', 'Security', and 'Trap Sender'. The 'Security' sub-menu is selected, and the 'User' tab is active. The main content area shows a table of users with columns for User Name, Auth Type, Authentication Passphrase, Priv Type, Privacy Passphrase, and Delete. An 'Update' button is located at the top right of the table.

User Name	Auth Type	Authentication Passphrase	Priv Type	Privacy Passphrase	Delete
jimh	MD5	jimsauth	DES	jimspriv	<input 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 ValuPoint 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.

### 9.4 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.

User Agent ID

System Name: Babel Buster BB3-6101

System Location: St. Paul, Minnesota

System Contact: www.csimn.com

SNMPv3

Engine ID: 80000EE702FE8000000000000002409DFFFE45464E

Engine Boots: 7  Set New

Seconds Since Start: 36713

SNMPv1/v2c  Allow SNMPv1/v2c

v1/v2c Community: 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.

iReasoning MIB Browser

File Edit Operations Tools Bookmarks Help

Address: 192.168.1.125 Advanced... OID: .1.3.6.1.2.1.1.7.0 Operations: Get Next Go

SNMP MIBs

MIB Tree

- iso.org.dod.internet

Result Table

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...

## 9.5 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	Agent ID			
System Name	Babel Buster 883-6101			Update
System Location	St. Paul, Minnesota			
System Contact	www.csinn.com			
<b>SNMPv3</b>				
Engine ID	80000EE702FE8000000000000002409DFFFE45464E			
Engine Boots	7	<input type="checkbox"/>	Set New	
Seconds Since Start	36713			
<b>SNMPv1/v2c</b>				
	<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.

## 9.6 SNMPv2 Community

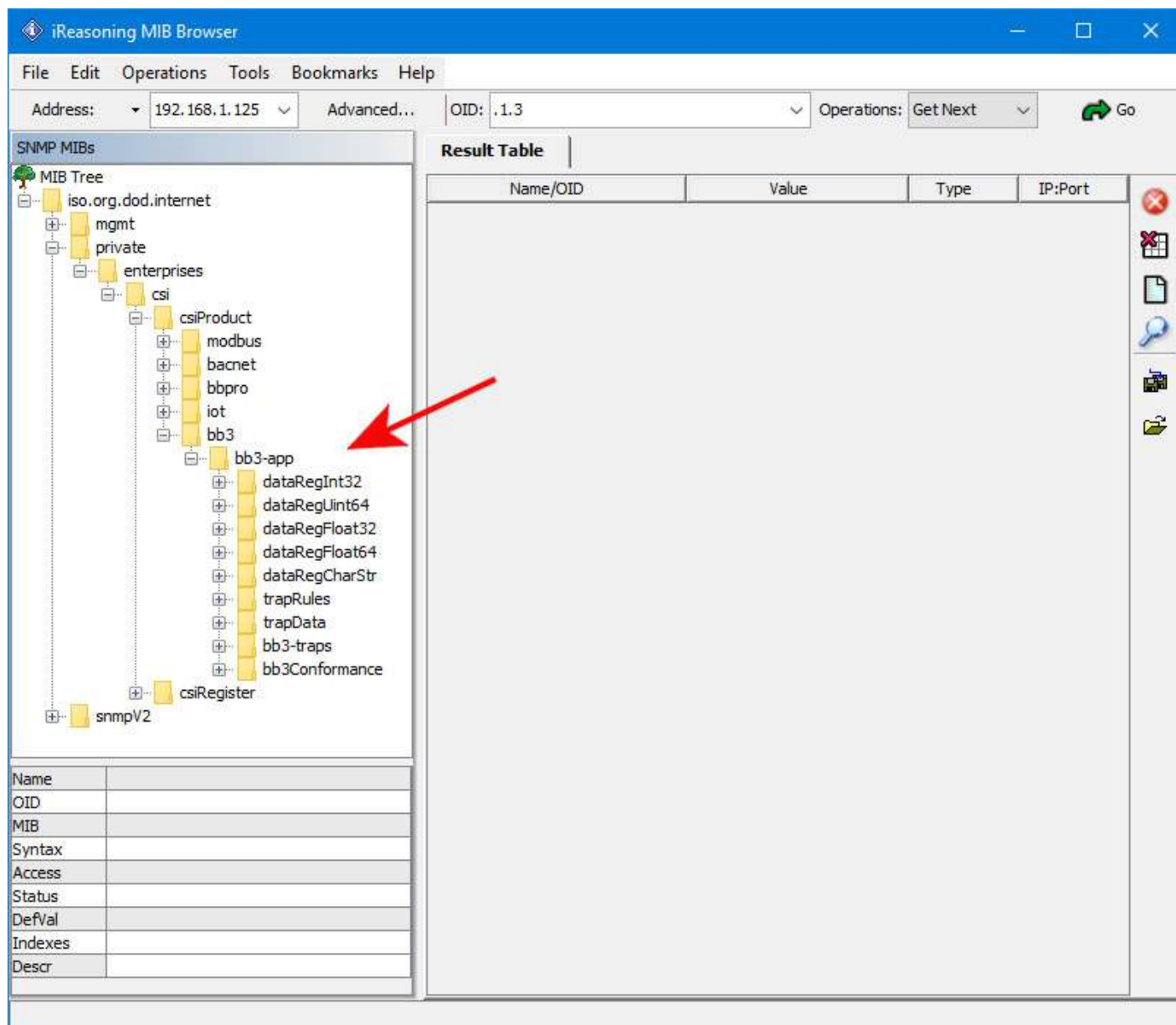
Check Allow if SNMP v1/v2c should be permitted to access the MIB in this device. The v1/v2c community must be used by the remote SNMP v1/v2c client to Get or Set this device. The name "public" is no longer accepted (unless you explicitly enter it here - not recommended). Changes to the allow/disallow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

User	Agent ID			
System Name	Babel Buster 883-6101			Update
System Location	St. Paul, Minnesota			
System Contact	www.csimn.com			
SNMPv3				
Engine ID	80000EE702FE8000000000000000002409DFFFE45464E			
Engine Boots	7	<input type="checkbox"/>	Set New	
Seconds Since Start	36713			
SNMPv1/v2c				
	<input type="checkbox"/>	Allow SNMPv1/v2c		
v1/v2c Community	private			

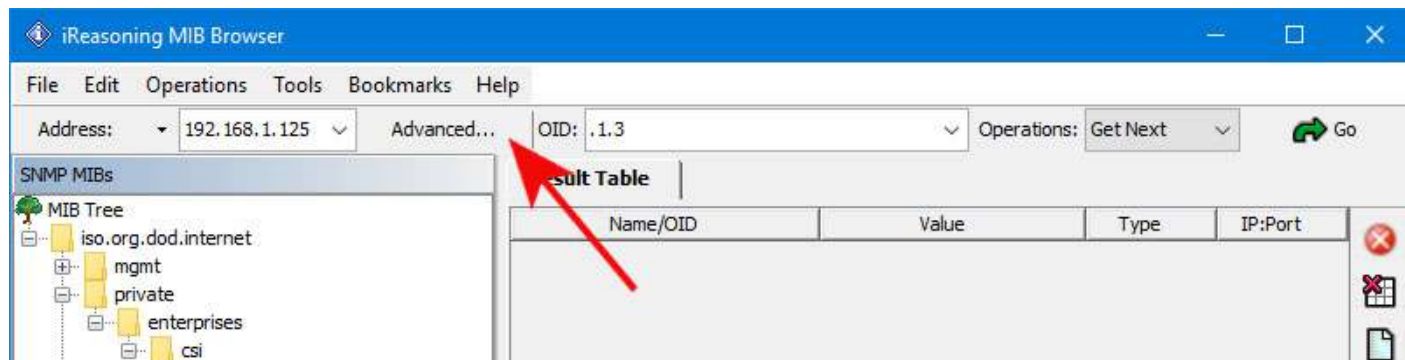
## 9.7 Testing the SNMP Agent

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 ValuPoint. 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 VP6-1460-WNMP 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 ValuPoint VP6-1460-WNMP 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 ValuPoint.

Advanced Properties of SNMP Agent

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:

Ok Cancel

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

The screenshot shows the iReasoning MIB Browser interface. The left pane displays the SNMP MIBs tree, with the following structure expanded:

- iso.org.dod.internet
  - private
    - enterprises
      - csi
        - csiProduct
          - modbus
          - bacnet
          - bbpro
          - iot
          - bb3
            - bb3-app
              - dataRegInt32
                - dataRegInt32Table
                  - dataRegInt32Entry
                    - dataRegInt32Index
                    - dataRegInt32Register** (highlighted with a red arrow)
                    - dataRegInt32Name
                  - dataRegUInt64
                  - dataRegFloat32
                  - dataRegFloat64
                  - dataRegCharStr
                  - trapRules

The right pane shows the Result Table with the following columns: Name/OID, Value, Type, and IP:Port. The table is currently empty. Below the tree view, a detailed view of the selected MIB entry is shown:

|         |                               |
|---------|-------------------------------|
| Name    | dataRegInt32Register          |
| OID     | .1.3.6.1.4.1.3815.1.6.1.1.1.2 |
| MIB     | CSI-BB3-MIB                   |
| Syntax  | INTEGER.32                    |
| Access  | read-write                    |
| Status  | current                       |
| DefVal  |                               |
| Indexes | dataRegInt32Index             |
| Descr   | Register's data value.        |

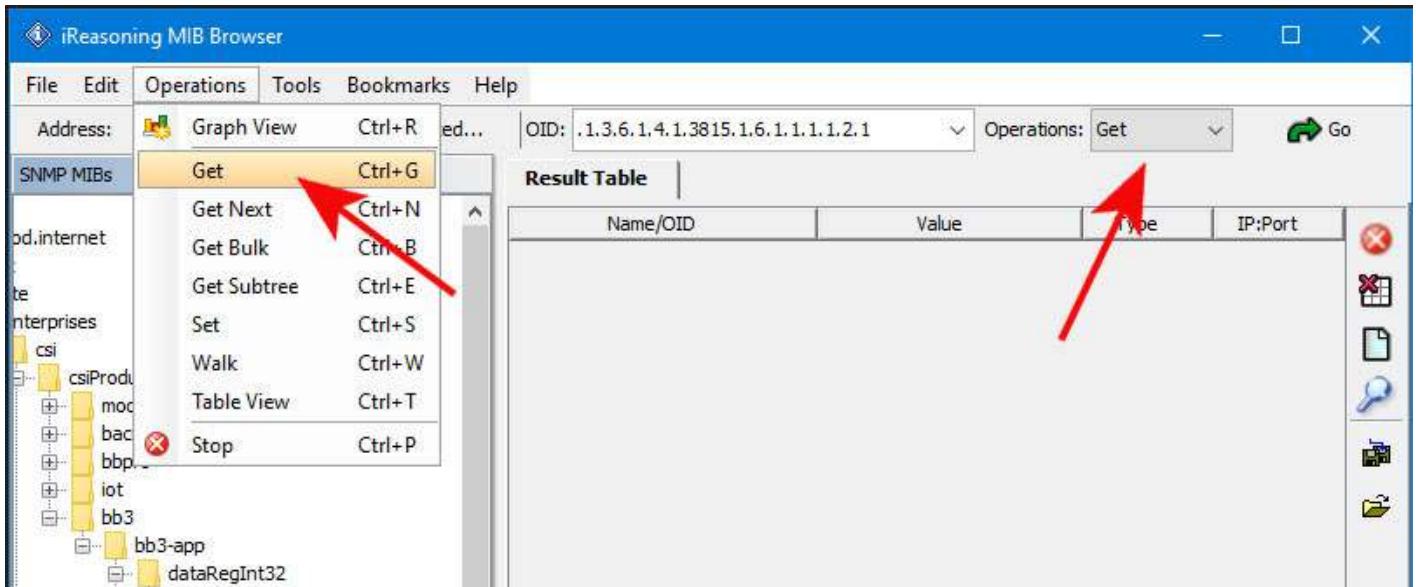
The status bar at the bottom shows the full path: .iso.org.dod.internet.private.enterprises.csi.csiProduct.bb3.bb3-app.dataRegInt32.dataRegInt32Table.dataRegInt32Entry.dataRegInt32Register

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.

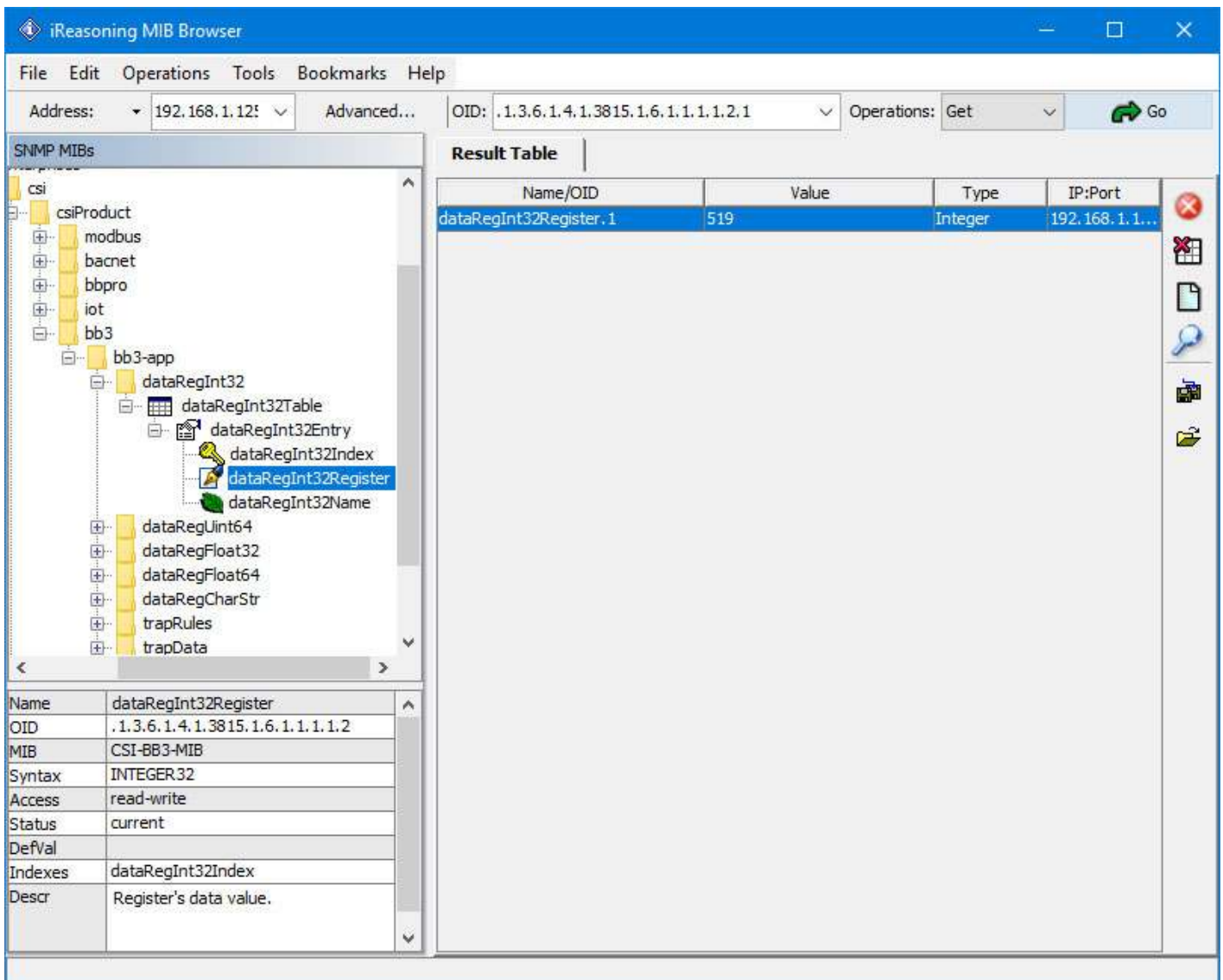
The screenshot shows the iReasoning MIB Browser interface after the OID has been updated. The OID field now contains: .1.3.6.1.4.1.3815.1.6.1.1.1.2.1. The Operations menu is set to 'Get'. A red arrow points to the 'Go' button.

To cause the MIB browser to Get a value from the ValuPoint, 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 left pane shows the MIB tree with 'dataRegInt32Register' selected. The right pane shows a 'Result Table' with two rows:

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

Below the tree, the details for 'dataRegInt32Register' are shown:

|         |                               |
|---------|-------------------------------|
| Name    | dataRegInt32Register          |
| OID     | .1.3.6.1.4.1.3815.1.6.1.1.1.2 |
| MIB     | CSI-BB3-MIB                   |
| Syntax  | INTEGER.32                    |
| Access  | read-write                    |
| Status  | current                       |
| DefVal  |                               |
| Indexes | dataRegInt32Index             |
| Descr   | Register's data value.        |

The status bar at the bottom shows 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 shows the iReasoning MIB Browser interface. The top menu bar includes File, Edit, Operations, Tools, Bookmarks, and Help. The Address field is set to 192.168.1.12! and the OID is .1.3.6.1.4.1.3815.1.6.1.3.1.1.2.1. The Operations dropdown is set to Get. The left pane shows a tree view of SNMP MIBs, with the following structure expanded:

- erprises
  - csi
    - csiProduct
      - modbus
      - bacnet
      - bbpro
      - iot
      - bb3
        - bb3-app
          - dataRegInt32
          - dataRegUInt64
          - dataRegFloat32
            - dataRegFloat32Table
              - dataRegFloat32Entry
                - dataRegFloat32Index
                - dataRegFloat32Register (selected)
                - dataRegFloat32Name
              - dataRegFloat64
              - dataRegCharStr
              - trapRules

The right pane displays a Result Table with the following data:

| Name/OID                 | Value         | Type        | IP:Port        |
|--------------------------|---------------|-------------|----------------|
| dataRegFloat32Register.1 | 0x42 A4 83 12 | OctetString | 192.168.1.1... |

Below the tree view, a detailed view of the selected MIB object is shown:

|         |                                    |
|---------|------------------------------------|
| 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: .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

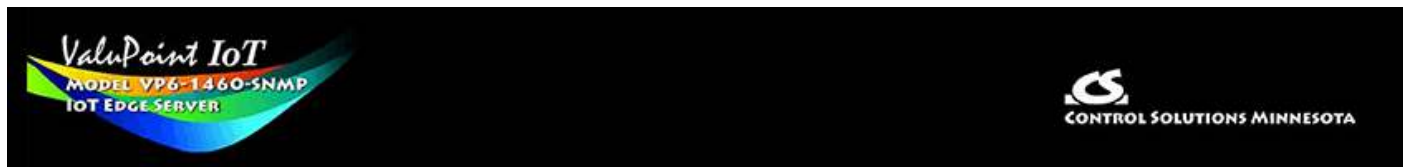
- enterprises
  - csi
    - csiProduct
      - modbus
      - bacnet
      - bbpro
      - iot
      - bb3
        - bb3-app
          - dataRegInt32
          - dataRegUInt64
          - dataRegFloat32
          - dataRegFloat64
          - dataRegCharStr
            - dataRegCharStrTable
            - 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



## 10. Configuring SNMP Trap Sender

### 10.1 SNMP Trap Destinations

The first step in sending traps is to tell ValuPoint 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.

## 10.2 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 ValuPoint 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.

The screenshot displays the configuration page for a trap trigger. At the top, there are tabs for 'Devices', 'Trap Trigger', and 'Trap Summary'. The 'Trap Trigger' tab is active, showing 'Trap # 1' and 'Trigger rule presently tests FALSE'. There are 'Update', '< Prev', and 'Next >' buttons. The main configuration area includes:

- 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**

At the bottom, it shows '# Trap Trigger Rules Enabled: 2' and 'Insert' and 'Delete' buttons.

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.

### 10.3 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.

| 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       | ---         | ---          |

Showing 1 to 2 of 2

Update < Prev Next >

1 Trap rule vars loaded 1 Trap data vars loaded Reload SNMP

### 10.4 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 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 ValuPoint.

The screenshot shows the iReasoning MIB Browser interface. The main window is titled "Trap Receiver x" and displays a table of trap data. Below the table, detailed information is shown, including Source, Timestamp, SNMP Version, Trap OID, and Variable Bindings.

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

**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:**

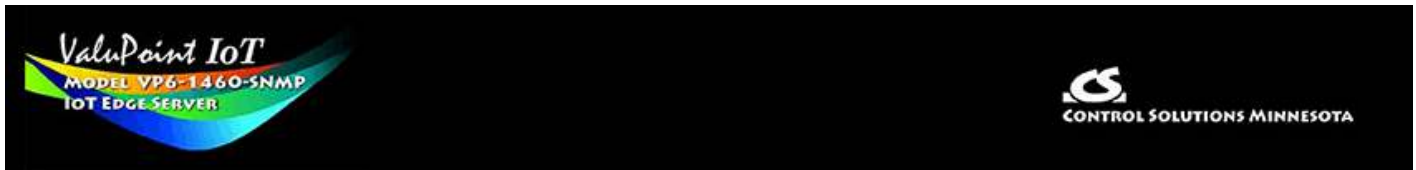
**Name:** .iso.org.dod.internet.mgmt.mib-2.system.sysUpTime.0  
**Value:** [TimeTicks] 53 minutes 59 seconds (323961)

**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

.iso.org.dod.internet.mgmt.mib-2.system.sysServices.0



## 11. Configuring the Scheduler

The ValuPoint becomes more useful when control functions can be combined with monitoring. One element of control that is often useful is the ability to schedule things to happen at certain times on certain days. The scheduler makes that possible.

Scheduling is done in a very generic and simple way. A register you select will change value according to a schedule you provide. From there, you can use the client to write that register to some external Modbus device to cause action according to your schedule.

The scheduler does require access to an SNTP server in order to know what the current time and date are. Be sure to configure NTP on the Network setup page. If SNMP is not an option, you can also use the internal battery backed real time clock but it will be up to you to be sure it is set correctly.

### 11.1 Weekly Schedule

The weekly schedule allows you to specify that something should happen at a certain time of certain days of the week. It can be one day, multiple days, or every day.

The days of the week start with Sunday in the left column. Simply check the boxes for those days you want action. Then select on and off time of day using 24-hour format. Select a register number, and its "on" and "off" value.

The "on" state will be that period that falls between On Time and Off Time. Any other time is "off". If multiple lines are used for the same day and same register, they should be organized with later times last and they will be processed sequentially.

Using the example illustrated below, local register 27 will be set to a value of 10 from 10:00AM until noon on Sunday, and be set to a value of 2 at all other times. And so forth.

Valupoint IoT  
MODEL VP6-1460-SNMP  
IOT EDGE SERVER

CONTROL SOLUTIONS MINNESOTA

Local Data Modbus SNMP System

System Setup Scheduler

Weekly Schedule On Demand Holidays

Showing 1 to 11 of 11 Update < Prev Next >

| #  | S...M...T...W...T...F...S   | Holidays | On Time  | Off Time | Register Number | "On" Value | "Off" Value | Register Name |
|----|---|----------|----------|----------|-----------------|------------|-------------|---------------|
| 1  | <input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>   | Holidays | 10:00:00 | 12:00:00 | 27              | 10.00000   | 2.000000    | Data Value 1  |
| 2  | <input type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>   | Holidays | 10:00:00 | 12:00:00 | 28              | 10.00000   | 2.000000    | Data Value 2  |
| 3  | <input type="checkbox"/> <input type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>   | Holidays | 10:00:00 | 12:00:00 | 29              | 10.00000   | 2.000000    | Data Value 3  |
| 4  | <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>   | Holidays | 10:00:00 | 12:00:00 | 30              | 10.00000   | 2.000000    | Data Value 4  |
| 5  | <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>   | Holidays | 10:00:00 | 12:00:00 | 31              | 10.00000   | 2.000000    | Data Value 5  |
| 6  | <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/>   | Holidays | 10:00:00 | 12:00:00 | 32              | 10.00000   | 2.000000    | Data Value 6  |
| 7  | <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input checked="" type="checkbox"/>   | Holidays | 10:00:00 | 12:00:00 | 33              | 10.00000   | 2.000000    | Data Value 7  |
| 8  | <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>  | Holidays | 10:00:00 | 12:00:00 | 34              | 10.00000   | 2.000000    | Data Value 8  |
| 9  | <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> | Holidays | 10:00:00 | 12:00:00 | 35              | 10.00000   | 2.000000    | Data Value 9  |
| 10 | <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> | Holidays | DUSK-30  | DAWN+30  | 36              | 10.00000   | 2.000000    | Data Value 10 |
| 11 | <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>  | Holidays | 0:00:00  | 0:00:00  | 0               | 0.00       | 0.00        |               |

Events Enabled: 11 ReSync Insert Delete

Note the ReSync button at the bottom. If you have made changes to the scheduler, ReSync will cause everything about the schedule to be re-evaluated and registers updated accordingly. Normally, registers are written only when the schedule says it is time to change. The evaluation is made at the start of the configured time period. Therefore, if you have made a new schedule entry that says a register should be "on" now, you will need to hit the ReSync button to cause that to happen now.

Click Update to register your changes. The "Events Enabled" simply sets the scope of the web page display. If you are just starting out and want to see a page of 10 unused entries, set this to 10 and update. If you have many entries, use Next and Prev to scroll through the list. Insert will insert a new blank entry before the entry number in the Showing box at the top. Delete will delete the entry number you enter in the Showing box at the top.

## 11.2 On Demand Scheduled Events

The scheduler also provides the opportunity to schedule something to happen just one time on a given day or days. Instead of day of week, a date is provided here. Other than selection of day, the On Demand scheduler works the same as Weekly scheduler (except there are no holidays for On Demand).



Using the example illustrated below, local register 27 will be set to a value of 100 starting at 3:00PM February 27, 2024, and remain at that value until 10:00AM February 28. At all other times, local register 27 will be set to 1. And so forth.

| Weekly Schedule |          | On Demand        |          | Holidays          |                    |            |             |               |  |
|-----------------|----------|------------------|----------|-------------------|--------------------|------------|-------------|---------------|--|
| #               | On Time  | On Date<br>Y-M-D | Off Time | Off Date<br>Y-M-D | Register<br>Number | "On" Value | "Off" Value | Register Name |  |
| 1               | 15:00:00 | 2024-02-27       | 10:00:00 | 2024-02-28        | 27                 | 100.0000   | 1.000000    | Data Value 1  |  |
| 2               | 10:00:00 | 2024-02-27       | 15:00:00 | 2024-02-28        | 28                 | 200.0000   | 2.000000    | Data Value 2  |  |
| 3               | 14:00:00 | 2024-02-27       | 14:30:00 | 2024-02-27        | 29                 | 300.0000   | 3.000000    | Data Value 3  |  |
| 4               | 15:00:00 | 2024-02-27       | 15:30:00 | 2024-02-27        | 29                 | 310.0000   | 4.000000    | Data Value 3  |  |
| 5               | 16:00:00 | 2024-02-27       | 16:30:00 | 2024-02-27        | 29                 | 320.0000   | 5.000000    | Data Value 3  |  |
| 6               | 0:00:00  | 0000-00-00       | 0:00:00  | 0000-00-00        | 0                  | 0.00       | 0.00        |               |  |

Showing 1 to 6 of 6

Update < Prev Next >

# Commands Enabled: 6

Insert Delete

Entries applied to the same register number will be processed sequentially. Register 29 in the above example will be set to a value of 3 prior to 2:00PM Feb. 27. Then from 2:00PM to 2:30PM on Feb. 27, it will be set to a value of 300. From 2:30PM to 3:00PM, the value will be 3. From 3:00PM to 3:30PM, the value will be 310. From 3:30PM to 4:00PM, the value will be 4. From 4:00PM to 4:30PM, the value will be 320. Any time after 4:30PM Feb. 27, the value will be 5. On October 1, the value in register 29 will still be 5.

Click Update to register your changes. The "# Commands Enabled" simply sets the scope of the web page display. If you are just starting out and want to see a page of 10 unused entries, set this to 10 and update. If you have many entries, use Next and Prev to scroll through the list. Insert will insert a new blank entry before the entry number in the Showing box at the top. Delete will delete the entry number you enter in the Showing box at the top.

### 11.3 Holidays

Sometimes you want a weekly schedule to not apply on a holiday, or maybe you want something to only happen on a holiday (although that would be nearly the same as On Demand). The holiday processing in the scheduler allows exceptions to the weekly schedule.

Start by creating a Holiday on the Holidays tab. Give it a name, start time and date, and end time and date. Most often the start time for a holiday will be 0:00:00 and end time will be 23:59:59 so that it means "all day". You may create up to 32 holidays.

| Weekly Schedule                       |              | On Demand |                  | Holidays |                   |
|---------------------------------------|--------------|-----------|------------------|----------|-------------------|
| <input type="button" value="Update"/> |              |           |                  |          |                   |
| #                                     | Holiday Name | On Time   | On Date<br>Y-M-D | Off Time | Off Date<br>Y-M-D |
| 1                                     | Test Holiday | 0:00:00   | 2024-02-14       | 23:59:00 | 2024-02-14        |
| 2                                     |              | 0:00:00   | 0000-00-00       | 0:00:00  | 0000-00-00        |
| 3                                     |              | 0:00:00   | 0000-00-00       | 0:00:00  | 0000-00-00        |

To incorporate a holiday into a weekly schedule entry, click on that line's Holidays link.

| Weekly Schedule   |   | On Demand                |          | Holidays |                 |   |             |               |
|---|---|--------------------------|----------|----------|-----------------|---|-------------|---------------|
| Showing 1 to 11 of 11 <input type="button" value="Update"/> <input type="button" value=" &lt; Prev"/> <input type="button" value=" Next &gt;"/> |   |                          |          |          |                 |   |             |               |
| #   | S...M...T...W...T...F...S   | Holidays                 | On Time  | Off Time | Register Number | "On" Value  | "Off" Value | Register Name |
| 1   | <input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>   | <a href="#">Holidays</a> | 10:00:00 | 12:00:00 | 27              | 10.00000  | 2.000000    | Data Value 1  |
| 2   | <input type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>   | <a href="#">Holidays</a> | 10:00:00 | 12:00:00 | 28              | 10.00000  | 2.000000    | Data Value 2  |
| 3   | <input type="checkbox"/> <input type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>   | <a href="#">Holidays</a> | 10:00:00 | 12:00:00 | 29              | 10.00000  | 2.000000    | Data Value 3  |
| 4   | <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>   | <a href="#">Holidays</a> | 10:00:00 | 12:00:00 | 30              | 10.00000  | 2.000000    | Data Value 4  |
| 5   | <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>   | <a href="#">Holidays</a> | 10:00:00 | 12:00:00 | 31              | 10.00000  | 2.000000    | Data Value 5  |
| 6   | <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/>   | <a href="#">Holidays</a> | 10:00:00 | 12:00:00 | 32              | 10.00000  | 2.000000    | Data Value 6  |
| 7   | <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input checked="" type="checkbox"/>   | <a href="#">Holidays</a> | 10:00:00 | 12:00:00 | 33              | 10.00000  | 2.000000    | Data Value 7  |
| 8   | <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>  | <a href="#">Holidays</a> | 10:00:00 | 12:00:00 | 34              | 10.00000  | 2.000000    | Data Value 8  |
| 9   | <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> | <a href="#">Holidays</a> | 10:00:00 | 12:00:00 | 35              | 10.00000  | 2.000000    | Data Value 9  |
| 10  | <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> | <a href="#">Holidays</a> | DUSK-30  | DAWN+30  | 36              | 10.00000  | 2.000000    | Data Value 10 |
| 11  | <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>  | <a href="#">Holidays</a> | 0:00:00  | 0:00:00  | 0               | 0.00  | 0.00        |               |
| Events Enabled: 11 <input type="button" value="ReSync"/>  |   |                          |          |          |                 | <input type="button" value="Insert"/> <input type="button" value="Delete"/> |             |               |

The available holidays will be listed. To add a holiday, click on the holiday in the Available list and click the Add button. To remove a holiday previously added, click on the holiday in the Selected list and then click the Remove button. Once you have added a holiday or two, select whether to include or exclude.

The effect of exclude is to temporarily, effectively, uncheck that day of the week. The effect of include is to temporarily, effectively, check that day of the week. In the example below, regardless of what day of the week it is, if this day happens to be the holiday, the "On" value will not be applied between 10:00AM and noon.

Note that in the following example, no days of the week are selected but a holiday is selected as included. This is effectively an On Demand scheduled event for that holiday. The "On" value will be applied on this holiday, regardless of day of week, between 10:00AM and noon (assuming the holiday is defined as all day - if the holiday starts at 3:00PM, then the "On" value would not be applied and this entry in the schedule will never do anything.)

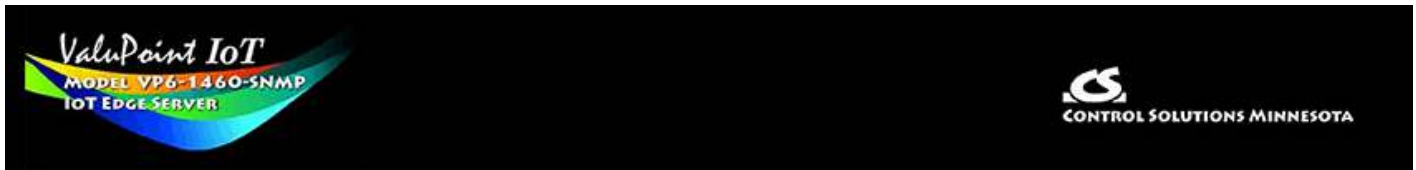
### 11.4 Astronomical Clock

If you were looking closely at the first example in this section, you may have noticed one peculiar entry.

Suppose you are scheduling lights to come on when it gets dark outside. One way of doing that is with a light sensor. Another way is by scheduling, but then you have to keep changing the on and off times throughout the seasons. The astronomical clock feature of this scheduler will keep changing the on and off times for you when you use "DUSK" and "DAWN" as entries. In the example above, register 36 will be set to a value of 10 thirty minutes before sundown, and returned to a value of 2 thirty minutes after sunrise. The more likely scenario would be an on value of 1 and off value of 0 to

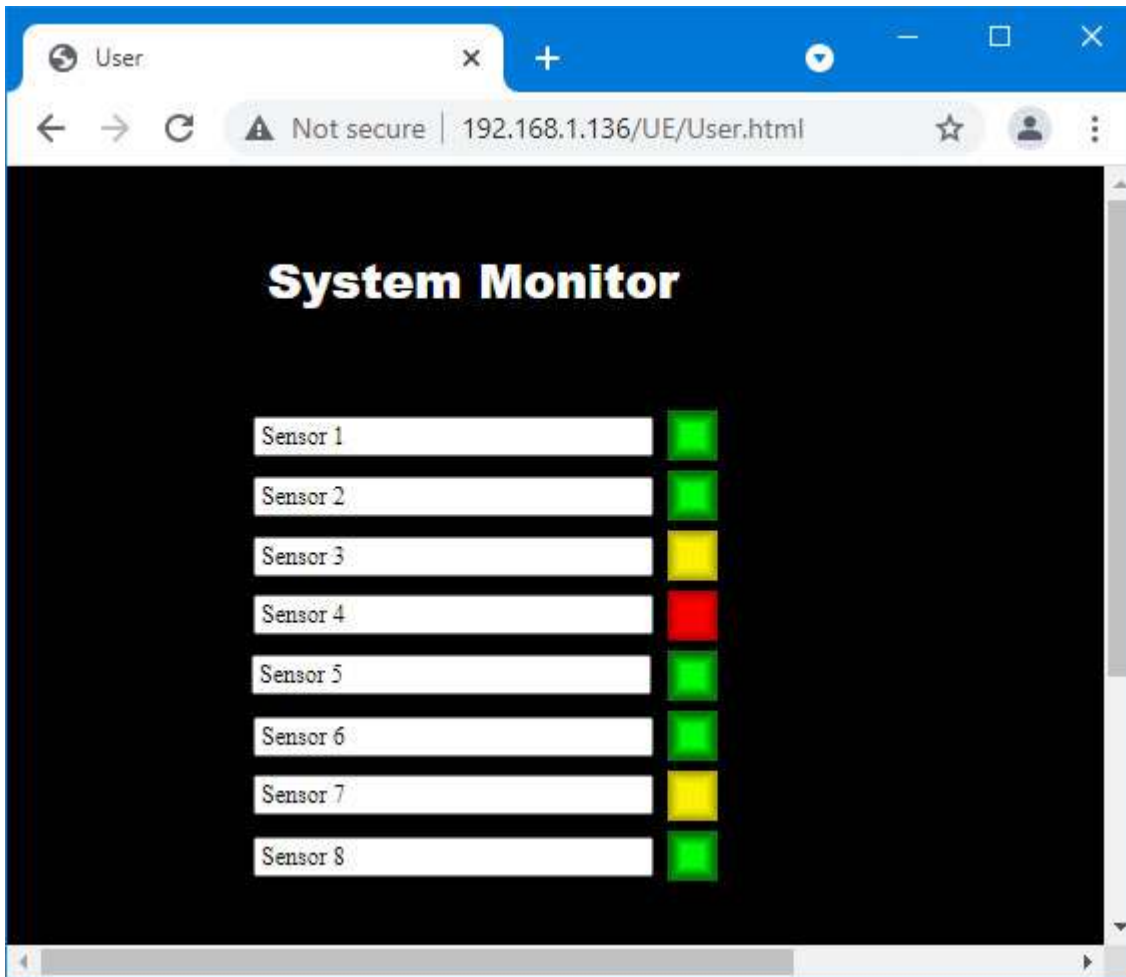
switch a switch somewhere.

Note that in order for the astronomical clock to work correctly in your location, you must set the latitude and longitude for the location on the Network setup page, NTP section. You will also see the currently calculated sunrise and sunset times displayed there.



## 12. User HTML

This section is an updated version of the user\_html\_cgi.pdf document found on the introduction page under User HTML Tutorials at <https://info.csimn.com>. This information has been updated per implementation in ValuPoint. Except for changing the link "../index.html" to "/html/index.html" where applicable, the various demos found in the knowledgebase should work in a ValuPoint.



The "naked pages" referenced in the 2008 version of the CGI overview are not implemented in ValuPoint. They were an attempt at allowing some level of "private labeling" the ValuPoint. If private labeling is desired, Control Solutions does offer that option which will result in fully rebranding the entire preprogrammed web UI.

### 12.1 Static HTML

User HTML may be installed as a "wrapper" around the default web pages. To install user HTML, use the File Manager (filter set to \*.\* ) to upload any combination of .html,

.txt, .gif, and .jpg files. You can also include .pdf, .xml, .css and JavaScript .js files. Note that PHP, ASP, etc., are not supported.

You can also use FTP to upload files to the /FLASH0 (that's flash zero) directory and this will be the default directory upon opening an FTP session. An FTP session in Linux, Windows command prompt, or Windows PowerShell work well. Some smart FTP clients work, but some try to be too helpful and simply screw things up. We tested FileZilla at one point and it seemed to work ok, but we do not keep testing every revision of FileZilla.

The top level user file must be named User.html (case sensitive). If this file is present in /FLASH0, it will be served instead of the default index.html page pre-programmed in the device any time you simply browse to the device's IP address. Once this page is open, it may link to any other html files in the /FLASH0 directory. All user HTML is filtered as it is served to provide dynamic content.

There are a handful of tricks that must be observed to make user html work. All references to other user pages and to user image files must have the file names preceded with /UE/ as in /UE/User2.html or /UE/mypicture.gif. The UE stands for "user escape" and is treated as a virtual directory that actually points to /FLASH0. All preprogrammed pages are found in /html/ and preprogrammed images are found in /img/.

## 12.2 Dynamic Data Tags in User HTML

### Dynamic Data – Creating a Form

Dynamic access to register data is provided. Dynamic updates of register contents is also supported via the form post method. The form must be defined using the following tags:

```
<form id="UserForm" action="/UE/icanForm" method="post"
name="UserForm">

</form>
```

The submit button causing the post must be defined as:

```
<input type="submit" name="submitChange" value="Change">
```

Any submit buttons other than those recognized as noted here will simply result in a page refresh. Only the submit button named "submitChange" with a value of "Change" will result in parsing of the form data. Only a form with action as named above will be parsed.

If you want to redefine the appearance of the button, you can implement a graphic button by including an image as follows, and then including the JavaScript function as shown:

```

```

```
<script type="text/javascript"><!--
function sendMeAway() {
    document.UserForm.submitChange.value="Change";
    document.UserForm.submit();
}
//--></script>
```

### Input Types: <input type="text">

Two types of data input are recognized by CGI processing of the user post: Text input and option select. The search string keyed upon for text is **<input type="text">** and the search string keyed upon for the select option is **<select name=**

A text input should be constructed as follows:

```
<input type="text" name="reg22" value="%d" readonly size="8">
```

The contents of the register number included in the name ("regX") will be displayed when the page is served, and the data will be taken dynamically from the register at that point in time, and again each time the page is refreshed. The data will be formatted using the C format string found in the value tag. Integer formats (%d, %04d, %x, etc) should be used for integer registers, and floating point (%f, %.2f, etc) should be used for floating point registers. If "readonly" is specified, data will only be displayed in this window. Otherwise the data returned by the post will be parsed, and the result placed back into the register.

The following keywords are recognized as text input "names":

- regX – references the value in local register number X
- namX – references the name of register number X
- site – references the Thing Name (Thing ID page)

All of these data elements may be read, and will be written unless you specify "readonly". The definition of read means take data from the local register when serving the page, and write means write data to the local register if the form was submitted by the appropriately named submit button (see Form above).

### Input Types: <select>

An option select should be constructed as follows:

```
<select name="reg25" size="1">
    <option selected value="0">OFF</option>
    <option value="1">ON</option>
</select>
```

The strings corresponding to the values given will be displayed when the register named matches that value, otherwise "---" will be displayed. When an option is selected and the form posted, the value corresponding to the new selection will be written back into the register. The "selected" tag shown above is not required since it is automatically inserted in the appropriate place (moved around) when the page is served.

## Input Types: <input type="hidden">

An additional form of input has been added to filtered HTML. Hidden variables may be defined using the following syntax:

```
<input type="hidden" name="reg22" value="%d" readonly>
```

This will be processed the same as "text" input except the value is not displayed. This is useful as a means of providing non-displayed data to a JavaScript function. Hidden data upon return will be parsed and put back into registers unless `readonly` is specified. Omit `readonly` if hidden data should be parsed. This provides a means for JavaScript to get data back into registers.

## Page Links

To create a link on the user page to get into the default preprogrammed pages, define a link to `/html/index.html`, for example:

```
<a href="/html/index.html">Log In</a>
```

To link to another user page in the FLASH0 directory, use a link such as:

```
<a href="/UE/pwUserP3.html">Room #1</a>
```

Links to graphic images you want shown on the page are created in similar fashion:

```

```

Note that you preface the page name with `/UE/` when the file is located in the FLASH0 directory, but preface the name with `/html/` when accessing a preprogrammed page.

## Password Protection

There are 3 levels of password protection: Restricted, Maintenance, Administrator (root is a special form of administrator). User pages may be password protected at the "Restricted" level. To password protect a user page, simply insert the letters "pw" in front of the name. Therefore, if *User2.html* is a page you wish to protect, rename it *pwUser2.html*.

The top level page for User HTML must still be named *User.html* (and not *user.html* or *USER.html*). If you don't want anything useful to be completely unrestricted, simply put a plain dumb page in *User.html* with a link that says "log in" and link it to *pwUser.html*.

Note that "restricted" level of password protection means the user can access any "pw" user pages, but cannot access any pages beyond index in the pre-programmed page set.

## Other Input Types



Radio buttons and other forms of input are not supported at this time. The HTML will be passed through, but not filtered and associated with register data. Therefore, you can use radio buttons, etc., with JavaScript, but you must explicitly associate the resulting data with hidden input variables in order to return the data to registers.

### Additional Special Submit Buttons

The submit button causing the post for changing data values must be defined as:

```
<input type="submit" name="submitChange" value="Change">
```

Two additional button actions are available to save the configuration file or restart the device:

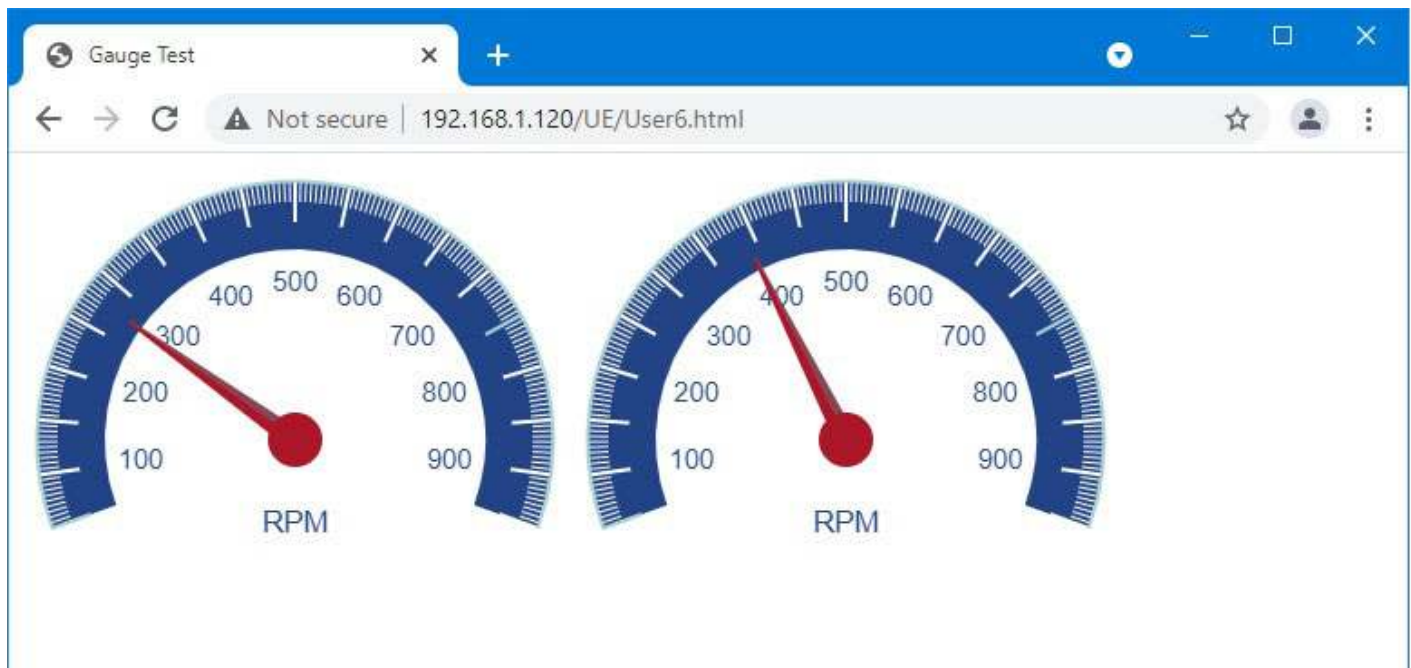
```
<input type="submit" name="submitSave" value="Save">  
<input type="submit" name="submitRestart" value="Restart">
```

## 12.3 Live JavaScript Gauges

An external JavaScript library can be specified. The JS file should be loaded into the /FLASH0 directory along with User.html, etc. In the HTML file, the script file is referenced as illustrated here by the first few lines of User.html that generated the gauges pictured in the screen shot.

```
<!doctype html>  
<html>  
<head>  
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  
<title>Gauge Test</title>  
<link rel="stylesheet" href="/FS/FLASH0/fonts.css">  
<script src="/FS/FLASH0/gauge.min.js"></script>  
</head>  
<body style="background: #fff" onload="animateGauges()">  
  
<canvas id="canvasPressure"></canvas>  
<canvas id="canvasPressure2"></canvas>  
  
<script>  
var gaugePressure = new RadialGauge({
```

(entire project is available on [csimn.com](http://csimn.com) web site)



Many different JavaScript gauges are available at <https://canvas-gauges.com/>.

To cause the gauges to automatically update in real time, you need two things: The `animateGauges()` function in the JavaScript, and something that is going to retrieve real time data into the HTML document. The real time data retrieval is done by a hidden iframe, included in the User.html like this:

```
<iframe name="phantom" src="UserGetData.html" frameborder="0"
height="50" width="50"></iframe>
```

The UserGetData.html is constructed like this:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">

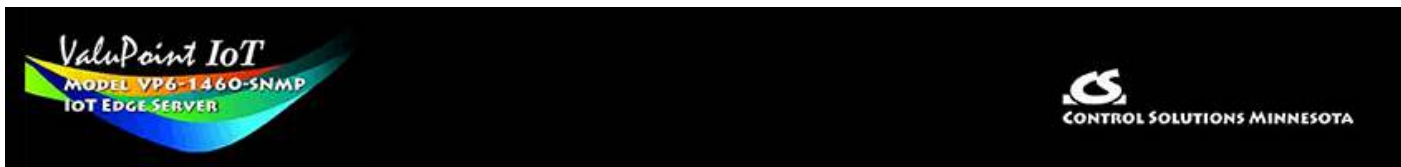
<html>
<head>
<meta http-equiv="content-type" content="text/html; charset=iso-8859-1">
<title>Untitled Page</title>
<meta http-equiv="refresh" content="1">
</head>

<body bgcolor="#ffffff">
<form id="UserForm" action="/UE/icanform" method="post" name="UserForm">
<div align="left">
<input type="hidden" name="reg1" value="%d"><input type="hidden"
name="reg2" value="%d"></div>
</form>
<p></p>
</body>

</html>
```

**NOTE:** The above example using an iframe to retrieve real time data was created prior to adding the REST API capability to ValuPoint. The gauge animation could be

restructured to take advantage of the REST API if you know your way around JavaScript.



## Appendix A Hardware Details

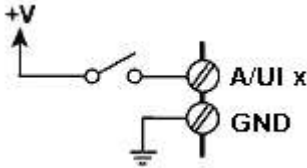
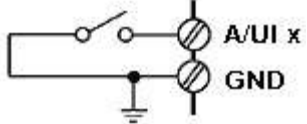
### A.1 Wiring, Physical I/O Connections

#### Connection of Inputs

The VP6-1460 contains no configuration jumpers for configuring I/O points. There is no need to open the enclosure for configuration of I/O. Input types are switched under software control.

Input points should be connected as indicated in the various diagrams below. In addition to selecting a wiring diagram, the corresponding selections should be made on the I/O Config page as discussed in section 4.2 of this user guide.

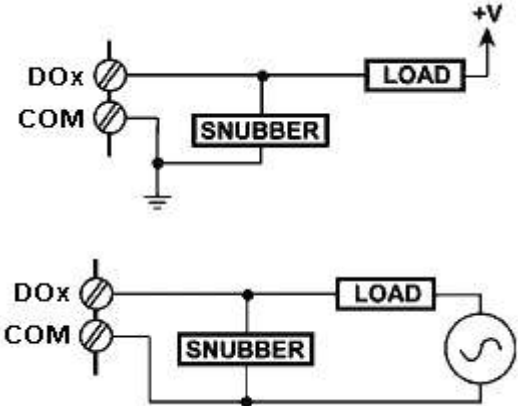
| I/O Point Type   | Wiring Guide | Additional Information  |
|--|--------------|---|
| <b>A/UI Analog Input:</b><br><b>0-10VDC</b><br><b>Voltage Input</b><br><b>or Pulse Count</b> |              | <p>A/UI inputs 1-12 will accept voltage inputs of up to 10VDC. Voltages to 11VDC will be measured. Voltages to 24VDC will be tolerated, but measurement is internally limited to a reading of 11VDC.</p> <p>This wiring diagram is also applicable to pulse counter input for counting pulses from an active pulse generator.</p> |
| <b>A/UI Analog Input:</b><br><b>0-20mA</b><br><b>Current Input</b>                           |              | <p>A/UI inputs 1-12 will accept current inputs of 0-20mA with the addition of a 500 ohm 1/2 watt external resistor. A 500 ohm resistor will produce a 0-10VDC signal. A 250 ohm resistor may also be used to produce a 0-5VDC signal.</p>   |
| <b>A/UI Analog Input:</b><br><b>Thermistor Input</b>   |              | <p>A/UI inputs 1-12 will accept thermistors of 3k, 5k, 10k, or 20k ohms. Linearization via interpolation of a 56-point table is performed internally.</p>   |

|   |   |  |
|---|---|--|
| <p><b>A/UI or DI<br/>Discrete Input:<br/>Discrete Voltage</b></p>   |  | <p>A/UI inputs 1-12 may be connected as discrete voltage sensing inputs. Inputs up to 24VDC are tolerated, but threshold sensing only functions over the 0-10VDC range.</p>  |
| <p><b>A/UI or DI<br/>Discrete Input:<br/>Dry<br/>Contact Closure<br/>to Ground or<br/>Pulse Count</b></p> |  | <p>A/UI inputs 1-12 may be connected as discrete inputs sensing dry contact closure to ground. Internal excitation of 0.3mA is provided. The excitation current may be increased by the addition of an external resistor (pullup to DC power).</p> <p>This wiring diagram is also applicable to pulse counter input for counting pulses from a switch closure.</p> |

**Connection of Outputs**

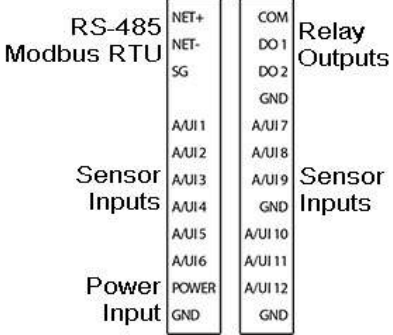
The VP6-1460 contains no configuration jumpers for configuring outputs. There is no need to open the enclosure.

Output points should be connected as indicated in the diagrams below. The discrete (relay) outputs are SPST N.O. with the common side connected together to the COM terminal. The COM terminal for relays is NOT electrically common to any of the GND terminals.

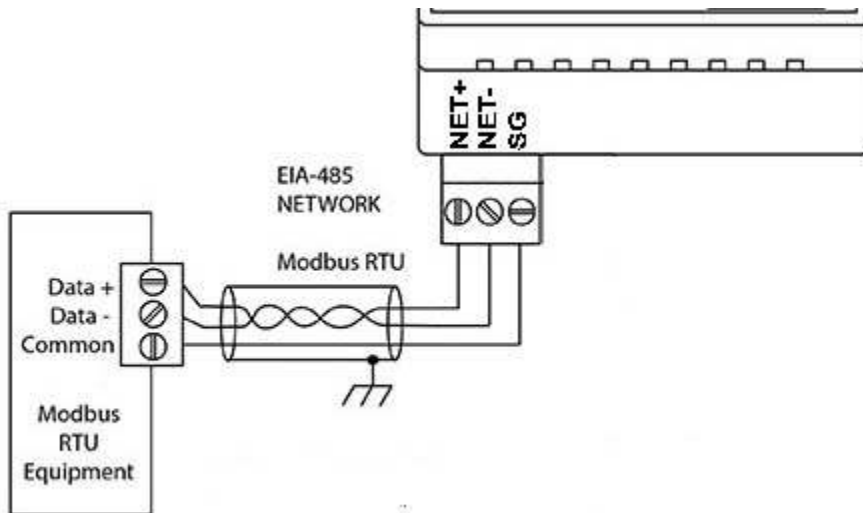
| I/O Point Type                                      | Wiring Guide  | Additional Information  |
|---|---|---|
| <p><b>Discrete<br/>Output:<br/>Form A Relay</b></p> |  | <p>DO outputs 1-2 are Form A dry contact relays rated for 2A @ 120VAC (resistive). Snubbers should be used with inductive loads. The relays are also rated for 2A @ 30VDC. Note: Relays are rated higher, but UL listing is for 2A.</p> |

**Connection of Power and Communications**

Power and communications should be connected as indicated below.

| I/O Point Type          | Wiring Guide  | Additional Information  |
|-------------------------|---|---|
| <b>Power</b>            | Connect AC or +DC power to Power terminal. Connect common or -DC power to GND terminal. GND is common to all terminals labeled COM.   | Nominal power consumption is 2.4 watts, or 0.1A @ 24VDC, with all relays on.  |
| <b>Communications</b>   | Connect Modbus RTU RS-485 data lines to NET+/- terminals. The RS-485 port is electrically isolated from all other terminals including power. Therefore, the SG must be connected to the ground reference for the RS-485 signal in order to communicate.   | Communication signals comply with EIA-485 standard.   |
| <b>Wiring Terminals</b> |  <p>The diagram shows a terminal block with the following connections:</p> <ul style="list-style-type: none"> <li><b>RS-485 Modbus RTU:</b> NET+ (top), NET- (middle), SG (bottom).</li> <li><b>Sensor Inputs:</b> A/UI1, A/UI2, A/UI3, A/UI4, A/UI5, A/UI6.</li> <li><b>Power Input:</b> POWER, GND.</li> <li><b>Relay Outputs:</b> COM, DO 1, DO 2, GND.</li> <li><b>Sensor Inputs (continued):</b> A/UI7, A/UI8, A/UI9, GND, A/UI10, A/UI11, A/UI12, GND.</li> </ul> | <p>Screw terminals ratings are substantially in excess of any I/O point ratings.</p> <p>Screw terminals are pluggable. They unplug from the unit in 3, 8, and 12-position blocks.</p> |

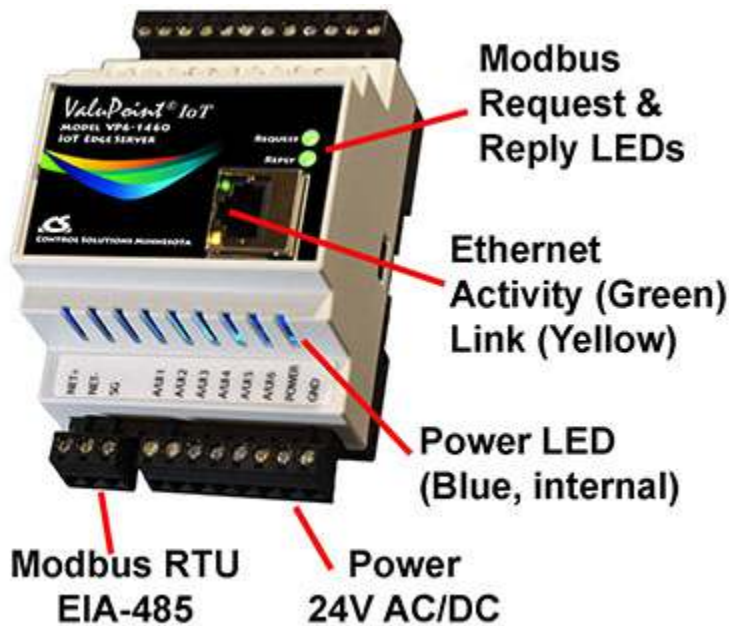
The RS-485 should be wired as illustrated below.



## A.2 Front Panel LED Indicators

VP6-1460 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.



VP6-1460 Request and Reply LEDs reflect Modbus RTU traffic, and the Ethernet activity LED will indicate network traffic in general. If Modbus RTU is not being used at all, then the Request and Reply LEDs will indicate TCP traffic. If Modbus RTU is in use, then the Request and Reply LEDs will indicate Modbus RTU traffic while the Ethernet LEDs will be the only indication of TCP traffic.

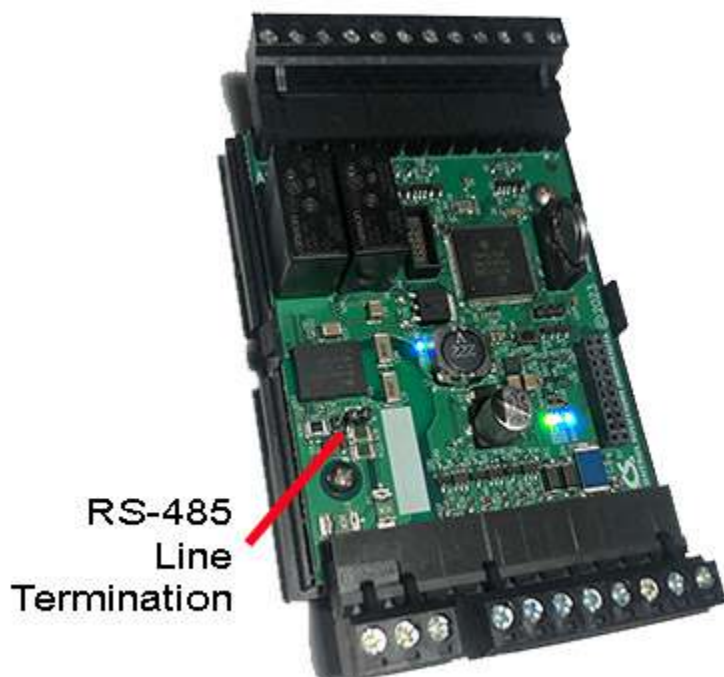
VP6-1460 LEDs indicate as follows (LEDs are bi-color):

|                   |  |
|-------------------|--|
| REQUEST           | Flashes yellow each time a request is sent when operating as Modbus Master, or each time a request is received when operating as Modbus Slave.   |
| REPLY             | Operating as Modbus Master, flashes green each time a good response is received, or red when an error code is received, the request times out, or there is a flaw in the response such as CRC error.<br><br>Operating as Modbus Slave, flashes green each time a good response is sent, or red if an exception code is sent (meaning the received request resulted in an error). |
| Ethernet Activity | Green LED is on solid during portions of the boot-up process, and then flashes briefly when Ethernet network traffic is detected.  |
| Ethernet Link     | Yellow LED indicates an Ethernet link is present. This indicator will light if a link is present regardless of processor or network activity. If not lit, check network wiring.  |

|        |  |
|--------|--|
| Status | Blue LED (internal) on any time power is present and internal power supply is functioning. In addition, a green "heartbeat" LED on the base board will blink about once every 2 seconds. |
|--------|--|

### A.3 RS-485 Line Termination

RS-485 line termination jumper is located as indicated in the photo. The line termination is "on" when the jumper block is aligned with the screened white block on the circuit board. Termination is required when the VP6-1460 is the last device on the daisy chain.



### 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 ValuPoint'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 ValuPoint 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 ValuPoint is still functional. Connect the ValuPoint 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 ValuPoint.

Upon power up, VP6-1460 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 ValuPoint 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 ValuPoint 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 ValuPoint is located at 192.168.1.42.

The image shows a Wireshark capture window titled "(Untitled) - Wireshark". The main pane displays a list of captured packets. Packet 12 is selected and highlighted in blue. The details pane below shows the structure of this packet: Ethernet II, Src: Digiboar\_2e:de:3f (00:40:9d:2e:de:3f), Dst: Broadcast (ff:ff:ff:ff:ff:ff), and Address Resolution Protocol (request). The ARP request details show the Sender MAC address as Digiboar\_2e:de:3f (00:40:9d:2e:de:3f), the Sender IP address as 192.168.1.42 (192.168.1.42), the Target MAC address as 00:00:00:00:00:00 (00:00:00:00:00:00), and the Target IP address as 192.168.1.43 (192.168.1.43). Red arrows point to the source MAC and IP in the packet list, and the sender IP in the details pane. The packet bytes pane at the bottom shows the raw data in hexadecimal and ASCII.

| No. | Time      | Source            | Destination     | Protocol | Info  |
|-----|-----------|-------------------|-----------------|----------|---|
| 1   | 0.000000  | Ibm_5e:b7:30      | Broadcast       | ARP      | Gratuitous ARP for 192.168.1.25 (Request)   |
| 2   | 0.999898  | Ibm_5e:b7:30      | Broadcast       | ARP      | Gratuitous ARP for 192.168.1.25 (Request)   |
| 3   | 2.001429  | Ibm_5e:b7:30      | Broadcast       | ARP      | Gratuitous ARP for 192.168.1.25 (Request)   |
| 4   | 3.031837  | Ibm_5e:b7:30      | Broadcast       | ARP      | who has 192.168.1.1? Tell 192.168.1.25      |
| 5   | 3.032390  | 192.168.1.25      | 239.255.255.250 | IGMP     | V2 Membership Report / Join group 239.255.2 |
| 6   | 4.004269  | 192.168.1.25      | 239.255.255.250 | IGMP     | V2 Membership Report / Join group 239.255.2 |
| 7   | 4.024360  | Ibm_5e:b7:30      | Broadcast       | ARP      | who has 192.168.1.1? Tell 192.168.1.25      |
| 8   | 5.005791  | 192.168.1.25      | 239.255.255.250 | IGMP     | V2 Membership Report / Join group 239.255.2 |
| 9   | 5.025819  | Ibm_5e:b7:30      | Broadcast       | ARP      | who has 192.168.1.1? Tell 192.168.1.25      |
| 10  | 5.034508  | Ibm_5e:b7:30      | Broadcast       | ARP      | who has 192.168.1.1? Tell 192.168.1.25      |
| 11  | 39.073317 | Digiboar_2e:de:3f | Broadcast       | ARP      | Gratuitous ARP for 192.168.1.42 (Request)   |
| 12  | 39.289914 | Digiboar_2e:de:3f | Broadcast       | ARP      | who has 192.168.1.43? Tell 192.168.1.42     |
| 13  | 43.994394 | 192.168.1.42      | 224.0.0.128     | IGMP     | V2 Membership Report / Join group 224.0.0.1 |

Frame 12 (60 bytes on wire, 60 bytes captured)  
 Ethernet II, Src: Digiboar\_2e:de:3f (00:40:9d:2e:de:3f), Dst: Broadcast (ff:ff:ff:ff:ff:ff)  
 Address Resolution Protocol (request)  
 Hardware type: Ethernet (0x0001)  
 Protocol type: IP (0x0800)  
 Hardware size: 6  
 Protocol size: 4  
 Opcode: request (0x0001)  
 Sender MAC address: Digiboar\_2e:de:3f (00:40:9d:2e:de:3f)  
 Sender IP address: 192.168.1.42 (192.168.1.42)  
 Target MAC address: 00:00:00:00:00:00 (00:00:00:00:00:00)  
 Target IP address: 192.168.1.43 (192.168.1.43)

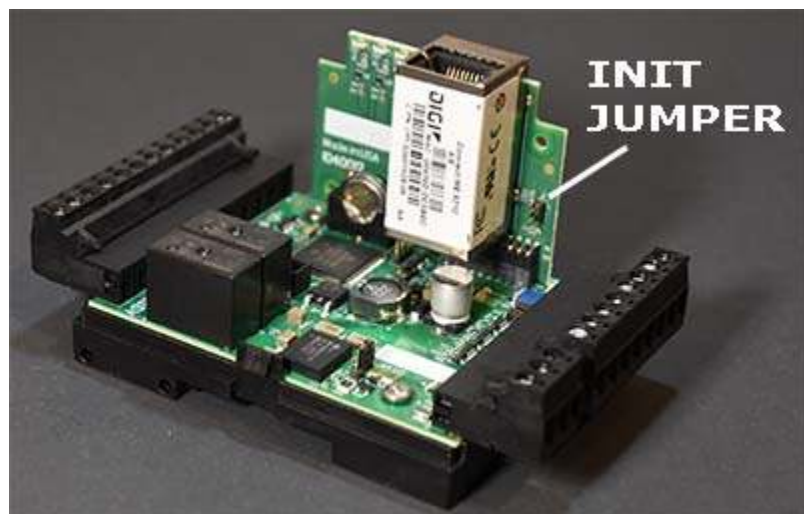
0000 ff ff ff ff ff ff 00 40 9d 2e de 3f 08 06 00 01  
 0010 08 00 06 04 00 01 00 40 9d 2e de 3f c0 a8 01 2a  
 0020 00 00 00 00 00 00 c0 a8 01 2b a4 d6 d5 d9 d1 d1  
 0030 00 00 00 00 00 00 00 00 00 00 00 00

Frame (frame), 60 bytes      Packets: 13 Displayed: 13 Marked: 0 Dropped: 0      Profile: Default

## A.6 Forced Hard Configuration Reset

**IMPORTANT:** Before considering the forced hard reset, be sure you have considered soft configuration reset, or discovering lost IP address if applicable.

The "Init" jumper inside the VP6-1460 serves two purposes, and what it does depends on whether you apply the jumper before or after the ValuPoint boots up.



### **Hard Configuration Reset:**

Installing the jumper after bootup causes the ValuPoint 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 ValuPoint will begin to flash its LEDs about once a second indefinitely until you remove the jumper, and then it will automatically restart.

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 ValuPoint'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.



The screenshot shows a web interface for a ValuPoint IoT Model VP6-1460-SNMP IOT Edge Server. The header includes the ValuPoint IoT logo and the Control Solutions Minnesota logo. The main content area has a dark teal background. There are two input fields with corresponding buttons:

- A text input field labeled "Valid MAC Address" containing the value "00:40:9D:45:45:EA" and a "Restore" button.
- A text input field labeled "Reformat Flash file system" which is currently empty, and a "Wipe" button.

### 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.

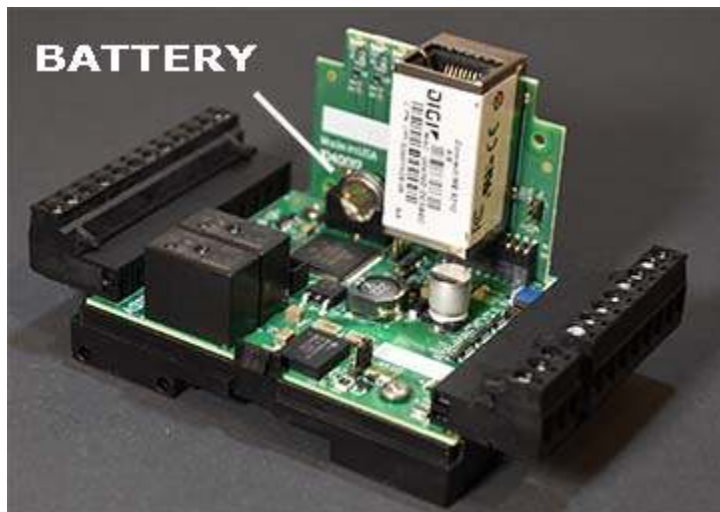
The brute force approach to updating firmware using TFTP as noted in the section above is always available, but the more graceful approach is to use FTP to upload the new image.bin file. There is one minor problem: The upload wants to buffer the entire file in RAM while it proceeds to reprogram the Flash 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 ValuPoint 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>.

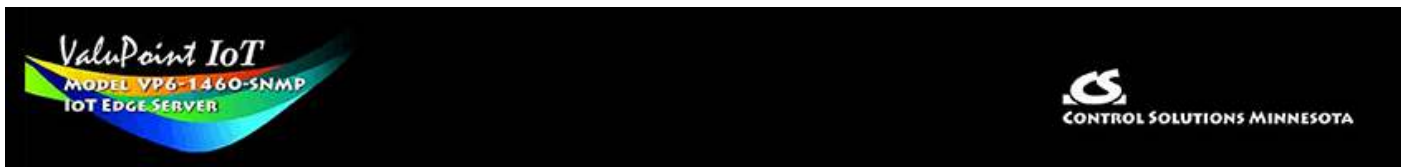
## A.8 Battery Replacement

No action is required of the user to activate the battery that backs up the real time clock. The battery should have a 10-year life in normal operation with "normal" meaning that the device is normally continuously powered for use with only intermittent dependency on the battery backup.



Replace battery with BR1225A only. Use of another battery may present a risk of fire or explosion. Replace with battery polarity as marked on the circuit board. Reverse polarity of the battery will not damage the board, but the clock will not be backed up if the battery is reversed.

**CAUTION:** The lithium battery contained in this device may explode if mistreated. DO NOT recharge, disassemble, or dispose of in fire.



## Appendix B Modbus CSV Import Files

### B.1 Modbus RTU Master Read/Write Maps

The CSV file for configuring Modbus TCP client read and write maps should contain a single header line with the labels indicated below, and content as applicable.

| Header Line Label | Notes | Description of Use   |
|-------------------|-------|--|
| RW                | -     | Enter 'R' to Read from a remote device, or 'W' to Write to a remote device.  |
| TYPE              | -     | Use this column to specify remote registers by type (see 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)  |
| SLAVE             | -     | Provide the slave address, ID, or unit number, of the Modbus RTU slave to be polled.   |
| SWAP              | -     | Any data item that occupies more than one Modbus register, e.g. 32-bit or Float, needs to have the register order defined since this is not standardized by Modbus protocol. The ValuPoints default to the high order register first. If the remote Modbus slave has its registers ordered with low order first, then select 'T' (True) to "swap" the register order (or 'F' to keep the default order). |
| SCALE             | -     | Data is multiplied by this scale factor after read from a remote device or before being written to a remote device.  |
| OFFSET            | -     | This offset is added to the data value after read from a remote device or before being written to a remote device.   |
| POLL              | -     | Specify a periodic poll time in seconds (fractions of sections are recognized).  |
| 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              | -     | 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   |

|          |   |   |
|----------|---|---|
|          |   | <p>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 RTU must include RW, LOCALREG, TYPE, REG, FORMAT, and SLAVE. All other columns are optional.

This is an example of a minimum CSV file as it would appear in a spread sheet program:

|    | A  | B    | C   | D      | E     | F        | G | H | I | J |
|----|----|------|-----|--------|-------|----------|---|---|---|---|
| 1  | RW | TYPE | REG | FORMAT | SLAVE | LOCALREG |   |   |   |   |
| 2  | R  | HOLD | 40  | S16    | 22    | 1        |   |   |   |   |
| 3  | R  | HOLD | 42  | U32    | 22    | 2        |   |   |   |   |
| 4  | R  | HOLD | 44  | U32    | 22    | 3        |   |   |   |   |
| 5  | R  | HOLD | 46  | U32    | 22    | 4        |   |   |   |   |
| 6  | R  | HOLD | 48  | U16    | 22    | 5        |   |   |   |   |
| 7  | R  | HOLD | 50  | U16    | 22    | 6        |   |   |   |   |
| 8  | R  | HOLD | 52  | U16    | 22    | 7        |   |   |   |   |
| 9  | R  | HOLD | 54  | S16    | 22    | 8        |   |   |   |   |
| 10 | R  | HOLD | 56  | S32    | 22    | 9        |   |   |   |   |
| 11 | R  | HOLD | 58  | S32    | 22    | 10       |   |   |   |   |
| 12 | R  | HOLD | 60  | S16    | 22    | 11       |   |   |   |   |

This is an example of how the minimum CSV file looks as just plain text:

```

rtu.csv - Notepad
File Edit Format View Help
RW,TYPE,REG,FORMAT,SLAVE,LOCALREG
R,HOLD,40,S16,22,1
R,HOLD,42,U32,22,2
R,HOLD,44,U32,22,3
R,HOLD,46,U32,22,4
R,HOLD,48,U16,22,5
R,HOLD,50,U16,22,6
R,HOLD,52,U16,22,7
R,HOLD,54,S16,22,8
R,HOLD,56,S32,22,9
R,HOLD,58,S32,22,10
R,HOLD,60,S16,22,11
R,HOLD,62,U32,22,12
R,HOLD,64,U32,22,13
R,HOLD,66,U32,22,14
    
```

## B.2 Modbus TCP Client Read/Write Maps

The CSV file for configuring Modbus TCP client read and write maps should contain a single header line with the labels indicated below, and content as applicable.

The only difference between TCP and RTU formats is DEVNUM and UNIT in TCP, versus just SLAVE in RTU. Everything else is identical.

| Header Line | Notes | Description of Use |
|-------------|-------|--------------------|
|-------------|-------|--------------------|

| Label    |   |  |
|----------|---|--|
| 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. (BB2-6010, BB2-7010 only)  |
| SCALE    | - | Data is multiplied by this scale factor after read from a remote device or before being written to a remote device.  |
| OFFSET   | - | This offset is added to the data value after read from a remote device or before being written to a remote device.   |
| POLL     | - | Specify a periodic poll time in seconds (fractions of sections are recognized).  |
| 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.3 Register Types

The content of the TYPE column should contain one of the following CSV Labels:

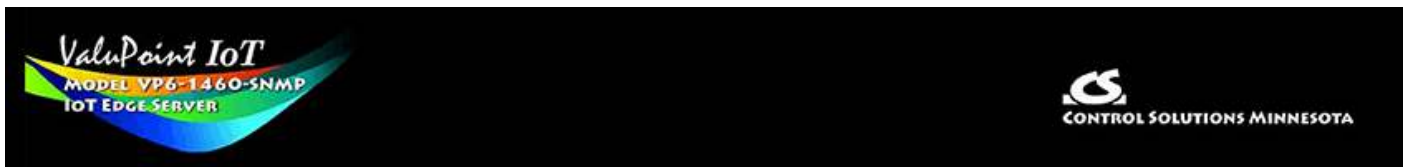
| CSV Label | Modbus Register Type       | Function Code for Read | Function Code for Write |
|-----------|----------------------------|------------------------|-------------------------|
| COIL      | Coil (1 bit)               | 1                      | 5 or 15                 |
| DISC      | Discrete Input (1 bit)     | 2                      | n/a                     |
| INPUT     | Input Register (16 bits)   | 4                      | n/a                     |
| HOLD      | Holding Register (16 bits) | 3                      | 6 or 16                 |

### B.4 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      Trouble Shooting

### C.1      Modbus RTU Trouble Shooting

You will find message and error counters listed on the Error Counts page under RTU Data. If the ValuPoint is configured as Modbus master, then the Error Counts page will list counts by slave address. If the ValuPoint is configured as Modbus slave, then errors show up on the first line (Unit # 1) regardless of what address the ValuPoint is configured to be.

The Errors: Read Maps and Errors: Write Maps pages will tell you exactly which maps are getting errors when the ValuPoint is configured as Modbus Master.

The most frequent problem is "no response" or timeout. This means the master and slave are not connecting for any of several possible reasons: (a) There is a wiring problem; (b) Port parameters are not configured the same (baud rate, etc); (c) Master's timeout setting is too short.

When it comes to wiring, remember that RS-485 is NOT truly a 2-wire interface as it is commonly referred to. Refer to the RS-485 FAQ under Support at csimn.com if you have questions or concerns about wiring.

If you are getting CRC errors, that is almost always a wiring problem, but can be a port problem such as mismatched parity setting. A CRC error will not be caused by incorrect configuration of a Read Map or Write Map.

If you are getting exception errors, that is somewhat good news - it means that at least you are successfully communicating. An exception error most often means the master is asking the slave for a register that the slave does not have. If the ValuPoint is configured as Modbus master, this means the Read Map or Write Map is not configured correctly.

### C.2      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 ValuPoint 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 ValuPoint 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 ValuPoint is configured as Modbus master, this means the Read Map or Write Map is not configured correctly.

### C.3 Modbus Reference Information

#### 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 normally 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.  |

### 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 ValuPoints, 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 ValuPoints 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 ValuPoints 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 ValuPoints 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 devices, 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). ValuPoints 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. ValuPoints 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 ValuPoint 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 ValuPoint 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.

## Can I put 2 Valupoints on the same Modbus network?

You can not have more than one Master on a Modbus RTU (RS-485) network.



Therefore, if the ValuPoint is to be configured as the Master, you can only have 1 ValuPoint. You cannot use multiple ValuPoints to read more points from the same Modbus slave device.

Multiple ValuPoints configured as slaves can reside on the same Modbus RS-485 network.

If you are using RS-232 devices, you can have only two devices total, regardless of how they are configured. RS-232 is not multi-drop.

### **How many devices can I have on a Modbus RTU network?**

Logically you can address over 250 devices; however, the RS-485 transceivers are not capable of physically driving that many devices. Modbus protocol states that the limit is 32 devices, and most RS-485 transceivers will agree with this. Only if all devices on the network have low load transceivers can you have more than 32 devices.

## **C.4 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.

## **C.5 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).



## C.6 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.

The image shows a Wireshark network traffic analysis window titled "\*Ethernet". The interface includes a menu bar (File, Edit, View, Go, Capture, Analyze, Statistics, Telephony, Wireless, Tools, Help) and a toolbar with various icons. A filter bar at the top displays "ip.addr==192.168.1.125".

| No. | Time     | Source        | Destination   | Protocol | Length | Info                          |
|-----|----------|---------------|---------------|----------|--------|-------------------------------|
| 304 | 4.741960 | 192.168.1.112 | 192.168.1.125 | SNMP     | 113    | get-request                   |
| 307 | 4.744061 | 192.168.1.125 | 192.168.1.112 | SNMP     | 174    | report 1.3.6.1.6.3.15.1.1.4.0 |
| 308 | 4.744505 | 192.168.1.112 | 192.168.1.125 | SNMP     | 205    | encryptedPDU: privKey Unknown |
| 309 | 4.751867 | 192.168.1.125 | 192.168.1.112 | SNMP     | 205    | encryptedPDU: privKey Unknown |

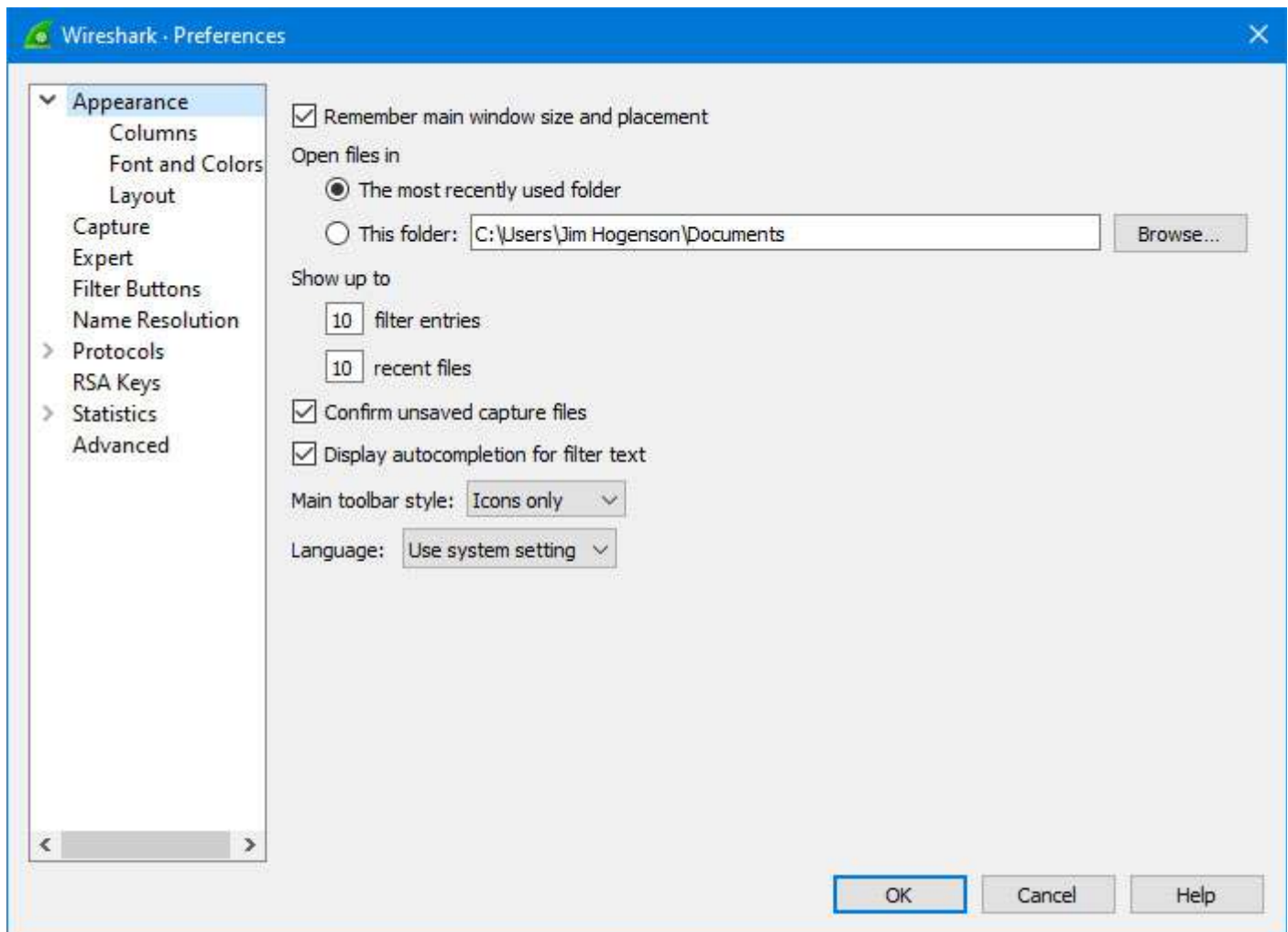
Below the packet list, the details pane for frame 304 is expanded to show the following structure:

- > Frame 304: 113 bytes on wire (904 bits), 113 bytes captured (904 bits) on interface \Device\NPF\_{193A9AF7-FE82-4902-9A83-D8CB2D41E36} ^
- > Ethernet II, Src: Dell\_a2:b9:f6 (a4:bb:6d:a2:b9:f6), Dst: Digiboar\_45:46:4e (00:40:9d:45:46:4e)
- > Internet Protocol Version 4, Src: 192.168.1.112, Dst: 192.168.1.125
- > User Datagram Protocol, Src Port: 50294, Dst Port: 161
- > Simple Network Management Protocol
  - msgVersion: snmpv3 (3)
  - > msgGlobalData
    - msgAuthoritativeEngineID: <MISSING>
    - msgAuthoritativeEngineBoots: 0
    - msgAuthoritativeEngineTime: 0
    - msgUserName: initial
    - msgAuthenticationParameters: <MISSING>
    - msgPrivacyParameters: <MISSING>

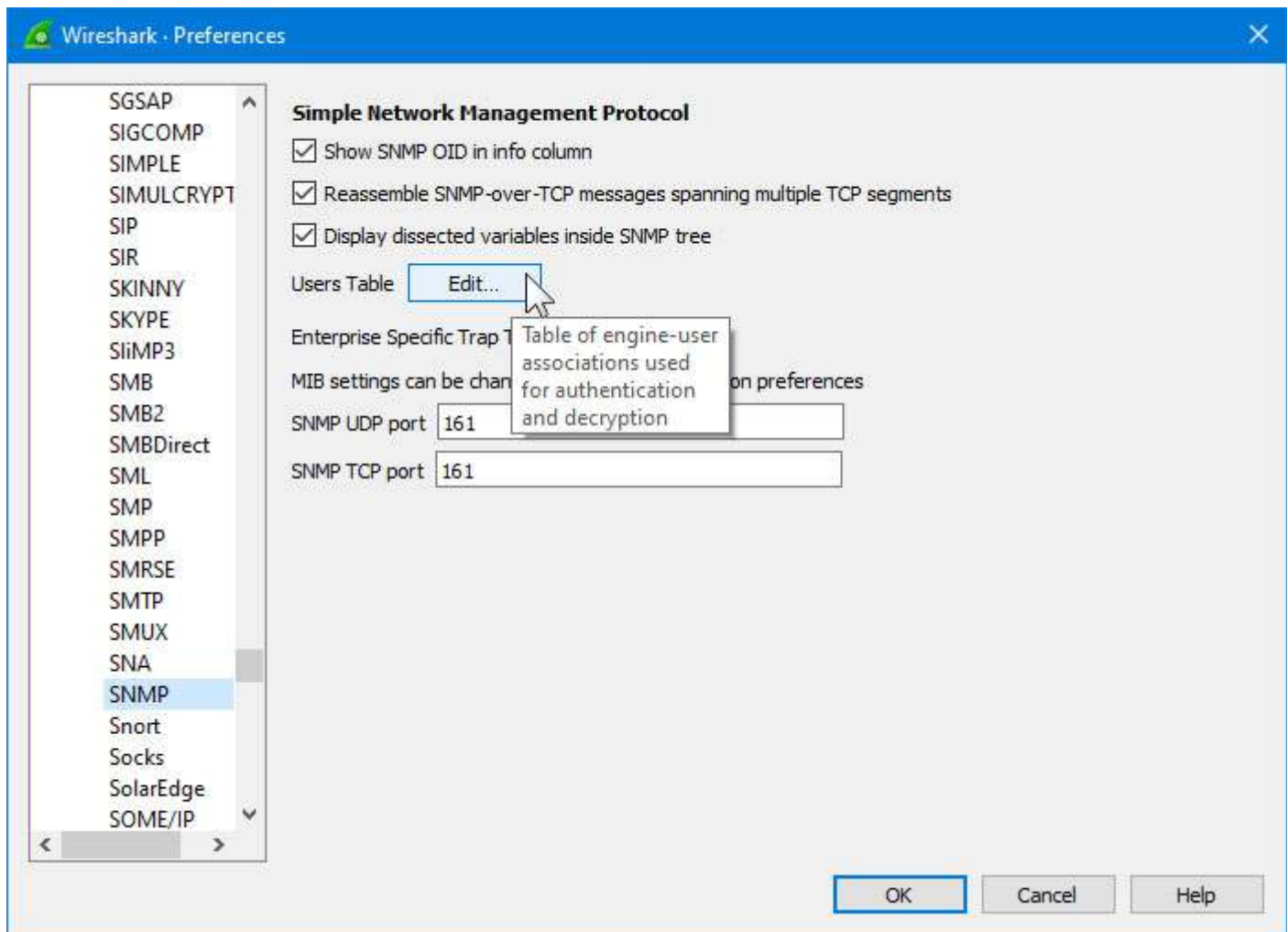
The packet bytes pane at the bottom shows the raw data in hexadecimal and ASCII. The ASCII column contains the text "@.EFN..m...E" and "c(...p" and "}.v..0..0E..0" and "...7.GG....." and "...0....." and "initial...0...." and "...7.GG.....".

The status bar at the bottom indicates "wireshark\_Ethernet\_20200819100913\_a10188.pcapng" and "Packets: 460 · Displayed: 4 (0.9%)" and "Profile: Default".

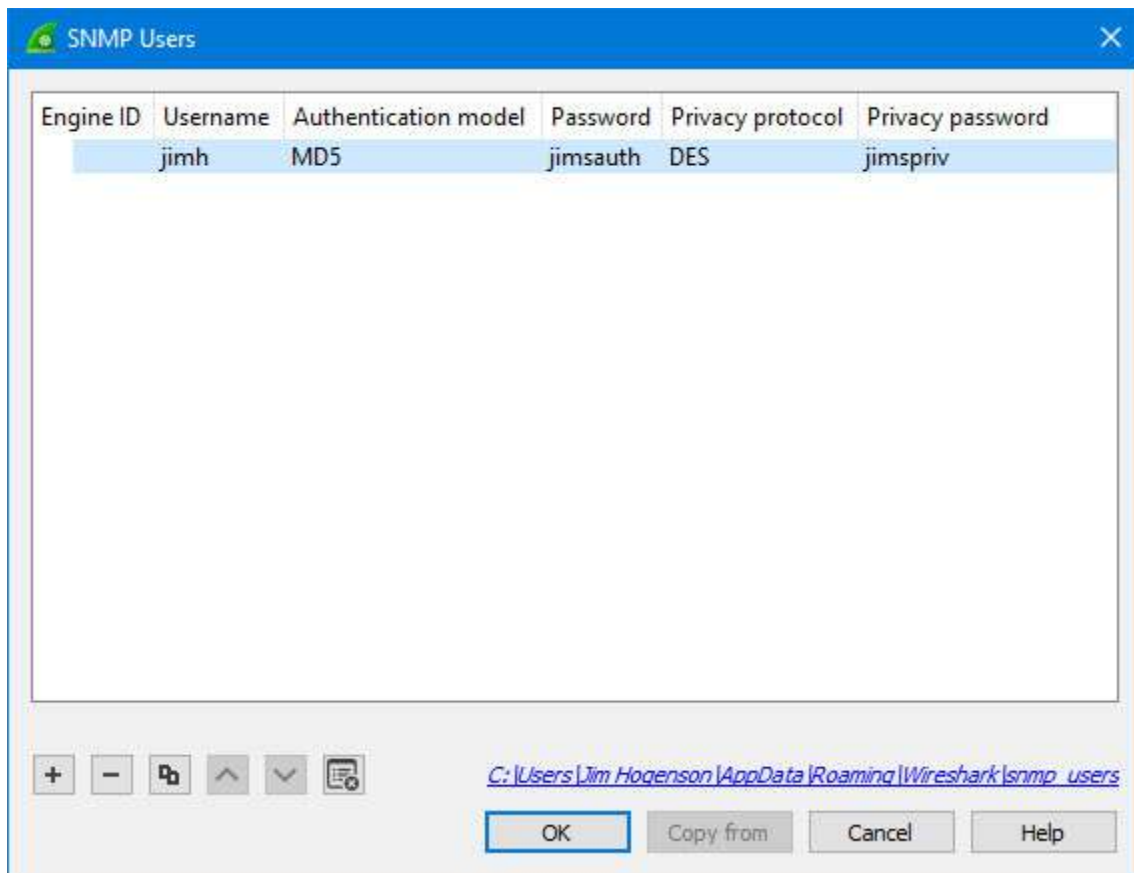
Under the Edit menu, select Preferences.



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.

The image shows a Wireshark capture of an SNMP message. The packet list pane displays the following data:

| No.  | Time       | Source        | Destination   | Protocol | Length | Info  |
|------|------------|---------------|---------------|----------|--------|---|
| 309  | 53.333473  | 192.168.1.125 | 192.168.1.112 | SNMP     | 205    | get-response 1.3.6.1.4.1.3815.1.6.1.1.1.1.2.1 |
| 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 1.3.6.1.4.1.3815.1.6.1.1.1.1.2.1  |
| 3210 | 324.685085 | 192.168.1.125 | 192.168.1.112 | SNMP     | 205    | get-response 1.3.6.1.4.1.3815.1.6.1.1.1.1.2.1 |

The packet details pane for the selected packet (No. 3210) shows the following structure:

- 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: 80000ee702fe8000000000000002409dffffe45464e
  - msgAuthoritativeEngineBoots: 9
  - msgAuthoritativeEngineTime: 6897
  - msgUserName: jimh
  - msgAuthenticationParameters: 4347f30b21c0b4b1e347fb11
  - msgPrivacyParameters: 000000900004473
  - msgData: encryptedPDU (1)
    - encryptedPDU: d4068f6f390d4106bd0e5cca5ca04707a494ae1e1c93c3c7...
      - Decrypted ScopedPDU: 3041041580000ee702fe800000000000002409dffffe4546...
        - contextEngineID: 80000ee702fe8000000000000002409dffffe45464e
        - contextName:
        - data: get-response (2)
          - get-response
            - request-id: 931153739
            - error-status: noError (0)
            - error-index: 0
            - variable-bindings: 1 item
              - 1.3.6.1.4.1.3815.1.6.1.1.1.1.2.1: 110500
                - Object Name: 1.3.6.1.4.1.3815.1.6.1.1.1.1.2.1 (iso.3.6.1.4.1.3815.1.6.1.1.1.1.2.1)
                - Value (Integer32): 110500

The packet bytes pane shows the raw hex and ASCII data for the selected packet:

```

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...7.GK .....
  
```

At the bottom of the interface, the status bar shows: Frame (205 bytes) | Decrypted ScopedPDU (72 bytes) | wireshark\_Ethernet\_20200819100913\_a10188.pcapng | Packets: 3463 · Displayed: 8 (0.2%) | Profile: Default



## Appendix D      SSL Certificates for Secure Web (HTTPS)

The secure web server (HTTPS) requires SSL certificates in order to establish secure connections. These certificates are for the use of the web server. The HTTPS certificates are only required if HTTPS is enabled on the Network configuration page in the ValuPoint.

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

The ValuPoint 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.

A screenshot of a web configuration interface with a dark teal background. At the top, it shows 'Web Server' with a checked checkbox, 'HTTPS Enabled (on 443)' with a checked checkbox, and 'HTTP Enabled' with a checked checkbox. Below this are input fields for 'HTTP Port' (80), 'Modbus Port' (502), and 'MIB Offset' (0). A 'Set Ports' button is to the right of the HTTP Port field. Below the input fields is 'FTP Server' with a checked checkbox and the text 'Enabled'. At the bottom, it shows 'MAC Address: 00:40:9D:DC:0D:DD' and 'System Uptime: 0,00:55:34'. 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.



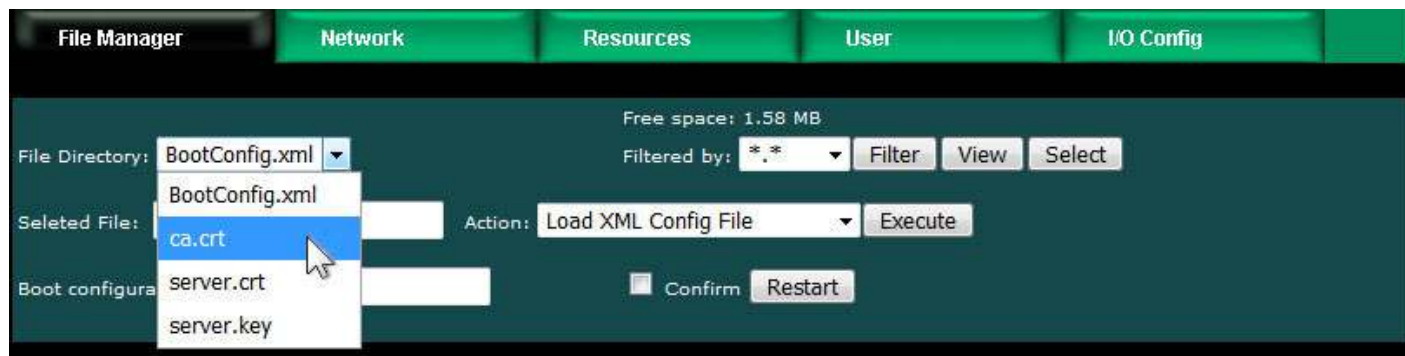
```

C:\Users\Jim Hogenson\My Documents\bb-mq-61\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>

```

## D.2 External Certificates

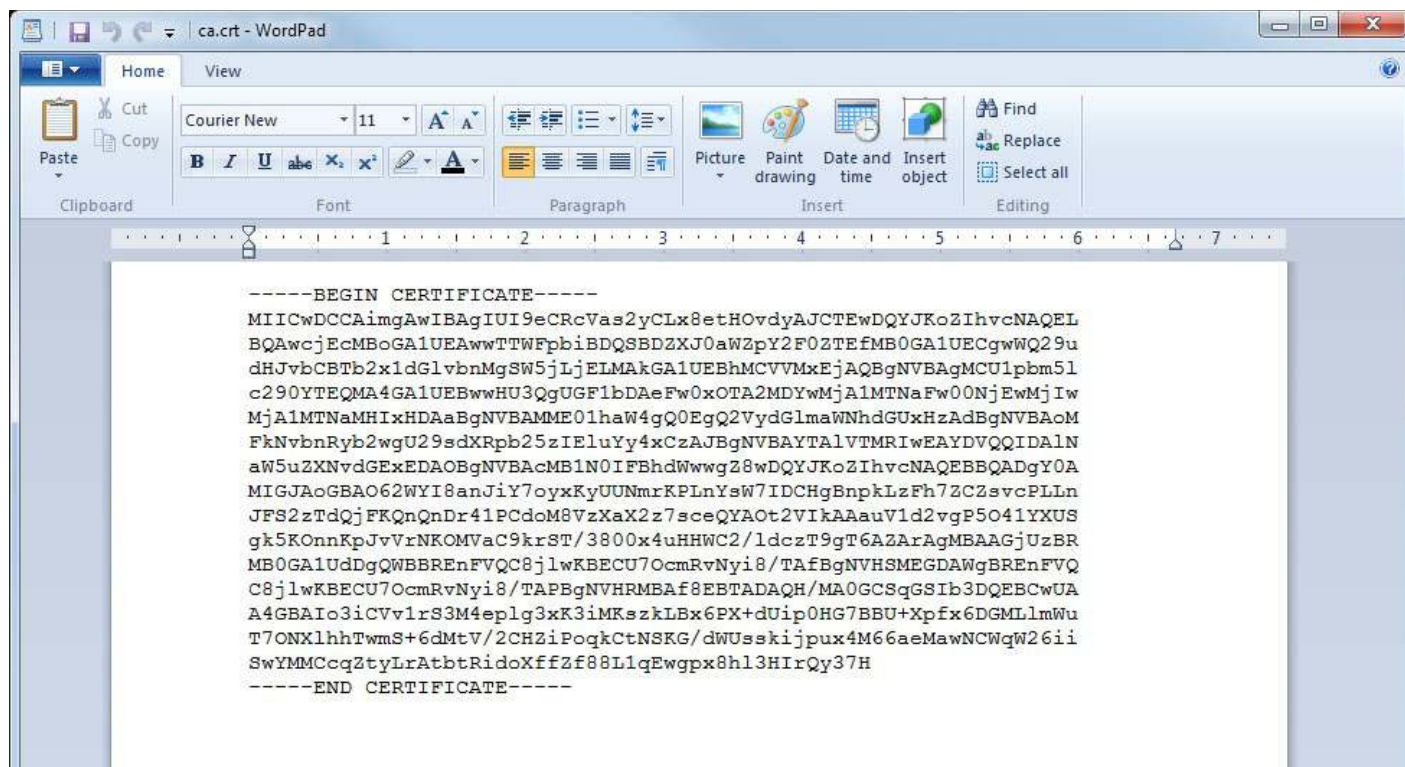
There are three certificates that you must generate and upload to use SSL certificates other than the self-generated X.509 certificates.



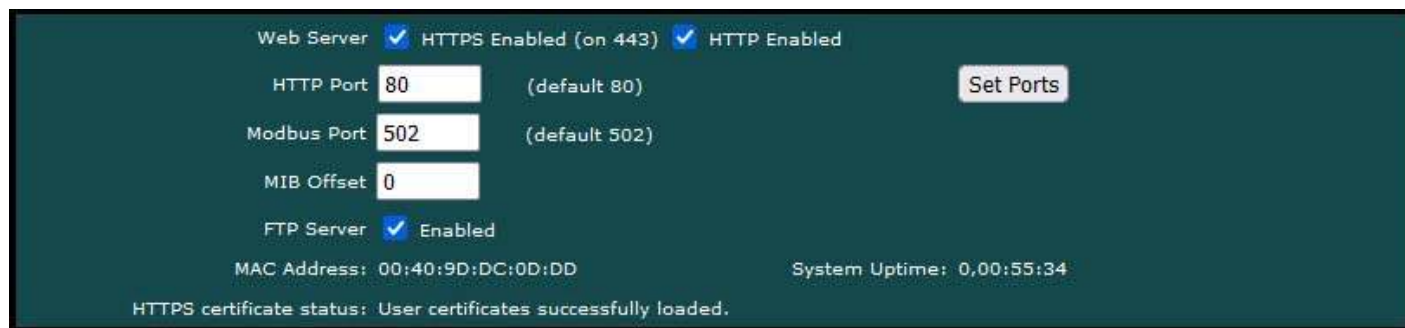
The required certificates are as follows, and must use exactly these names.

|            |                                   |
|------------|-----------------------------------|
| 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).



If external certificates were loaded successfully, that will be indicated at the bottom of the Network configuration page.



### D.3 Certificate Generation Script (Linux)

The art and science of generating SSL certificates is beyond the scope of this document. An example SSL certificate generation script is provided here as a reference.

The following script, run on a Linux system with OpenSSL installed, will generate the three required SSL certificate files. It will generate a number of intermediate files as well - you don't need to upload them. Replace references to Control Solutions in this script with your own company name.

```
#!/bin/bash
echo hello
# This will create some self signed certs, using one master CA.
#
# these can be the webserver DNS name, or an IP address, however you
access
```

```

# the resource, this needs to match.
if [ -z "$1" ] || [ -z "$2" ]; then
echo 'Usage: gen.sh <server-name> <client-name>'
echo ' <server-name> and <client-name> can be IP addresses'
echo ' or DNS names.'
exit 1
fi
SNAME=$1
CNAME=$2
#
# Bits for strength, 1024, 2048, 4096, etc.. (suggest 2k or 4k for web
servers)
BITS=1024
#
# HASH - Options are sha256, sha512, sha1, md5
HASH="sha256"
SN=`date +%Y%m%d%H%M%S`
#####
# below is the entry for the CRL
# Do not use http://www.csimn.com/crl.pem for production keys and
certificates
# cat <<EOF >> extensions.cnf
# [ extensions_section ]
# crlDistributionPoints = URI:http://www.csimn.com/crl.pem
#
# basicConstraints = CA:FALSE
# keyUsage = nonRepudiation, digitalSignature, keyEncipherment
# subjectAltName = DNS:${SNAME},IP:${SNAME}
# EOF
#####
#####
# first, lets generate some private keys...
openssl genrsa -out server.key ${BITS}
openssl genrsa -out client.key ${BITS}
# ok, and now the MAIN CA
openssl req -x509 -${HASH} -nodes -days 10000 -newkey rsa:${BITS} -keyout
ca.key -out ca.crt -subj "/CN=Main CA Certificate/O=Control Solutions
Inc./C=US/ST=Minnesota/L=St Paul"
#####
#
# Create a CSR for both server and client
# Replace these values with one appropriate for your organization
openssl req -out server.csr -key server.key -new -subj "/CN=${SNAME}/
O=Control Solutions Inc./C=US/ST=Minnesota/L=St Paul"
openssl req -out client.csr -key client.key -new -subj "/CN=${CNAME}/
O=Control Solutions Inc./C=US/ST=Minnesota/L=St Paul"
#
#
#####
# Sign the keys with the CA
openssl x509 -req -days 3650 -in server.csr -CA ca.crt -CAkey ca.key -
set_serial ${SN}01 -out server.crt -${HASH}
openssl x509 -req -days 3650 -in client.csr -CA ca.crt -CAkey ca.key -

```

```
set_serial ${SN}02 -out client.crt -${HASH}
# Create a windows file to import the client keys if needed in this
format
openssl pkcs12 -export -clcerts -in client.crt -inkey client.key -out
client.p12
# Create the client keys as a complete pem file if needed in this format
openssl pkcs12 -in client.p12 -out client-full.pem -clcerts
# mv -f server.key svrkey.pem
# mv -f server.crt svrcert.pem
# mv -f client.key clntkey.pem
# mv -f client.crt clntcert.pem
# cp -f ca.crt cacert.pem
###
# cleanup
# rm -f client.csr server.csr
#DLS 20160420
echo '*****'
echo '* WARNING: Do not use this script to generate production *'
echo '* keys and certificates. This script is for *'
echo '* demonstration purposes only. *'
echo '*****'
```