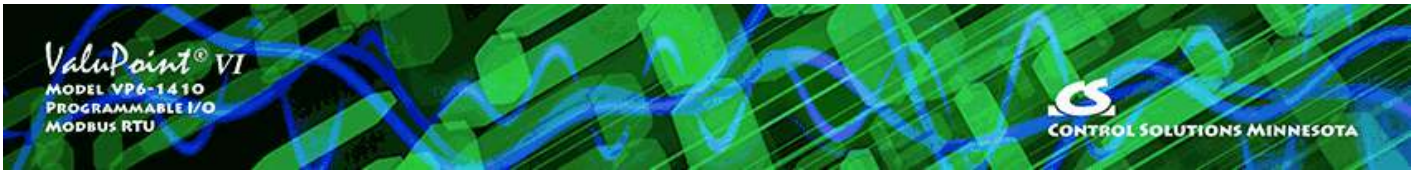



# User Guide

**Model VP6-1410**
**ValuPoint VI**
**Programmable I/O**
**for Modbus RTU**

**Rev. 1.0 – Dec. 2023**

**VP6-1410 User Guide Contents**




# User Guide

# ValuPoint® VI


**Model VP6-1410**
**Programmable I/O for Modbus RTU Networks**
**Rev. 1.0 – Dec. 2023**



# 1 Overview

## 1.1 How to Use This Guide

Section 1 gives an overview of the VP6-1410 programmable I/O device. Section 2 talks about installing the configuration software and connecting the VP6-1410. Sections 3 through 11 are guides for each of the tabs found on the screen of the configuration software. Appendix A through D are reference material.

## 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 Overview of the VP6-1410

Control Solutions' Model VP6-1410 ValuPoint® Programmable I/O for Modbus RTU operates as a master or slave for I/O expansion, or as a controller with soft PLC capability. This 5th generation ValuPoint® platform features 12 universal context sensitive inputs. When configured as an analog input, the A/D converter produces 12-bit resolution. When configured as a discrete or pulse input, inputs are treated as state inputs. The VP6-1410 also includes 2 Form A relay outputs. High speed pulse counting capability is provided on 4 inputs.

### Hardware Features of Model VP6-1410

- 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 RTU at 1200 to 115200 baud
- Isolated RS-485 port
- 512KB non-volatile EEPROM configuration file capacity
- 64K Flash for User Program
- ARM 32-bit processor, 1MB Flash, 256K RAM
- Powered by 18-30VDC or 24VAC 50/60 Hz Class 2, 0.3A max.
- DIN rail mounting, 100mm H x 105mm W x 60mm D
- -40C to +80C, 5%-95% RH non-condensing
- Certifications: FCC, CE, UL 916 Listed

The VP6-1410 is configured by writing various special Modbus registers accessible as holding registers. The VP6-1410 configuration tool simplifies this process by providing a graphical tool for automatically writing all of the necessary registers by selecting various options on the screen and then clicking the 'Write' button. Configuration is saved in non-volatile memory. Once configured, you can save your entire configuration to an XML format file on your PC for later re-use. To replicate the same configuration, simply load the XML file back into the configuration tool and click the 'Write All' button.

## 1.4 Warranty

**This configuration software and 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 or in the product(s) and/or the program(s) described in this documentation at any time. This product could include software bugs, technical inaccuracies, typographical errors, and the like. Changes are periodically made to the information herein; these changes may be incorporated in new editions of the software.

**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.

## 1.5 Required License Information

The VP6-1410 configuration and line programming tools include the SmartWin library (http://smartwinlib.org) under the following terms:

License agreement for SmartWin++ (BSD license)

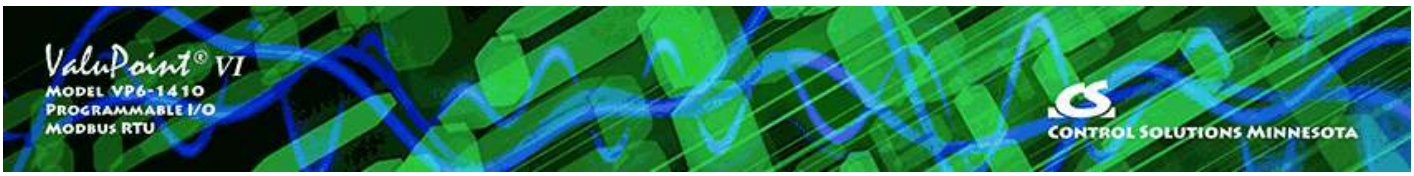
Copyright (c) 2005, Thomas Hansen All rights reserved.
Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
* Redistributions in binary form must reproduce the above copyright notice, this list of

conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
* Neither the name of the SmartWin++ nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

i.CanDrawIt includes, licensed under LGPL, TinyCAD, Copyright 1994-2009 Matt Pyne. Source code is available at http://tinycad.sourceforge.net. Open source products included under either GPL or LGPL include TinyCAD v2.70; Unicode/Font Conversions: iconv.dll version 1.9.0.0; PNG Image Support: libpng13.dll version 1.2.8.0; Image compression support: zlib1.dll version 1.2.1.0.




# 2 Installation and Connections

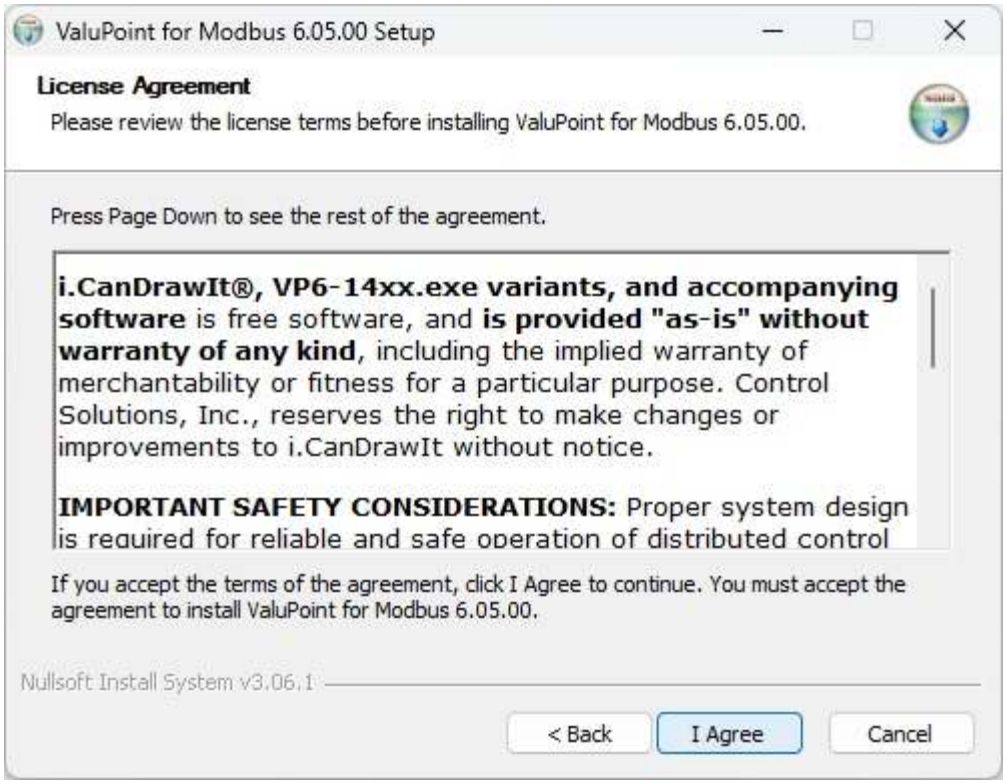
## 2.1 Installing the configuration software

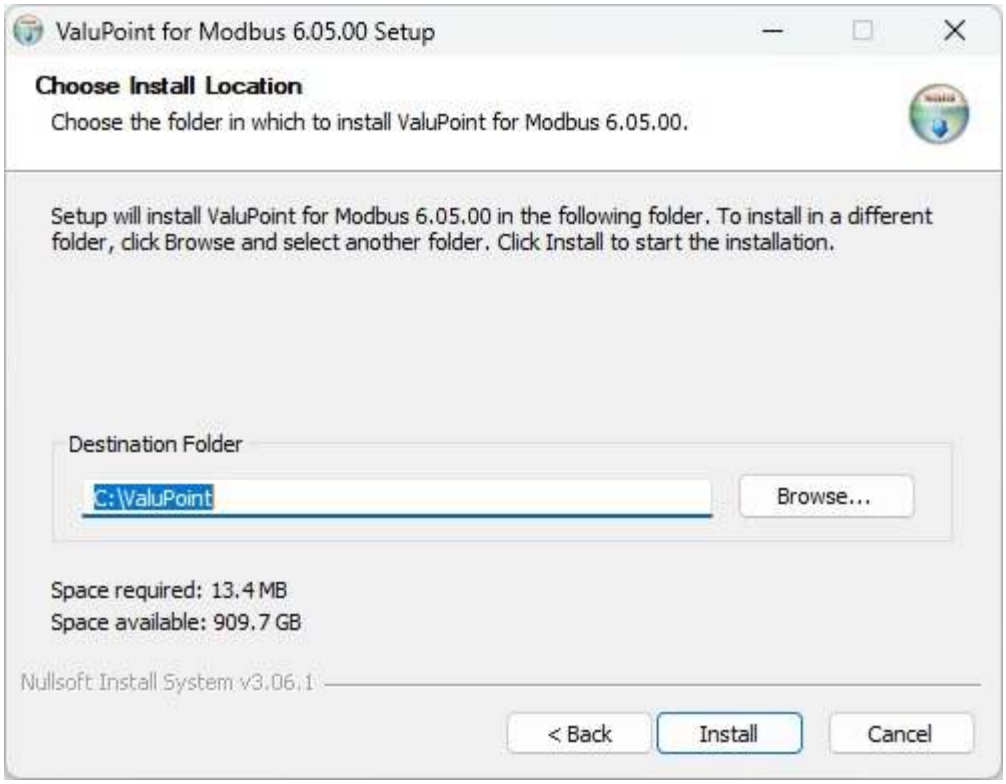
Look for the installer icons in the directory where you unzipped the download that got you to this document. The installer icons look like this:

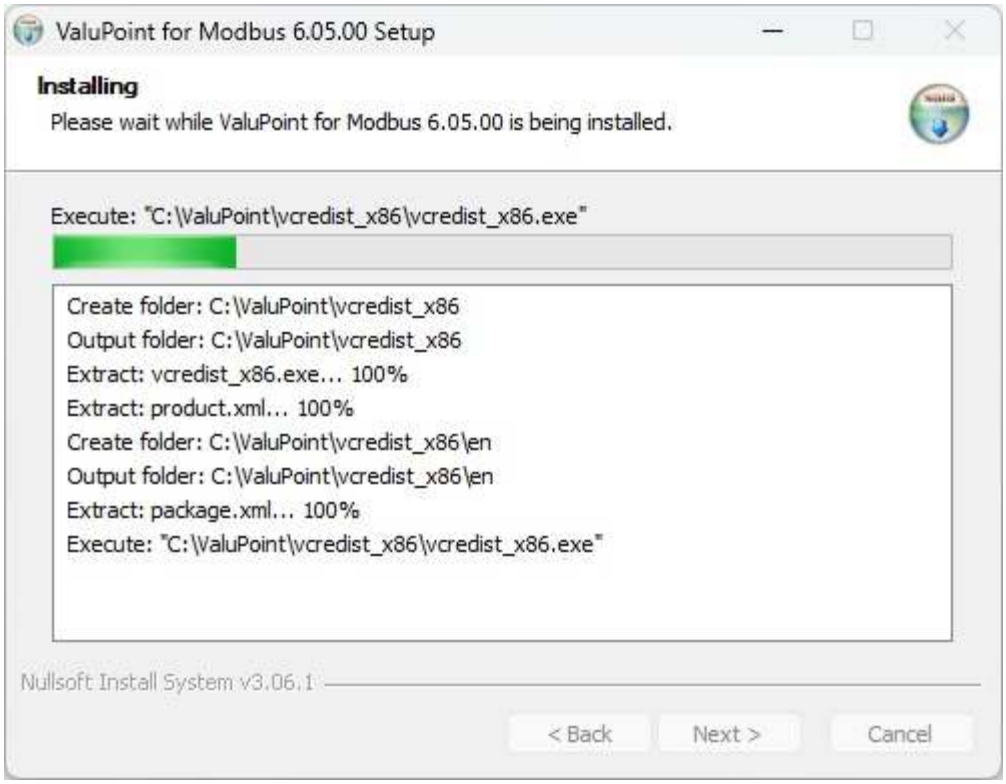

The installation is a 2-step installation. Install VP6-1410 first. Then install i.CanDrawIt second.

Double click the icon to run the VP6-1410_setup.exe. You will be questioned about whether to continue because Windows cannot verify the publisher of the software. Permit installation to continue. The sequence of installer screens include the following on Windows 11:

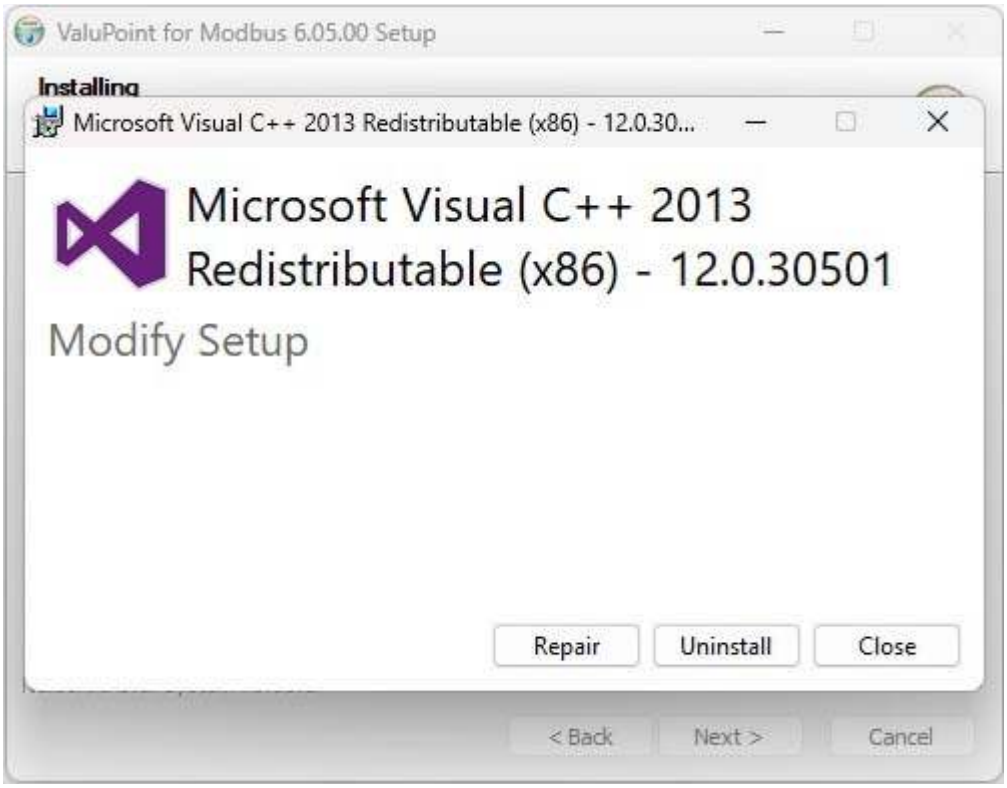

ValuPoint for Modbus 6.05.00 Setup
Welcome to ValuPoint for Modbus 6.05.00 Setup
Setup will guide you through the installation of ValuPoint for Modbus 6.05.00.
It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.
Click Next to continue.
Next >
Cancel


ValuPoint for Modbus 6.05.00 Setup
License Agreement
Please review the license terms before installing ValuPoint for Modbus 6.05.00.
Press Page Down to see the rest of the agreement.
i.CanDrawIt®, VP6-14xx.exe variants, and accompanying software is free software, and is provided "as-is" without warranty of any kind, including the implied warranty of merchantability or fitness for a particular purpose. Control Solutions, Inc., reserves the right to make changes or improvements to i.CanDrawIt without notice.
IMPORTANT SAFETY CONSIDERATIONS: Proper system design is required for reliable and safe operation of distributed control
If you accept the terms of the agreement, click I Agree to continue. You must accept the agreement to install ValuPoint for Modbus 6.05.00.
Nullsoft Install System v3.06.1
< Back
I Agree
Cancel

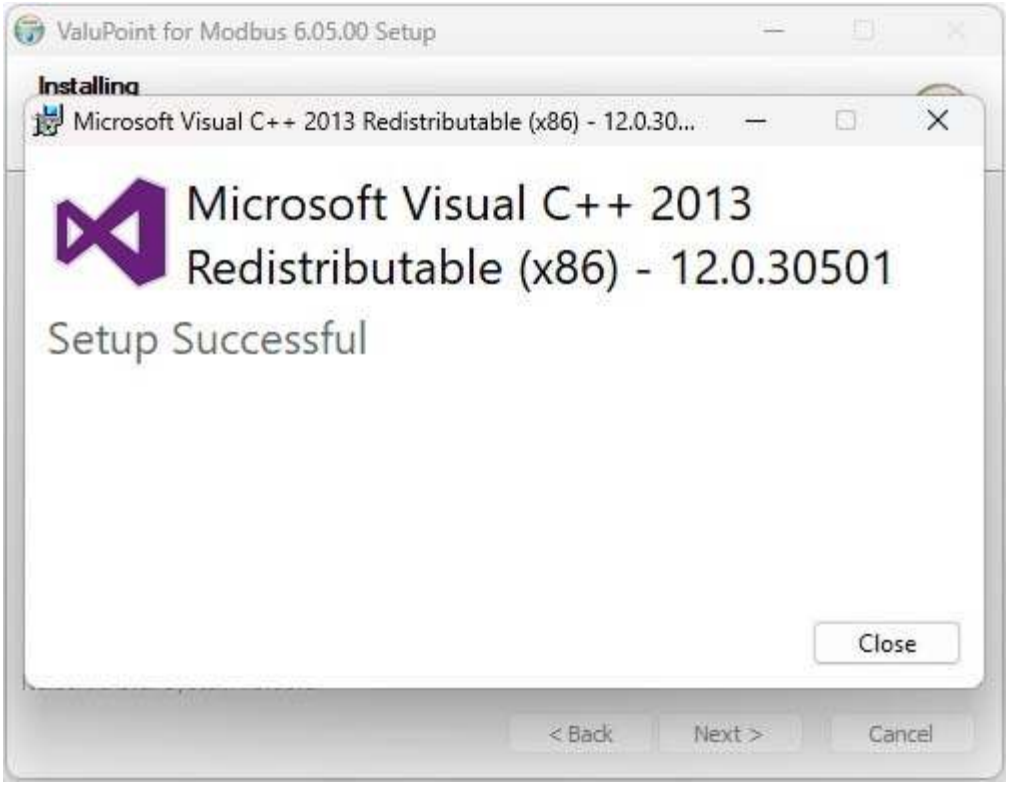


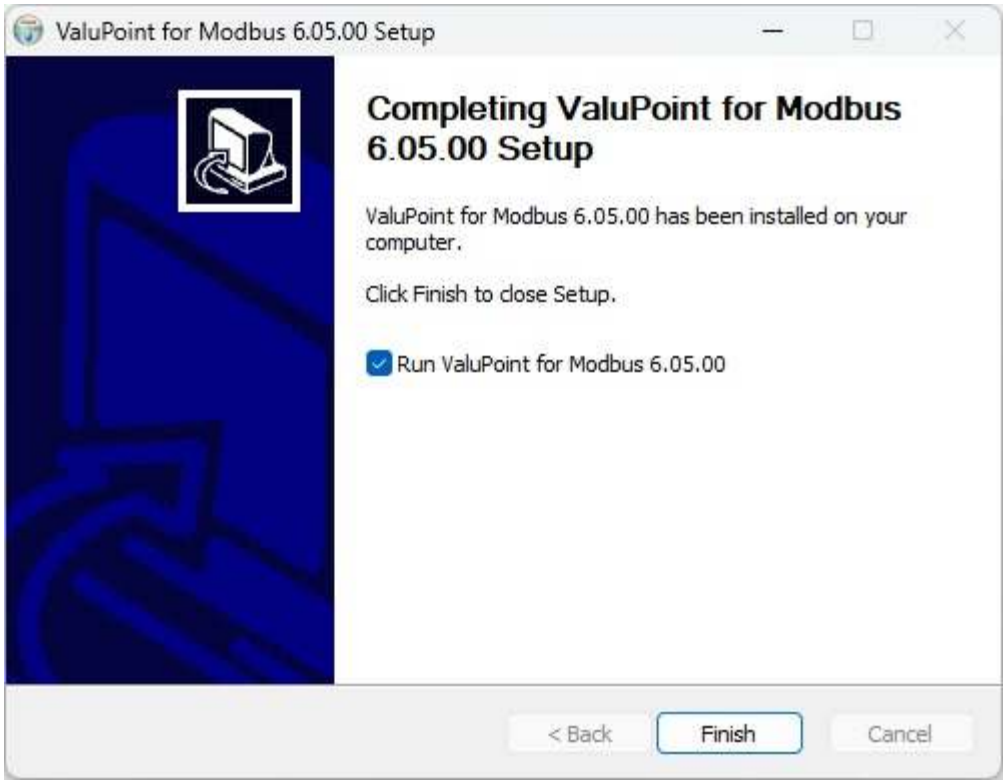

The installer will check to see whether Visual C++ support is already installed on your system, and install it if not. This is standard software provided by Microsoft. If it is already there, it will give you the option to "Repair". Select Repair (which in most cases will not really do anything other than verify the installation).



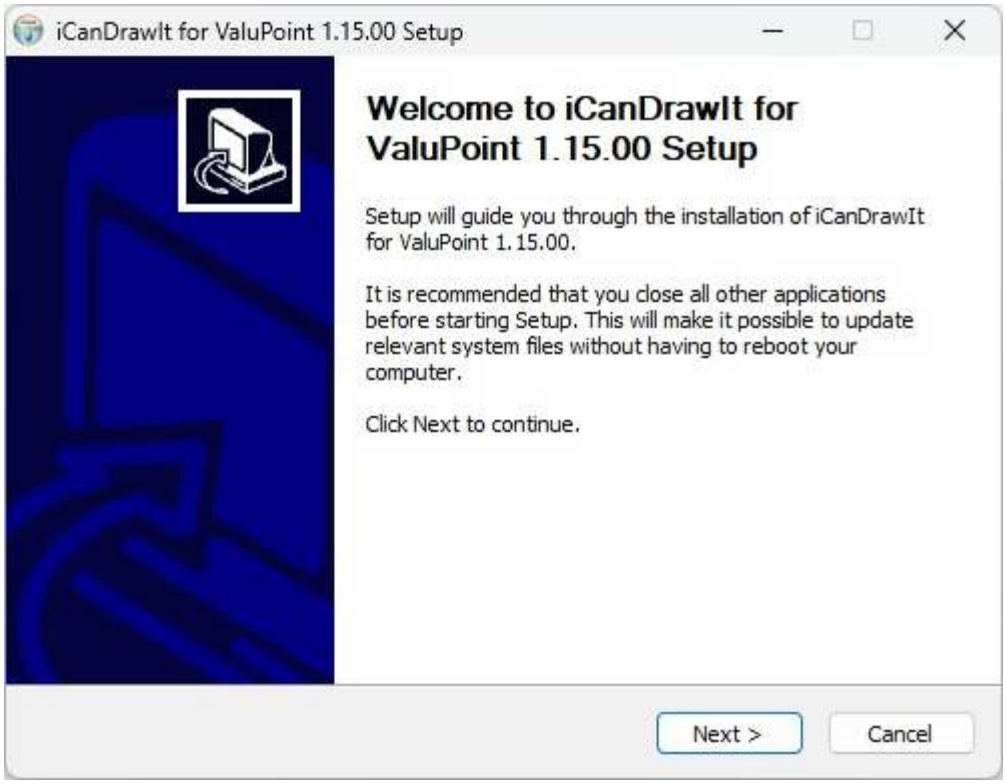

Click Close to continue the installation.

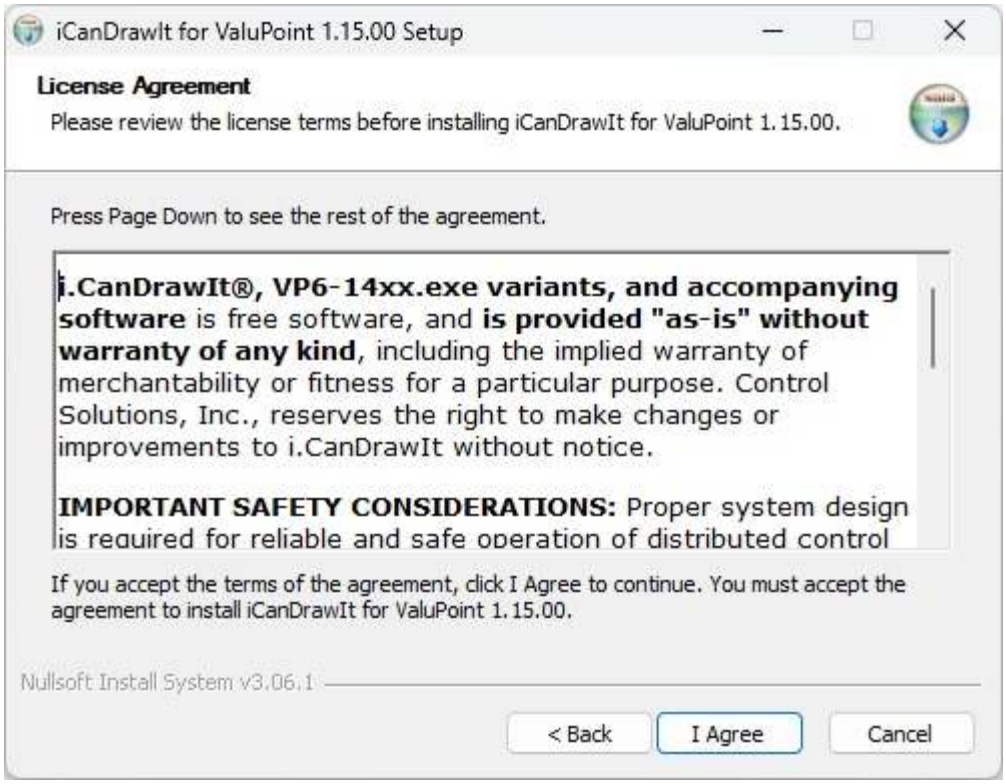



When you get to the "Finish" screen, you are ready to go.

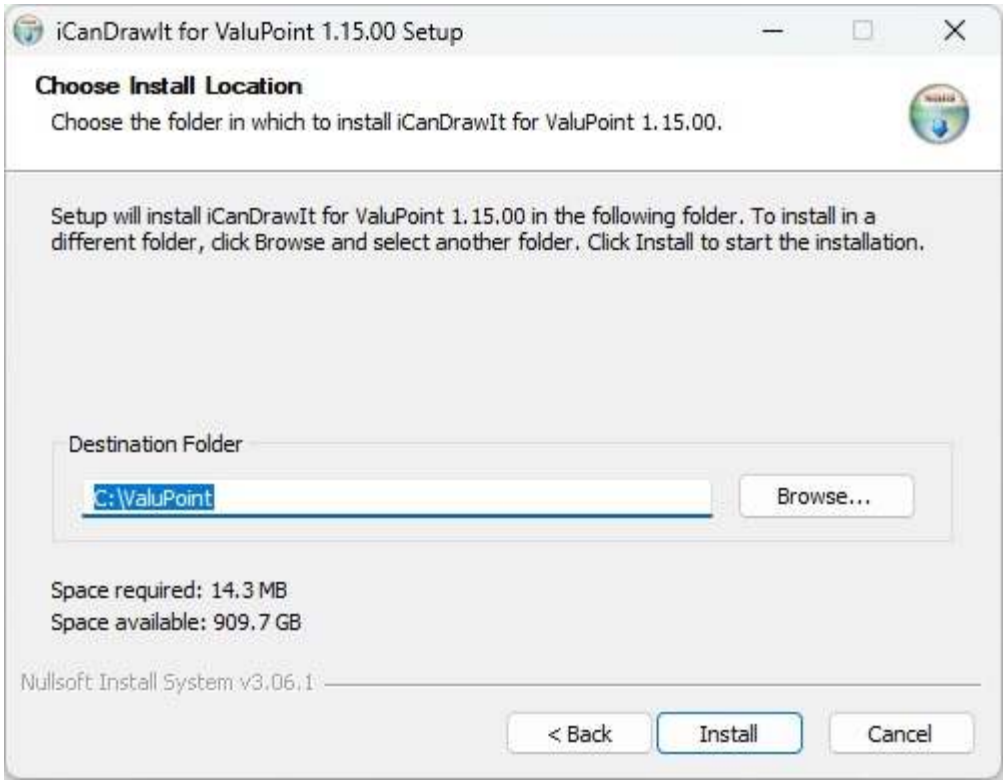



Next, proceed to install i.CanDrawIt. This part is optional. If you will not be using the VP6-1410 as a programmable controller, you can skip this step. The first installer screens look like this:

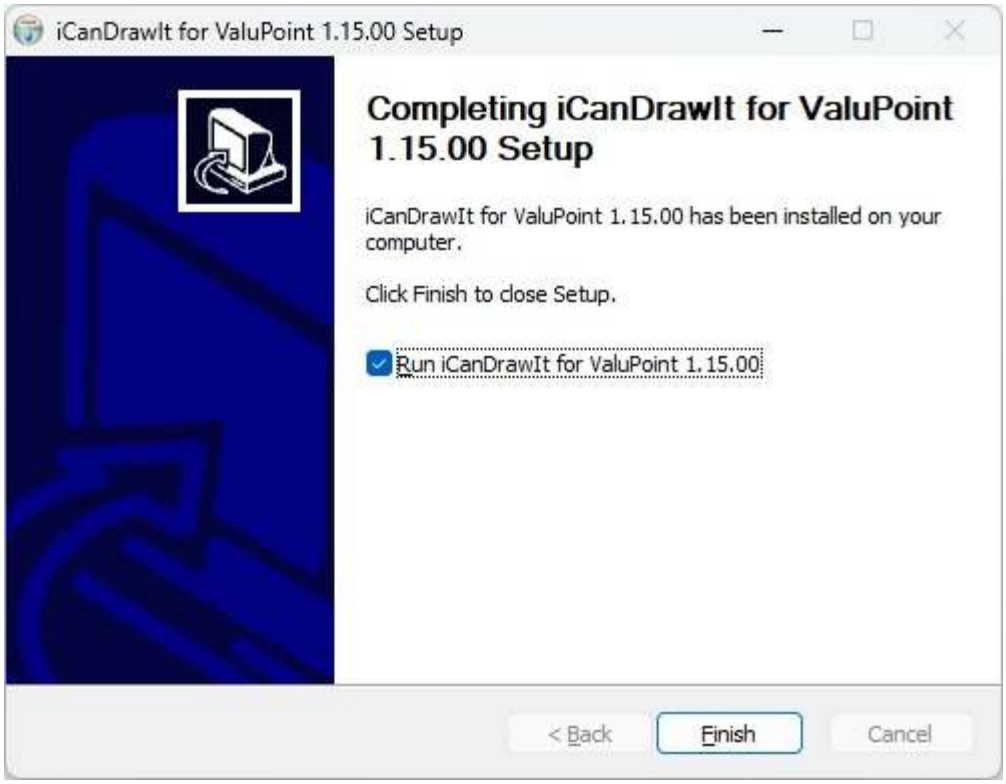





The installation directory should be the same directory that VP6-1410 was installed into.



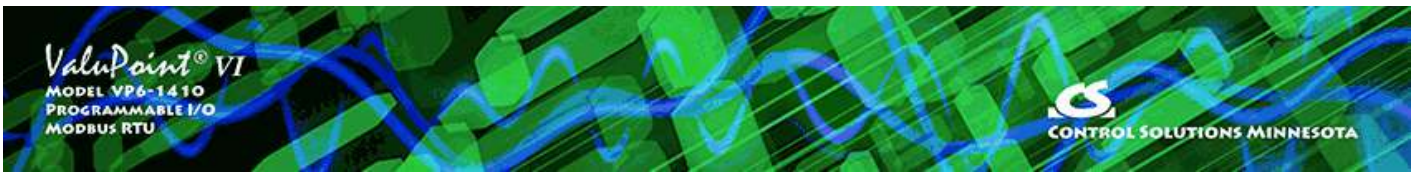

After a few more screens that look much like the VP6-1410 screens above, you will get the familiar 'done' screen.




## 2.2 Serial Port Connection

The configuration and programming tools communicate with the ValuPoint using Modbus RTU via any COM port on your PC, with an RS232 to RS485 adapter (unless you have a native RS485 COM port on your PC, which is unlikely). You do not need to install any special drivers to use an RS485 adapter on your COM port. The supercom.dll that gets installed with your tools takes care of connecting the tools to your COM port.

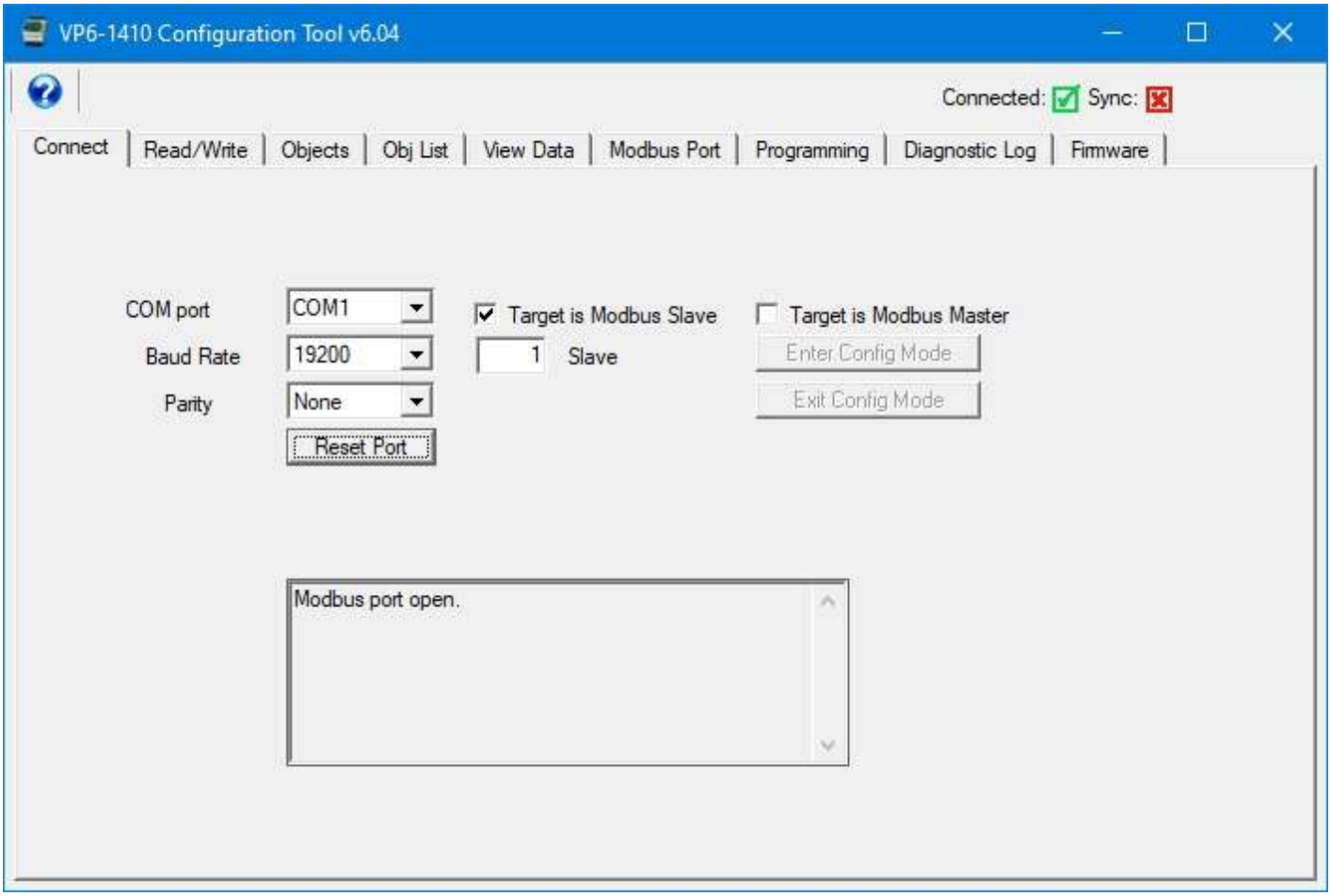
Connect power to the terminals marked POWER and GND. Connect the RS485 network to NET+, NET-, and SG. The serial communication signal ground marked SG is NOT electrically common to any other GND terminal on the device, and therefore must be connected to the network signal ground in order to complete the communication connection.




# 3 Connect

## 3.1 Connect to Target Running as Slave

To connect to a device configured as Modbus slave, simply select COM port, baud rate, parity, and slave address and enter these on the Connect page of the configuration tool. You may click Reset Port to open the COM port. If you don't do this, the port will be automatically opened the first time you try to communicate with the device. Simply resetting the port does not communicate with the device, therefore when the Connected indicator in the upper right corner changes to a green check mark, this only means the COM port was successfully opened. To test actual communications with the device, go to the Read/Write tab and simply test reading a single holding register.



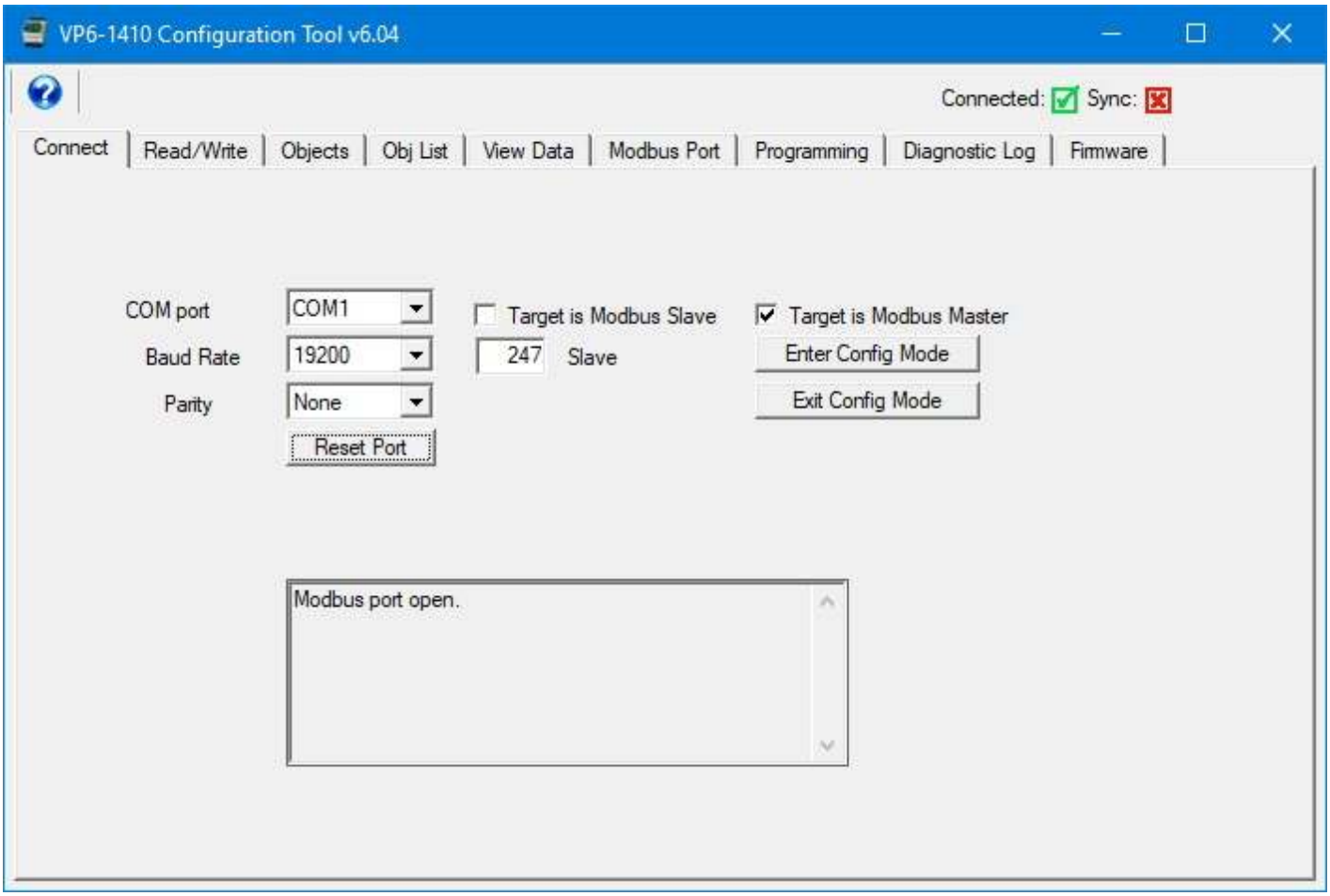

## 3.2 Connect to Target Running as Master

Connecting to a device configured to be Modbus master is a little tricky because the

device is already a master, but the configuration tool also wants to be master. You can never have more than one master on a Modbus RTU (RS485) link. To begin the process of getting the device's attention if it is running as master, check the 'Target is Modbus Master' box. The Enter/Exit Config Mode buttons will now come alive.

Click 'Enter Config Mode'. During configuration mode, the ValuPoint normally acting as master will become a slave at address 247. When you click the 'Enter Config Mode' button, the first thing that happens is that the tool will wait for the ValuPoint to query the tool for the mode change. The ValuPoint, when running as master, will query a register at slave address 247 every few seconds. If the configuration tool responds to that request with the appropriate configuration mode code, then the ValuPoint will 'switch gears' and become a slave temporarily. Once the ValuPoint has switched to slave mode, you can proceed to configure it. You cannot configure a ValuPoint Modbus device that is not functioning as a slave at least temporarily. To end the temporary slave operation, click 'Exit Config Mode'.

You will know you are in configuration mode when the LEDs stop indicating outgoing Modbus traffic and begin behaving as they normally would in slave mode.

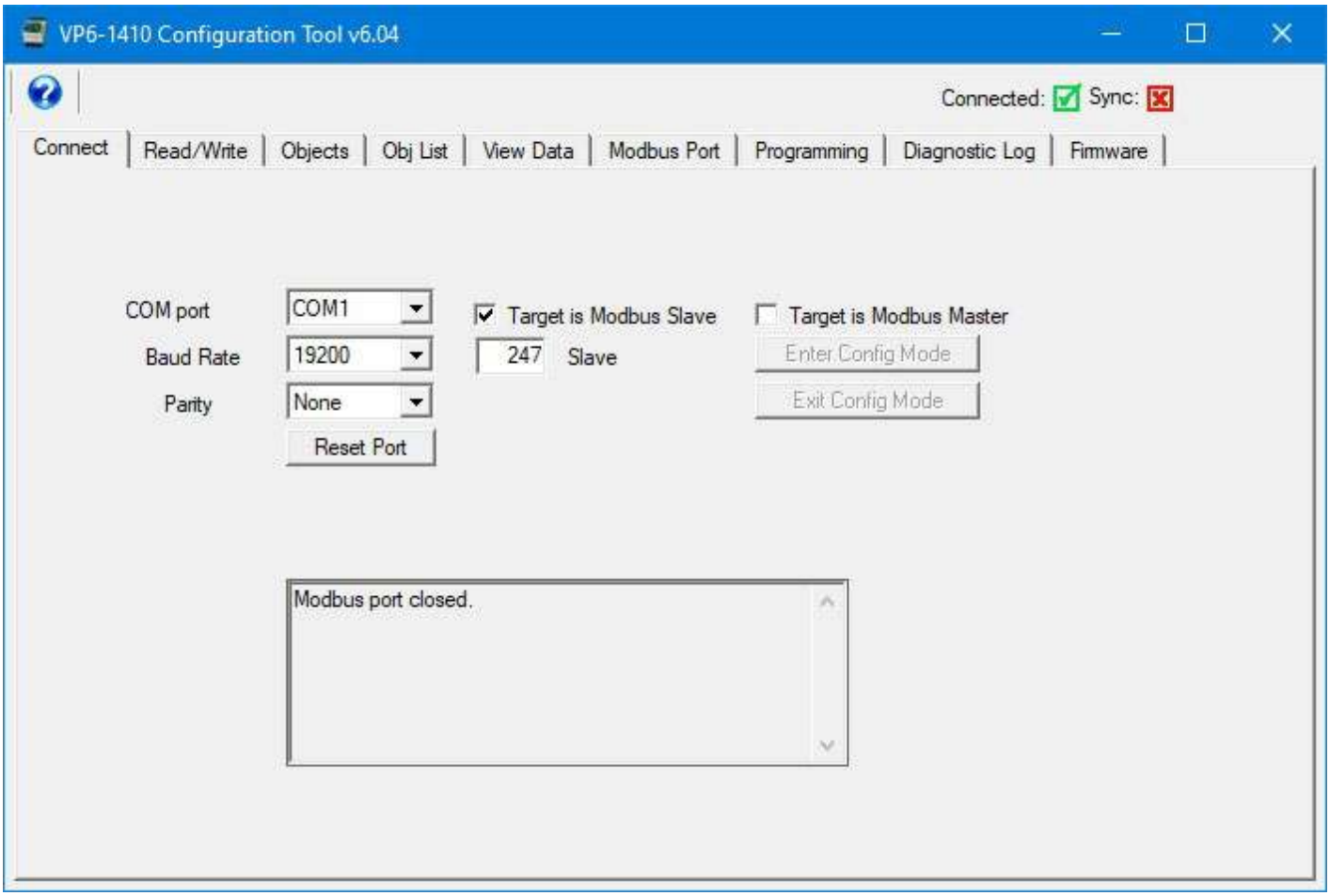



**IMPORTANT:** Do not change device mode to Modbus master until AFTER you have created some Modbus register mappings for the master to act upon. These are set up on the Objects page. If you switch to master mode without anything for the master to do, you will have trouble entering configuration mode. See Modbus Port page for more about this.

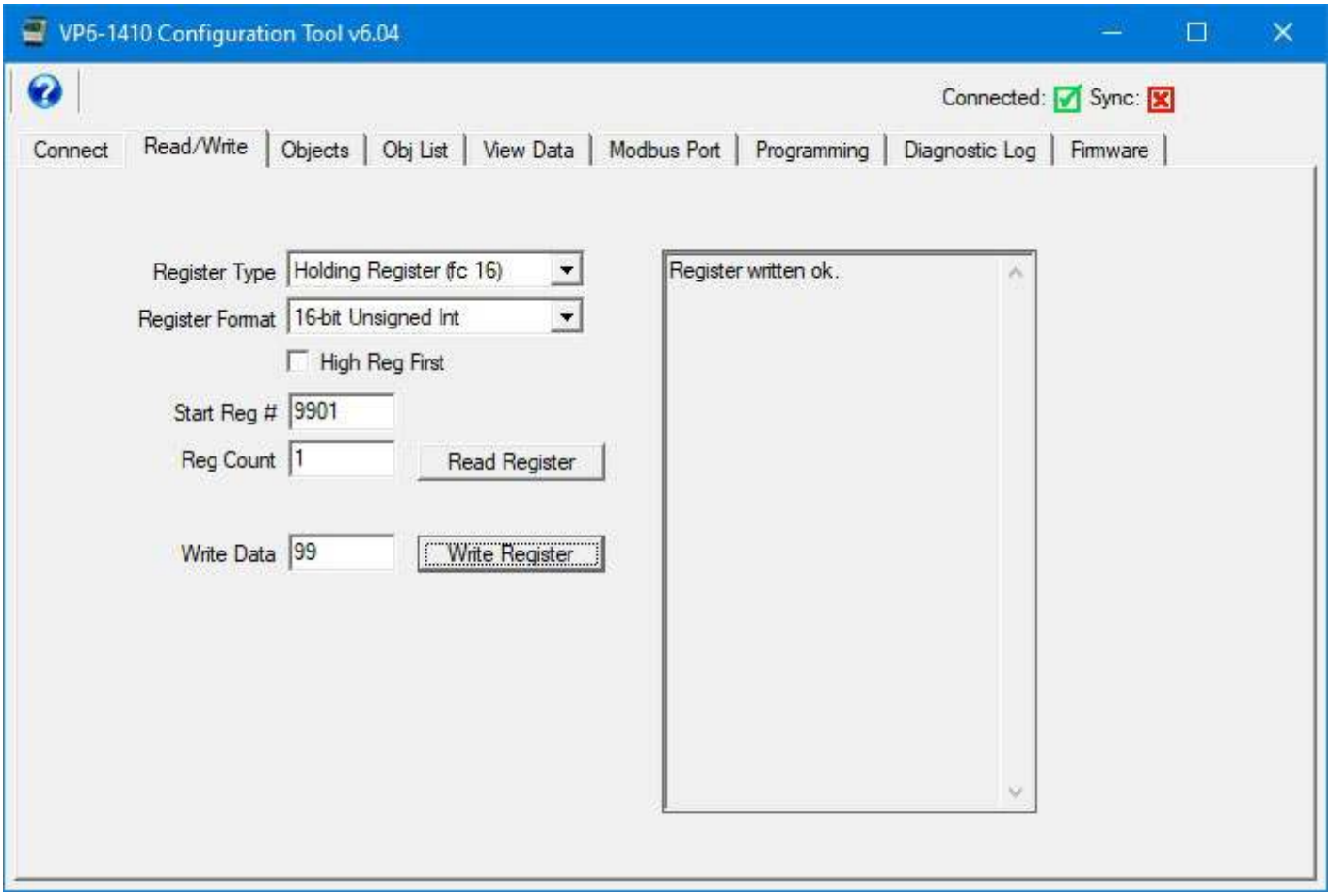
## 3.3 Recovery Mode

There is a brief window of opportunity to recover the communication port if you have lost track of baud rate, slave address, etc. During the first 3 seconds after power up, the VP6-1410 will be a Modbus slave set to 19200 baud, N81, slave address 247. If you then write the value 99 to holding register 9901, the VP6-1410 will remain in slave mode at that port setting so that you can go to the Modbus Port page to read what the current settings are (and change them if need be).

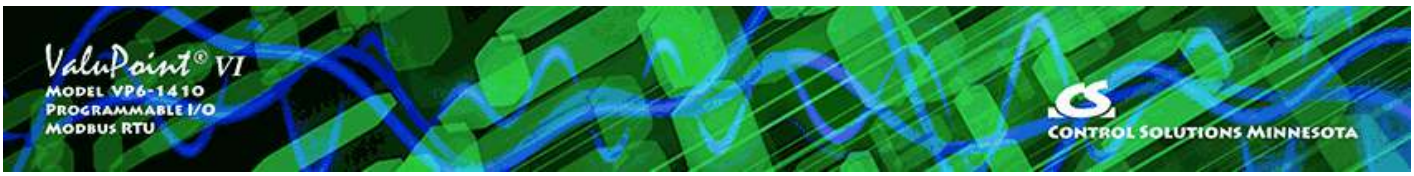
To prepare for recovery mode, set up the Connect page as follows.




Then prepare the Read/Write page as follows.

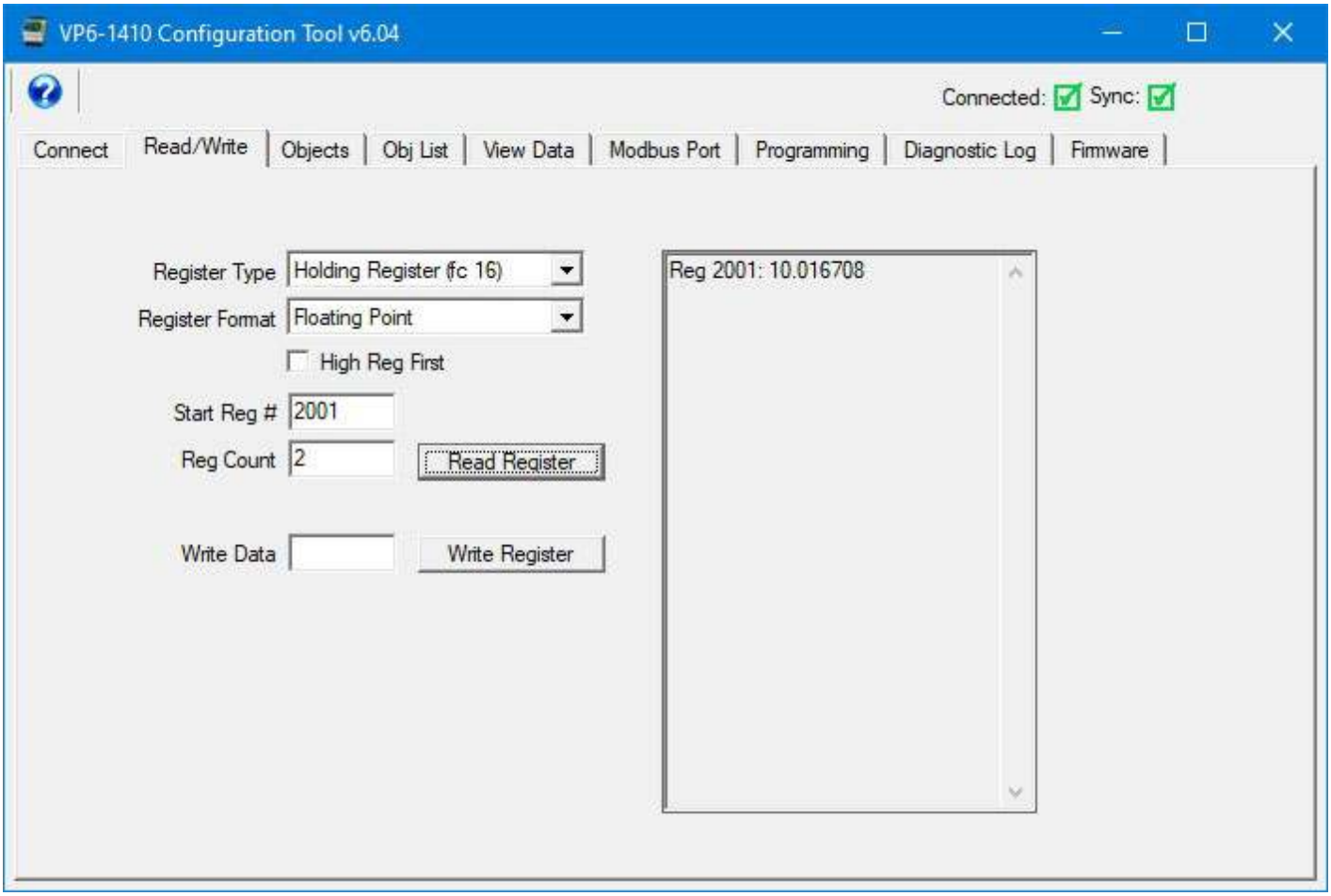



Now power up the VP6-1410 and click Write Register, and repeat until "Register written ok" appears. If more than a few seconds have passed without success, you will need to power down and try again.




# 4 Read/Write

## 4.1 Read Registers




Select 'Register Type'. This determines the Modbus function code that will be sent to the Modbus device. Selecting 'Register Format' only tells the configuration tool how to interpret the data and format for you to read. The format has nothing to do with construction the query sent to the device. If reading a 32-bit data value, it will occupy two consecutive Modbus registers. The order is not standardized from one manufacturer to the next, so the option of swapping them is provided here. Check the 'High Reg First' box if the most significant data is found in the first of the two registers (must check with manufacturer of device). Again, this is only used to interpret the data and has nothing to do with constucting the query to the device.

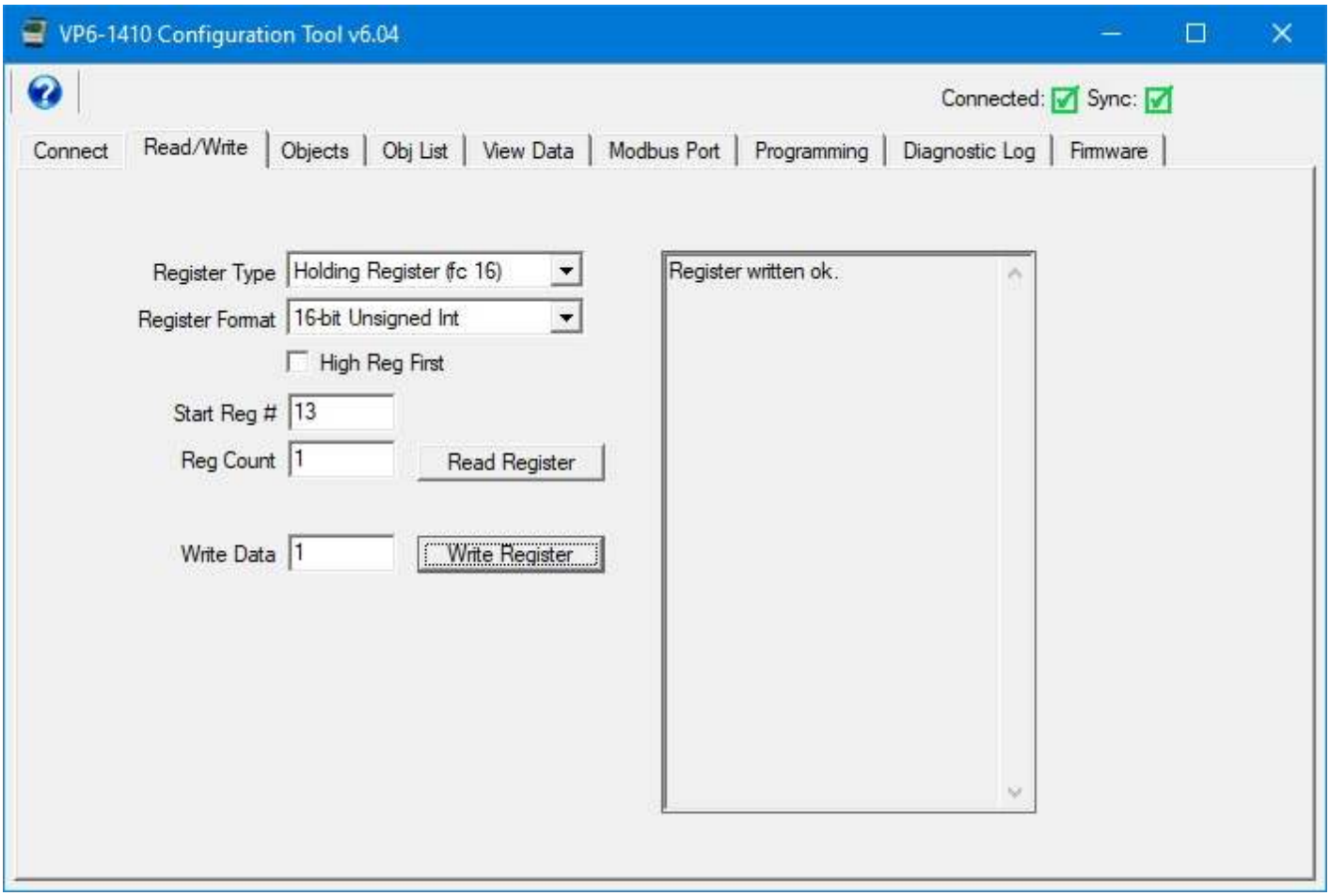
Select starting register number and count. Note that 32-bit values will always occupy 2 registers each. The starting register and register count are important pieces of

information used to construct the query to the Modbus slave. Click 'Read Register' to cause the query to be created and transmitted over the RTU network. The results of the query will be displayed in the log window on the right.

The Read/Write functionality provided here is completely generic, and can be used to read/write any Modbus slave connected to the same RS485 network that your PC is connected to.

**NOTE:** When reading data from some other manufacturer's device, you need to check with that manufacturer to see what order the registers are in for reading 32-bit values. When reading/writing a ValuPoint, the 'High Reg First' box on the Read/Write page should match whatever was configured on the Modbus Port page. The default as initially shipped is to not check this box.

## 4.2 Write Registers




Select all of the same parameters that you would for reading a register. In addition, enter a data value to be written, and then click 'Write Register'. The 'Register Format' will be used to determine how the data you enter is converted to raw binary form to be transmitted to the device. The result of your 'Write Register' attempt will be displayed in the log window. The example illustrated above will turn on relay DO 1 in the VP6-1410.

## 4.3 Errors

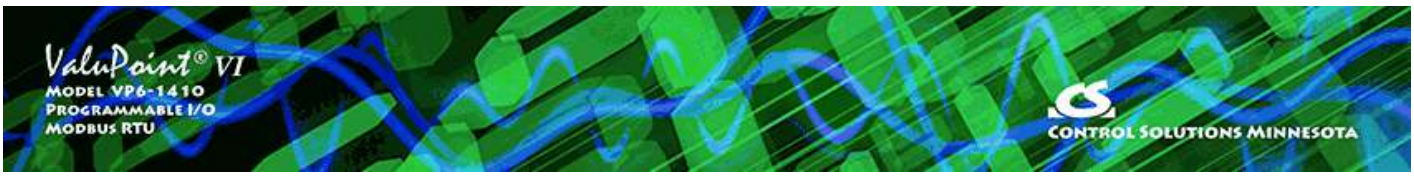
One of the most common Modbus RTU errors is simply 'Timeout' or 'No Response'. This means the slave device did not respond for any of the following reasions: (a) Slave not powered up; (b) slave not connected correctly; (c) baud rate mismatch. In the case of a poorly behaved Modbus slave, it may ignore a request it does not like, but Modbus protocol says it is supposed to return an exception error if it received a query but does not like it for some reason.

If communication is successful, but the Modbus slave returns an exception error, the most common errors are listed below. If you get an error other than these, it will be unusual and you will need to consult both the Modbus protocol definition of error codes as well as the manufacturers data to see what caused it.

The most common error will be 'illegal data address'. This simply means you requested a register that does not exist in the slave device, usually due to a typo or a misinterpretation of the manufacturers data. Illegal function will happen only if you attempt to send a request that is not at all supported by the Modbus slave. If this happens, it will most likely be associated with a Write request, and most often happens trying to use 'write multiple' when only 'write single' is supported, or vice versa, as it applies to coils or holding registers.

| | | |
|---|---|---|
| 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. |

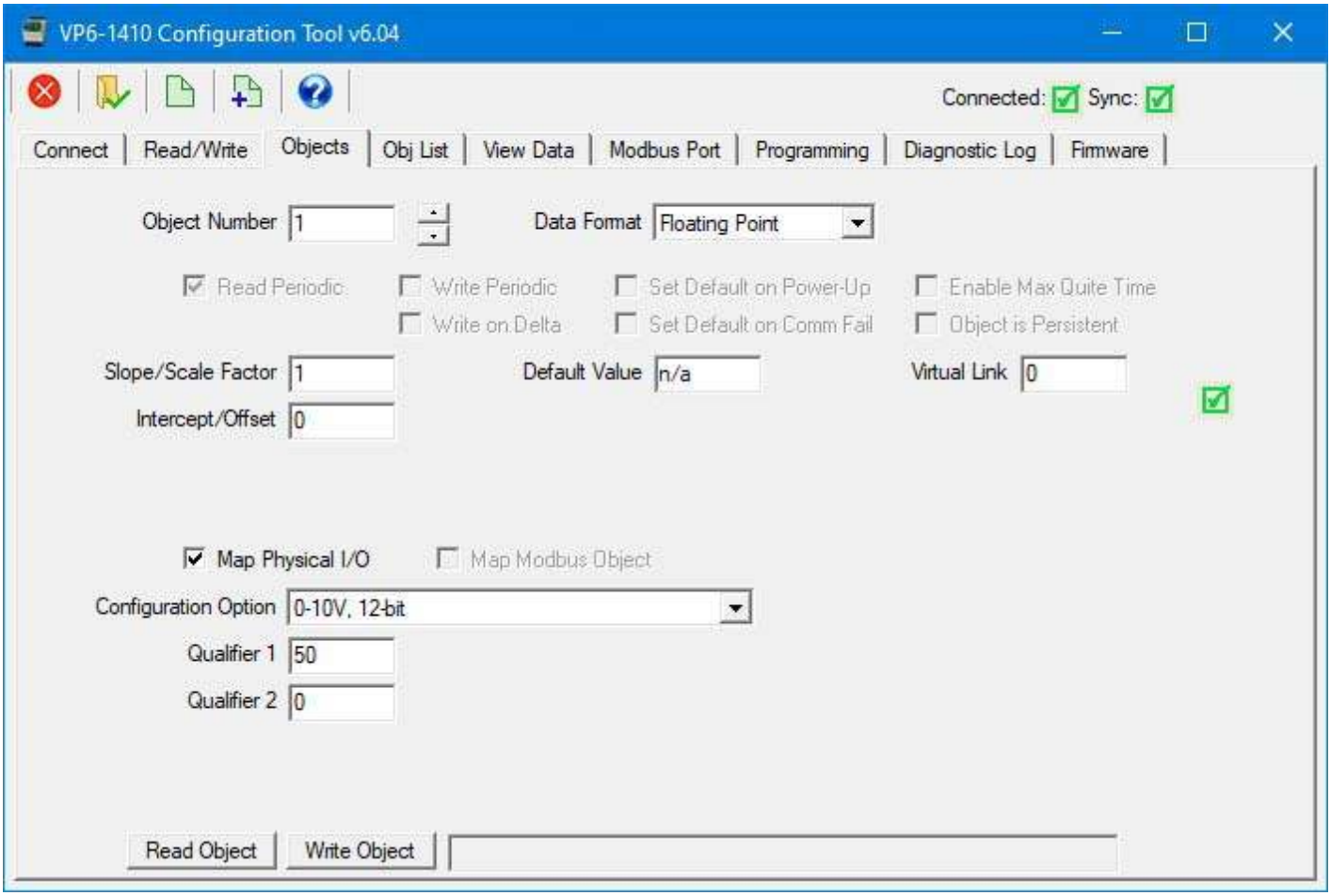
If the error you are getting is CRC and this happens frequently, the problem might be parity setting, but is most often related to wiring problems such that excessive noise is getting onto the data lines. Often the CRC errors are mixed in with frequent occurrences of 'no response', but the meaning is the same. Check port settings and wiring.




# 5 Object Map Page

The Object Map page is used for configuring the I/O objects as well as optional Modbus objects. When the configuration tool's serial port is open, the Connected icon at the top will display a green check rather than red X. If all configuratoin displayed by the configuration tool matches what is actually stored in the VP6-1410, then the Sync icon will be a green check mark rather than red X. If changes are made to an object, then the icon in the middle of this page will change from green check to red X and will return to green when the Write Object button is clicked and the configuration is successfully written via Modbus to the VP6-1410. You can click Read Object to cancel any changes and retrieve configuration from the VP6-1410.

Configuration changes made in the configuration tool are not automatically sent to the VP6-1410 device. You need to explicitly tell the tool to send changes by using the Write Object button. The icons that change back and forth between green check mark and red X will show you whether the configuration you see in the tool's pages matches what is stored in the VP6-1410 device.




## 5.1 Common Parameters

Object Number – The ValuPoint has a collection of data "objects" and each of these may be configured using the Objects page. The object number is not necessarily the Modbus register number. If you do use the object number as holding register number, you will read and write the object as a 16-bit integer even if the object's native data format is floating point. You need to reference the object using the correct floating point register number to access it as floating point.

NOTE: Objects 1-14 are dedicated to physical hardware I/O points and cannot be mapped to anything other than physical I/O. Conversely, the remaining objects cannot be mapped to physical I/O. The remainder of the 500 available objects are for optional use as additional Modbus registers.

Data Format – Select the native data format for this data object. This determines how the data is maintained internally. The format in which you access the object as a Modbus register depends on the Modbus register number you use. There are multiple Modbus register numbers that access each data object.

NOTE: The data format refers to internal storage format. The Modbus register number you use determines data format for reading the data as a Modbus register, and data type conversion is done automatically on the fly when you read Modbus registers using an external Modbus client (master). Guidelines for selecting internal storage type:

When monitoring analog sensors, the most logical choice is floating point. For discrete inputs and for controlling the relay outputs, there is nothing gained by using floating point and thus integer makes most sense. When the device is acting as a Modbus master and reading data from other Modbus devices, or writing data to other Modbus devices, then it is most logical to make the internal storage format match that of the remote register being read or written.

Slope/Scale Factor – Scaling applies the formula y=mx+b. When reading from a hardware input or from another Modbus device, the raw data as read is multiplied by the scale factor, then the offset is added to produce the resulting Present Value. When writing to the remote Modbus register, the offset is first subtracted from Present Value, and that result is divided by the scale factor to produce the raw data actually written to the remote device. NOTE: If no scale factor is given (zero is entered), no scaling will be done, as if slope=1 and intercept=0.

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

Default Value – This is the value that should become the Present Value upon power-up or upon communications failure, if either of these options are selected by the appropriate check boxes above.

**CHECK BOXES** will be enabled when applicable to the data object being configured, and these are as follows:

Read Periodic (check box) – Check this box if the data object will be periodically reading from a slave Modbus device. The box is always checked (selected/enabled) for physical hardware inputs.

Write Periodic (check box) – Check this box if the data object will be periodically writing to a slave Modbus device. This box is always checked (selected/enabled) for physical hardware outputs.

Write on Delta (check box) – Check this box if the data object will be writing to the slave device only when the object's value changes by a specified 'Delta'. It is valid to check both Write Periodic and Write on Delta at the same time, or check just one or the other.

Set Default on Power-Up (check box) – Check this box if the object should assume the default value every time the ValuPoint powers up.

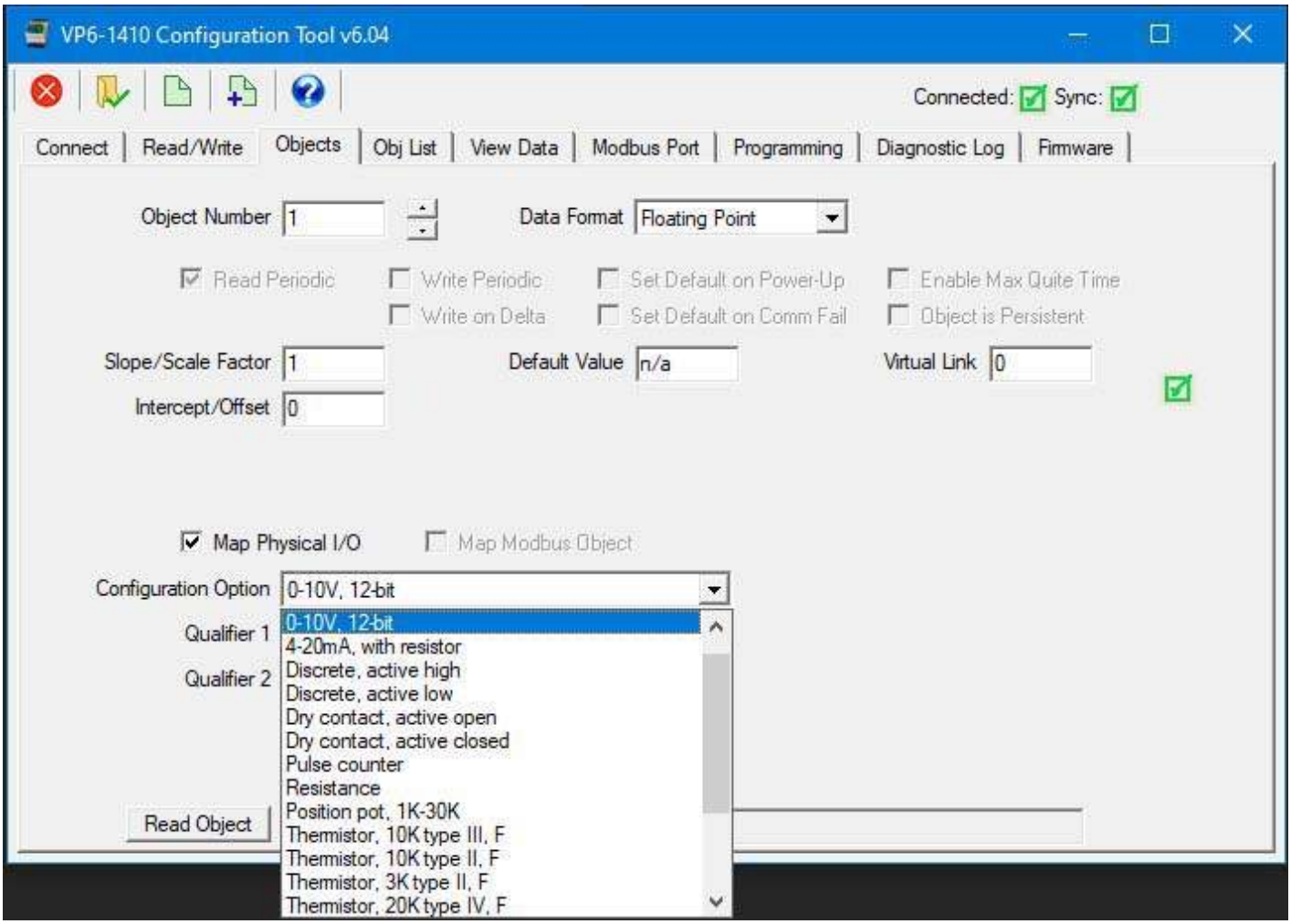
Set Default on Comm Fail (check box) – Check this box if the object should assume the default value when communication with the slave device has failed some number of times.

Enable Max Quiet Time (check box) – Check this box to enable maximum quiet time. This is applicable to an object which is set to Write on Delta. The result is that if there has been no change within the max quite time, the slave device will be re-written anyway.

Object is Persistent (check box) – Checking this box means the data value of the

object will be retained through restart or power cycle, rather than reset to zero. This is applicable to only certain objects. Data objects mapped to physical inputs, for certain configurations, will not honor the Persistent flag. For example, an Analog Input will always reflect the actual input value when configured for 0-10V input. However, the persistent option does apply when the input is configured for pulse counting. When the count is persistent, the count will be retained through restart or power cycle; however, to extend the life of the non-volatile memory, the persistent value is only saved once every 15 minutes.

## 5.2 Physical I/O Hardware Parameters




Objects 1-14 are permanently mapped to physical I/O points. While you cannot unmap them, you can change how the universal inputs are configured to operate.

Map Physical I/O (check box) – This box is forced to the selected/enabled state for those objects that map to physical I/O, as determined by the I/O features of the ValuPoint model.

Configuration Option – Select the I/O configuration desired for the physical point. This only applies to A/UI analog/universal input points. The A/UI inputs may function as any of the following:

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

Thermistor, 10K type III, F
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.

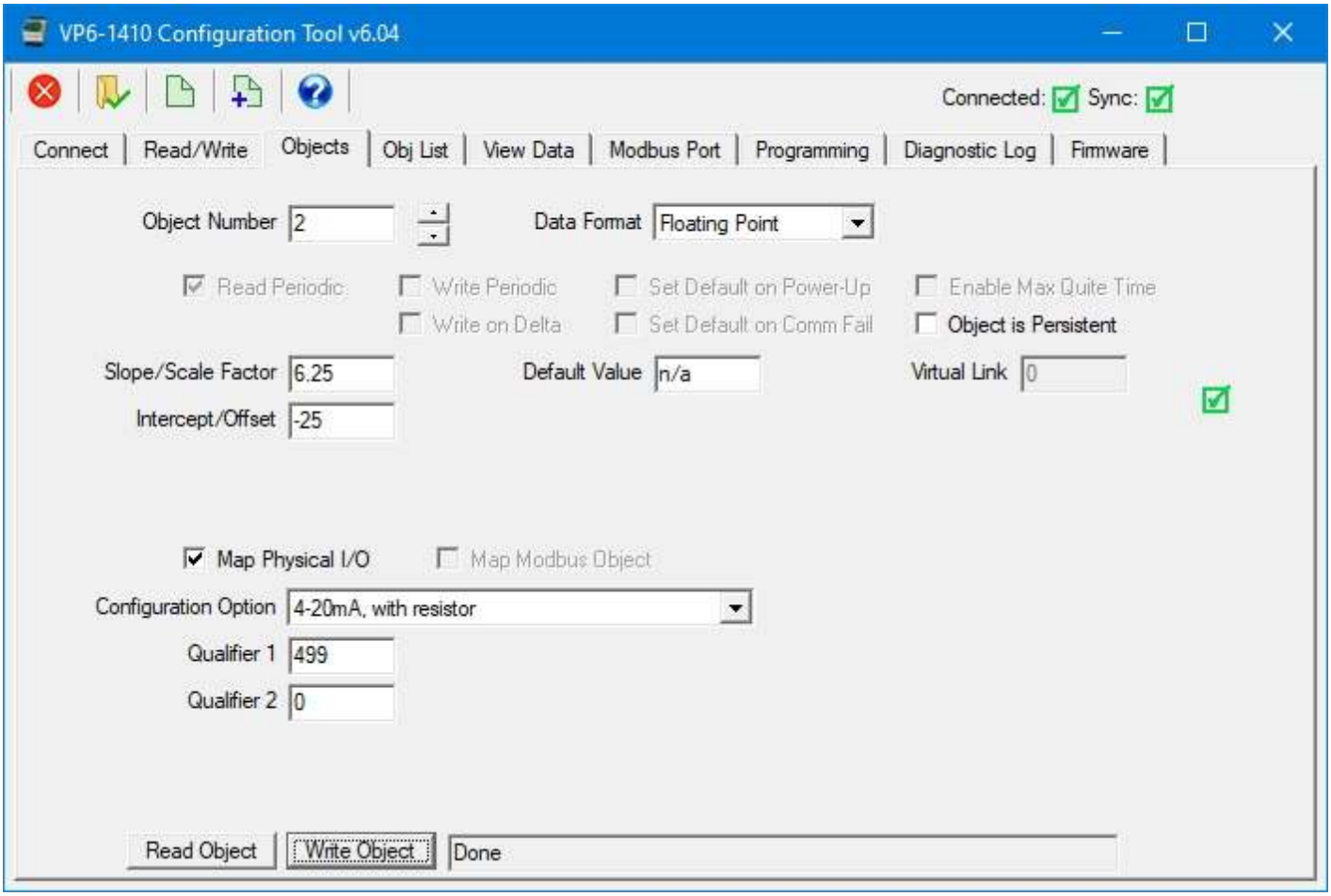
Qualifier 1 – Enter the configuration qualifier value, if applicable, for the selected configuration. Qualifiers are required only in the following modes:

*4-20mA modes:* 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.

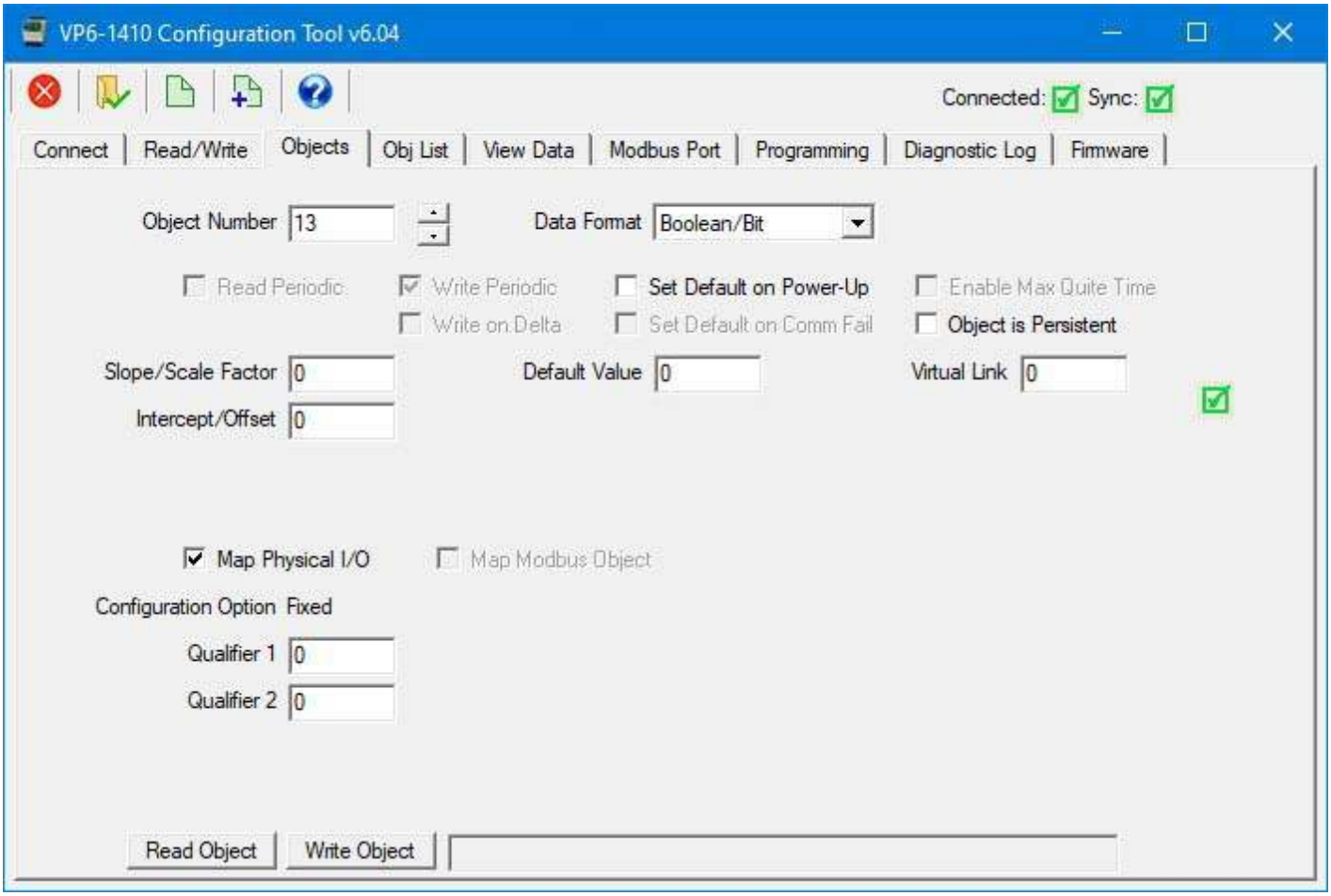
*Discrete and Dry Contact modes:* The qualifier is a threshold between 1% and 99% at which the input should trip from off to on or vice versa. The A/UI inputs are specified as 0-10V inputs. Therefore, since discrete inputs are sampled as analog values and compared to a threshold, the qualifier here is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

Qualifier 2 – Enter a second configuration qualifier value if instructed to do so. This qualifier is reserved for future use.



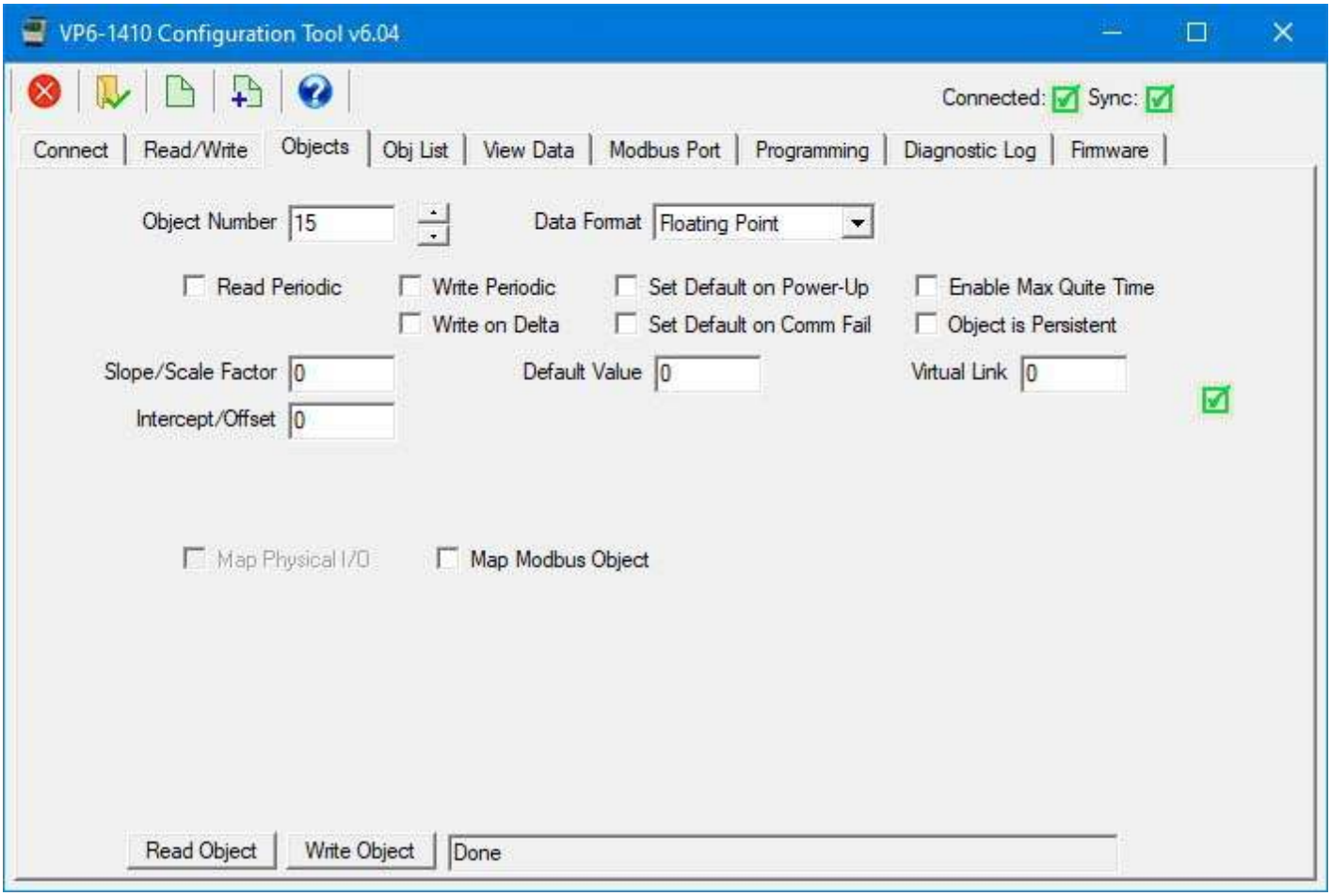

The above example illustrates configuration for a 4-20mA input with a 499 ohm dropping resistor connected externally. The configuration is denoted as 4-20mA, but in reality, it is actually a 0-20mA input. The scale factors illustrated here will produce a resulting value in the range of 0-100% where 0% is indicated at 4mA and 100% is indicated at 20mA.



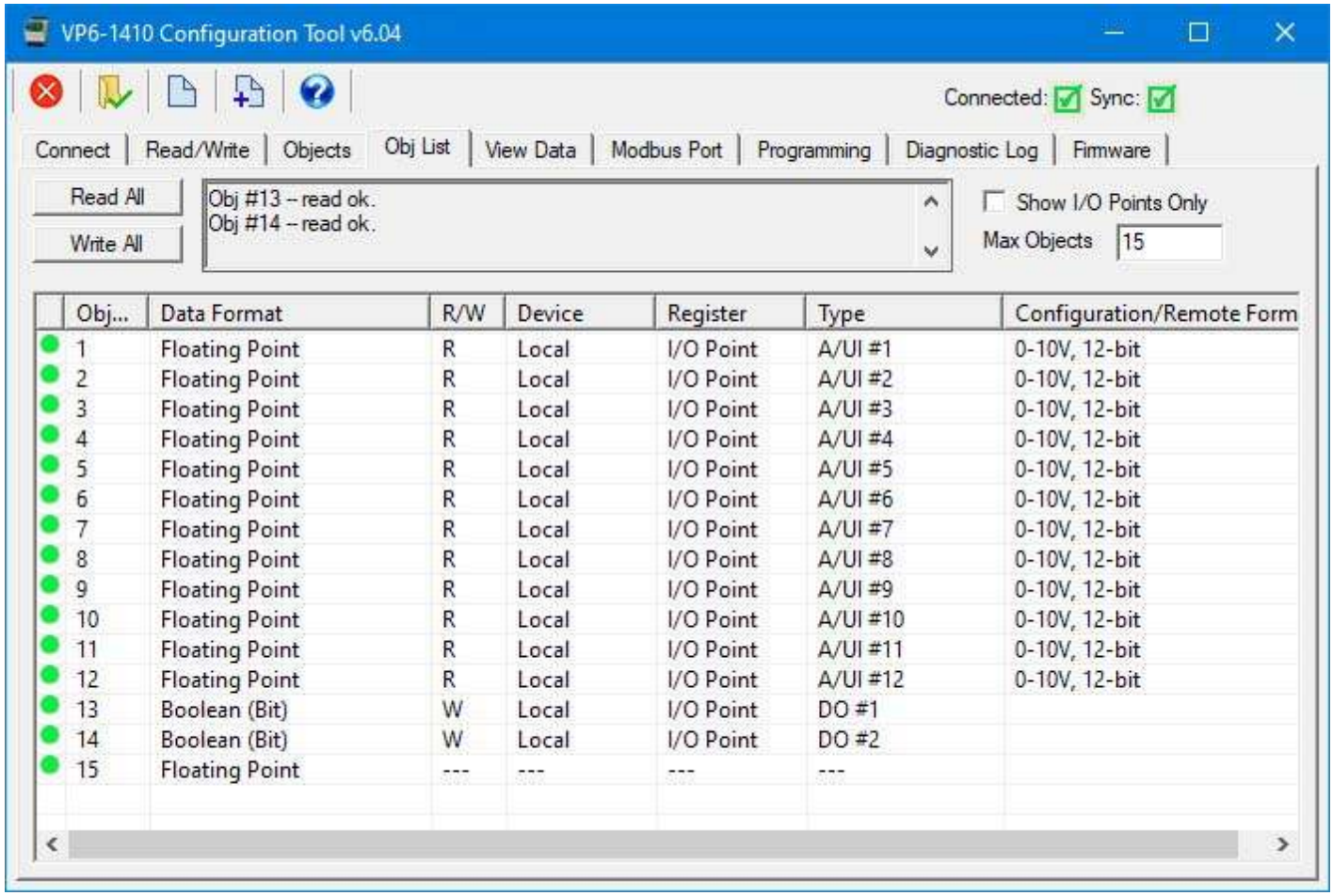

The above examle illustrates the configuraton for one of the relay outputs. There is really nothing to configure unless you want to do some special things like link the output to an input as illustrated later in this section.

## 5.3 Local Object

A "local object" is one which is not mapped to physical hardware or to a remote Modbus object. These can be used as internal storage registers for i.CanDrawIt programs. To allocate a local object, simply select a data format for the object as illustrated below.

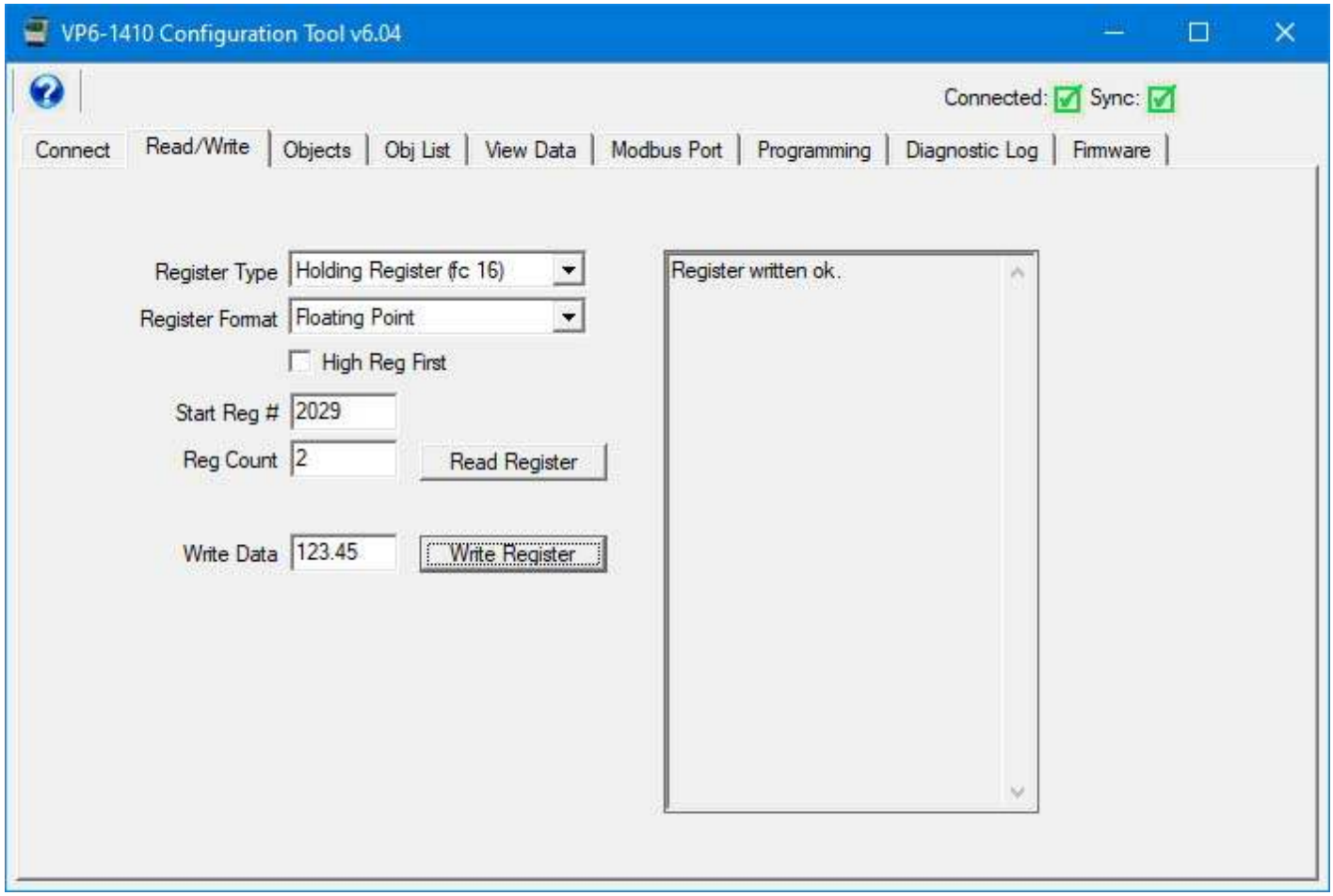



This new local object will now appear on the object list page.

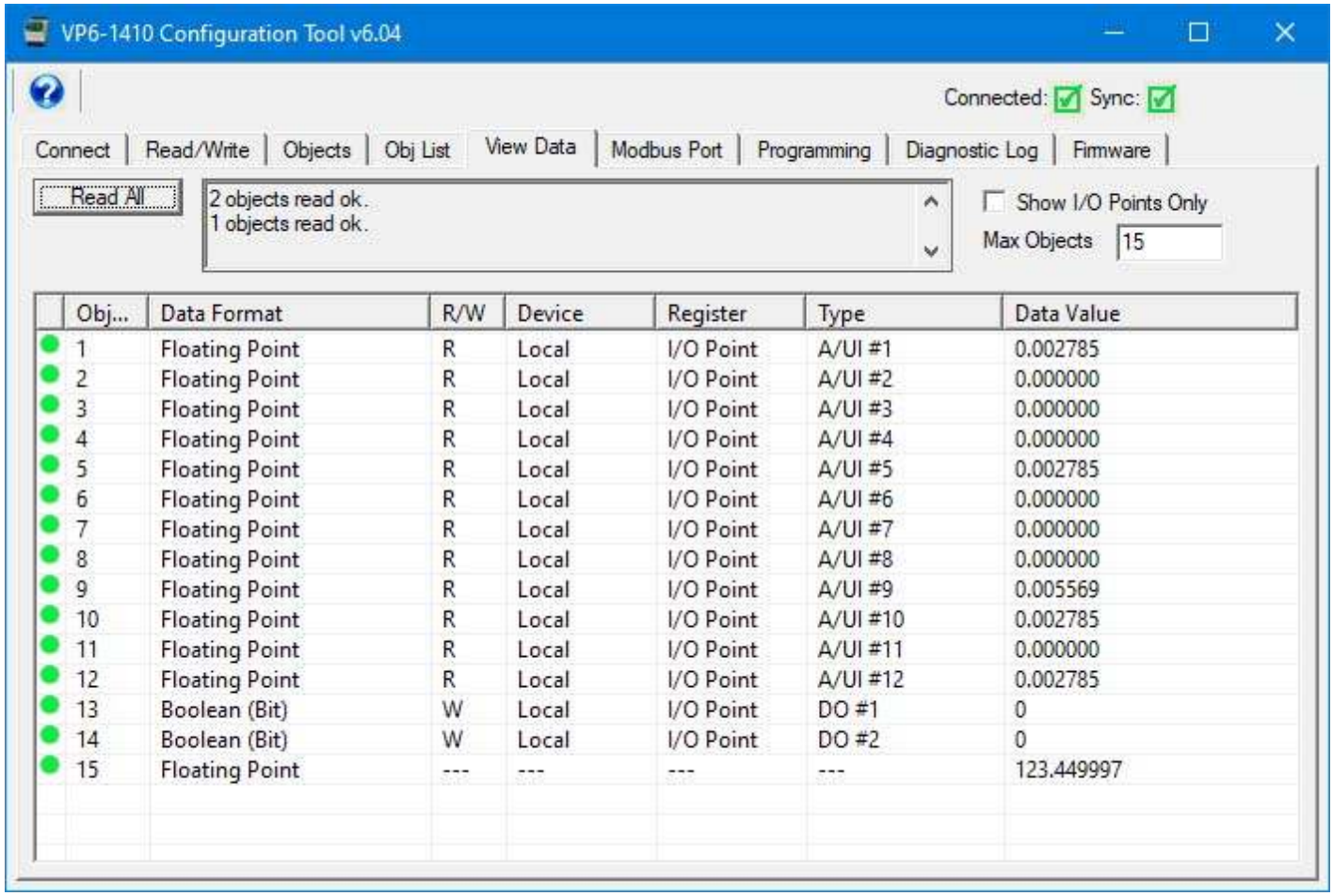



| Obj... | Data Format | R/W | Device | Register | Type | Configuration/Remote Form |
|---|---|---|---|---|---|---|
| 1 | Floating Point | R | Local | I/O Point | A/UI #1 | 0-10V, 12-bit |
| 2 | Floating Point | R | Local | I/O Point | A/UI #2 | 0-10V, 12-bit |
| 3 | Floating Point | R | Local | I/O Point | A/UI #3 | 0-10V, 12-bit |
| 4 | Floating Point | R | Local | I/O Point | A/UI #4 | 0-10V, 12-bit |
| 5 | Floating Point | R | Local | I/O Point | A/UI #5 | 0-10V, 12-bit |
| 6 | Floating Point | R | Local | I/O Point | A/UI #6 | 0-10V, 12-bit |
| 7 | Floating Point | R | Local | I/O Point | A/UI #7 | 0-10V, 12-bit |
| 8 | Floating Point | R | Local | I/O Point | A/UI #8 | 0-10V, 12-bit |
| 9 | Floating Point | R | Local | I/O Point | A/UI #9 | 0-10V, 12-bit |
| 10 | Floating Point | R | Local | I/O Point | A/UI #10 | 0-10V, 12-bit |
| 11 | Floating Point | R | Local | I/O Point | A/UI #11 | 0-10V, 12-bit |
| 12 | Floating Point | R | Local | I/O Point | A/UI #12 | 0-10V, 12-bit |
| 13 | Boolean (Bit) | W | Local | I/O Point | DO #1 | |
| 14 | Boolean (Bit) | W | Local | I/O Point | DO #2 | |
| 15 | Floating Point | --- | --- | --- | --- | |

You can use the Read/Write page to read and write that local object. Here we see a value of 123.45 being written as floating point to local object 15. Refer to the Appendix B, Modbus Register Map, to see why the register number is 2029 for object 15.



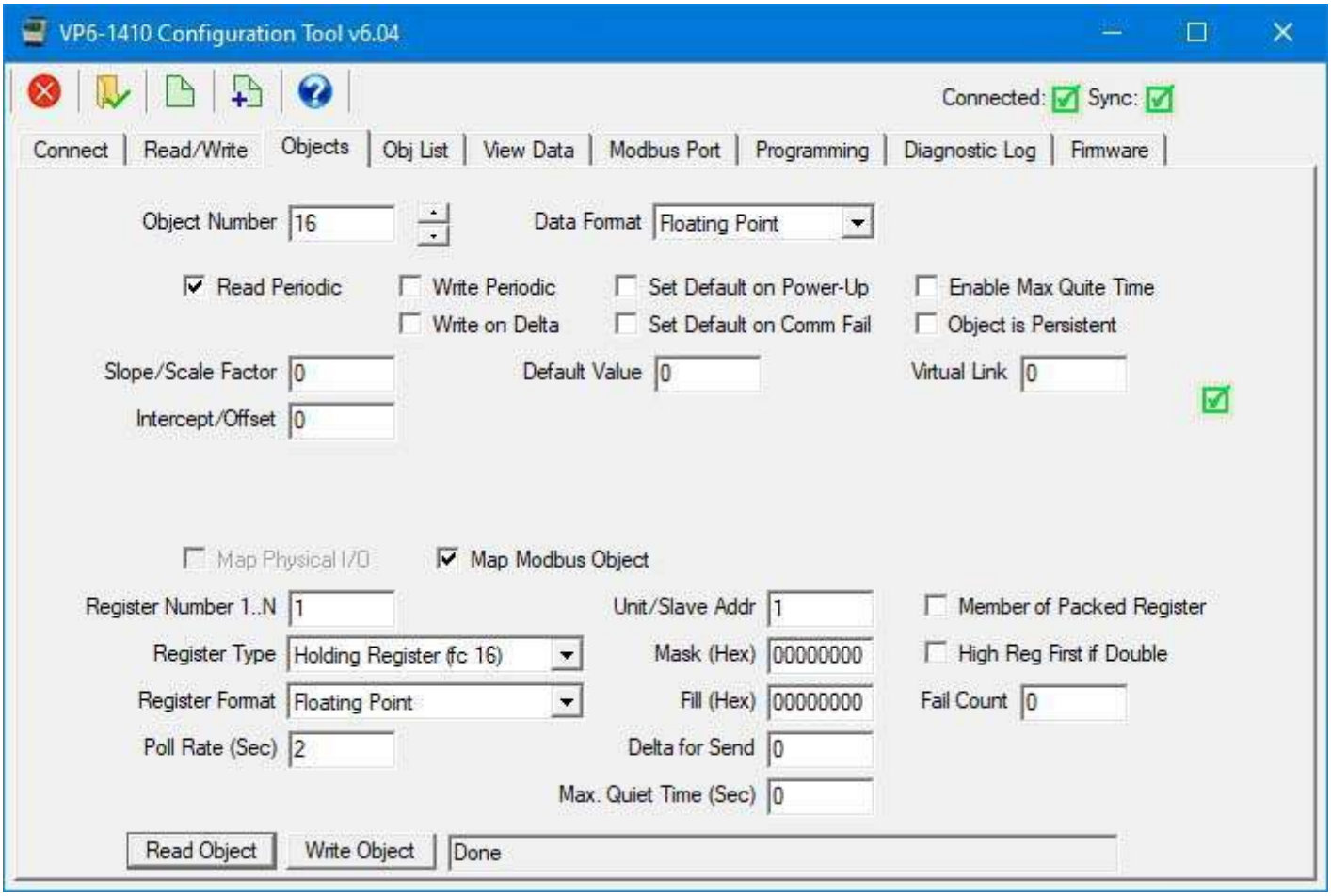

The updated value now appears on the View Data page.




## 5.4 Modbus RTU Master Parameters

The VP6-1410 has the ability to interact with other Modbus devices when it is configured to be a Modbus master. As a master, you can configure what registers are read from which other devices, or which registers are written to other devices. When reading, a copy of the register data obtained from the other device is stored in the local object. When writing, data is taken from the local object to write to the remote Modbus device.

To configure an object to interact with a remote Modbus device, start by checking the Map Modbus Object box. When you check this box, the remaining windows below that box will appear.




Map Modbus Object (check box) – Check this box when the ValuPoint should look for another Modbus device as a slave device to query for data. When this box is checked, the remaining boxes required for Modbus map setup will appear.

Register Number 1..N – Enter the Modbus register number between 1 and 65535. The register number is raw address plus one. Therefore if your Modbus device's documentation indicates that the first register is at address zero, add one to every address to get the register 'number'. DO NOT enter Modicon numbers like 40001 here.

Register Type – Select a Modbus register type from the list, such as Holding Register, etc. Coils and Holding Registers can be written as single or multiple, and some Modbus devices only recognize one function code or the other. If you are having trouble writing, check the device's documentation and see if you are using the correct function code. Both possible codes for each register type are included in the list, denoted '(fc x)'.

Register Format – Select a register format from the list. If the data format is 32-bit integer or floating point, the VP6-1410 will automatically read/write two consecutive registers to get the entire value. The designation 'high reg first' means the most significant part of a multi-register value will be in the first, or lowest numbered, Modbus register.

Format designations:

| CSV Notation | Description |
|---|---|
| SIGN | Signed 16-bit |
| UNSI | Unsigned 16-bit |
| SDBE | Signed 32-bit (high reg first) |
| UDBE | Unsigned 32-bit (high reg first) |
| FPBE | Floating Point (high reg first) |
| BBIT | Bit (coil/discrete) |
| SDLE | Signed 32-bit (low reg first) |
| UDLE | Unsigned 32-bit (low reg first) |
| FPLE | Floating Point (low reg first) |

Poll Rate (Sec) – This rate applies to periodic reading or writing, and simply specifies how often the slave device is read from or written to. (This parameter does not apply to physical I/O.)

Unit/Slave Addr – Enter the Modbus device's slave address here. This number is also often referred to as unit number or slave ID.

Mask (Hex) – When extracting a single bit or set of bits from a packed Modbus register, this mask specifies where in the register the desired bits are found. This mask is entered as an 8-digit hexadecimal number representing 32 bits. After masking data read from Modbus by performing a bit-wise And between the data and this mask, the resulting data is right shifted so that the least significant bit of interest becomes the LSB of data. When writing, this process is reversed. When applied to a 16-bit register, only the least significant 16 bits of the mask are used.

Fill (Hex) – Applicable only when writing, this value is optionally used to always set specified bits. This value is entered as an 8-digit hexadecimal number representing 32 bits. The data to be written to the Modbus slave is bit-wise logical Or-ed with this value just before being written to the device. When applied to a 16-bit register, only the least significant 16 bits of the fill are used.

Member of Packed Register (check box) – Check this box if the VP6-1410 should look at the next consecutive object map, or is included in the group started by a previous object map, to combine this and at least one other object map to collect multiple local objects into a single Modbus register (or vice versa when reading).

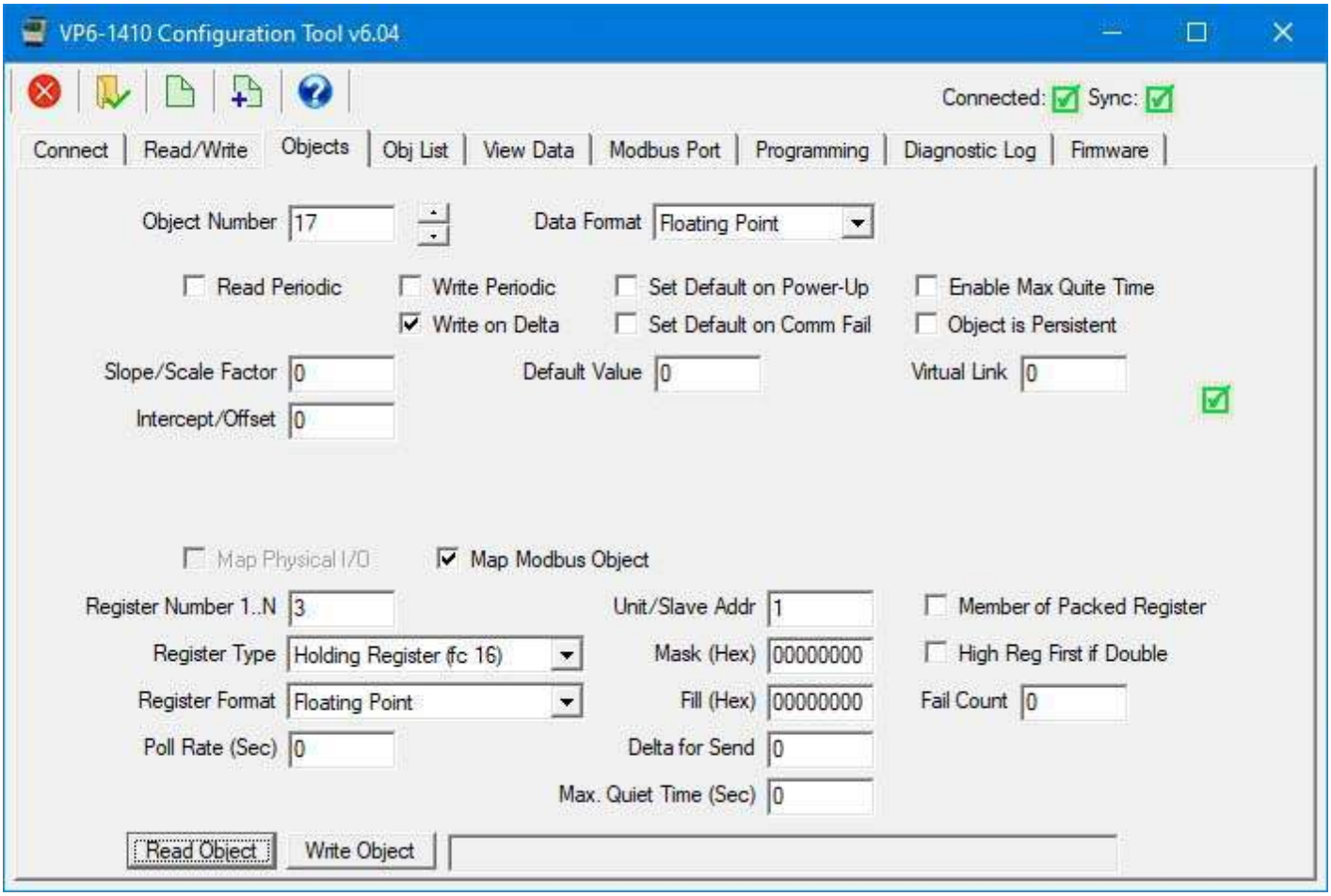
High Reg First if Double (check box) - Used only when reading/writing 32-bit data, check this box if the first register will contain the most significant half of the data.

Fail Count – When 'Set Default on Comm Fail' is checked, this entry will allow you to disregard a small number of spurious errors. At least this many errors must occur consecutively before a fault will actually be flagged. Causing a fault as a result of a single instance of spurious noise on a communication line can be a nuisance. This setting allows quieting the nuisance fault notifications. (This parameter does not apply

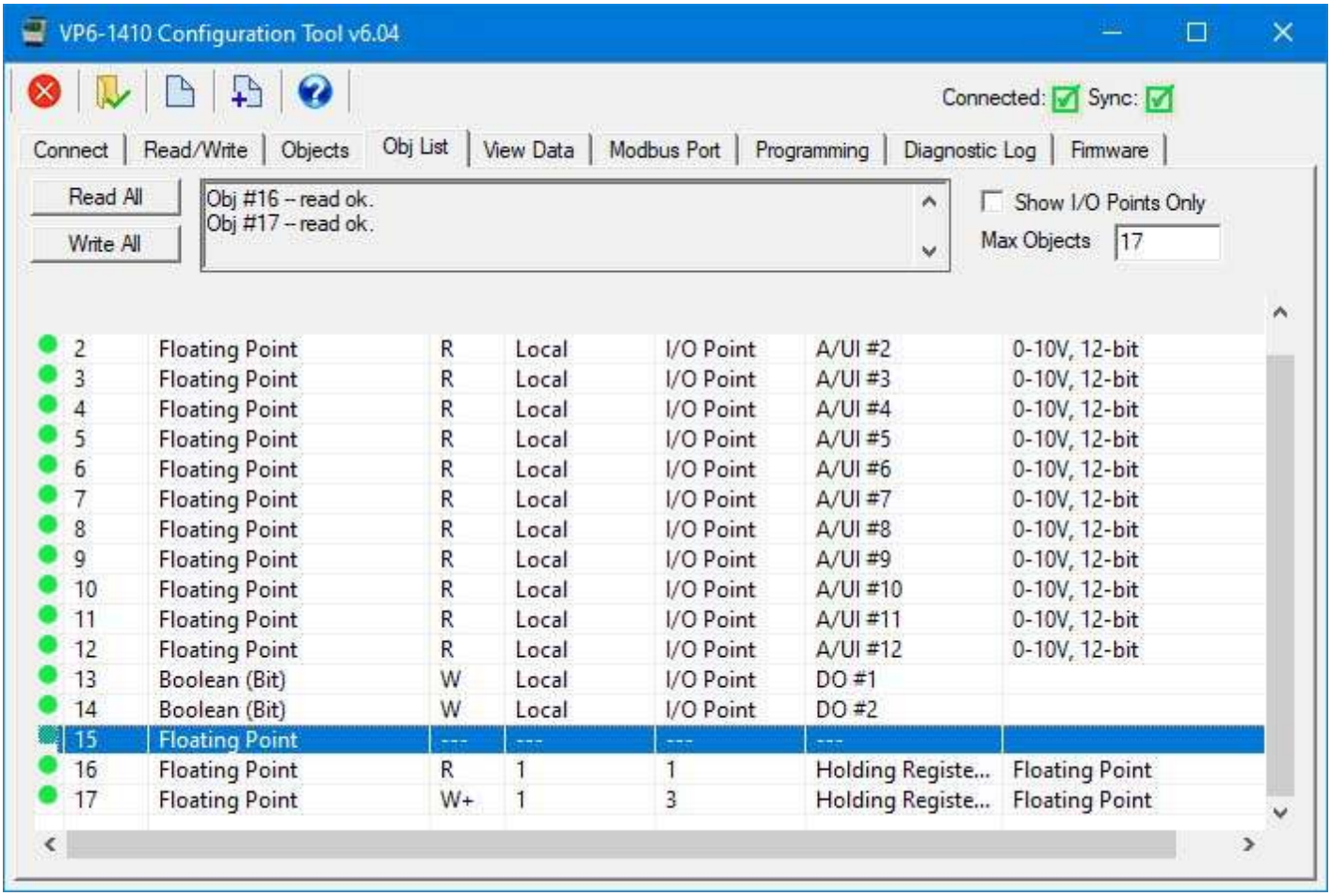
to physical I/O.)

Delta for Send – Enter the threshold for writing when configured to Write on Delta. If a value of 5.0 is entered, the value must change by more than 5.0 before the slave device will be written to.

Max. Quiet Time (Sec) – This is applicable to an object which is set to Write on Delta (writing to a remote Modbus device). The result is that if there has been no change within this amount of time, the slave device will be re-written anyway. In addition to entering a nonzero value here, you must check the 'Enable Max Quiet Time' box.

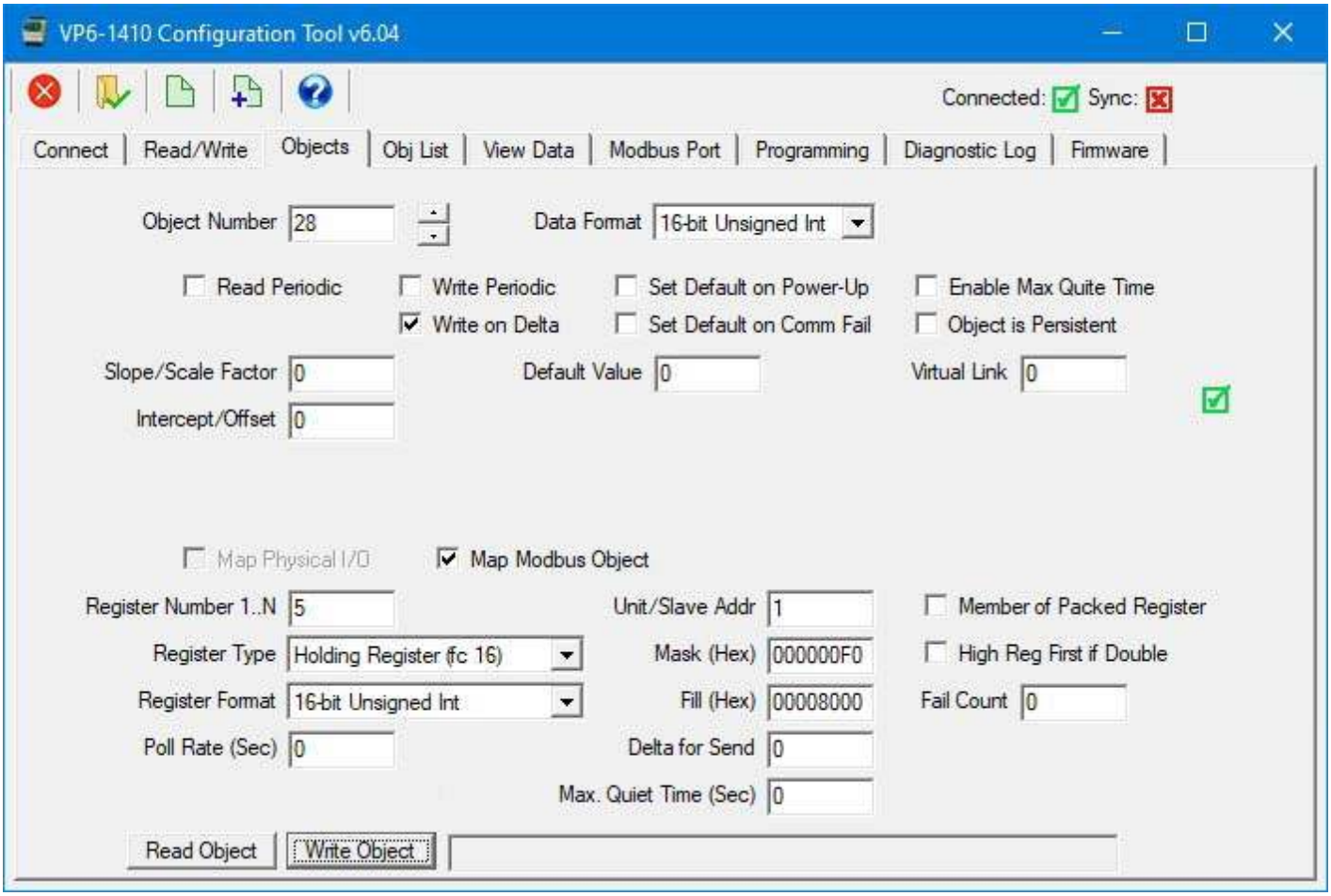



The two examples above show reading two floating point values from a Modbus RTU slave at address 1 and storing those values in local objects 16 and 17 respectively. These would appear on the object list page as illustrated below.

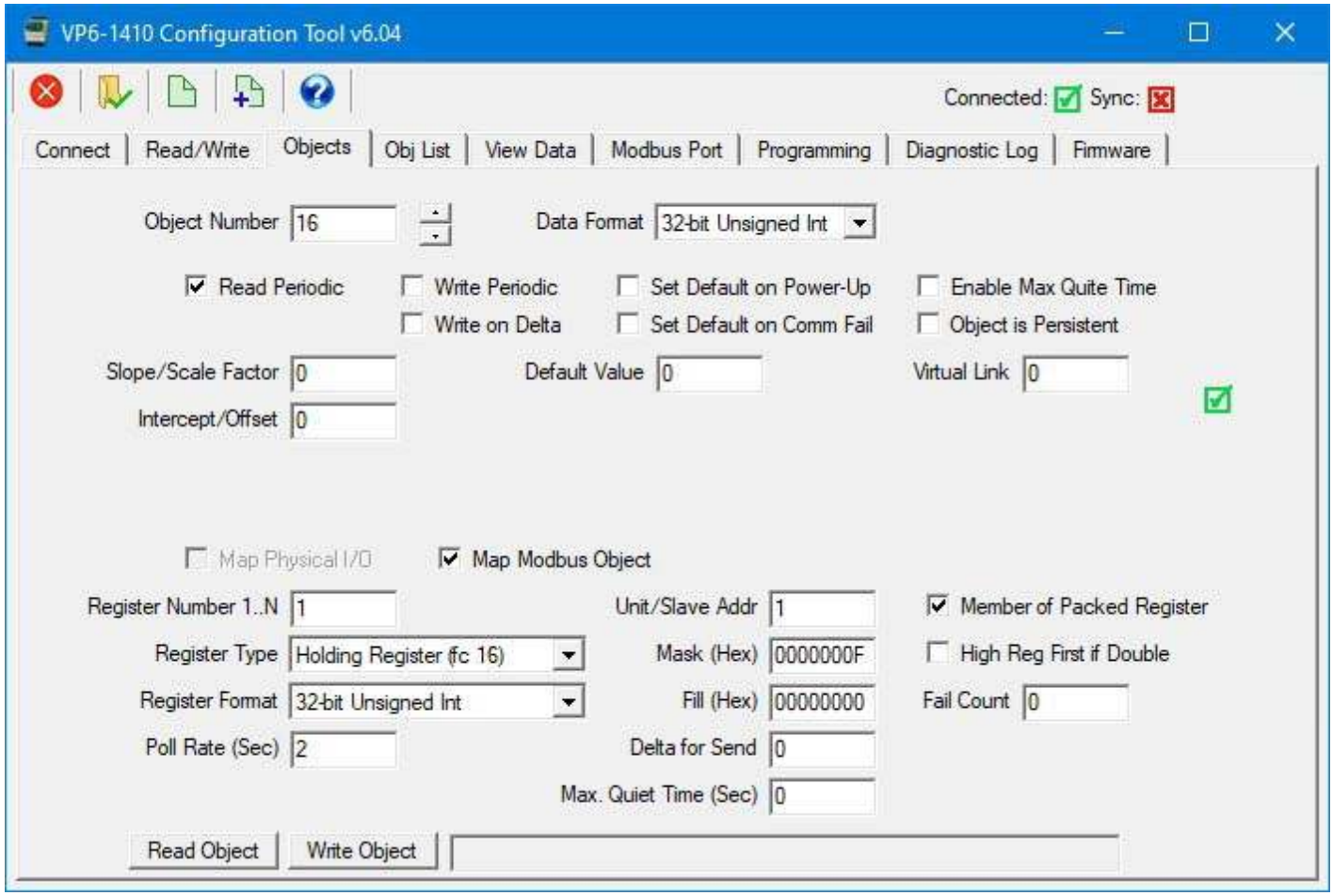


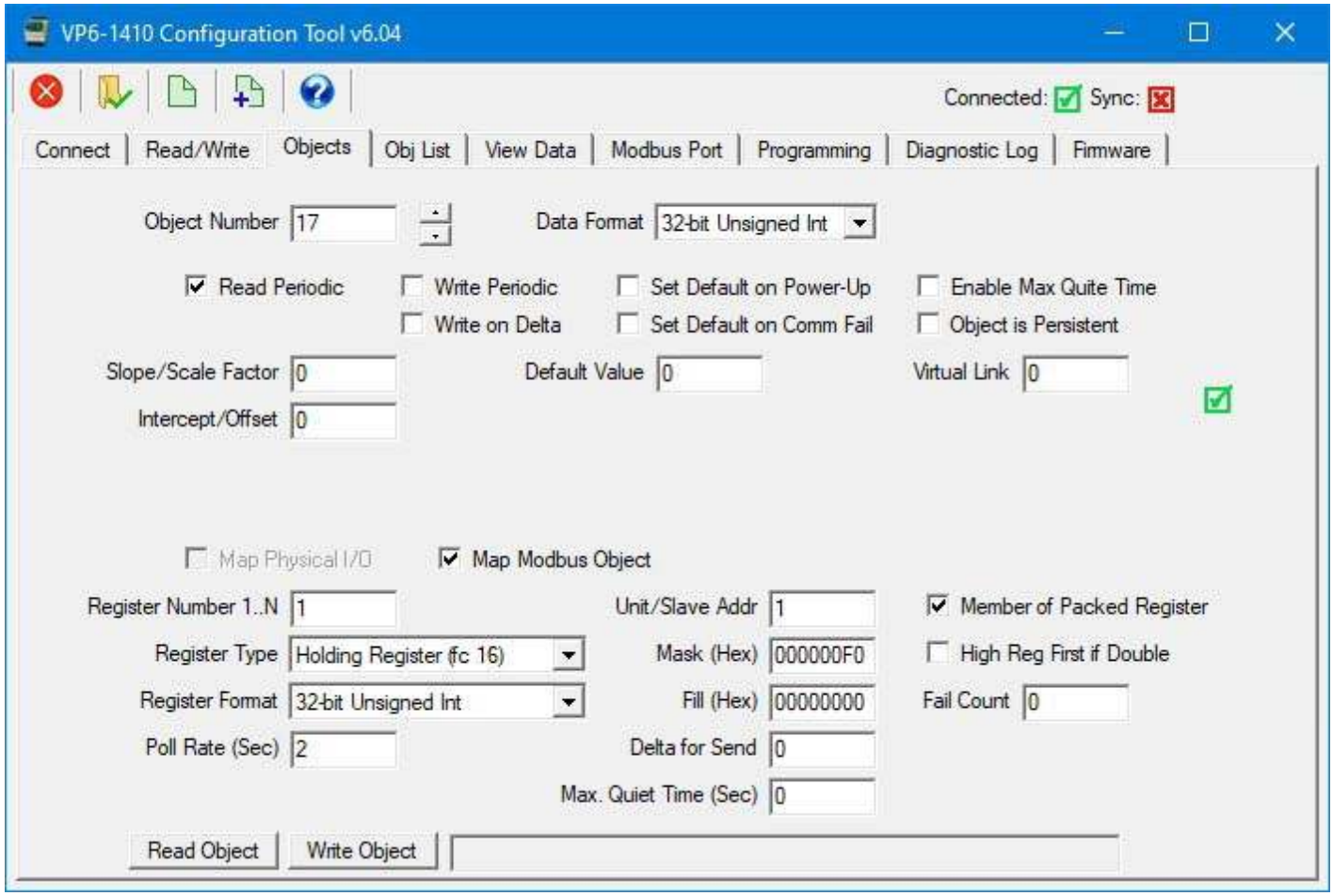

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 2 | Floating Point | R | Local | I/O Point | A/UI #2 | 0-10V, 12-bit |
| 3 | Floating Point | R | Local | I/O Point | A/UI #3 | 0-10V, 12-bit |
| 4 | Floating Point | R | Local | I/O Point | A/UI #4 | 0-10V, 12-bit |
| 5 | Floating Point | R | Local | I/O Point | A/UI #5 | 0-10V, 12-bit |
| 6 | Floating Point | R | Local | I/O Point | A/UI #6 | 0-10V, 12-bit |
| 7 | Floating Point | R | Local | I/O Point | A/UI #7 | 0-10V, 12-bit |
| 8 | Floating Point | R | Local | I/O Point | A/UI #8 | 0-10V, 12-bit |
| 9 | Floating Point | R | Local | I/O Point | A/UI #9 | 0-10V, 12-bit |
| 10 | Floating Point | R | Local | I/O Point | A/UI #10 | 0-10V, 12-bit |
| 11 | Floating Point | R | Local | I/O Point | A/UI #11 | 0-10V, 12-bit |
| 12 | Floating Point | R | Local | I/O Point | A/UI #12 | 0-10V, 12-bit |
| 13 | Boolean (Bit) | W | Local | I/O Point | DO #1 | |
| 14 | Boolean (Bit) | W | Local | I/O Point | DO #2 | |
| 15 | Floating Point | --- | --- | --- | --- | |
| 16 | Floating Point | R | 1 | 1 | Holding Registe... | Floating Point |
| 17 | Floating Point | W+ | 1 | 3 | Holding Registe... | Floating Point |

The following example illustrates the use of mask in fill in a write operation. The data found in local object 28 is going to be written to remote Modbus register 5 at RTU slave address 1. But only the least significant 4 bits of local object 28 will be written, and they will be shifted into the position indicated by the mask. In addition, the MSB of the 16 bit value written will always be set as indicated by the fill value.



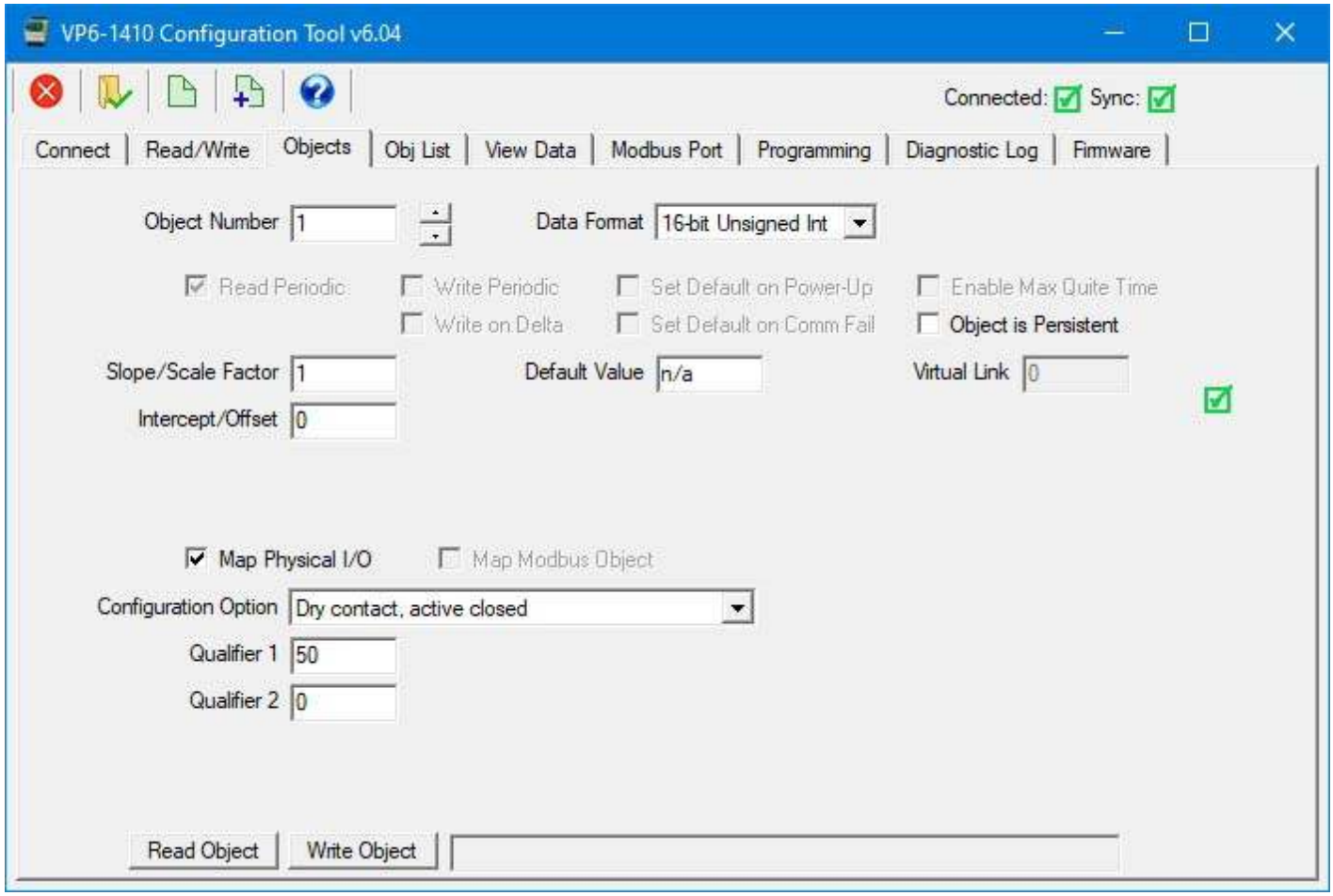

The following two screen shots show using masking and use of the packed register option. A 32-bit unsigned integer value will be read from remote Modbus register 1 at RTU slave address 1. The least significant 4 bits of the remote register value will be stored into local object 16. The next 4 bits (b4-b7) will be stored into local object 17. The "packed" register that was read from the remote Modbus device is being "unpacked" into two different registers here. Another very common scenario is to read a single remote register and unpack individual bits into a series of several local objects.


VP6-1410 Configuration Tool v6.04
Connected:
Sync:
Connect
Read/Write
Objects
Obj List
View Data
Modbus Port
Programming
Diagnostic Log
Firmware
Object Number
16
Data Format
32-bit Unsigned Int
Read Periodic
Write Periodic
Write on Delta
Set Default on Power-Up
Set Default on Comm Fail
Enable Max Quite Time
Object is Persistent
Slope/Scale Factor
0
Intercept/Offset
0
Default Value
0
Virtual Link
0
Map Physical I/O
Map Modbus Object
Register Number 1..N
1
Register Type
Holding Register (fc 16)
Register Format
32-bit Unsigned Int
Poll Rate (Sec)
2
Unit/Slave Addr
1
Mask (Hex)
0000000F
Fill (Hex)
00000000
Delta for Send
0
Max. Quiet Time (Sec)
0
Member of Packed Register
High Reg First if Double
Fail Count
0
Read Object
Write Object

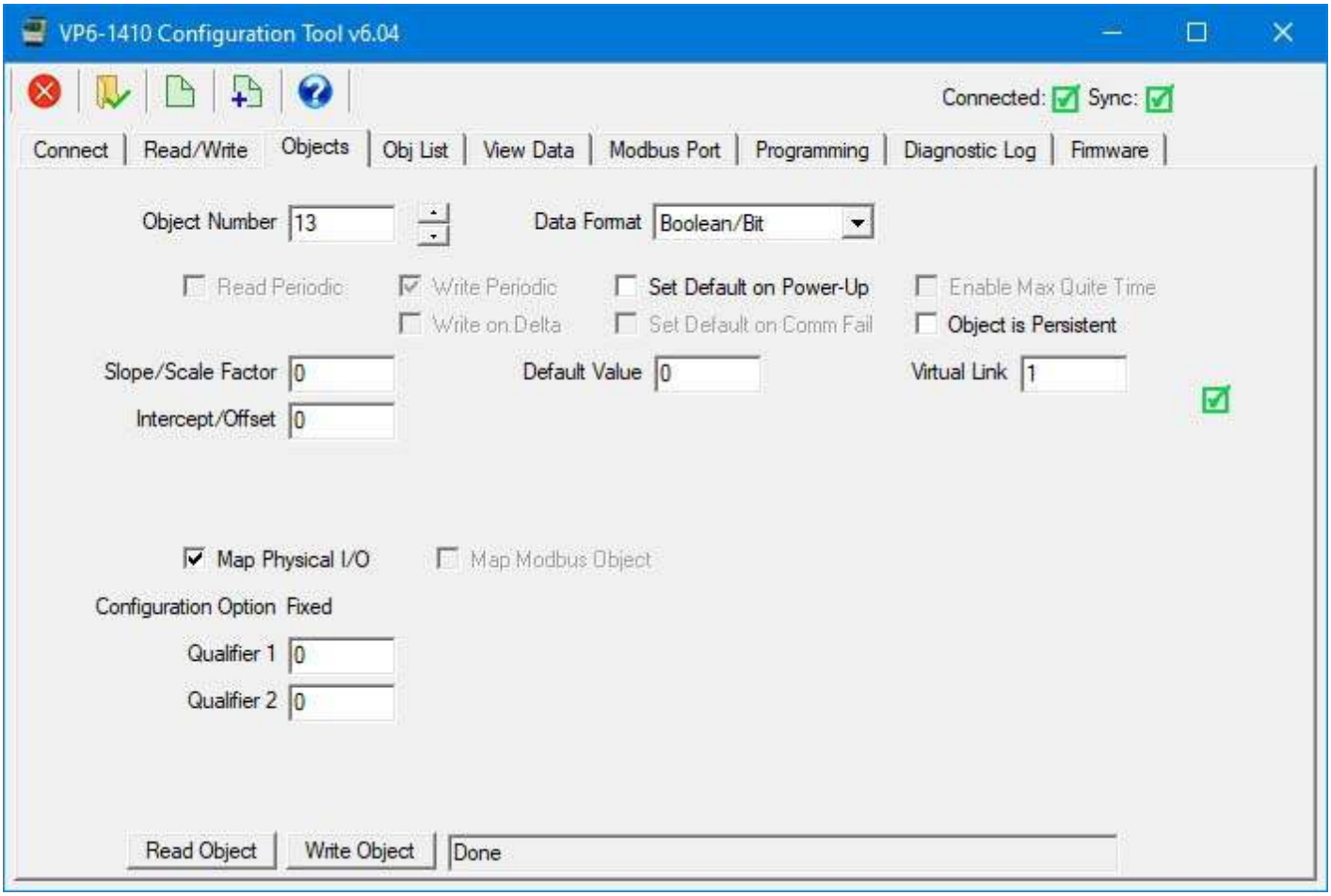



## 5.5 Virtual Links

One use of the Virtual Link is to make a virtual connection between an input and an output. Consider A/UI #1 configured as a dry contact, active closed, input as illustrated below. Its value will be either 0 or 1 for off or on.




Setting the Virual Link number to 1 for object 13, referencing the above input, will cause object 13, the relay output DO #1, to now be operated by A/UI #1.




There are many other possible virtual links not associated with physical I/O or mapped remote Modbus registers. These represent "virtual" inputs found in the system. These may be used as the source of an object's data.

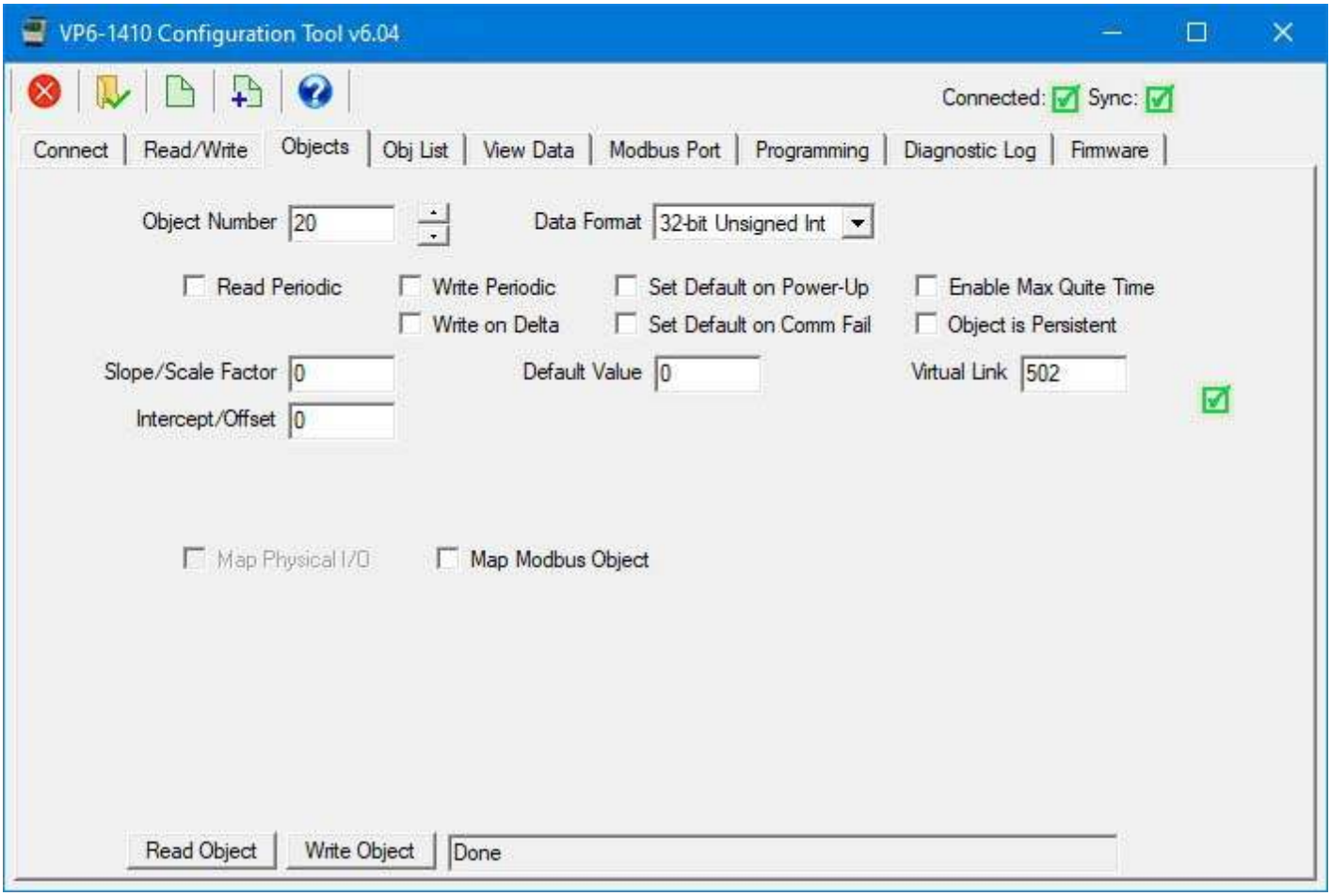
Both cycle count and on-time are retained in non-volatile memory which is written each time the output makes a transition, or every 15 minutes (for on time) if the output remains on for an extended period of time. These virtual link numbers are NOT Modbus register numbers, they are virtual input numbers, and cannot be read directly via Modbus. They must be used as the virtual link in an ordinary data object which may then be read/written via Modbus.

Normally the data objects representing cycle count or on-time would be "read only", but you can reset the counts to zero (with the exception of system up time or heartbeat) by writing zero to the object linked to those properties.

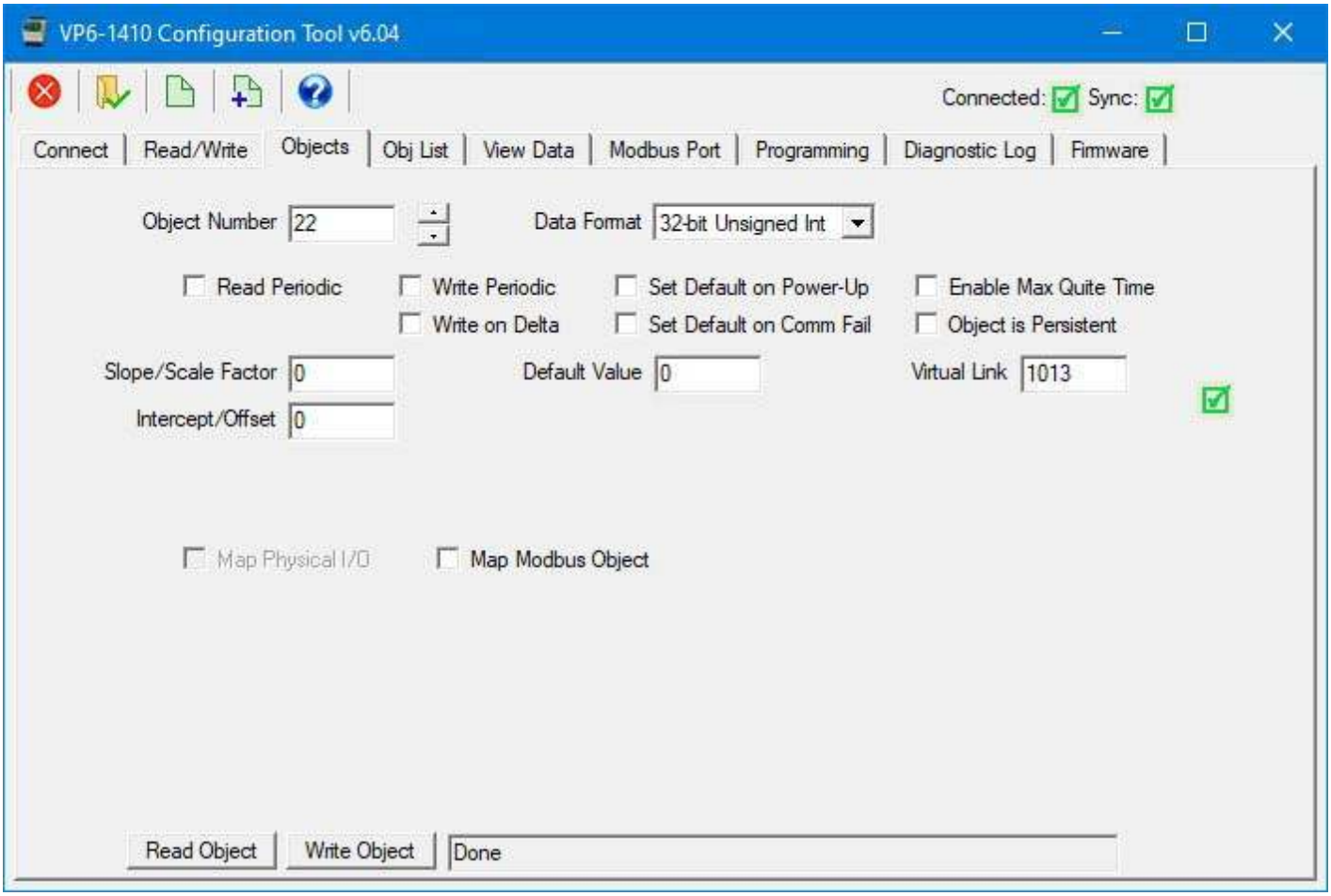
| Link number | Virtual Object Linked |
|---|---|
| 502 | Virtual link to system up time in seconds, 32-bit value |
| 503 | Virtual link to heartbeat counter, 32-bit value |
| 1001-1500 | Virtual link to cycle counter for objects 1-500, 32-bit value (automatically persistent) |
| 2001-2247 | Virtual link to Modbus error code by slave address, 16-bit value |

| | |
|---|---|
| 3001-3500 | Virtual link to on-time counter in seconds, objects 1-500 (automatically persistent) |

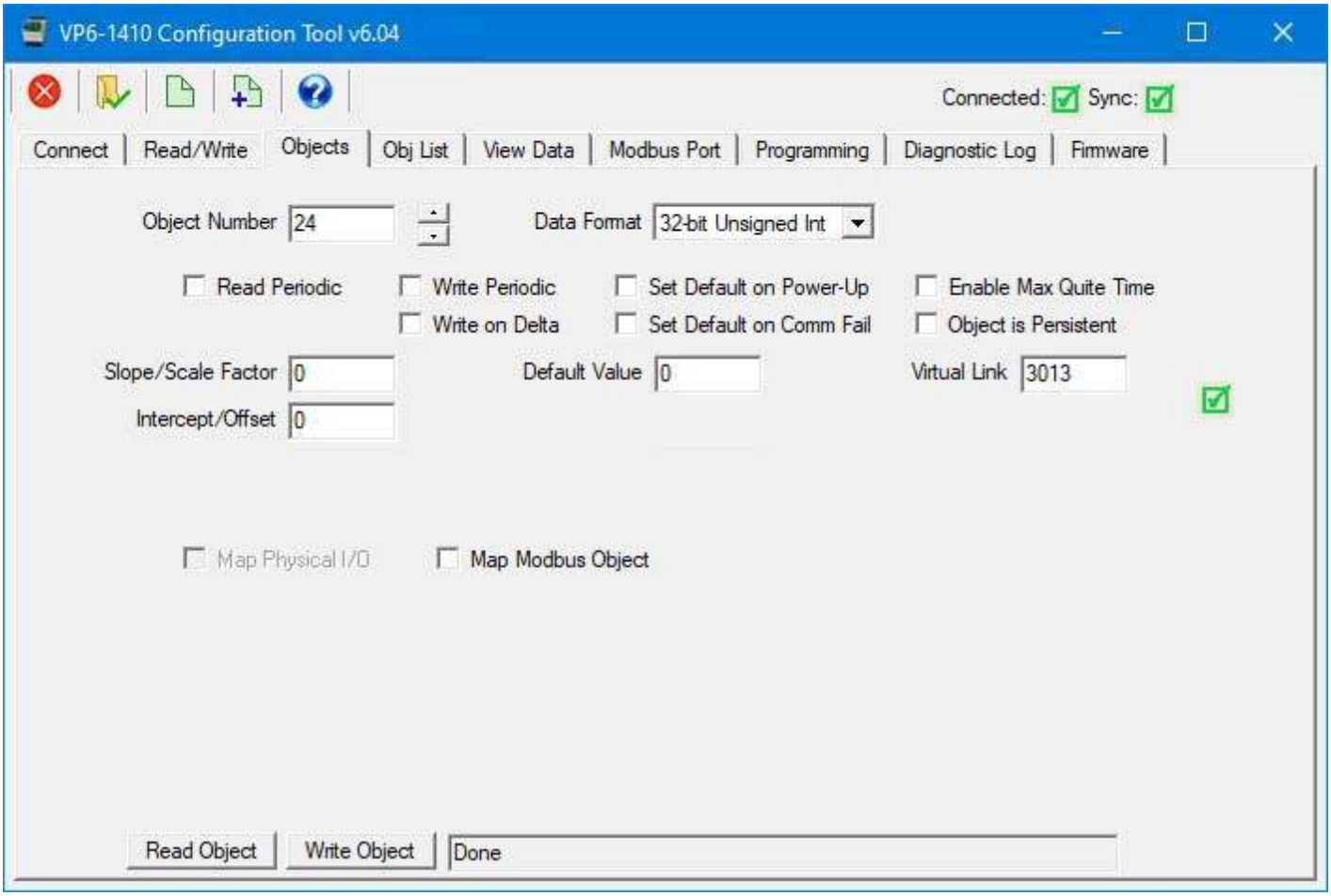
The following example illustrates making system up time accessible as a Modbus register value.



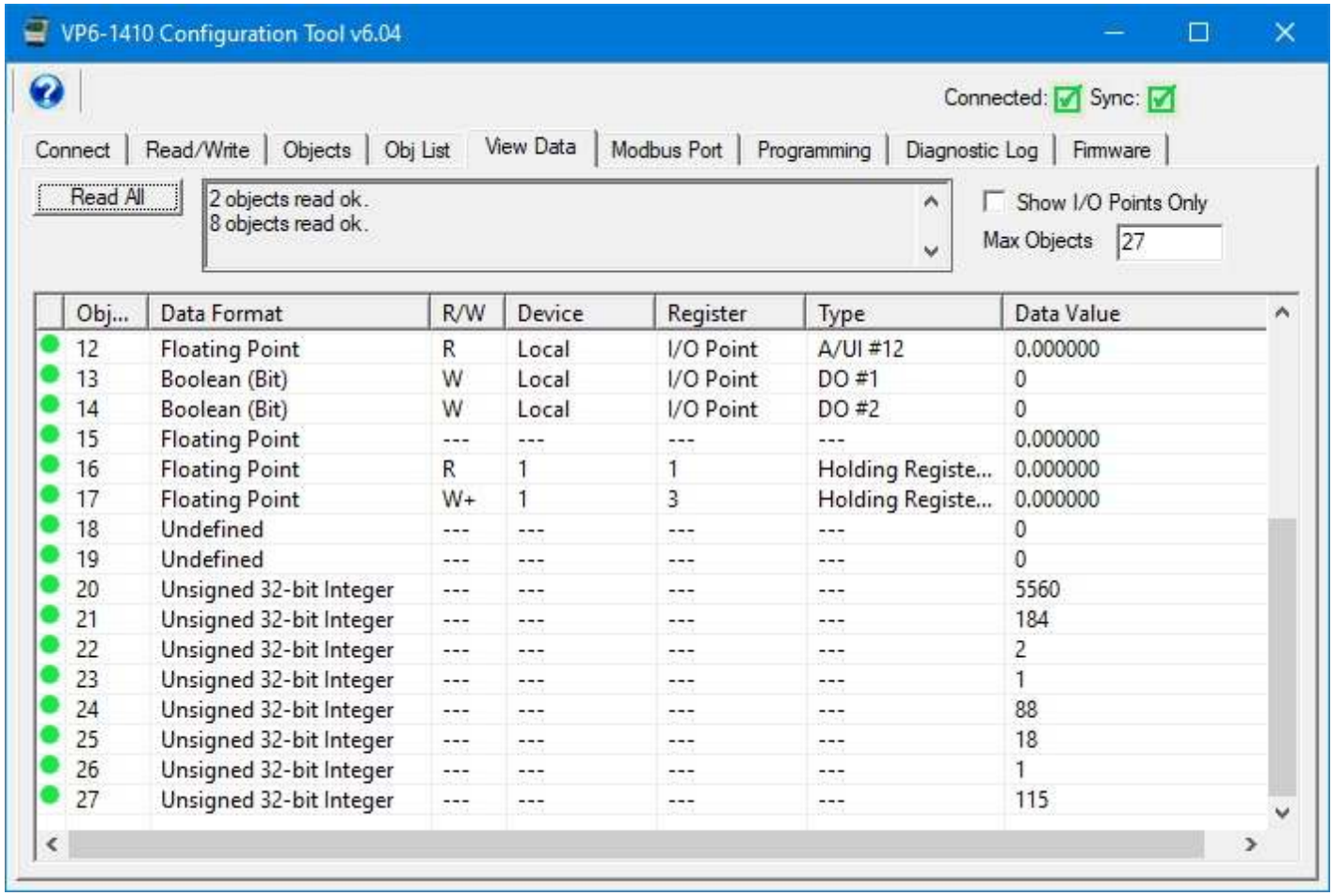

Example of usage: If you want to read via Modbus how many times the relay output DO 1 on the VP6-1410 (object 13) has turned on (cycle count), you would enter virtual link number 1013 in some otherwise unused object. Now read this object to see how many times the output has turned on.




Example of usage: If you want to read via Modbus how much time the relay output DO 1 on the VP6-1410 (object 13) has been in the on state, you would enter virtual link number 3013 in some otherwise unused object. Now read this object via Modbus to see how much time DO 1 has been in the on state.

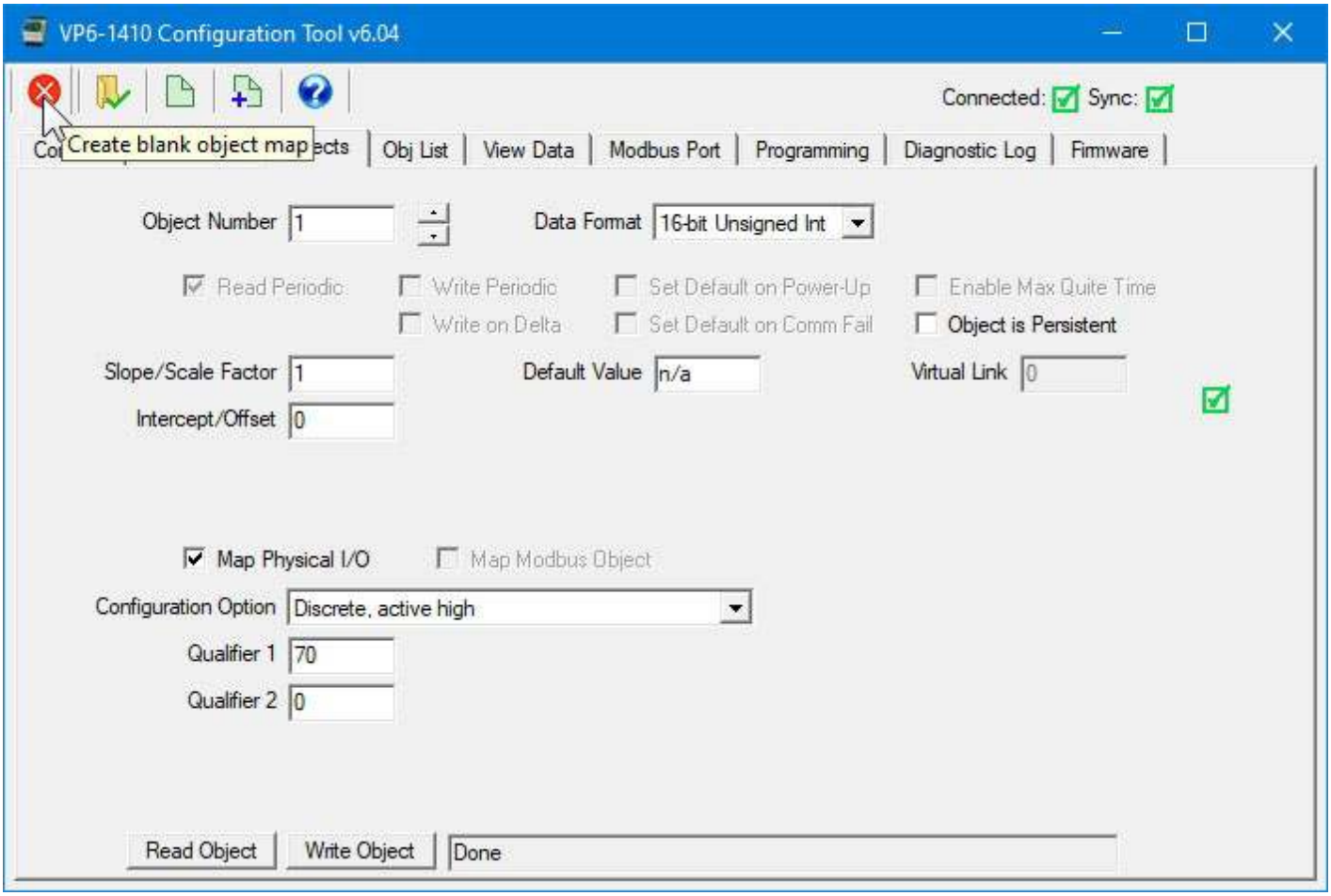



View of the above examples on the View Data page is illustrated below.

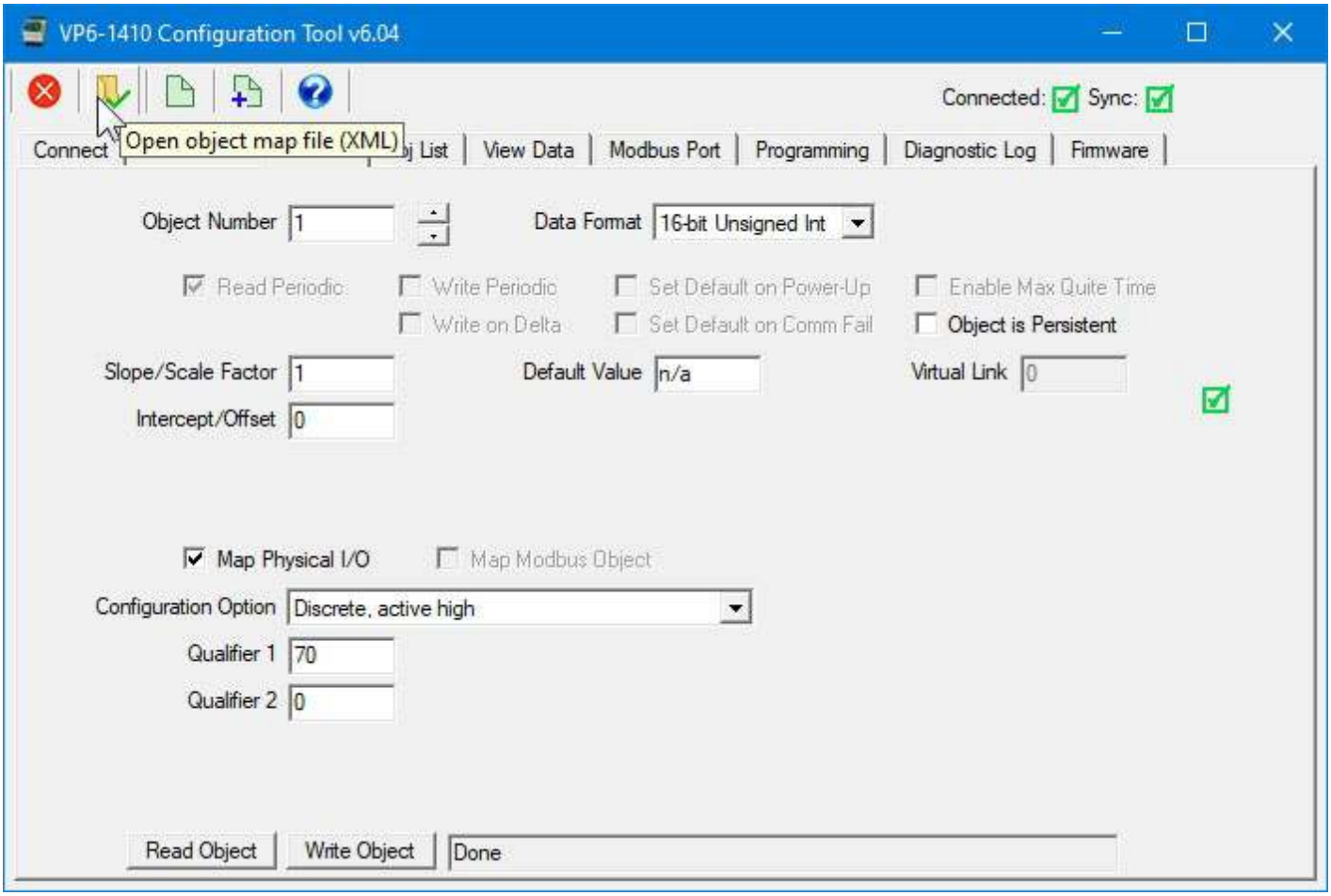



## 5.6 File Read/Write (Object Map XML)

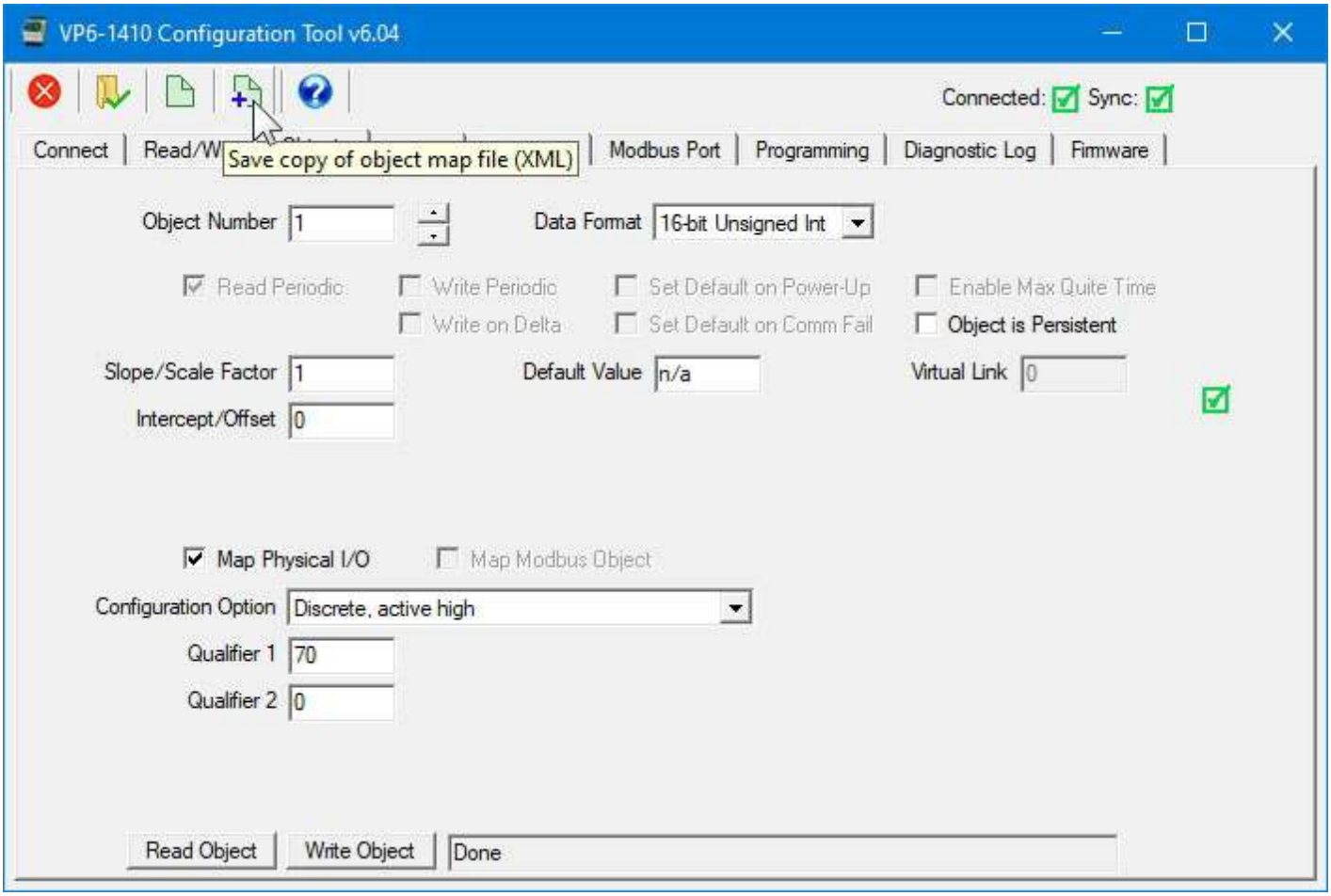
Click the "Clear" icon to erase all object mappings with the exception of physical I/O objects. The physical I/O objects will be restored to their default settings.



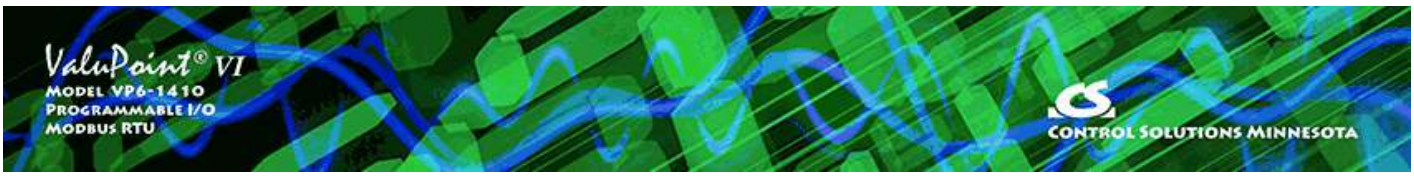

Click the File Open icon to import a previously saved XML file. Whatever object mappings had been saved are now loaded into the configuration tool. You may then subsequently write these settings to the VP6-1410 to replicate a previous configuration.




Click the "Save copy" icon to create a new XML file that contains all of the configuration settings currently displayed by the configuration tool. The icon just to the left will re-save an XML file that is already open.

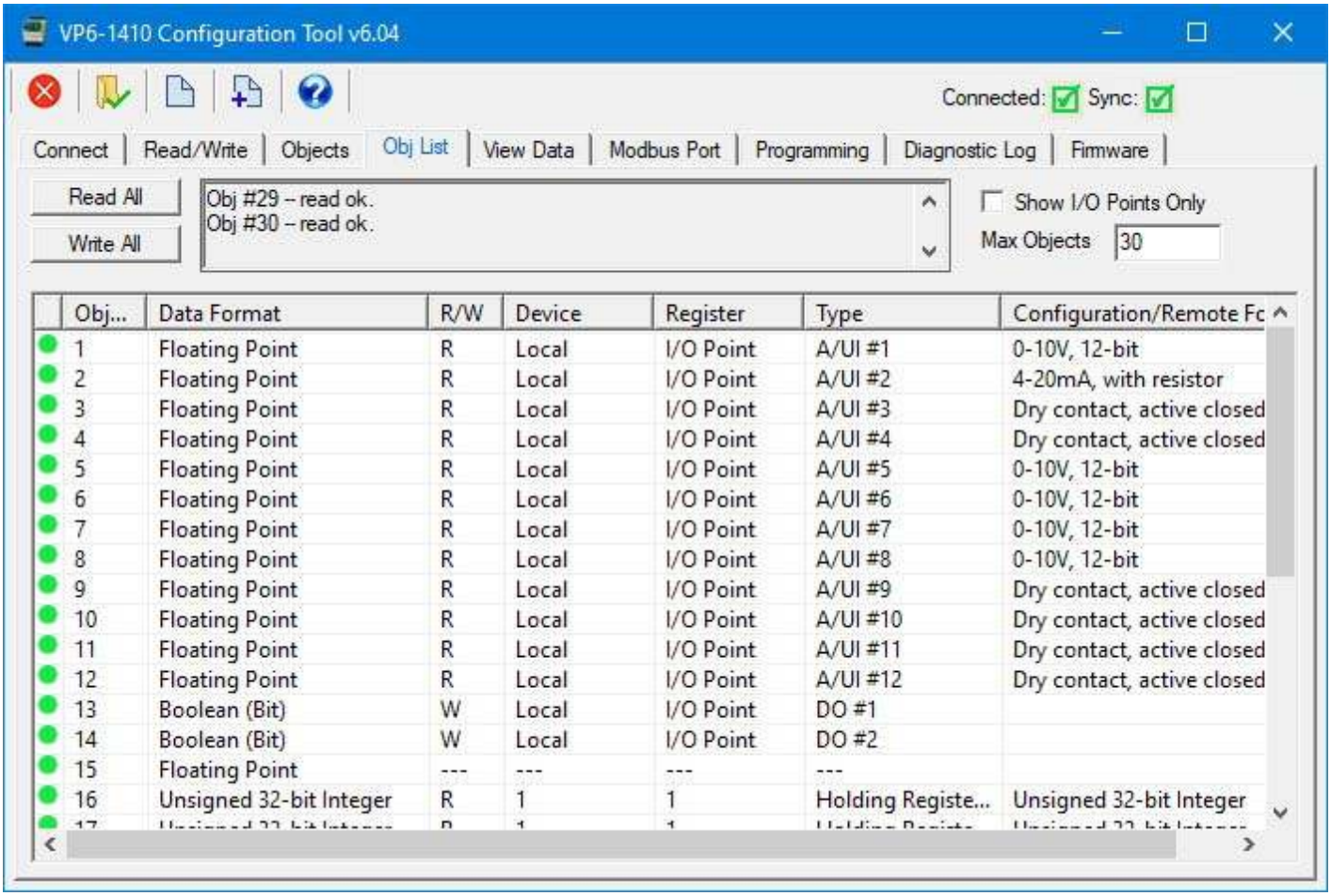



The file read or written from the Objects page will be an XML formatted file that contains all configuration information including Modbus port settings. The file read or written form the Obj List page will be a CSV file representing only the object mappings. This CSV file may be edited in a spread sheet program such as Excel to simplify the matter of creating a long list of mappings. Once a CSV file is imported on the Obj List page, it is a good idea to complete the configuration by making applicable settings on the Modbus Port page and then saving the complete configuration as an XML file on the Objects page (here).




# 6 Object List Page

## 6.1 Read All, Write All




Click 'Read All' to begin the process of reading all of the available object maps from the device. If the tool was not in sync with the device previously, the red dots in the first column will turn green as the map on that line is updated and synced with the device.

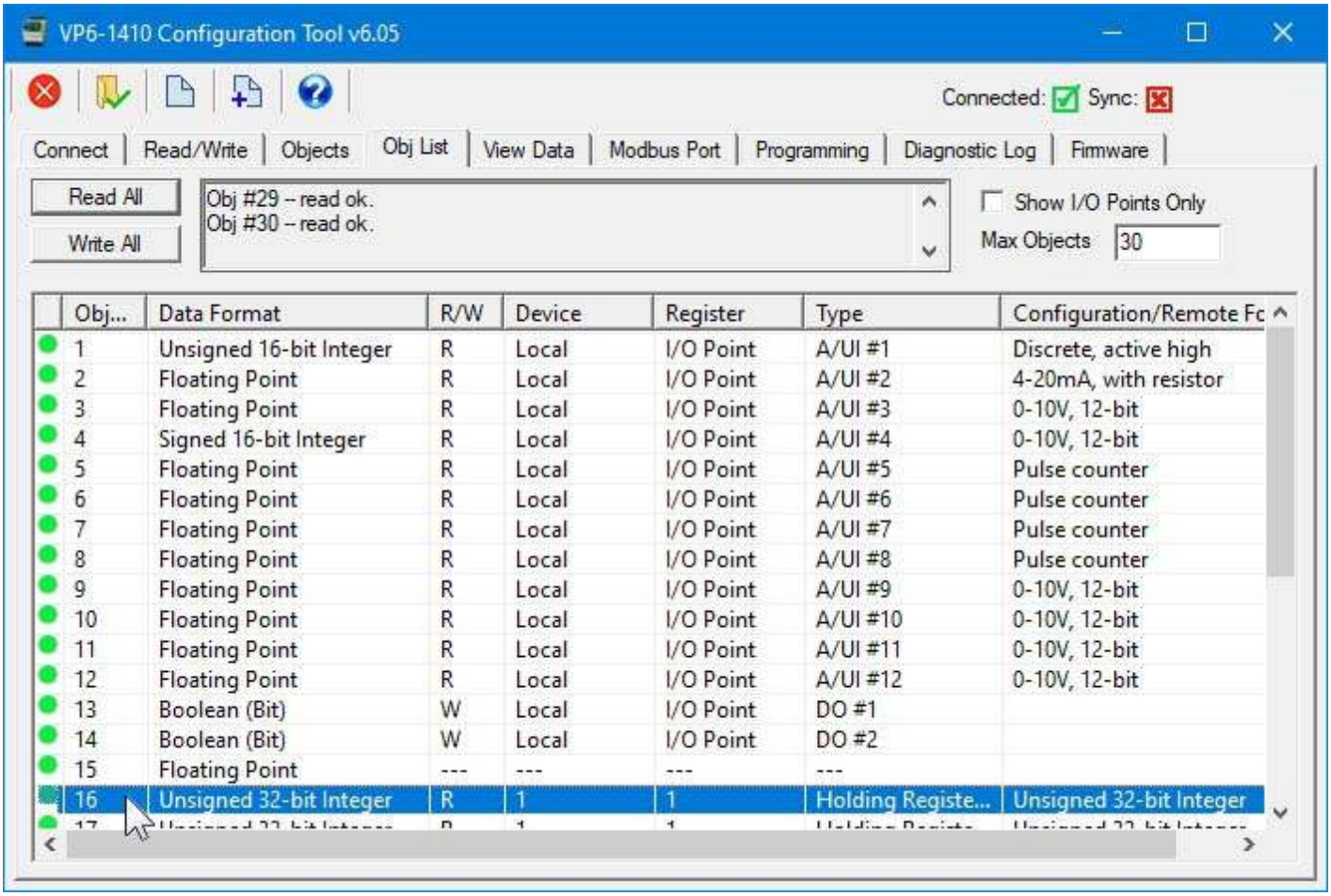
Click 'Write All' to begin the process of writing all object maps to the device. It would generally be assumed that you will only click this button after opening a file containing a previously defined configuration.

If you are only configuring the I/O points, check the "Show I/O Points Only" box and leave the Max Object count at 14. If you wish to show additional objects that are mapped as local objects or mapped for reading other Modbus devices, then uncheck

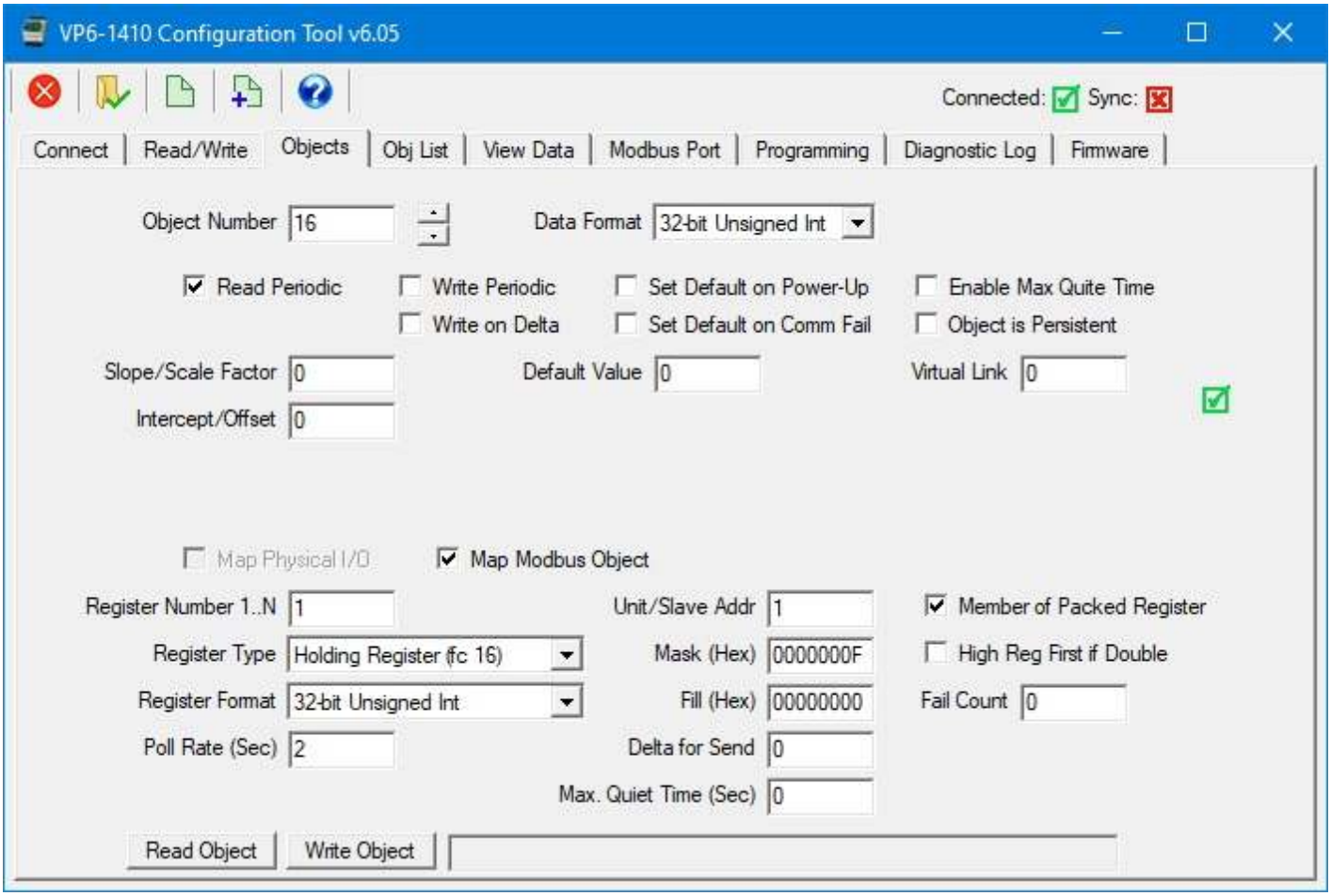
the box and enter the number of objects you would like to view. The configuration tool will not automatically read all 500 objects unless you tell it to because that will take some time and you are probably not using that many objects.

## 6.2 Select for Edit

Double click any line in the list of objects to be taken directly to the object map editing page for that object.




For example, double clicking object 16 above will take you to the Objects page for object 16 illustrated below.

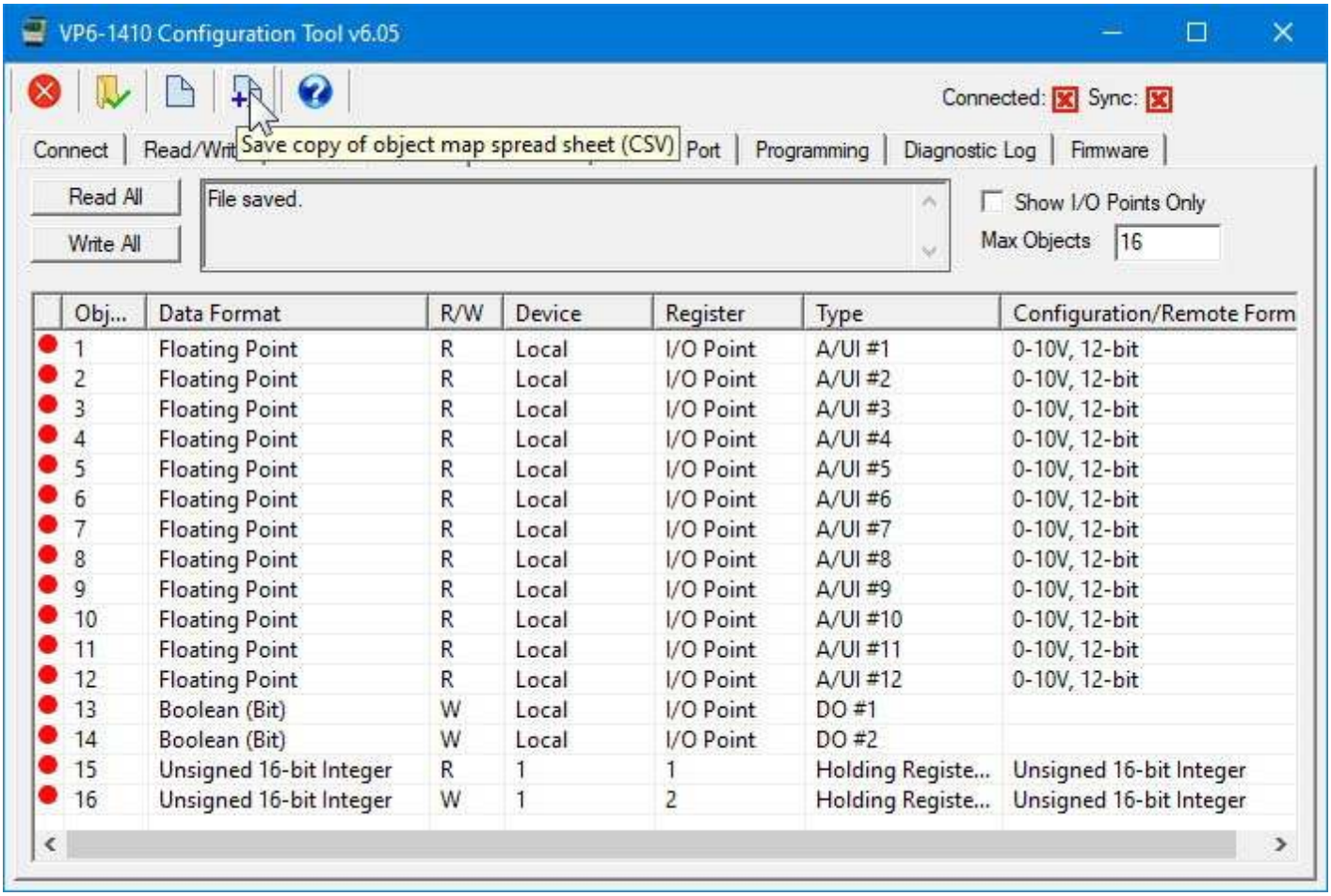



## 6.3 File Read/Write (Object Map CSV)

You have the option of using a spread sheet to create longer lists of object mappings when you want to read registers from other Modbus devices. Any spread sheet program (e.g. Excel) can be used to edit the spread sheet, but the columns must conform to a specific set of labels, and the file must be saved as a .CSV file in order to be imported into the configuration tool. Refer to Appendix C for the list of specific column labels required.

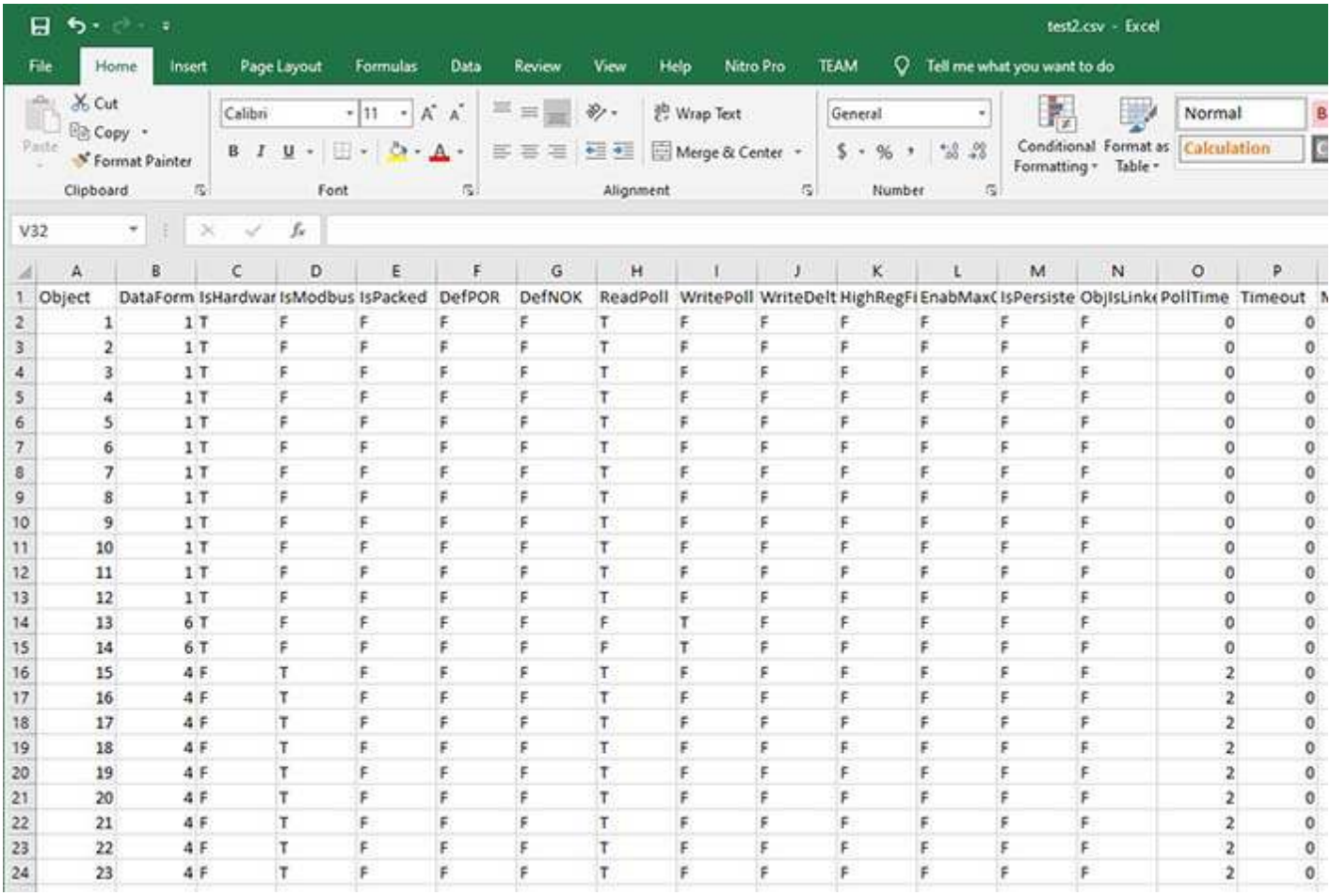
The easiest way to start the process is to open the configuration tool without necessarily connecting to any VP6-1410 device. The configuration tool will default to having all of the I/O points preconfigured. You will need to save these. In addition, to create a template for youself, create a mapped object or two that will read and write Modbus registers in other devices. In the example below, we have created one read map and one write map.

Once you have created a couple of maps using the configuration tool, click the Save icon to save your CSV file.




| Obj... | Data Format | R/W | Device | Register | Type | Configuration/Remote Form |
|---|---|---|---|---|---|---|
| 1 | Floating Point | R | Local | I/O Point | A/UI #1 | 0-10V, 12-bit |
| 2 | Floating Point | R | Local | I/O Point | A/UI #2 | 0-10V, 12-bit |
| 3 | Floating Point | R | Local | I/O Point | A/UI #3 | 0-10V, 12-bit |
| 4 | Floating Point | R | Local | I/O Point | A/UI #4 | 0-10V, 12-bit |
| 5 | Floating Point | R | Local | I/O Point | A/UI #5 | 0-10V, 12-bit |
| 6 | Floating Point | R | Local | I/O Point | A/UI #6 | 0-10V, 12-bit |
| 7 | Floating Point | R | Local | I/O Point | A/UI #7 | 0-10V, 12-bit |
| 8 | Floating Point | R | Local | I/O Point | A/UI #8 | 0-10V, 12-bit |
| 9 | Floating Point | R | Local | I/O Point | A/UI #9 | 0-10V, 12-bit |
| 10 | Floating Point | R | Local | I/O Point | A/UI #10 | 0-10V, 12-bit |
| 11 | Floating Point | R | Local | I/O Point | A/UI #11 | 0-10V, 12-bit |
| 12 | Floating Point | R | Local | I/O Point | A/UI #12 | 0-10V, 12-bit |
| 13 | Boolean (Bit) | W | Local | I/O Point | DO #1 | |
| 14 | Boolean (Bit) | W | Local | I/O Point | DO #2 | |
| 15 | Unsigned 16-bit Integer | R | 1 | 1 | Holding Registe... | Unsigned 16-bit Integer |
| 16 | Unsigned 16-bit Integer | W | 1 | 2 | Holding Registe... | Unsigned 16-bit Integer |

Now, using a spread sheet program, insert or add rows as necessary. Use the "fill down" feature to duplicate a number of read or write maps. Most of the column entries will remain the same from one map to the next. Once you have duplicated a bunch of maps, change only those columns that need to be changed, such as register number.



test2.csv - Excel

File Home Insert Page Layout Formulas Data Review View Help Nitro Pro TEAM Tell me what you want to do

Cut Copy Format Painter Paste Clipboard

Calibri 11 Font

Wrap Text Merge & Center Alignment

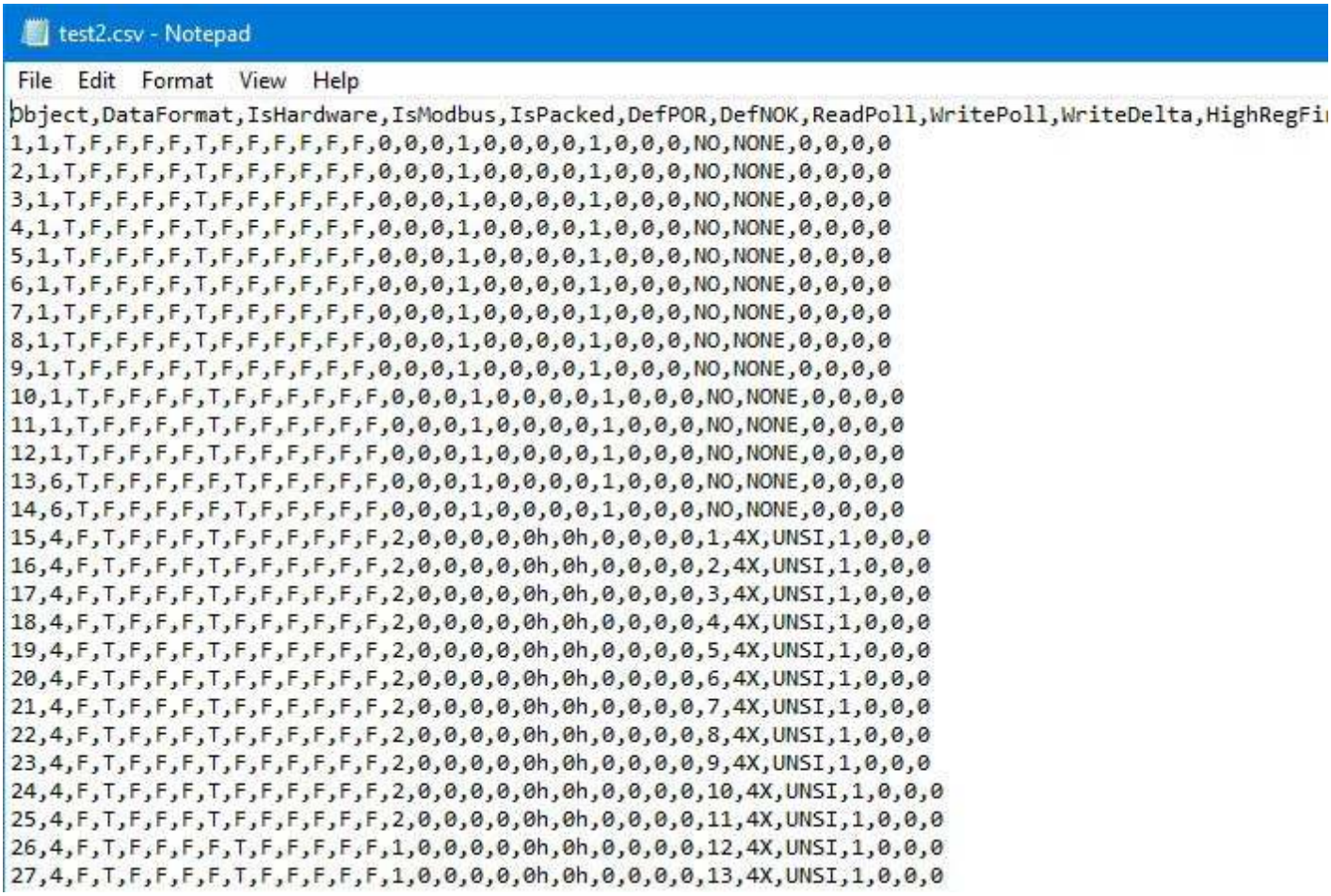
General Number

Conditional Formatting Format as Table Normal Calculation

V32

| | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Object | DataForm | IsHardwar | IsModbus | IsPacked | DefPOR | DefNOK | ReadPoll | WritePoll | WriteDelt | HighRegFi | EnabMaxC | IsPersiste | ObjIsLink | PollTime | Timeout |
| 2 | 1 | 1 | T | F | F | F | F | T | F | F | F | F | F | F | 0 | 0 |
| 3 | 2 | 1 | T | F | F | F | F | T | F | F | F | F | F | F | 0 | 0 |
| 4 | 3 | 1 | T | F | F | F | F | T | F | F | F | F | F | F | 0 | 0 |
| 5 | 4 | 1 | T | F | F | F | F | T | F | F | F | F | F | F | 0 | 0 |
| 6 | 5 | 1 | T | F | F | F | F | T | F | F | F | F | F | F | 0 | 0 |
| 7 | 6 | 1 | T | F | F | F | F | T | F | F | F | F | F | F | 0 | 0 |
| 8 | 7 | 1 | T | F | F | F | F | T | F | F | F | F | F | F | 0 | 0 |
| 9 | 8 | 1 | T | F | F | F | F | T | F | F | F | F | F | F | 0 | 0 |
| 10 | 9 | 1 | T | F | F | F | F | T | F | F | F | F | F | F | 0 | 0 |
| 11 | 10 | 1 | T | F | F | F | F | T | F | F | F | F | F | F | 0 | 0 |
| 12 | 11 | 1 | T | F | F | F | F | T | F | F | F | F | F | F | 0 | 0 |
| 13 | 12 | 1 | T | F | F | F | F | T | F | F | F | F | F | F | 0 | 0 |
| 14 | 13 | 6 | T | F | F | F | F | F | T | F | F | F | F | F | 0 | 0 |
| 15 | 14 | 6 | T | F | F | F | F | F | T | F | F | F | F | F | 0 | 0 |
| 16 | 15 | 4 | F | T | F | F | F | T | F | F | F | F | F | F | 2 | 0 |
| 17 | 16 | 4 | F | T | F | F | F | T | F | F | F | F | F | F | 2 | 0 |
| 18 | 17 | 4 | F | T | F | F | F | T | F | F | F | F | F | F | 2 | 0 |
| 19 | 18 | 4 | F | T | F | F | F | T | F | F | F | F | F | F | 2 | 0 |
| 20 | 19 | 4 | F | T | F | F | F | T | F | F | F | F | F | F | 2 | 0 |
| 21 | 20 | 4 | F | T | F | F | F | T | F | F | F | F | F | F | 2 | 0 |
| 22 | 21 | 4 | F | T | F | F | F | T | F | F | F | F | F | F | 2 | 0 |
| 23 | 22 | 4 | F | T | F | F | F | T | F | F | F | F | F | F | 2 | 0 |
| 24 | 23 | 4 | F | T | F | F | F | T | F | F | F | F | F | F | 2 | 0 |

Save your file as a CSV file. To verify that it was properly saved as CSV, you can open the .csv file using a simple text editor like Notepad. It should display as readable text only.

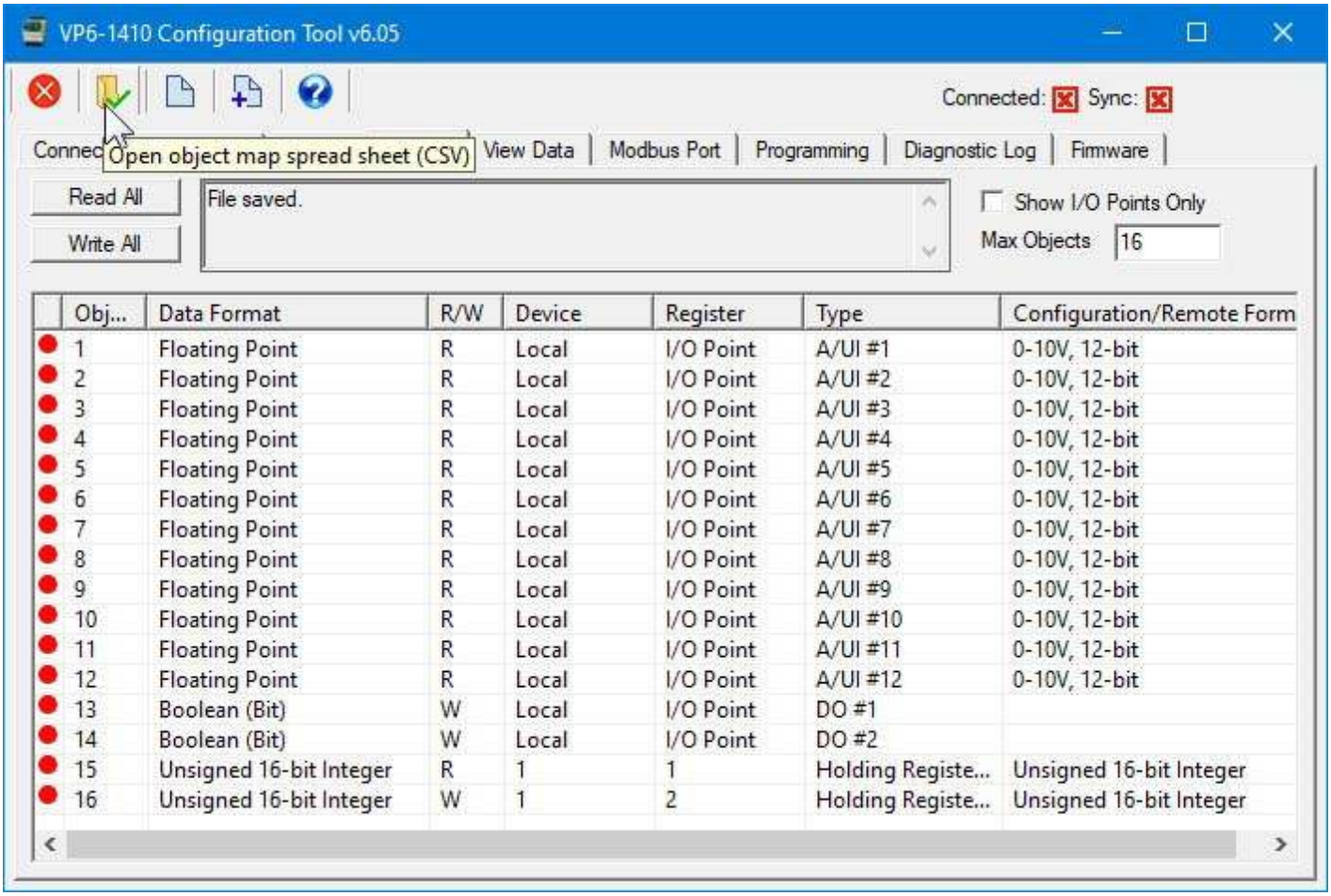


test2.csv - Notepad

File Edit Format View Help

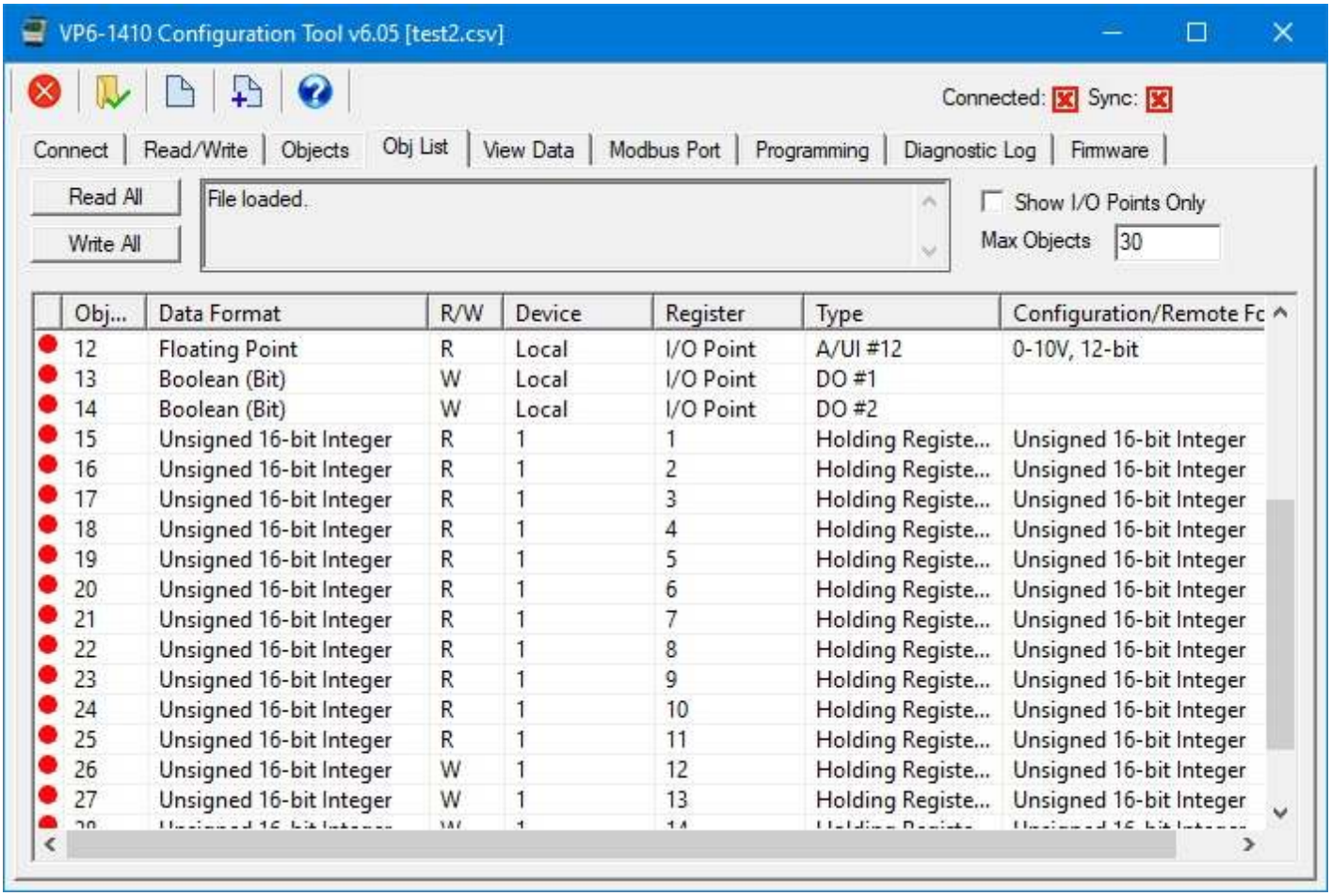
```
Object,DataFormat,IsHardware,IsModbus,IsPacked,DefPOR,DefNOK,ReadPoll,WritePoll,WriteDelta,HighRegFi
1,1,T,F,F,F,F,T,F,F,F,F,F,F,0,0,0,1,0,0,0,0,1,0,0,0,NO,NONE,0,0,0,0
2,1,T,F,F,F,F,T,F,F,F,F,F,F,0,0,0,1,0,0,0,0,1,0,0,0,NO,NONE,0,0,0,0
3,1,T,F,F,F,F,T,F,F,F,F,F,F,0,0,0,1,0,0,0,0,1,0,0,0,NO,NONE,0,0,0,0
4,1,T,F,F,F,F,T,F,F,F,F,F,F,0,0,0,1,0,0,0,0,1,0,0,0,NO,NONE,0,0,0,0
5,1,T,F,F,F,F,T,F,F,F,F,F,F,0,0,0,1,0,0,0,0,1,0,0,0,NO,NONE,0,0,0,0
6,1,T,F,F,F,F,T,F,F,F,F,F,F,0,0,0,1,0,0,0,0,1,0,0,0,NO,NONE,0,0,0,0
7,1,T,F,F,F,F,T,F,F,F,F,F,F,0,0,0,1,0,0,0,0,1,0,0,0,NO,NONE,0,0,0,0
8,1,T,F,F,F,F,T,F,F,F,F,F,F,0,0,0,1,0,0,0,0,1,0,0,0,NO,NONE,0,0,0,0
9,1,T,F,F,F,F,T,F,F,F,F,F,F,0,0,0,1,0,0,0,0,1,0,0,0,NO,NONE,0,0,0,0
10,1,T,F,F,F,F,T,F,F,F,F,F,F,0,0,0,1,0,0,0,0,1,0,0,0,NO,NONE,0,0,0,0
11,1,T,F,F,F,F,T,F,F,F,F,F,F,0,0,0,1,0,0,0,0,1,0,0,0,NO,NONE,0,0,0,0
12,1,T,F,F,F,F,T,F,F,F,F,F,F,0,0,0,1,0,0,0,0,1,0,0,0,NO,NONE,0,0,0,0
13,6,T,F,F,F,F,F,T,F,F,F,F,F,0,0,0,1,0,0,0,0,1,0,0,0,NO,NONE,0,0,0,0
14,6,T,F,F,F,F,F,T,F,F,F,F,F,0,0,0,1,0,0,0,0,1,0,0,0,NO,NONE,0,0,0,0
15,4,F,T,F,F,F,T,F,F,F,F,F,F,2,0,0,0,0,0h,0h,0,0,0,0,1,4X,UNSI,1,0,0,0
16,4,F,T,F,F,F,T,F,F,F,F,F,F,2,0,0,0,0,0h,0h,0,0,0,0,2,4X,UNSI,1,0,0,0
17,4,F,T,F,F,F,T,F,F,F,F,F,F,2,0,0,0,0,0h,0h,0,0,0,0,3,4X,UNSI,1,0,0,0
18,4,F,T,F,F,F,T,F,F,F,F,F,F,2,0,0,0,0,0h,0h,0,0,0,0,4,4X,UNSI,1,0,0,0
19,4,F,T,F,F,F,T,F,F,F,F,F,F,2,0,0,0,0,0h,0h,0,0,0,0,5,4X,UNSI,1,0,0,0
20,4,F,T,F,F,F,T,F,F,F,F,F,F,2,0,0,0,0,0h,0h,0,0,0,0,6,4X,UNSI,1,0,0,0
21,4,F,T,F,F,F,T,F,F,F,F,F,F,2,0,0,0,0,0h,0h,0,0,0,0,7,4X,UNSI,1,0,0,0
22,4,F,T,F,F,F,T,F,F,F,F,F,F,2,0,0,0,0,0h,0h,0,0,0,0,8,4X,UNSI,1,0,0,0
23,4,F,T,F,F,F,T,F,F,F,F,F,F,2,0,0,0,0,0h,0h,0,0,0,0,9,4X,UNSI,1,0,0,0
24,4,F,T,F,F,F,T,F,F,F,F,F,F,2,0,0,0,0,0h,0h,0,0,0,0,10,4X,UNSI,1,0,0,0
25,4,F,T,F,F,F,T,F,F,F,F,F,F,2,0,0,0,0,0h,0h,0,0,0,0,11,4X,UNSI,1,0,0,0
26,4,F,T,F,F,F,F,T,F,F,F,F,F,1,0,0,0,0,0h,0h,0,0,0,0,12,4X,UNSI,1,0,0,0
27,4,F,T,F,F,F,F,T,F,F,F,F,F,1,0,0,0,0,0h,0h,0,0,0,0,13,4X,UNSI,1,0,0,0
```

Now go back to the configuration tool and open your CSV file.



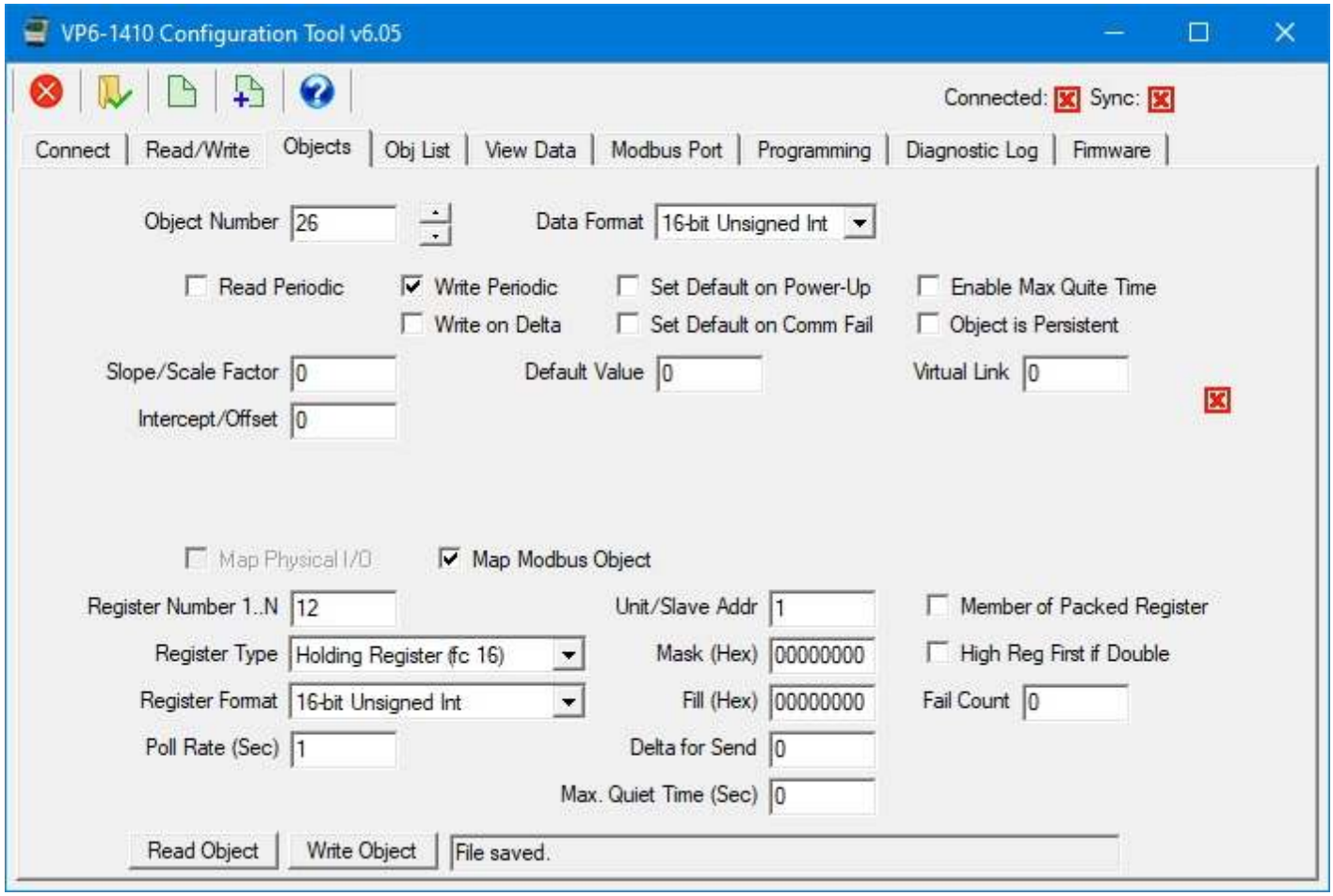

| Obj... | Data Format | R/W | Device | Register | Type | Configuration/Remote Form |
|---|---|---|---|---|---|---|
| 1 | Floating Point | R | Local | I/O Point | A/UI #1 | 0-10V, 12-bit |
| 2 | Floating Point | R | Local | I/O Point | A/UI #2 | 0-10V, 12-bit |
| 3 | Floating Point | R | Local | I/O Point | A/UI #3 | 0-10V, 12-bit |
| 4 | Floating Point | R | Local | I/O Point | A/UI #4 | 0-10V, 12-bit |
| 5 | Floating Point | R | Local | I/O Point | A/UI #5 | 0-10V, 12-bit |
| 6 | Floating Point | R | Local | I/O Point | A/UI #6 | 0-10V, 12-bit |
| 7 | Floating Point | R | Local | I/O Point | A/UI #7 | 0-10V, 12-bit |
| 8 | Floating Point | R | Local | I/O Point | A/UI #8 | 0-10V, 12-bit |
| 9 | Floating Point | R | Local | I/O Point | A/UI #9 | 0-10V, 12-bit |
| 10 | Floating Point | R | Local | I/O Point | A/UI #10 | 0-10V, 12-bit |
| 11 | Floating Point | R | Local | I/O Point | A/UI #11 | 0-10V, 12-bit |
| 12 | Floating Point | R | Local | I/O Point | A/UI #12 | 0-10V, 12-bit |
| 13 | Boolean (Bit) | W | Local | I/O Point | DO #1 | |
| 14 | Boolean (Bit) | W | Local | I/O Point | DO #2 | |
| 15 | Unsigned 16-bit Integer | R | 1 | 1 | Holding Registe... | Unsigned 16-bit Integer |
| 16 | Unsigned 16-bit Integer | W | 1 | 2 | Holding Registe... | Unsigned 16-bit Integer |

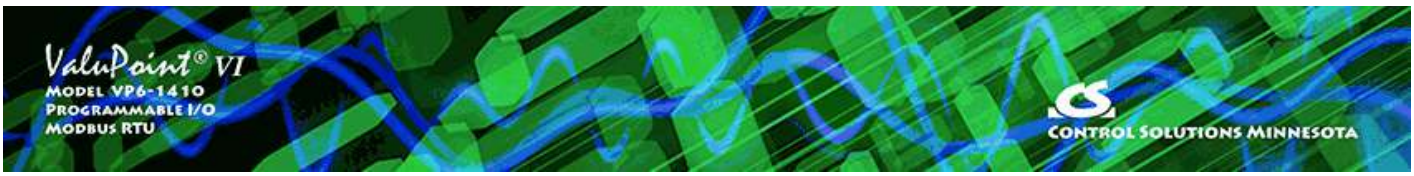
Scrolling down after importing the above example CSV file, you should now see the added read and write maps.




| Obj... | Data Format | R/W | Device | Register | Type | Configuration/Remote Fc |
|---|---|---|---|---|---|---|
| 12 | Floating Point | R | Local | I/O Point | A/UI #12 | 0-10V, 12-bit |
| 13 | Boolean (Bit) | W | Local | I/O Point | DO #1 | |
| 14 | Boolean (Bit) | W | Local | I/O Point | DO #2 | |
| 15 | Unsigned 16-bit Integer | R | 1 | 1 | Holding Registe... | Unsigned 16-bit Integer |
| 16 | Unsigned 16-bit Integer | R | 1 | 2 | Holding Registe... | Unsigned 16-bit Integer |
| 17 | Unsigned 16-bit Integer | R | 1 | 3 | Holding Registe... | Unsigned 16-bit Integer |
| 18 | Unsigned 16-bit Integer | R | 1 | 4 | Holding Registe... | Unsigned 16-bit Integer |
| 19 | Unsigned 16-bit Integer | R | 1 | 5 | Holding Registe... | Unsigned 16-bit Integer |
| 20 | Unsigned 16-bit Integer | R | 1 | 6 | Holding Registe... | Unsigned 16-bit Integer |
| 21 | Unsigned 16-bit Integer | R | 1 | 7 | Holding Registe... | Unsigned 16-bit Integer |
| 22 | Unsigned 16-bit Integer | R | 1 | 8 | Holding Registe... | Unsigned 16-bit Integer |
| 23 | Unsigned 16-bit Integer | R | 1 | 9 | Holding Registe... | Unsigned 16-bit Integer |
| 24 | Unsigned 16-bit Integer | R | 1 | 10 | Holding Registe... | Unsigned 16-bit Integer |
| 25 | Unsigned 16-bit Integer | R | 1 | 11 | Holding Registe... | Unsigned 16-bit Integer |
| 26 | Unsigned 16-bit Integer | W | 1 | 12 | Holding Registe... | Unsigned 16-bit Integer |
| 27 | Unsigned 16-bit Integer | W | 1 | 13 | Holding Registe... | Unsigned 16-bit Integer |

Double clicking on any line on the list page will take you to the Objects page where you can see the details of that map.

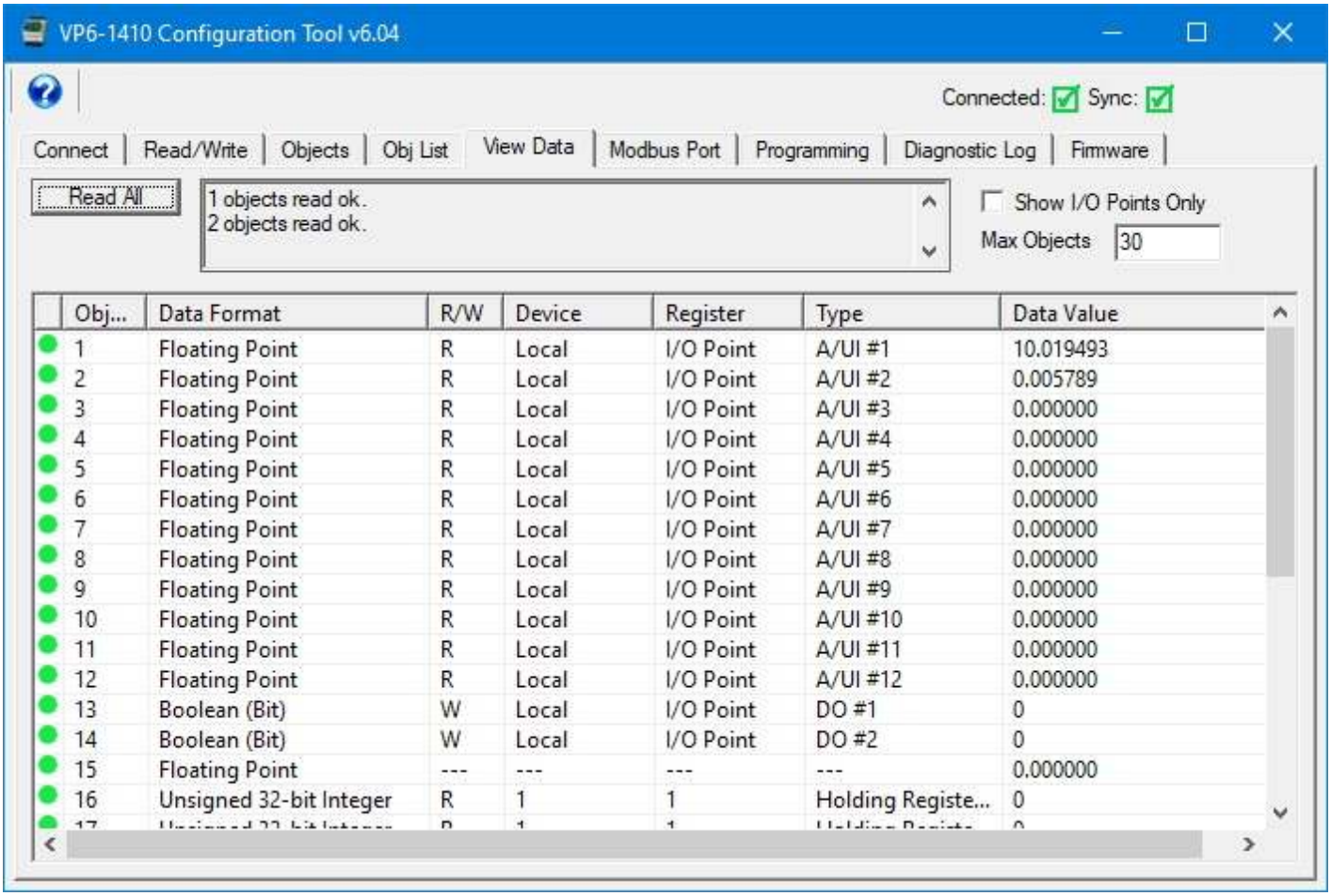

VP6-1410 Configuration Tool v6.05
Connected:
Sync:
Connect
Read/Write
Objects
Obj List
View Data
Modbus Port
Programming
Diagnostic Log
Firmware
Object Number
26
Data Format
16-bit Unsigned Int
Read Periodic
Write Periodic
Write on Delta
Set Default on Power-Up
Set Default on Comm Fail
Enable Max Quite Time
Object is Persistent
Slope/Scale Factor
0
Intercept/Offset
0
Default Value
0
Virtual Link
0
Map Physical I/O
Map Modbus Object
Register Number 1..N
12
Register Type
Holding Register (fc 16)
Register Format
16-bit Unsigned Int
Poll Rate (Sec)
1
Unit/Slave Addr
1
Mask (Hex)
00000000
Fill (Hex)
00000000
Delta for Send
0
Max. Quiet Time (Sec)
0
Member of Packed Register
High Reg First if Double
Fail Count
0
Read Object
Write Object
File saved.




# 7 View Data Page

## 7.1 Read All

Click 'Read All' to cause the tool to query all data objects in the ValuPoint device. A brief summary of the object's mapping is displayed along with its present data value. Progress of the 'Read All' process will scroll through the log window at the top.




| Obj... | Data Format | R/W | Device | Register | Type | Data Value |
|---|---|---|---|---|---|---|
| 1 | Floating Point | R | Local | I/O Point | A/UI #1 | 10.019493 |
| 2 | Floating Point | R | Local | I/O Point | A/UI #2 | 0.005789 |
| 3 | Floating Point | R | Local | I/O Point | A/UI #3 | 0.000000 |
| 4 | Floating Point | R | Local | I/O Point | A/UI #4 | 0.000000 |
| 5 | Floating Point | R | Local | I/O Point | A/UI #5 | 0.000000 |
| 6 | Floating Point | R | Local | I/O Point | A/UI #6 | 0.000000 |
| 7 | Floating Point | R | Local | I/O Point | A/UI #7 | 0.000000 |
| 8 | Floating Point | R | Local | I/O Point | A/UI #8 | 0.000000 |
| 9 | Floating Point | R | Local | I/O Point | A/UI #9 | 0.000000 |
| 10 | Floating Point | R | Local | I/O Point | A/UI #10 | 0.000000 |
| 11 | Floating Point | R | Local | I/O Point | A/UI #11 | 0.000000 |
| 12 | Floating Point | R | Local | I/O Point | A/UI #12 | 0.000000 |
| 13 | Boolean (Bit) | W | Local | I/O Point | DO #1 | 0 |
| 14 | Boolean (Bit) | W | Local | I/O Point | DO #2 | 0 |
| 15 | Floating Point | --- | --- | --- | --- | 0.000000 |
| 16 | Unsigned 32-bit Integer | R | 1 | 1 | Holding Registe... | 0 |

If you are only looking at the I/O points, check the "Show I/O Points Only" box and leave the Max Object count at 14. If you wish to show additional objects that are mapped as local objects or mapped for reading other Modbus devices, then uncheck the box and enter the number of objects you would like to view. The configuration tool will not automatically read all 500 objects unless you tell it to because that will take some time and you are probably not using that many objects.

## 7.2 Data Definitions

The Object column indicates the data object number.

The Data Format column indicates the native data format configured for this object. The object may be read in any of multiple formats depending on which Modbus holding register range you use to access the object.

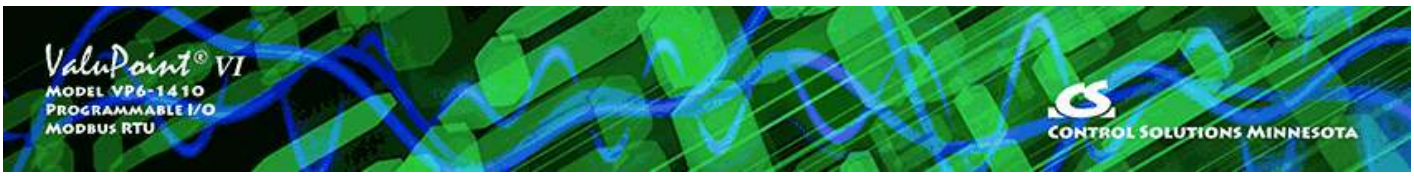
The R/W column indicates the following: R means periodic read, W means periodic write, W+ means write on delta. RW+ means a combination of those, etc.

The Device column will show 'Local', or show what remote slave device address this object is mapped to. 'Local' means a physical I/O point on the local ValuPoint.

The Register column is significant only when the object is mapped via the Modbus RTU master to a remote slave device. The register number that will be queried in the remote slave device is listed here..

Type shows the physical I/O point locally, or the remote register type, such as "Holding Register".

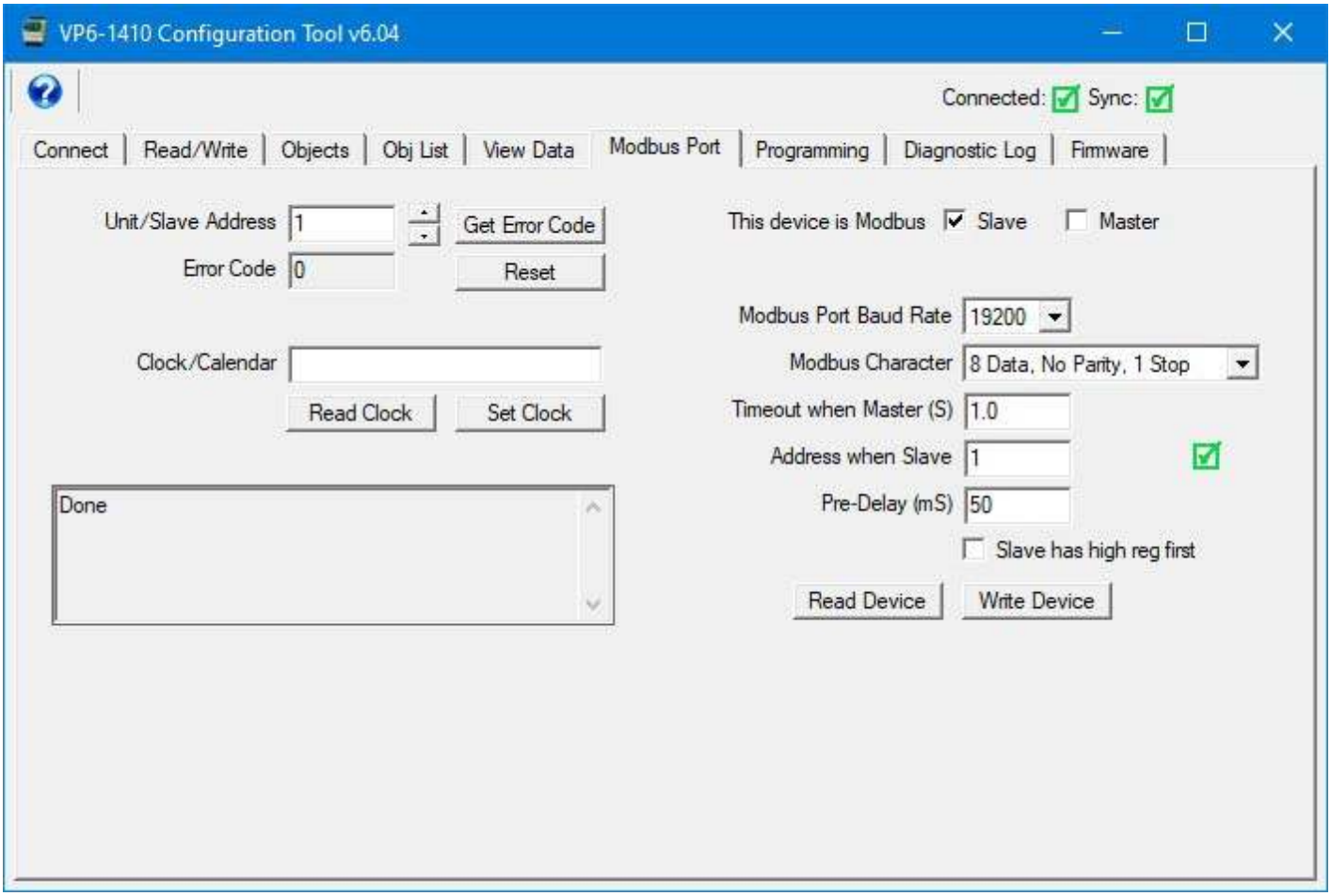
Data Value shows a representation of the present value of the data object in the ValuPoint.




# 8 Modbus PortPage

## 8.1 Set Modbus Port Parameters

Click 'Read Device' to get the tool synced to the device. Then set parameters as applicable. Once you have made the applicable changes, click Write Device.




Select whether the ValuPoint will be a Modbus Master or Slave. Remember that there can be only one master on a Modbus RTU network. If you switch from slave to master, upon clicking 'Write Device', you may need to go back to the Connect page and follow the process for entering configuration mode when device is configured as master. You are automatically always effectively in configuration mode when the device is configured to be a slave, but you will exit configuration mode when switching to master for the first time.

Select the Modbus port communication parameters, including baud rate, character

format, and timeout. The timeout is the amount of time the ValuPoint will wait for a Modbus slave to respond if configured as master.

The ‘Address when Slave’ only applies if Slave is checked and the ValuPoint is going to be a Modbus slave responding to another master on the Modbus RTU network.

The 'Slave has high reg first' should be checked of you want the most significant register of 32-bit register pair data to be read first when the VP6-1410 is operating as a slave.

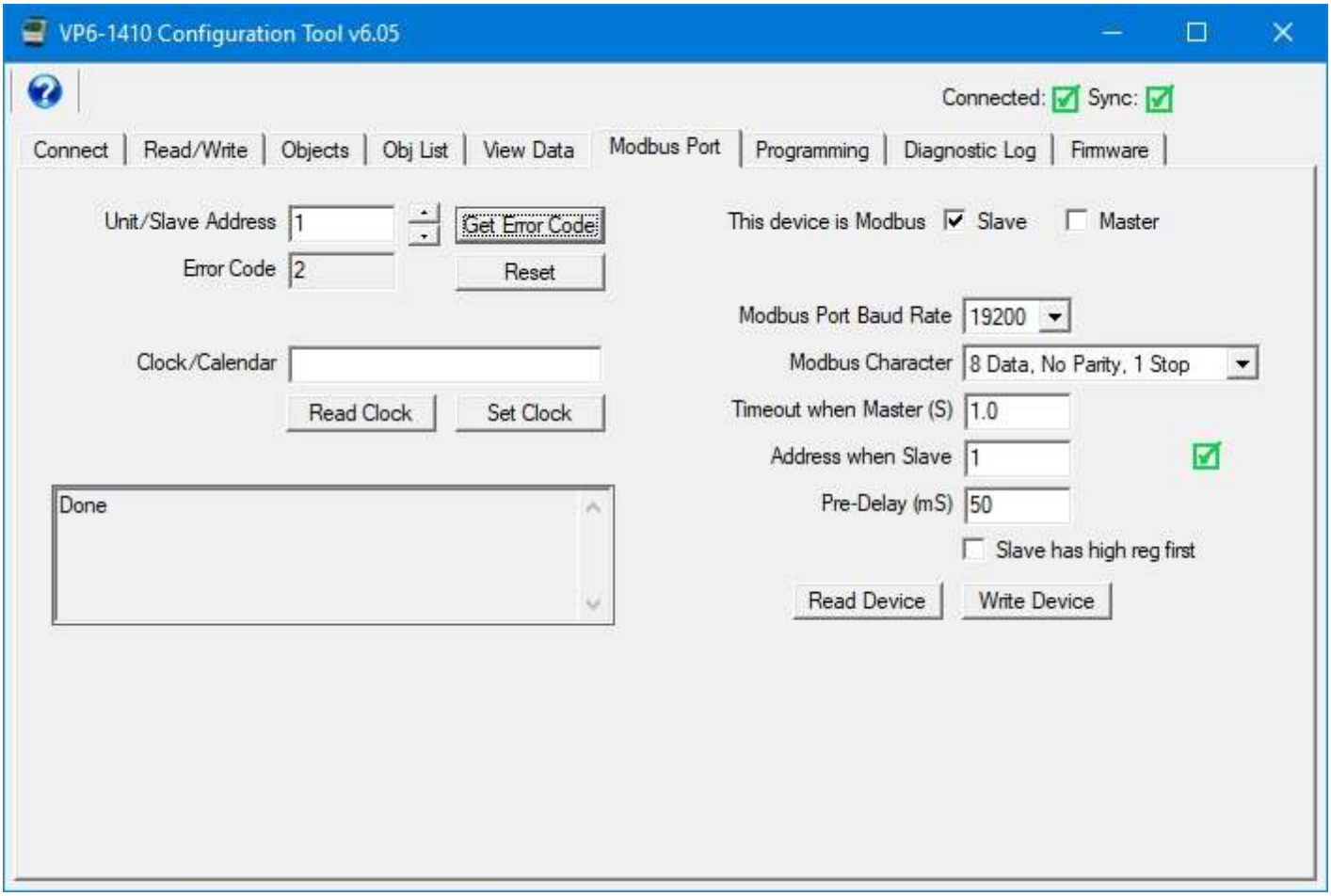
Pre-Delay (in milliseconds) specifies an amount of time the ValuPoint will wait before transmitting. The ValuPoint is fast enough to overrun some Modbus devices by either sending the next request too soon after the previous response, or by responding too fast as a slave. The packet overrun will manifest itself as no-response errors and/or CRC errors on the Modbus link. If you are seeing such errors and you have Pre-Delay set to zero, set it to at least 50 mS and continue testing.

**IMPORTANT:** Do not change device mode to Modbus master until AFTER you have created some Modbus register mappings for the master to act upon. These are set up on the Objects page. If you switch to master mode without anything for the master to do, you will have trouble entering configuration mode.

If you are having trouble communicating, and you suspect configuration parameters may be something other than what you expect, it should be noted that the device will ALWAYS be Modbus slave, communicating at 19200 baud N81, with slave address 247, for about 3 seconds after power up. During this time, you can click Read on the Modbus page to see what the port parameters in the device are. If they are not what you want, you can power cycle one more time and write new Modbus port parameters during that 3 second window. Once you have switched the device back to slave mode, you will be able to fully access the device.

## 8.2 Check/Reset Errors

Enter or scroll to a desired Modbus slave address and click ‘Get Errors’ to see the error counts recorded by ValuPoint for that device. When configured as slave, errors will be logged as slave address 1 for purposes of error counting regardless of what the actual slave address is. When configured as master, counts are kept for each slave address that the master is querying; however, the counts can only be read while in configuration mode. This means you can only observe error counts after the ValuPoint has operated as master for some time, and then temporarily placed in slave mode by entering configuration mode from the Connect page.




Click 'Reset' to clear the error counts and message count for the Modbus device whose address is currently displayed.

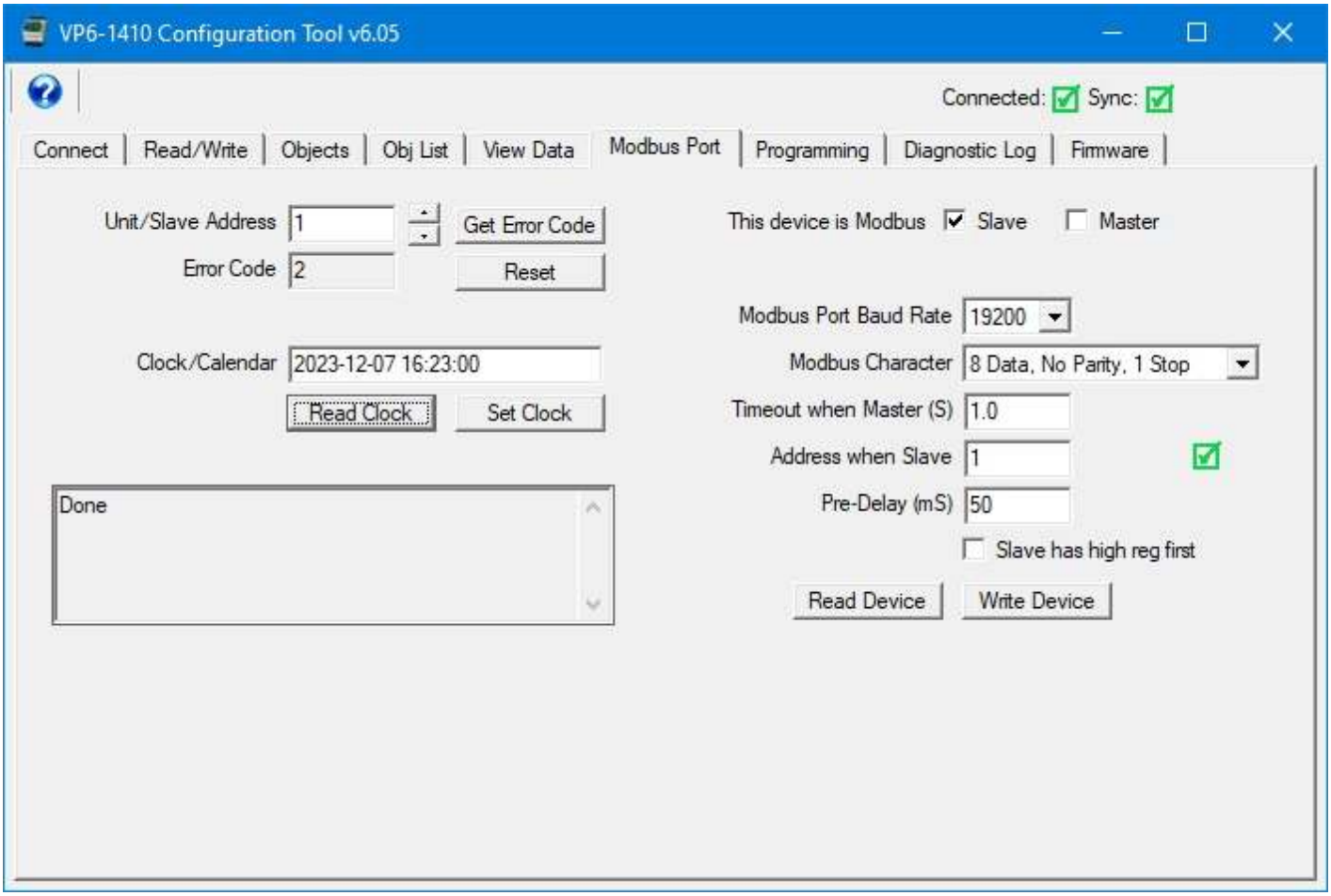
Error codes will be as follows, with lower numbered codes being the standard Modbus exception codes as defined by Modbus protocol, and the higher numbered codes being indicators of non-exception type errorsl.

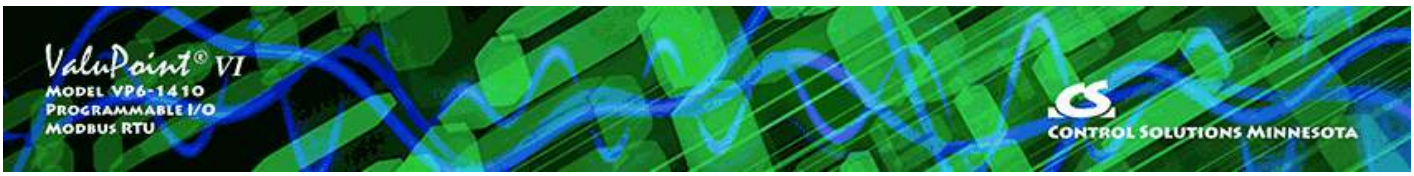
| Standard Modbus Exception Codes | | |
|---|---|---|
| 1 | Illegal Function | The function code received in the query is not recognized by the slave or is not allowed by the slave. |
| 2 | Illegal Data Address | The data address (register number) received in the query is not an allowed address for the slave, i.e., the register does not exist. If multiple registers were requested, at least one was not permitted. |
| 3 | Illegal Data Value | The value contained in the query's data field is not acceptable to the slave. |
| 4 | Slave Device Failure | An unrecoverable error occurred while the slave was attempting to perform the requested action |
| 6 | Slave Device Busy | The slave is engaged in processing a long-duration command. The master should try again later. |
| 10 | Gateway Path Unavailable | Specialized use in conjunction with gateways, usually means the gateway is misconfigured or overloaded |

| | | |
|---|---|---|
| 11 | Gateway Target Device Failed to Respond | Specialized use in conjunction with gateways, indicates no response was received from the target device. |
| ValuPoint Specific Codes (indicating non-exception errors) | | |
| 129 | No Response | Valid only if ValuPoint is Modbus master, indicates the addressed slave has failed to respond 1 or more times. |
| 130 | CRC Error | Valid only if ValuPoint is Modbus master, indicates that a CRC error in slave's reply has been found 1 or more times. |
| 131 | No Response/CRC | Simply indicates both of the above (129, 130) are true. |

## 8.3 Read/Write Clock/Calendar

Click the Read Clock button to see what the current clock/calendar value is. Note the format of the timestamp. To change the value, you must enter a new date and time in this format, then click Set Clock.



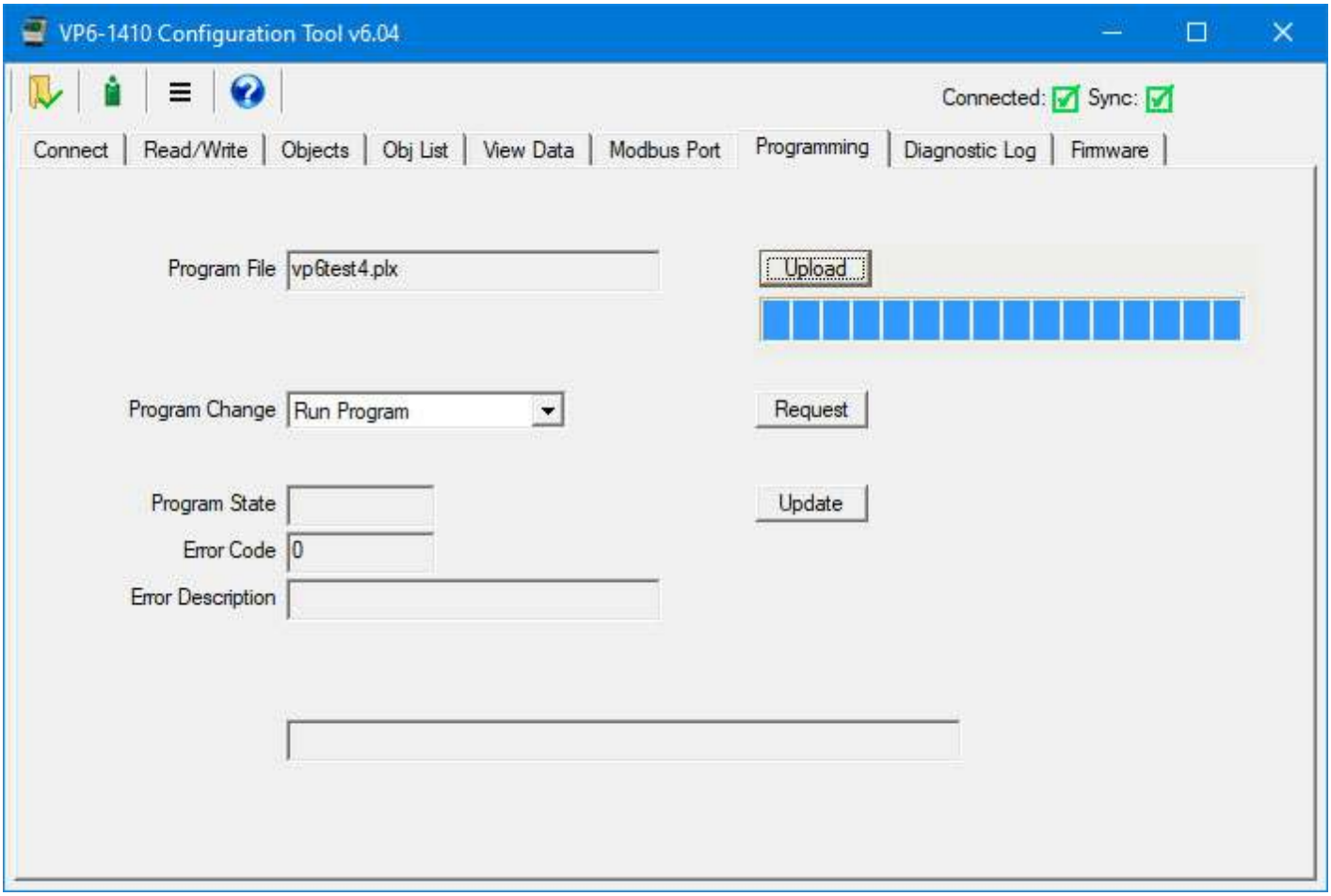



# 9 Programming Page

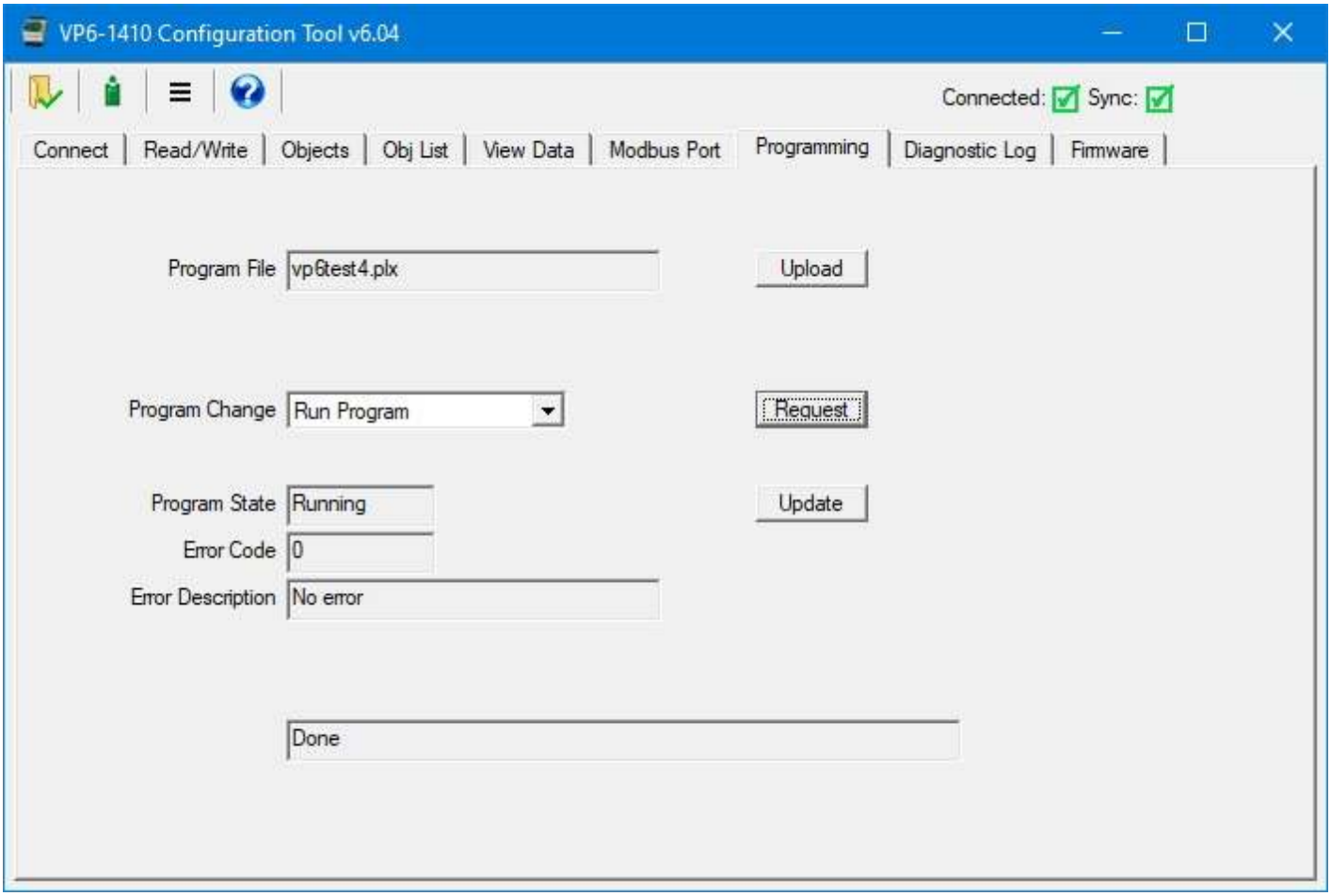
## 9.1 Program Loading and Execution

Click on the file folder icon at the top left to open a file. The file open dialog will appear. Select a .plx file from the list. If you do not yet have any programs compiled, you will need to use the program editing tools to create and compile a program.

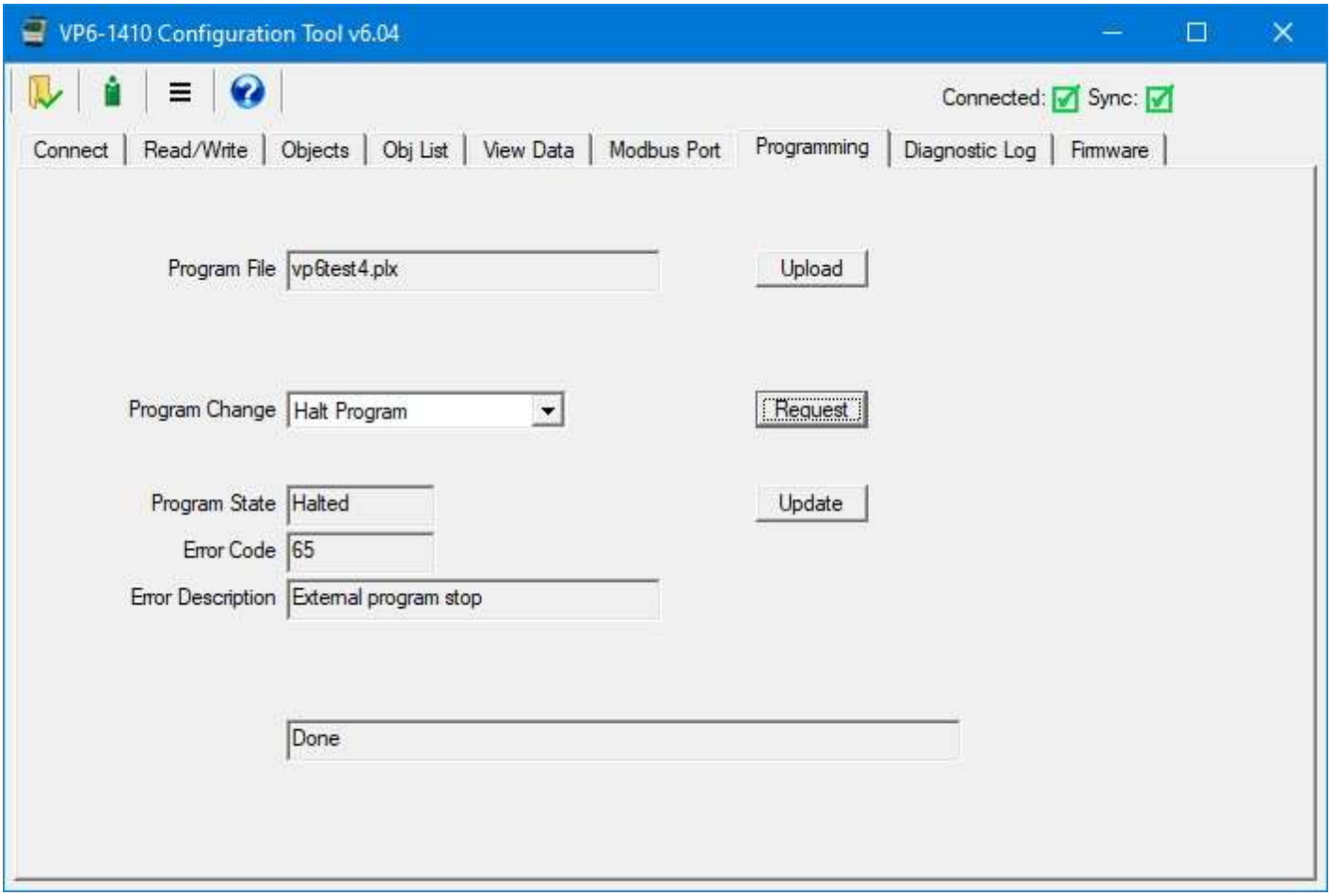
After a program (.plx file) has been opened, click the Upload button to send that program to the ValuPoint. A progress bar will indicate program loading progress.




To invoke execution of the program, select 'Run Program' from the Program Change list, and then click Request.

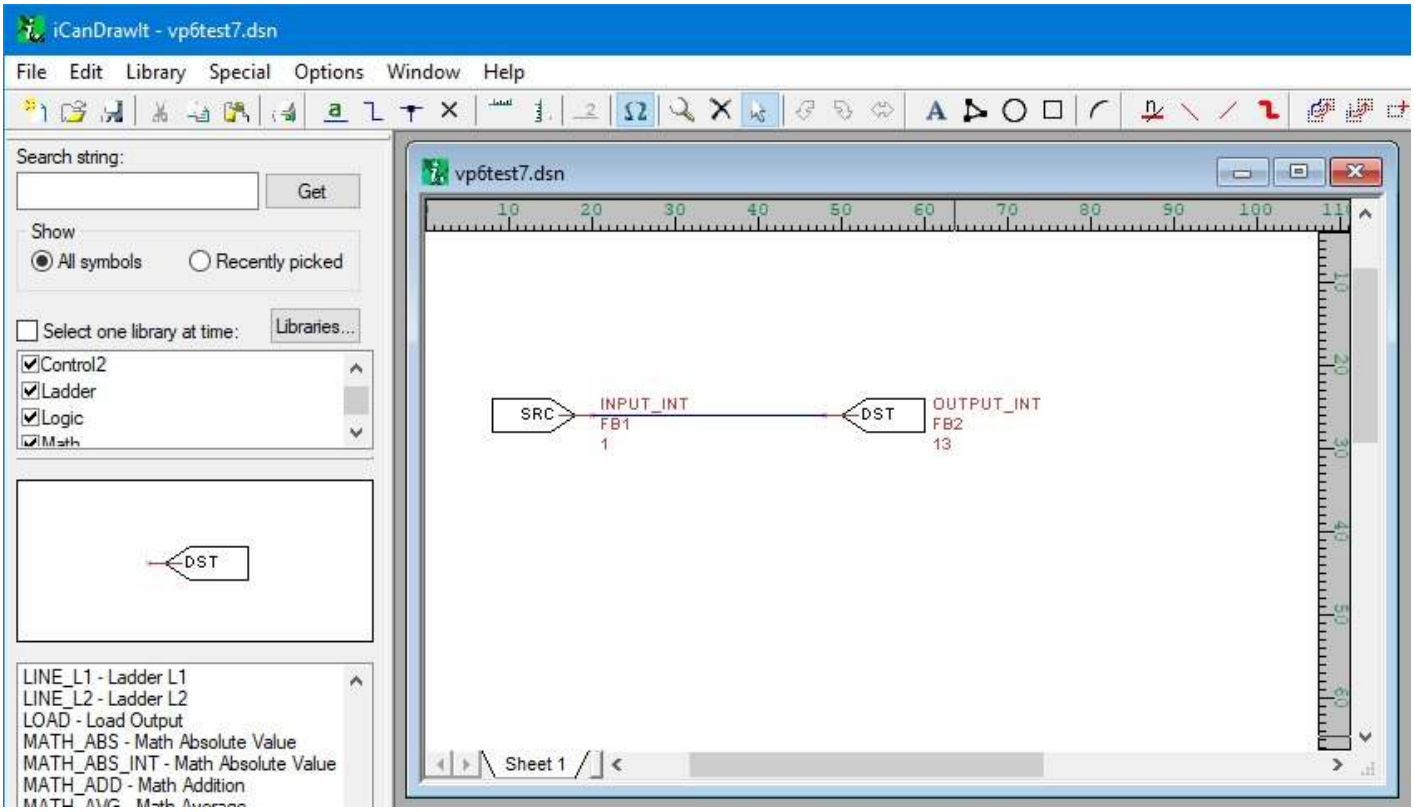



To stop the program, select Halt Program and click Request.




If the program encounters a fatal error during execution, its error code and description will be displayed after clicking the Update button. The Update button causes the tool to query the ValuPoint device for status.

## 9.2 Program Editing and Debugging

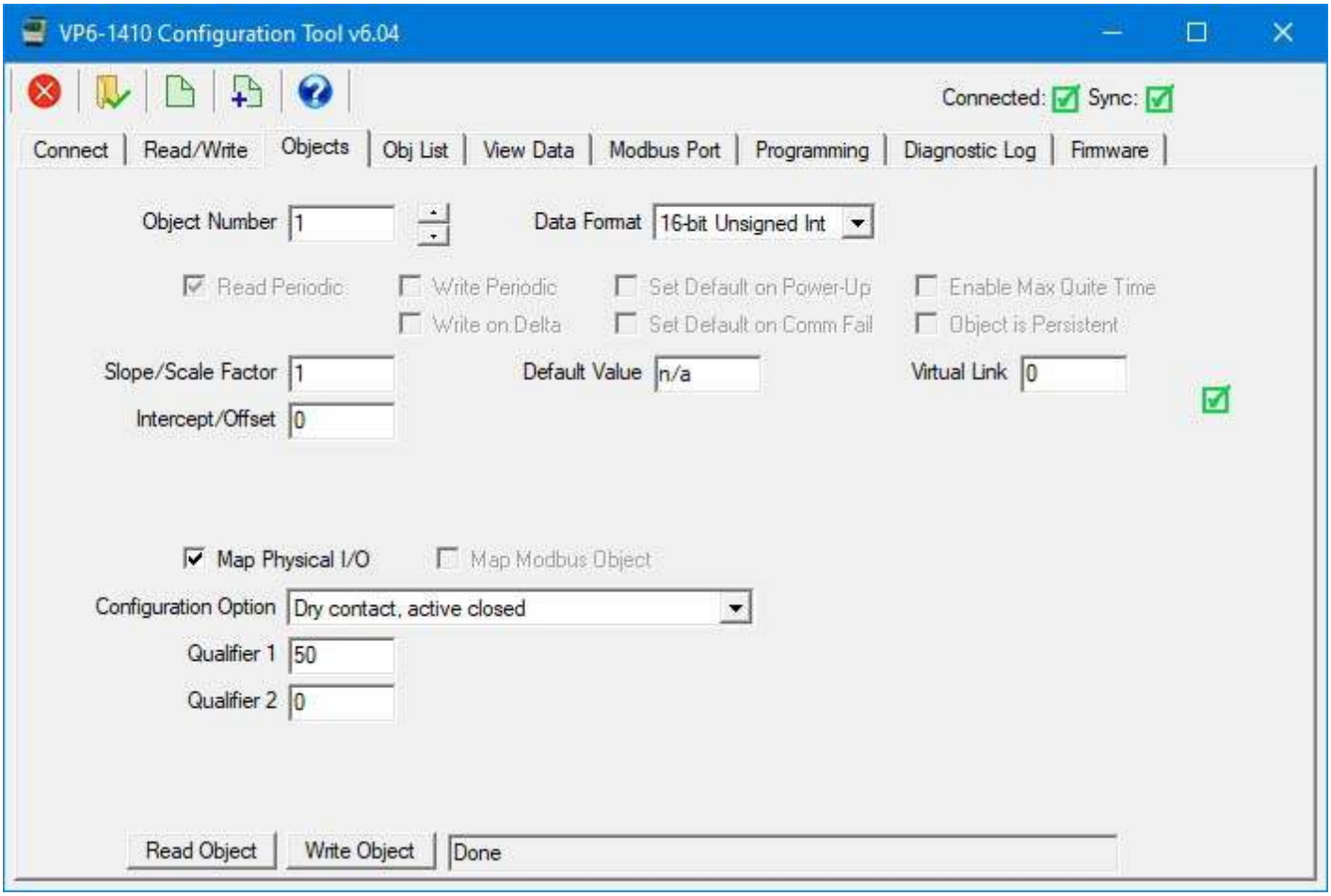



Click the green "i" icon next to the folder icon to open the i.CanDrawIt graphical programming tool that is illustrated below. It has its own set of help pages. Click on the "?" icon in that tool for more about programming. (Note that i.CanDrawIt is a second software package installed after the ValuPoint configuration tool - if you have trouble starting up i.CanDrawIt, be sure it was installed.)

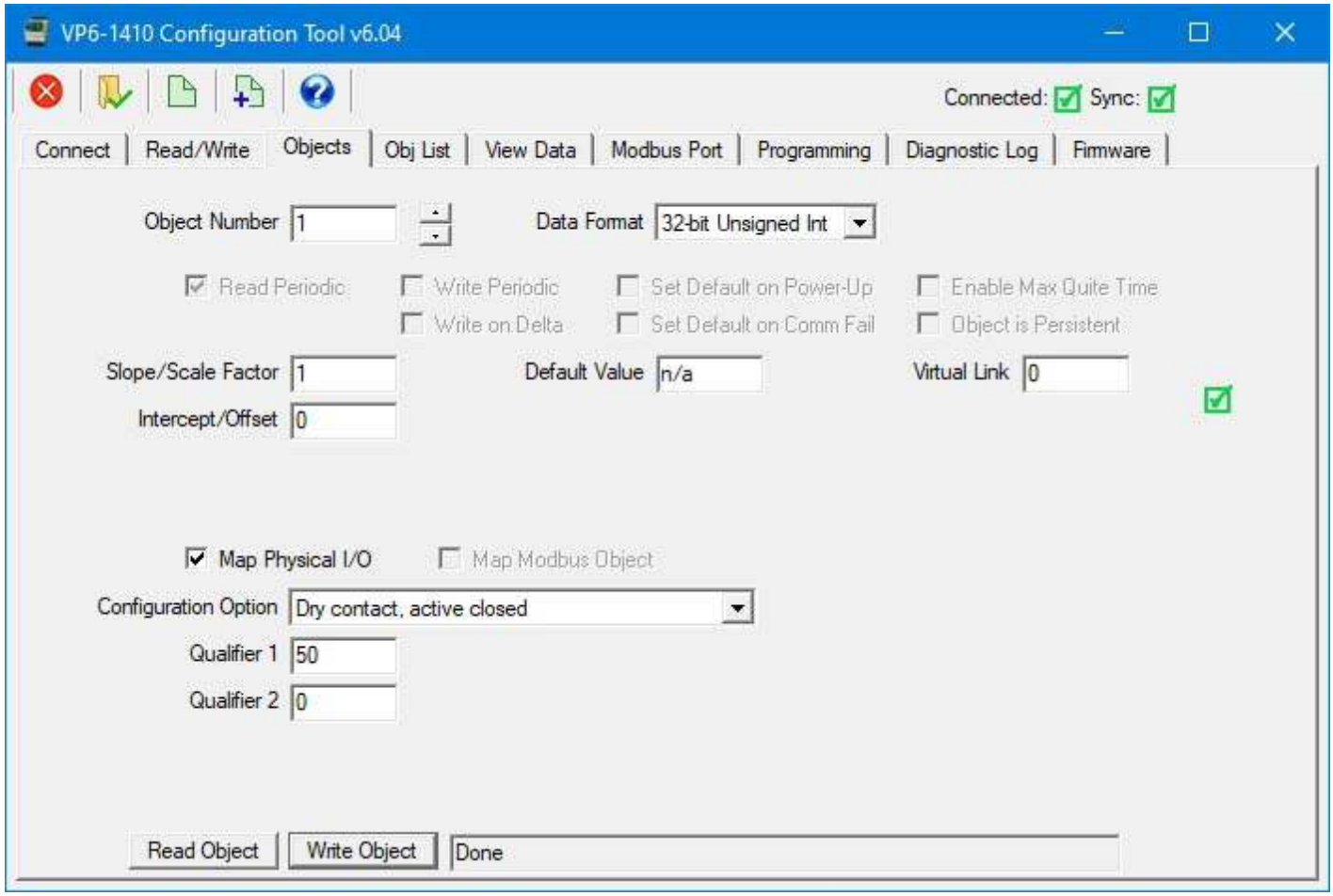
Click the "line" icon next to the "?" icon to open the line programming tool. If you do not want to "draw" a program, but would rather write a program using the native PL/i programming language, you can do this. The line programming tool also has its own set of help pages.

The PL/i programming language is a derivative of PL/1 but is not the same as PL/1. The language is referred to as PL/i with "i" as in i.CanDrawIt.

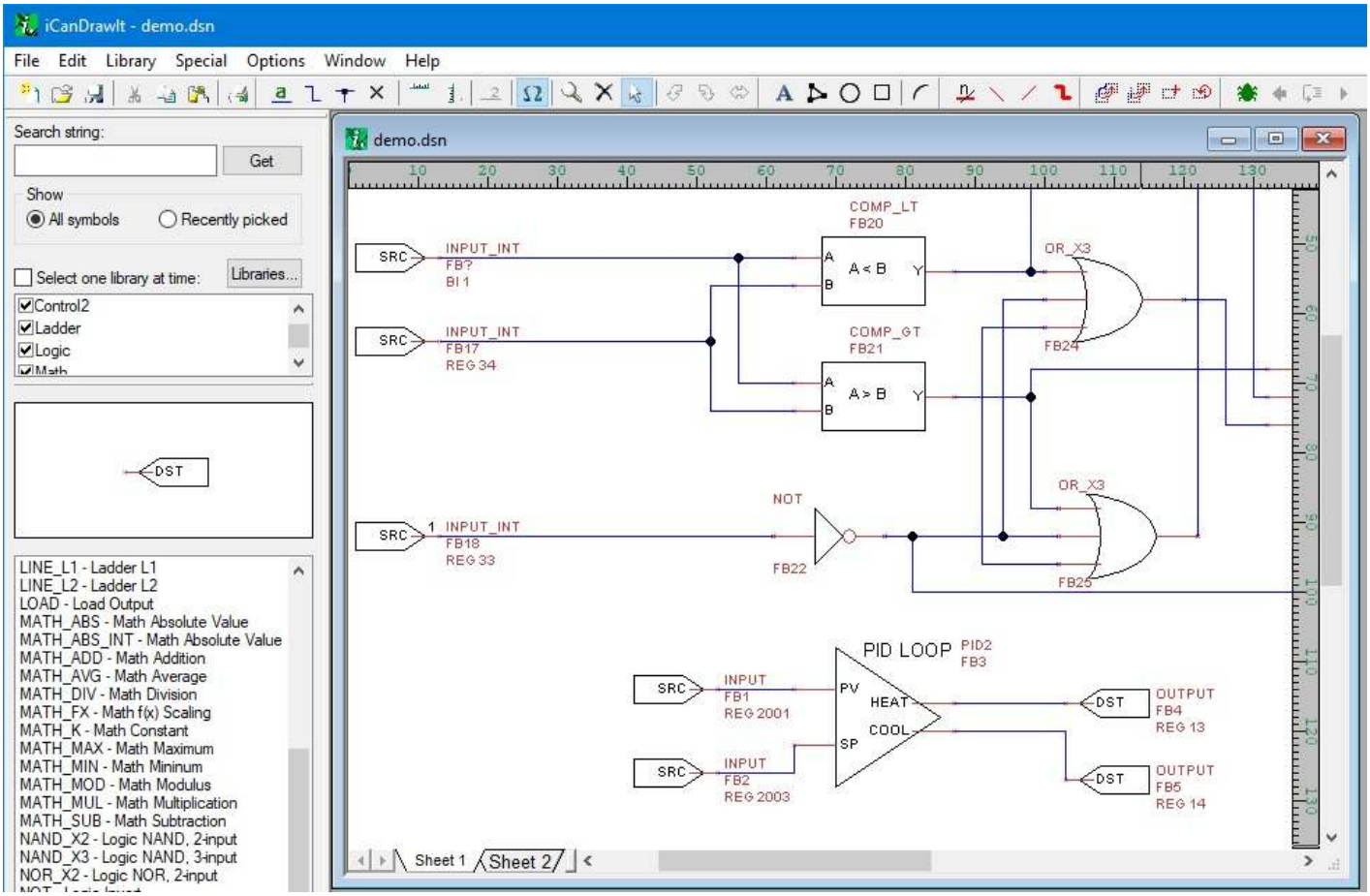
The above simple example program shows operating the relay output DO #1 (object 13) from input A/UI #1. To configure A/UI #1 for use in the above example, go to the Objects page and configure A/UI #1 for dry contact (or other applicable discrete type).




The programming environment only knows how to read/write 32-bit values. Therefore, select 32-bit integer as data format.




A more complex example is illustrated here:




## 9.3 Program Capacity

Maximum compiled program (.plx file) size: 65,280 bytes
Maximum RAM available for program variable and stack space: 16,384 bytes
Maximum EEPROM available: 256 bytes

## 9.4 Program States and Error Codes

Program State codes:

- 0 = idle
- 1 = loading
- 2 = running
- 3 = waiting
- 4 = halted
- 5 = unloading

Reason for Halt codes (Indicated as Error Code):

- 0 = no error (not halted)
- 1 = program load failed
- 64 = normal stop, end of program reached
- 65 = external stop via Program Change
- 66 = debug execution, suspended
- +n = error code (400 or above)

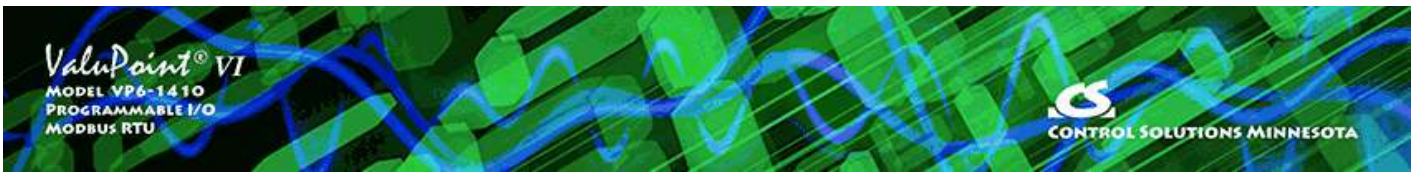
Non-fatal runtime errors:

401: subscript out of bounds, non-fatal
402: divide by zero, non-fatal
406: EEPROM address out of range, operation skipped
407: object instance out of bounds, operation skipped

Note: Error codes will show up as "Reason for Halt" even if the error was not necessarily fatal. This is because "Reason for Halt" is the only available standard Program Object property whose purpose is to report errors. Check toe Program State to determine if the program is actually halted.

Fatal runtime errors:

451: unrecognized opcode, fatal
452: stack overflow, fatal
453: stack underflow, fatal
454: program pc out of bounds, fatal

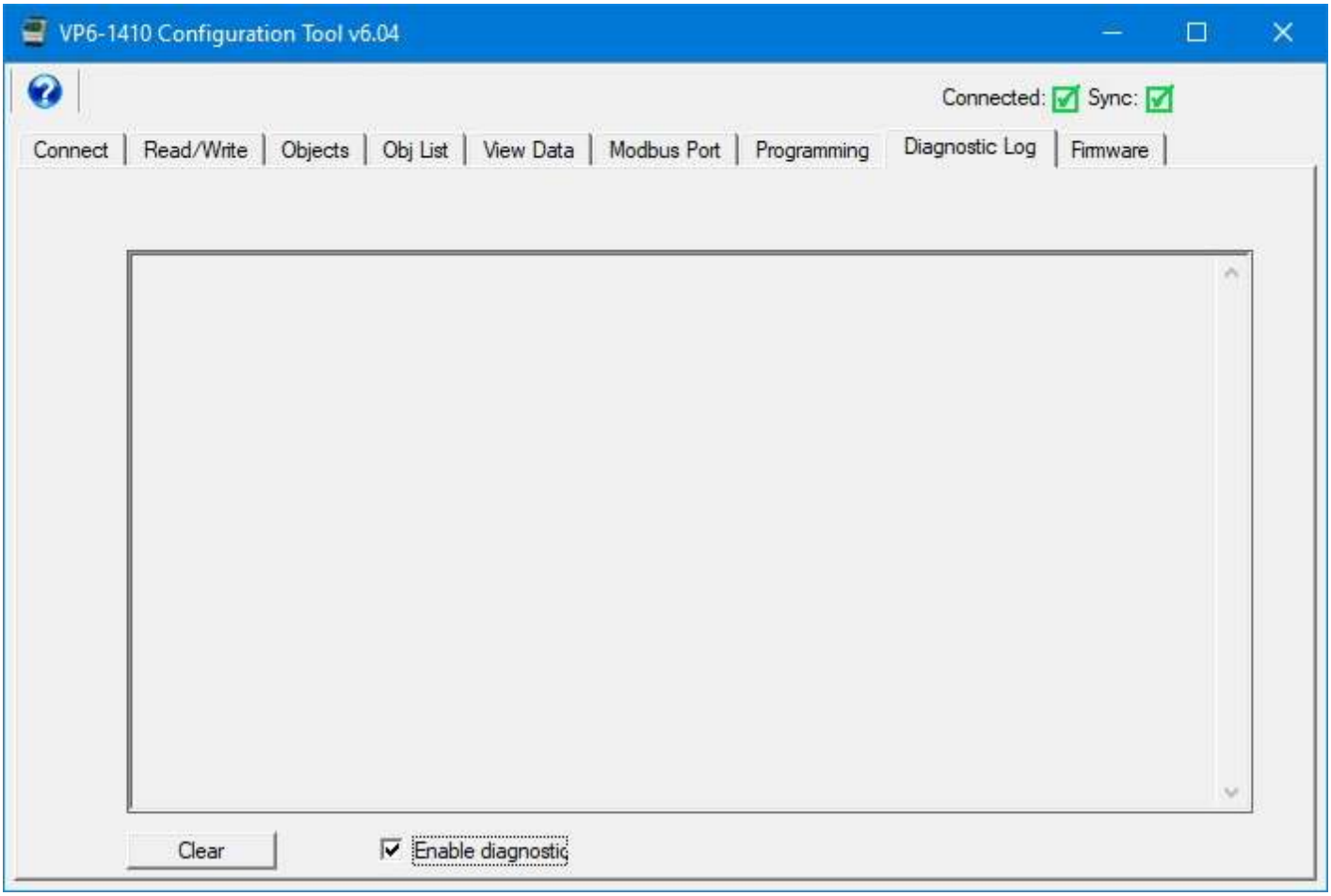



# 10 Diagnostic Log

## 10.1 Using the Diagnostic Log

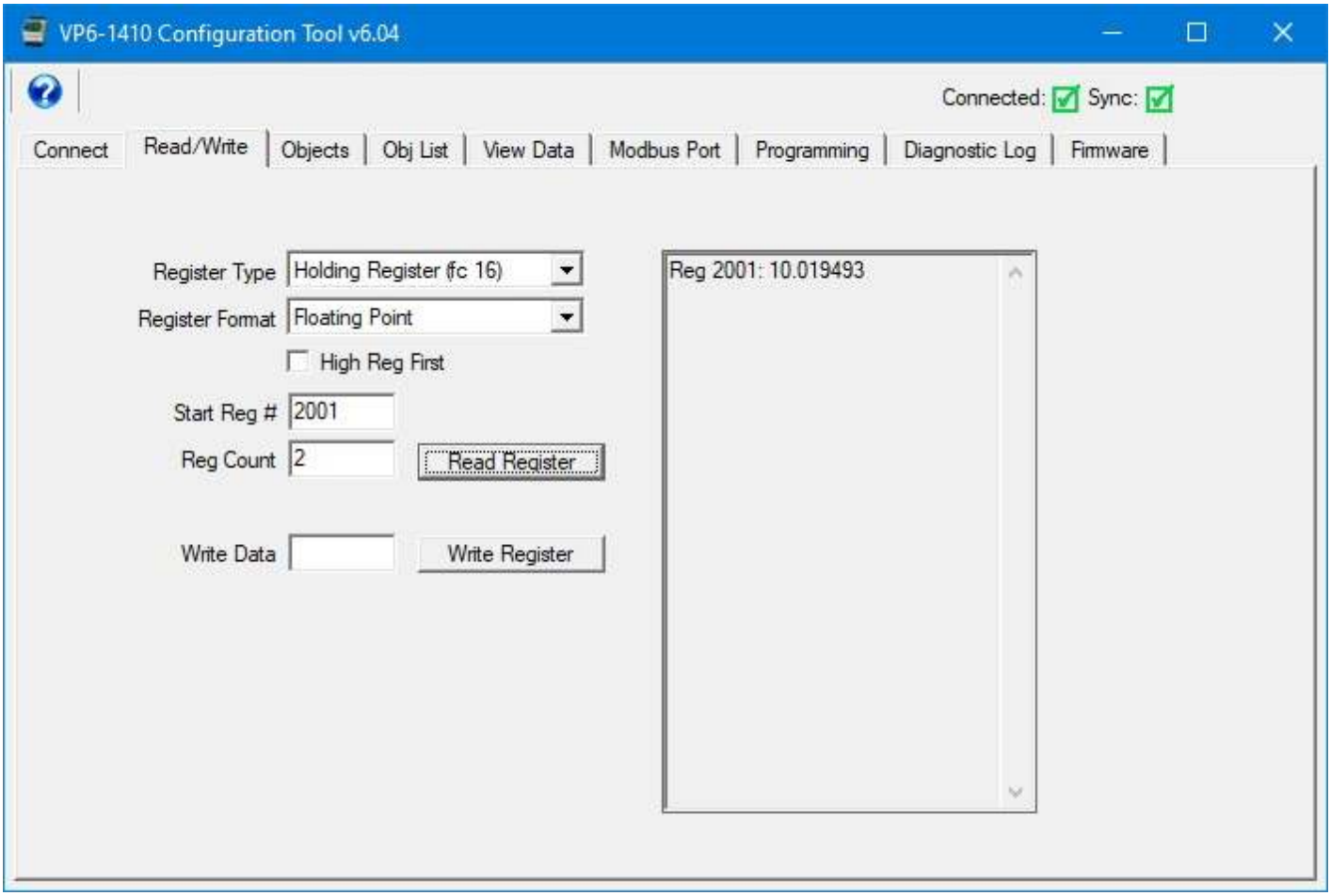
The diagnostic log is useful for verifying exactly what is going over the wire between the configuration tool and the device. The raw content of all Modbus packets sent to the ValuPoint device by the tool are displayed on a line labeled "Tx". The raw content of all Modbus packets received from the ValuPoint device by the tool are displayed on a line labeled "Rx". Each individual packet is displayed on a separate line.

Check the 'Enable diagnostic' box to enable the log. Click Clear to erase the contents of the window.

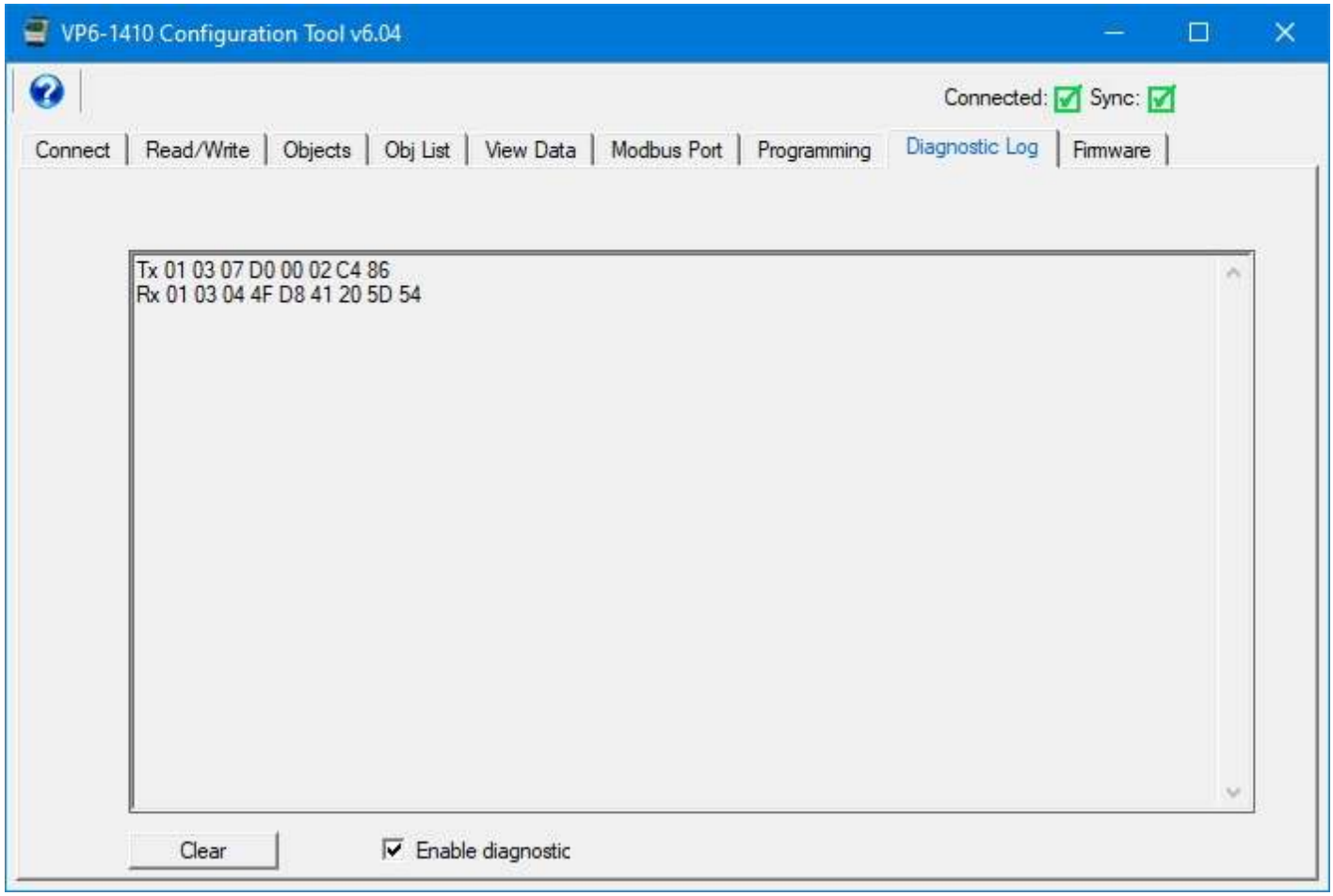



Note that the Read/Write page can be used to query any Modbus RTU device physically connected to the same network as the configuration tool. This means the ValuPoint tool can be a useful diagnostic tool for testing other Modbus devices. If you are using

this tool to query a device other than a ValuPoint, all communication with that device will also show up here on the Diagnostic Log.

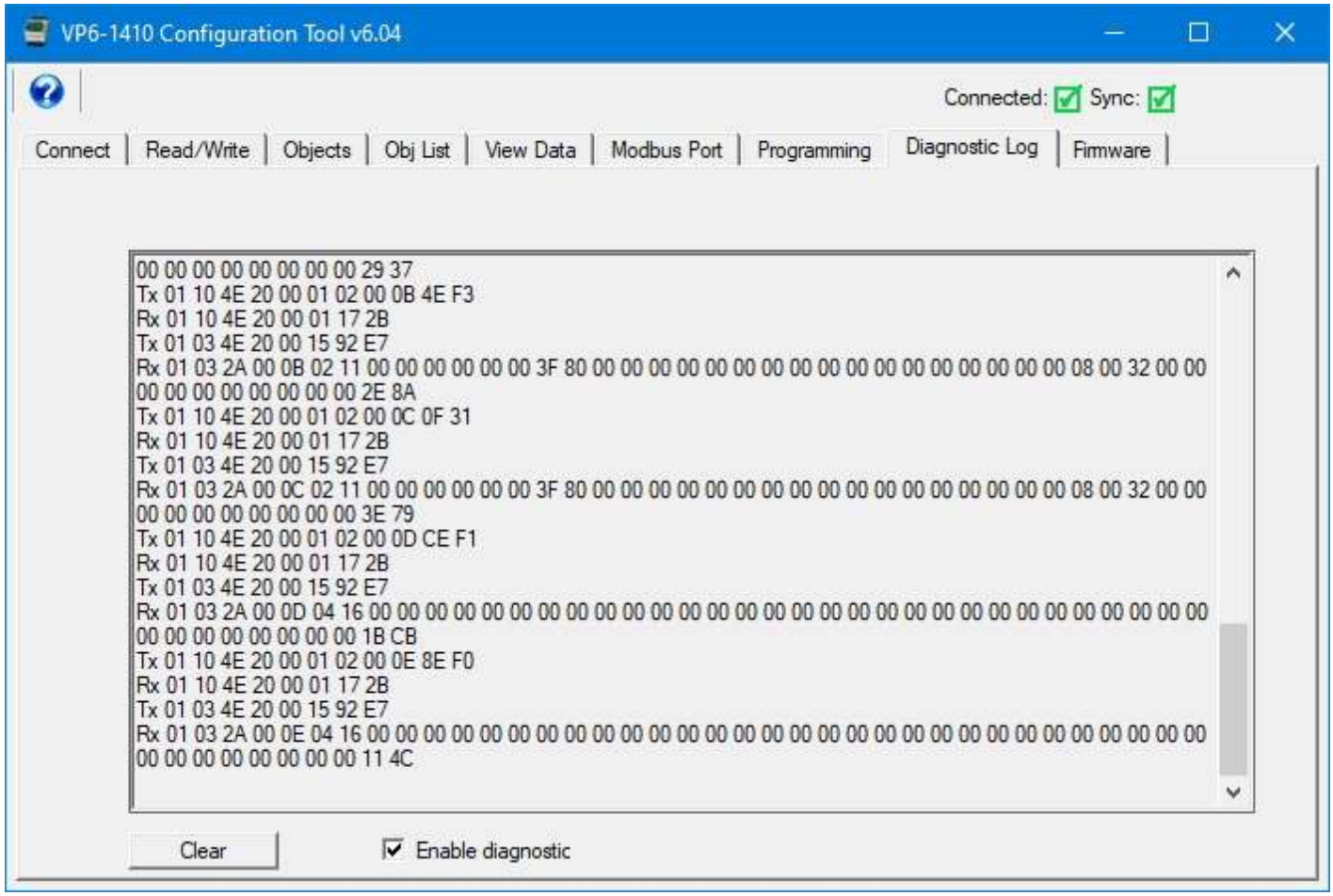



The above register read operation produced the following log.




Note that only communication between the configuration tool and some other Modbus device are displayed here. Communication between two other Modbus devices will not be captured here. Only communication involving the tool is displayed.

The following example log resulted from clicking Write All on the Object List page.


VP6-1410 Configuration Tool v6.04
Connected:
Sync:
Connect
Read/Write
Objects
Obj List
View Data
Modbus Port
Programming
Diagnostic Log
Firmware
00 00 00 00 00 00 00 00 29 37
Tx 01 10 4E 20 00 01 02 00 0B 4E F3
Rx 01 10 4E 20 00 01 17 2B
Tx 01 03 4E 20 00 15 92 E7
Rx 01 03 2A 00 0B 02 11 00 00 00 00 00 3F 80 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 08 00 32 00 00 00 00 00 00 00 00 00 00 2E 8A
Tx 01 10 4E 20 00 01 02 00 0C 0F 31
Rx 01 10 4E 20 00 01 17 2B
Tx 01 03 4E 20 00 15 92 E7
Rx 01 03 2A 00 0C 02 11 00 00 00 00 00 3F 80 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 08 00 32 00 00 00 00 00 00 00 00 00 00 3E 79
Tx 01 10 4E 20 00 01 02 00 0D CE F1
Rx 01 10 4E 20 00 01 17 2B
Tx 01 03 4E 20 00 15 92 E7
Rx 01 03 2A 00 0D 04 16 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 1B CB
Tx 01 10 4E 20 00 01 02 00 0E 8E F0
Rx 01 10 4E 20 00 01 17 2B
Tx 01 03 4E 20 00 15 92 E7
Rx 01 03 2A 00 0E 04 16 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 11 4C
Clear
Enable diagnostic


# 11 Firmware Update Page

Firmware update support is included as a page in the VP6-1410 configuration tool. The firmware update is done via Modbus RTU.

## 11.1 Firmware Organization

There are two copies of the application firmware - a primary and a backup image. The primary image is the one that is always running when the VP6-1410 is booted up. In the unlikely event that the primary image becomes corrupted, the backup image will replace it.

There are redundant copies of something known as a "boot information record" or "boot info" for short. There are two copies of the boot info for the primary image, and two copies of the boot info for the backup image. The primary and backup images are stored in the processor's flash memory. The boot info records are stored in on-chip EEPROM (non-volatile memory).

The boot info record contains the following information: Application code size, application image checksum, model number and variant, and firmware revision.

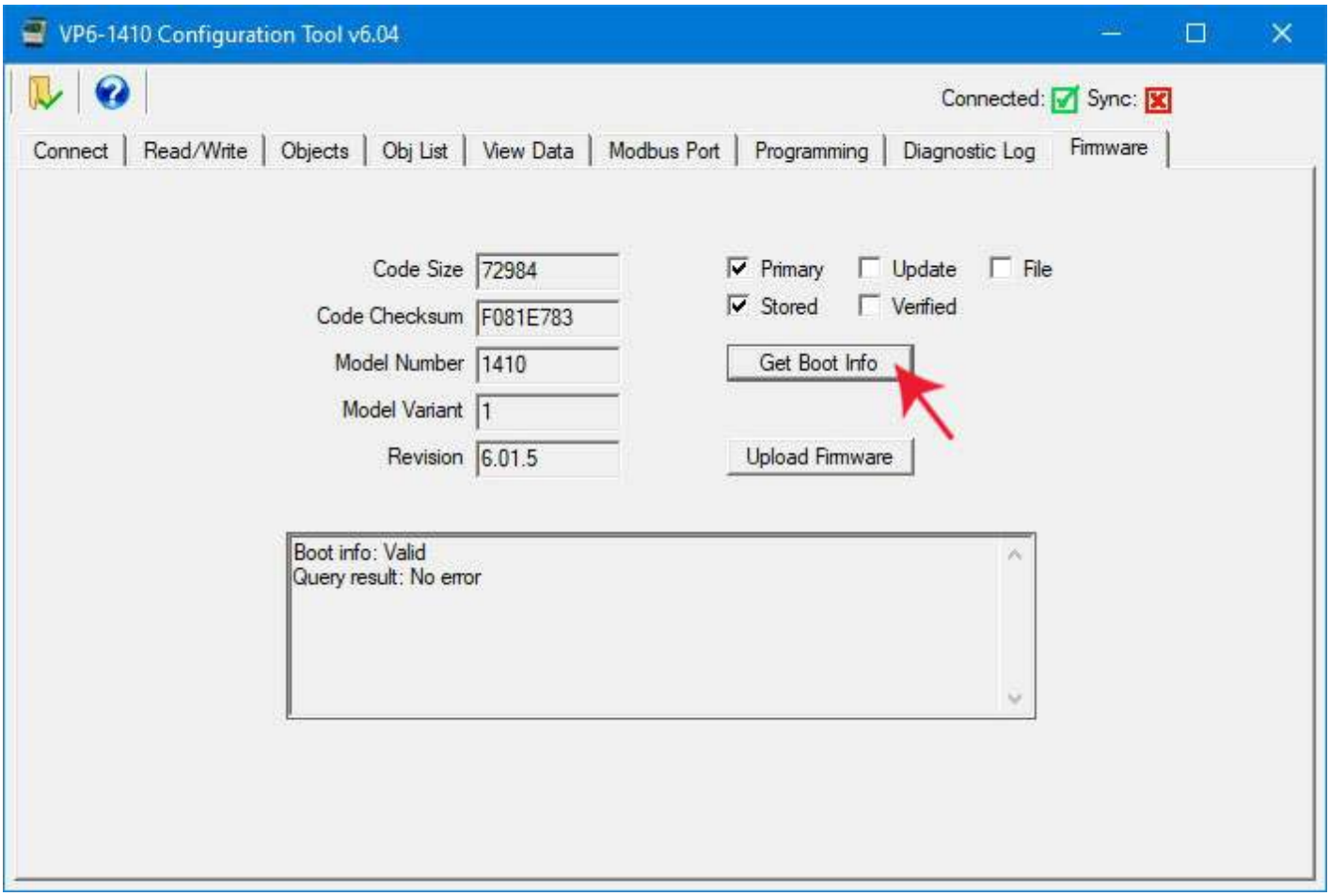
Upon bootup, the primary application image will be scanned and its checksum calculated. If this does not match the previously stored boot info record, it will be assumed that the application has somehow gotten corrupted. If the primary image is deemed to be corrupt (checksum does not match), then the backup image is scanned. If the backup image is deemed to be good, then the primary application area will be reprogrammed with a copy of the backup image.

It is generally highly unlikely that the application image flash area will become corrupted because the only time any write operations will be taking place with flash is in reprogramming firmware. If power is lost in the middle of a firmware update, for example, then the flash image will likely be corrupted.

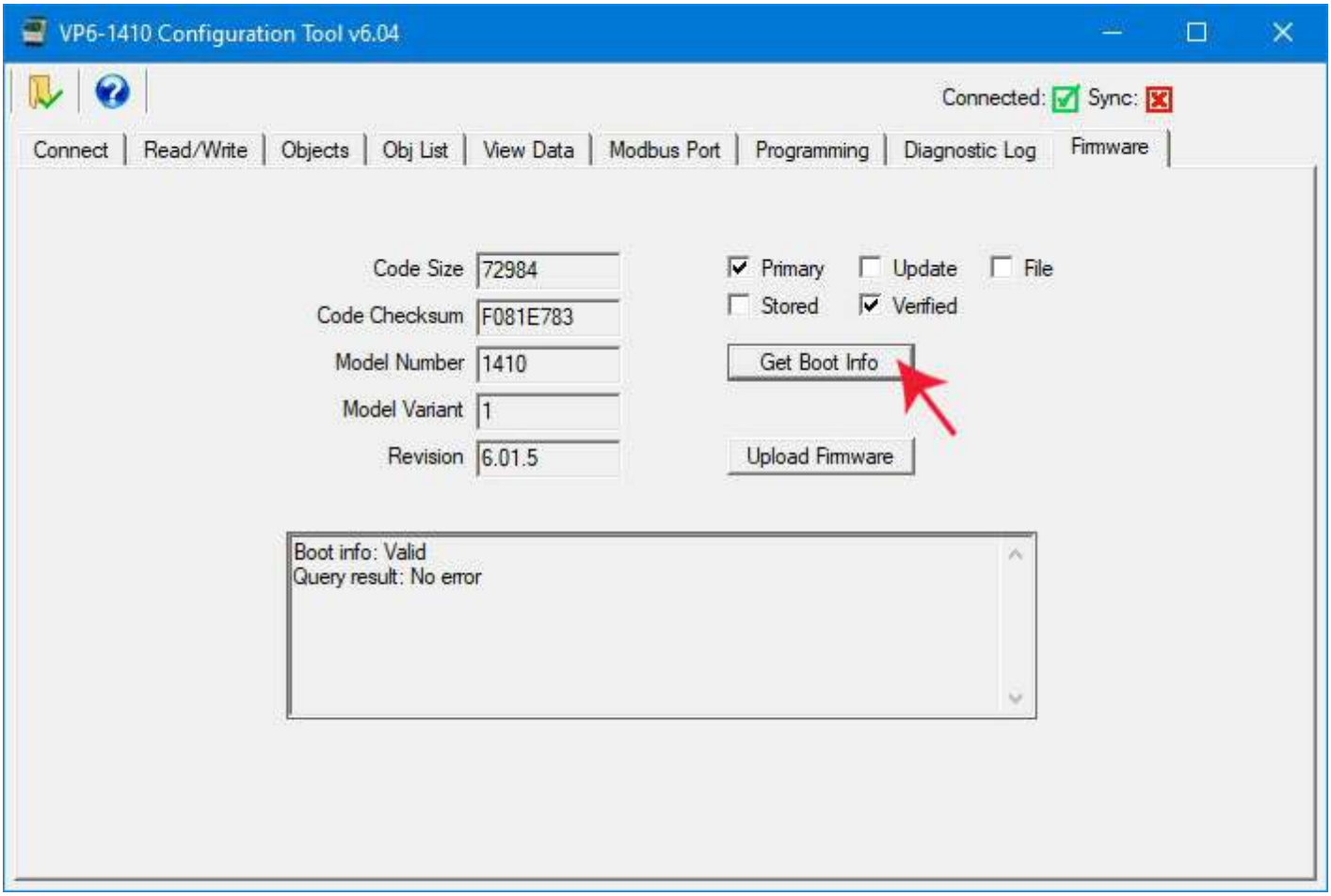
It is probably more likely that the EEPROM that stores the boot info may become corrupted by a loss of power while writing to EEPROM. Writes to EEPROM are somewhat more frequent as certain parts of the device's configuration are stored there. In the event both primary and backup images fail the checksum test, then a minimal validity check on the primary application is made. If it seems that there is a reasonable chance that the application may be valid, the boot loader will go ahead and start up the primary application, but with an error indication to alert you to the fact that this has happened.

## 11.2 Verifying Primary Application Image

If you check "Primary" and "Stored" and then click "Get Boot Info", you will retrieve the boot info record for the primary application image.

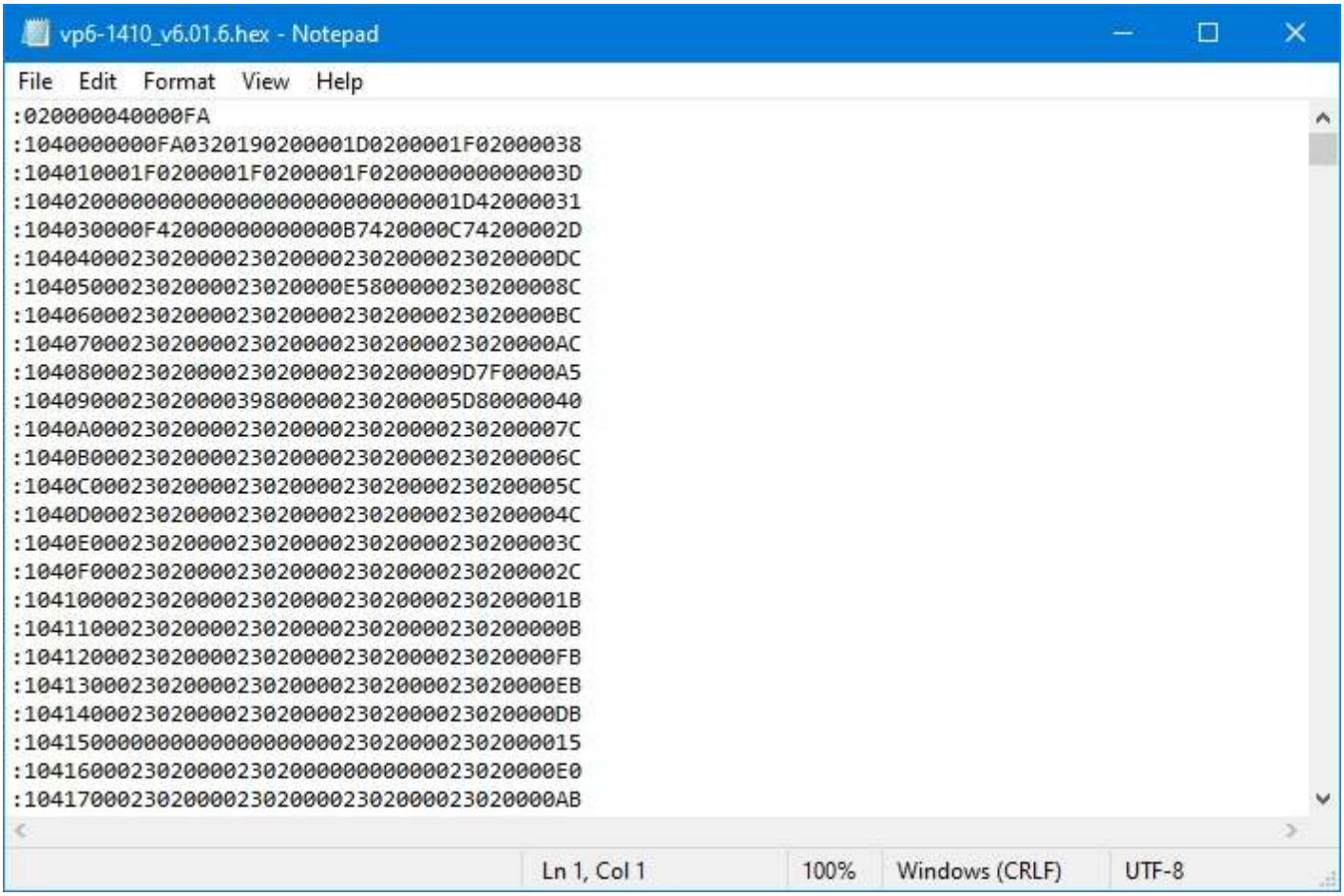



If you click "Primary" and "Verified" and then click "Get Boot Info", you will cause the device to scan its primary application image and compare it to the stored boot info record. It should always come back as valid, no error, and display the current firmware revision.



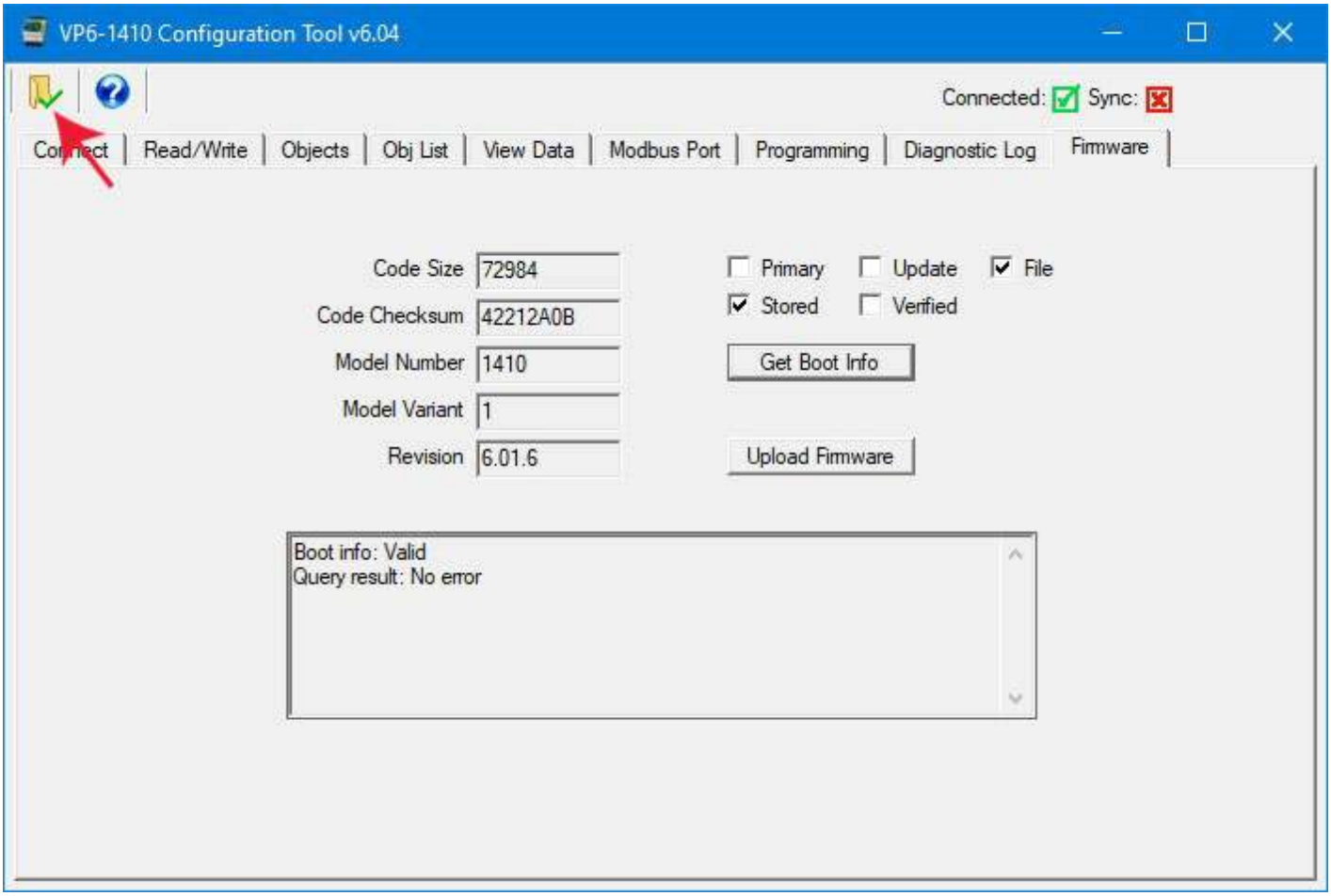

## 11.3 Firmware Update Process

If there is newer firmware and you have good reason to believe a firmware update would resolve some issue you're having, then start by obtaining a copy of the firmware update from Control Solutions support. The update will be in the form of a "hex" file, typically named something like "VP6-1410.hex" and its contents, which will always be printable, will look something like this (abbreviated view):



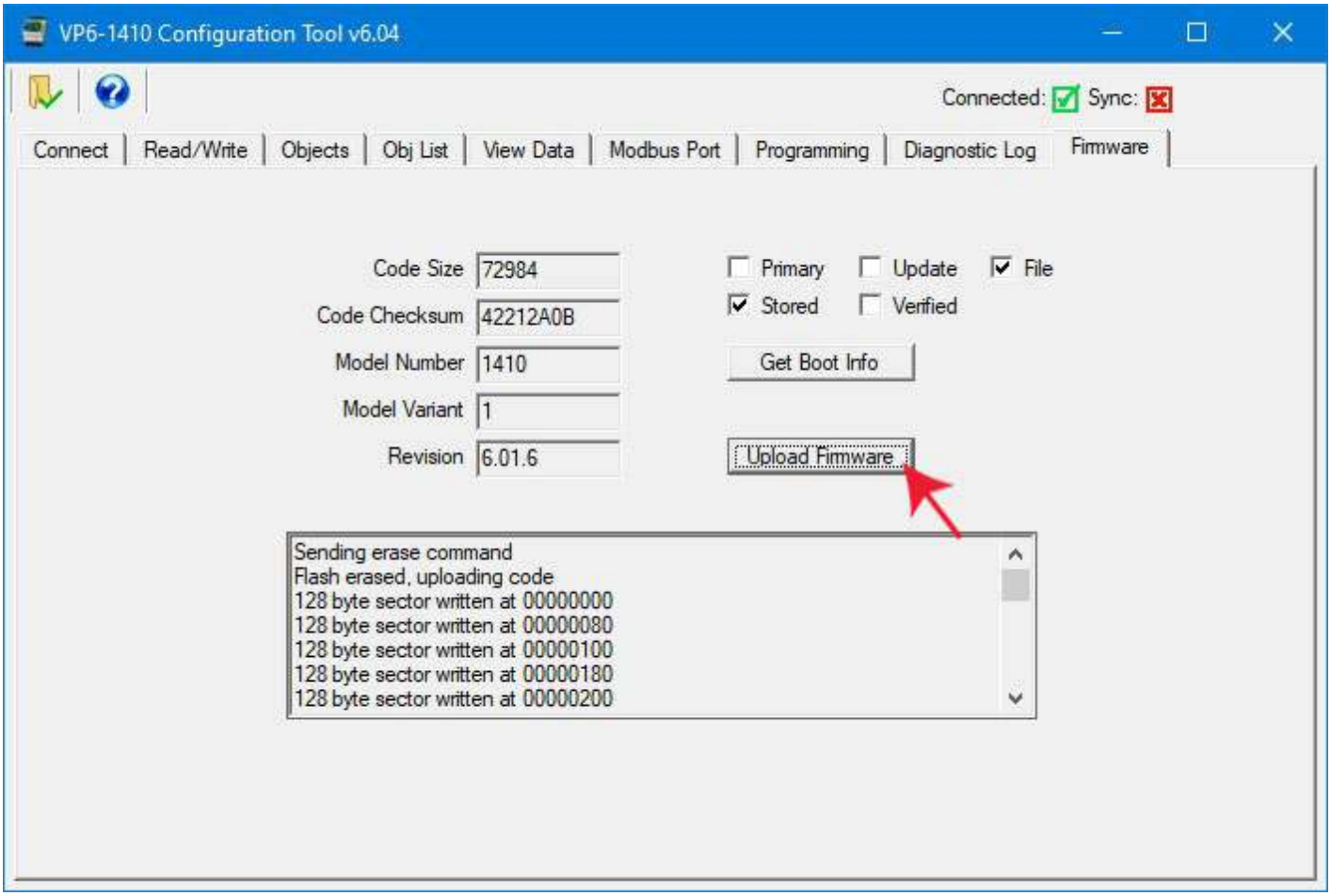

The VP6-1410 configuration tool should be connected in the same manner as for configuring I/O objects.

To upload a firmware update, start by clicking on the file open icon. Use the standard Windows file dialog that opens automatically to locate your copy of the .hex file and open it. After loading the file, the information displayed now is taken from the file just loaded. The File checkbox indicates that you are looking at boot info for the loaded code file rather than the connected device.

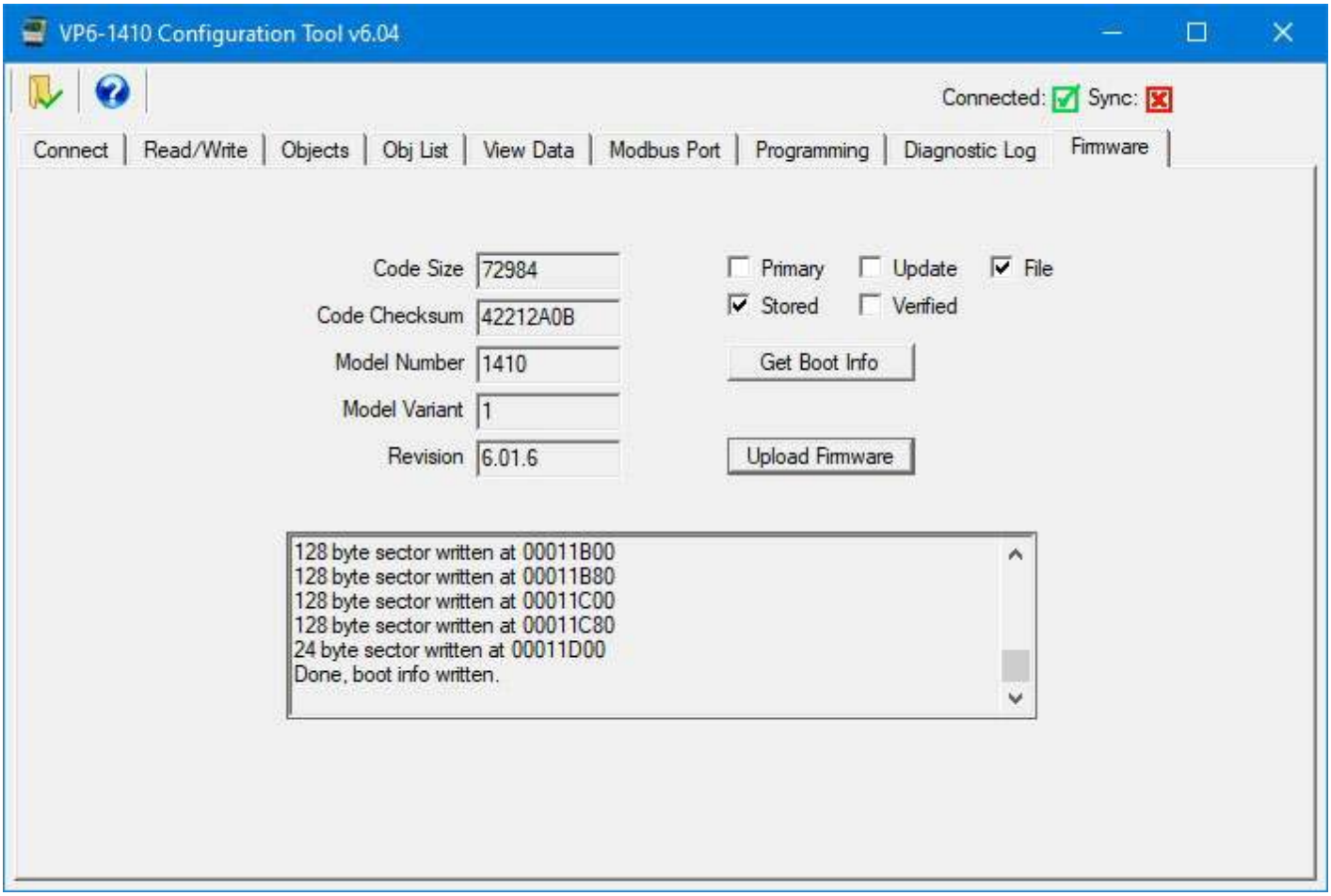



The upload process will take a couple of minutes. During this time, progress will continue to get posted to the status window. When the upload is complete, the boot info record for the update image is written.

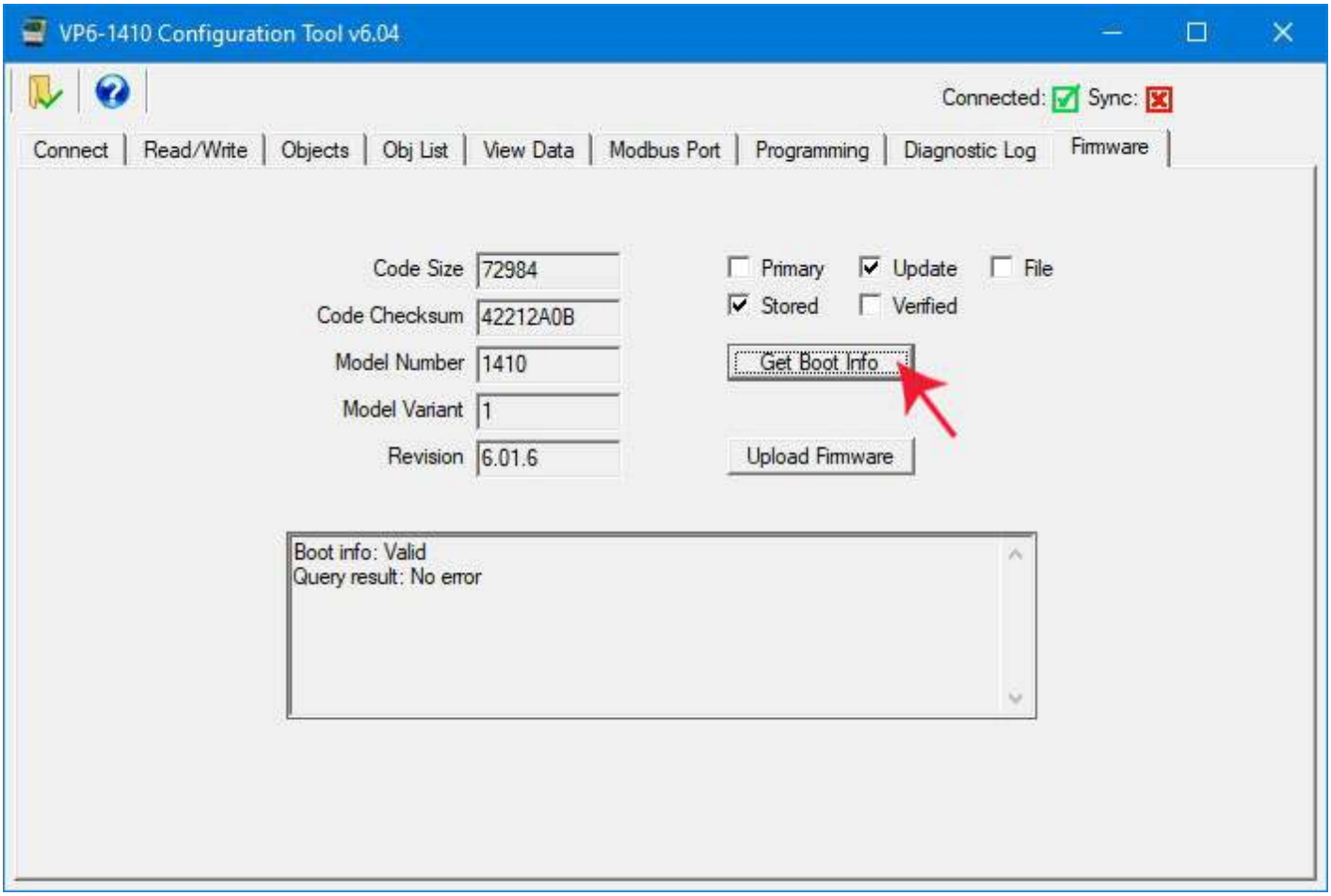
The firmware update is programmed into the backup application area of flash memory in the processor chip. Once the upload is complete, and verified, then all you need to do is restart the device to complete the process.




Once the upload process is complete, the "Done" message will appear.



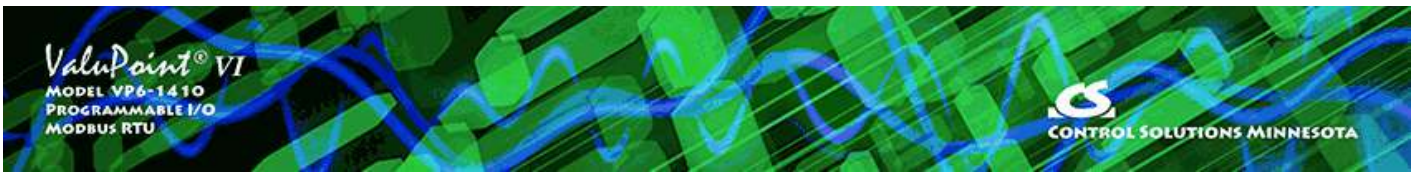

Following the upload, you can query the stored boot info record for the update image.




By selecting "Verified", you can cause the device to scan the code just uploaded and compare the checksum. You should get "Valid, No error" and this will indicate that the upload was successful and code has been verified.

Next, you will need to restart the device. Do this by simply power cycling the VP6-1410. The device will now restart, and upon bootup, and upon seeing that a newer image is available, the backup image will be copied into the primary application flash area of memory.

Following the restart, you should see the new firmware version displayed when you select Primary and Stored and then click the Get Boot Info button.

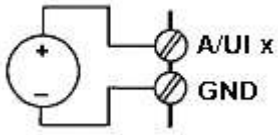
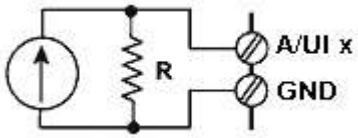
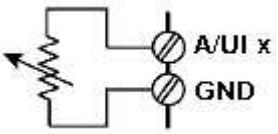


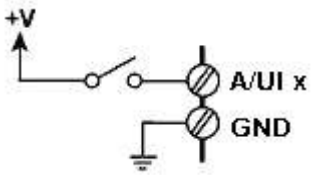
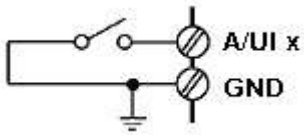

# Appendix A - Hardware Details

## A.1 Connection of Inputs

The VP6-1410 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 Objects page of the configuration tool.

| I/O Point Type | Wiring Guide | Additional Information |
|---|---|---|
| **A/UI Analog Input: 0-10VDC Voltage Input or Pulse Count** |   | 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.<br><br>This wiring diagram is also applicable to pulse counter input for counting pulses from an active pulse generator. |
| **A/ UI Analog Input: 0-20mA Current Input** |   | 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. |
| **A/UI Analog Input: Thermistor Input** |   | A/UI inputs 1-12 will accept thermistors of 3k, 5k, 10k, or 20k ohms. Linearization via interpolation of a 56-point |

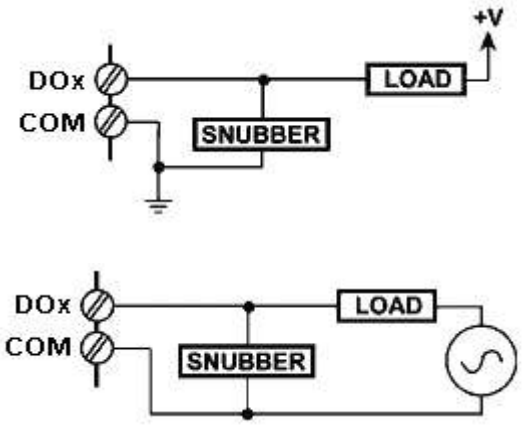
| | | |
|---|---|---|
| | | table is performed internally. |
| **A/UI or DI Discrete Input: Discrete Voltage** |  | A/UI inputs 1-12 may be connected as discrete voltage sensing inputs. Inputs up to 24VDC are tolerated, but threshold sensing only functions over the 0-10VDC range. |
| **A/UI or DI Discrete Input: Dry Contact Closure to Ground or Pulse Count** |  | 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).<br><br>This wiring diagram is also applicable to pulse counter input for counting pulses from a switch closure. |

## A.2 Connection of Outputs

The VP6-1410 contains no configuration jumpers for configuring outputs. There is no need to open the enclosure.

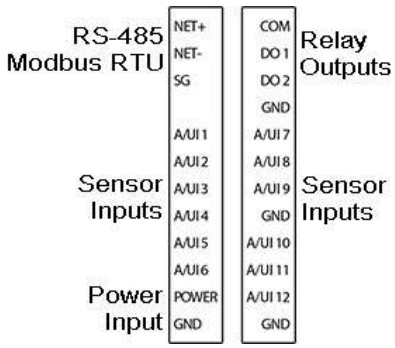
Output points should be connected as indicated in the diagrams below. The discrete (relay) outputs are SPST N.O. with the common side connected together to the COM terminal. The COM terminal for relays is NOT electrically common to any of the GND terminals.

| I/O Point Type | Wiring Guide | Additional Information |
|---|---|---|

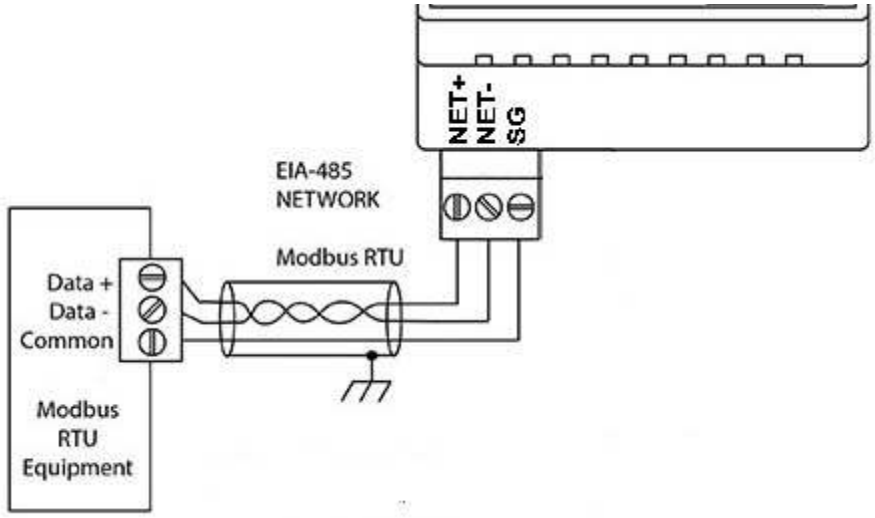
| | | |
|---|---|---|
| **Discrete Output: Form A Relay** |  | DO outputs 1-2 are Form A dry contact relays rated for 2A @ 120VAC (resistive). Snubbers should be used with inductive loads. The relays are also rated for 2A @ 30VDC. Note: Relays are rated higher, but UL listing is for 2A. |

## A.3 Connection of Power and Communications

Power and communications should be connected as indicated below.

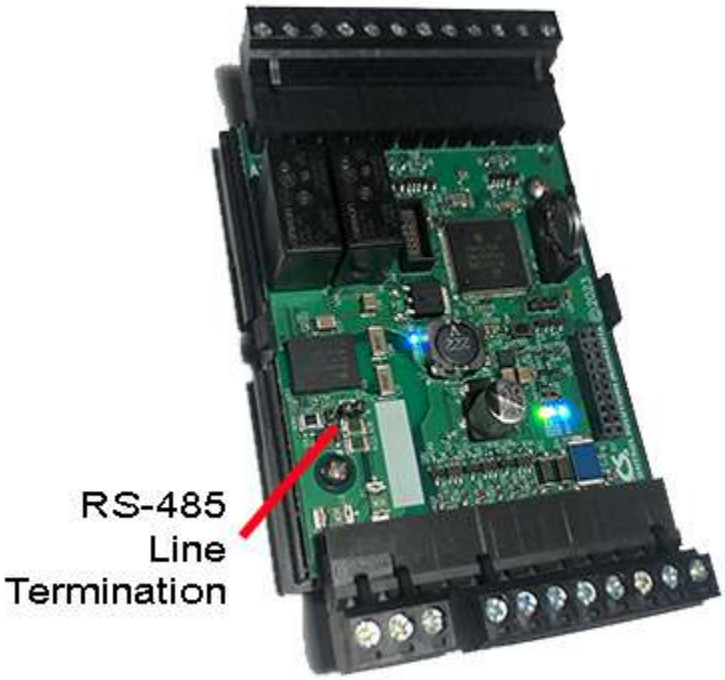
| I/O Point Type | Wiring Guide | Additional Information |
|---|---|---|
| **Power** | Connect AC or +DC power to Power terminal. Connect common or -DC power to GND terminal. GND is common to all terminals labeled COM. | Nominal power consumption is 2.4 watts, or 0.1A @ 24VDC, with all relays on. |
| **Communications** | 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. |
| **Wiring Terminals** |  | Screw terminals ratings are substantially in excess of any I/O point ratings.<br><br>Screw terminals are pluggable. They unplug from the unit in 3, 8, and 12-position blocks. |

The RS-485 should be wired as illustrated below.



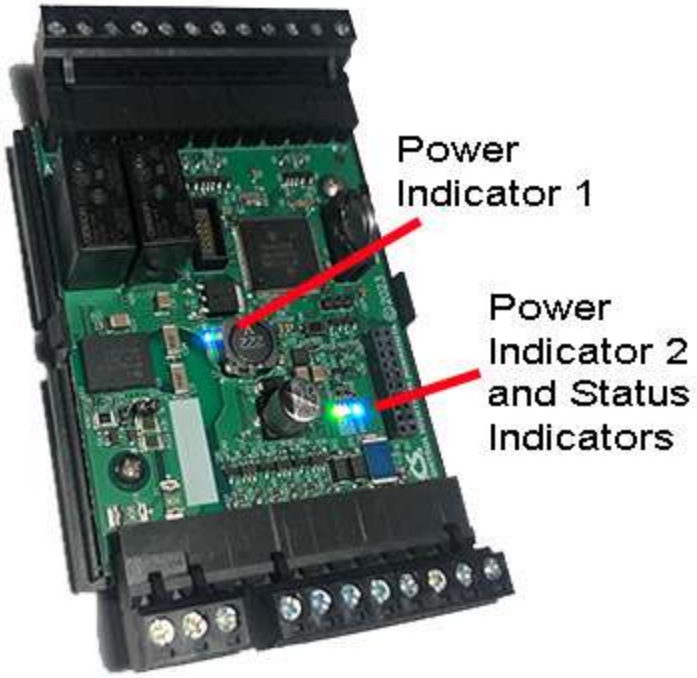

## A.4 RS-485 Line Termination

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 VP6-1410 is the last device on the daisy chain.




## A.5 LED Indicators

LED indicators are depicted in the following photo. Status indicators include green, red, and yellow LEDs which are visible through the vent slots of the enclosure. Two blue power indicators are also visible through the vent slots.




Status indicator functions for Modbus RTU:

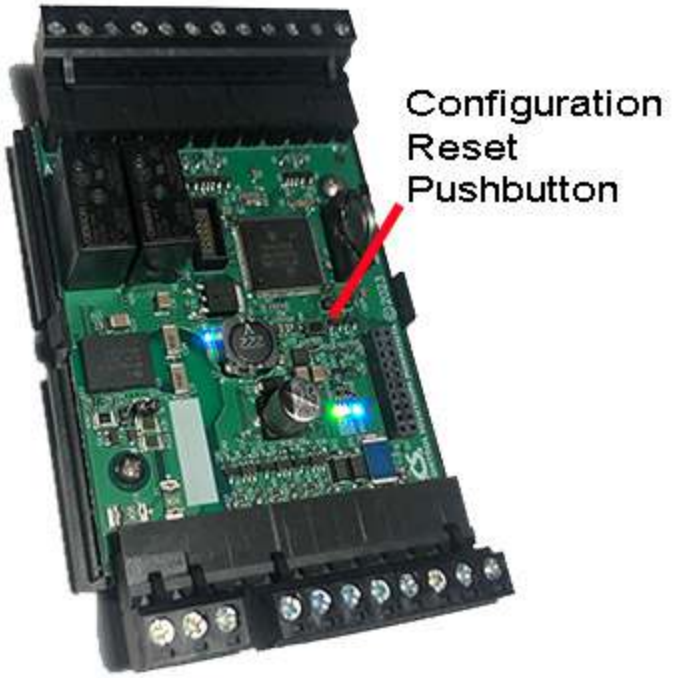
| Mode | Green LED | Yellow LED | Red LED |
|---|---|---|---|
| No traffic - Idle | Flashes "heartbeat" every 2 seconds | Off | Off |
| Modbus slave is receiving traffic | Flashes to indicate good response returned | Flashes when request received | Flashes to indicate error response |
| Modbus master is generating traffic | Flashes when good response received | Flashes when request sent | Flashes upon timeout or error response received |

NOTE: If only the red LED is flashing, and is primarily on while flashing briefly off a few times, count the number of "off" flashes. This is a system error code. Refer to the trouble shooting section.

### A.6 Restore Factory Defaults

The configuration reset pushbutton is located on the circuit board where indicated below.

IMPORTANT: Using the reset button should be your last resort. If you have simply forgotten what baud rate or address the VP6-1410 is set to, refer to Recovery Mode in section 3 of this user guide. Clearing all configuration should not really be required since you overwrite all old configuration every time you write a new configuration using the configuration tool.




There are two "codes" for restoring factory default.

Code 1: Reset communication parameters, but leave configuration intact.

Code 2: Reset communication parameters and erase all configuration.

Press and hold the button until you see code 1 flashing. If you wish to perform code 1, release the button while code 1 is flashing. Otherwise continue to hold the button until you see code 2 flashing and then release the button. If you do not release the button while code 2 is flashing, it will return to code 1 in a few seconds, and the codes will continue to alternate for as long as you hold the button down.

Upon releasing the button, you will see the code confirmation flashing. Press the button again to cancel the action. You must do the cancel within the next few seconds, otherwise the action will be performed. If you do not cancel the action, the action will be performed in a few seconds.

The LEDs that normally show Modbus traffic are temporarily used to display these codes while the reset is pending.

Code 1: Yellow LED flashes once and then pause (and then repeat).

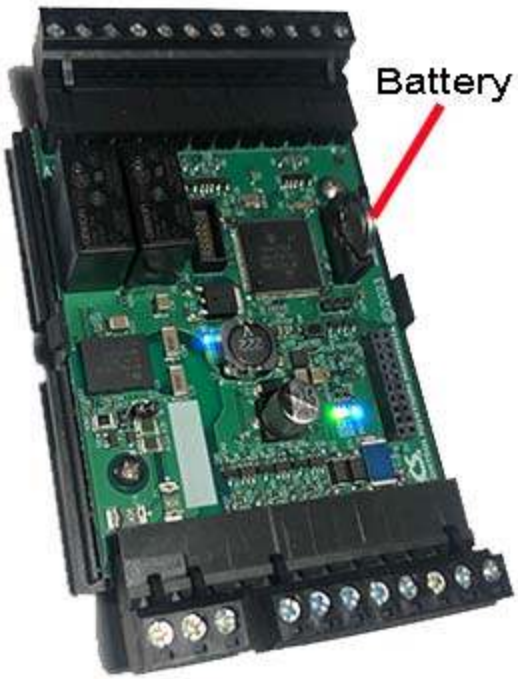
Code 2: Red LED flashes twice and then pause (and then repeat).

Confirmation: Green LED flashes showing either code 1 or code 2 (one or two flashes followed by a pause, and then repeat).

## A.7 Battery Replacement

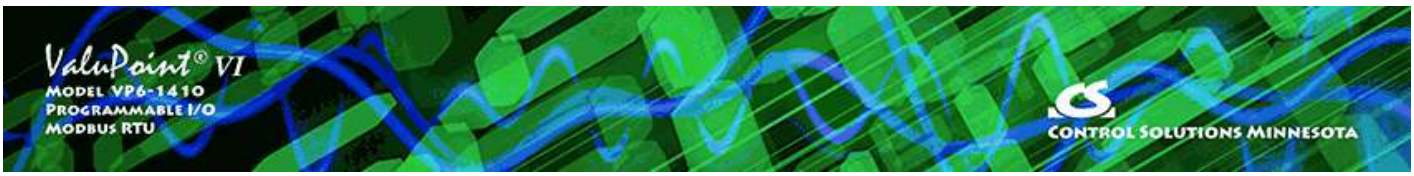
No action is required of the user to activate the battery that backs up the real time

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 Register Map

## B.1 Modbus Registers - Full Map

The following chart shows the available Modbus registers. Data objects are accessible as holding registers, as well as input register, discrete input, and coil. Special registers such as configuration, etc, are only accessible as holding registers, as indicated by the chart below.

Register numbers are shown as Modicon and Extended Modicon for reference, but the standard register numbers in the "Std. Reg. No." column must be used in the configuration tool software. Standard register numbers are raw address plus one.

| Modicon Std | Modicon Extd | Type | **Std. Reg. No.** | Description |
|---|---|---|---|---|
| Data object access registers (data registers) | | | | |
| 40001-40500 | 400001-400500 | Holding | 1-500 | Objects 1-500 accessed as signed integer, 16-bit, single Modbus register<br>VP6-1410: A/UI 1-12 = objects 1-12; DO 1-2 = objects 13-14 |
| | | | | |
| 40801 | 400801 | Holding | 801 | A/UI inputs accessed as packed register of bits, 1 bit per input, LSB=input 1 (if applicable) |
| 40803 | 400803 | Holding | 803 | DO outputs accessed as packed register of bits, 1 bit per output, LSB=input 1 (if applicable) |
| | | | | |
| 41001-41500 | 401001-401500 | Holding | 1001-1500 | Objects 1-500 accessed as unsigned integer, 16-bit, single Modbus register |
| 42001-43000 | 402001-403000 | Holding | 2001-3000 | Objects 1-500 accessed as IEEE754 floating point, 32-bit, double Modbus register |
| 43001-44000 | 403001-404000 | Holding | 3001-4000 | Objects 1-500 accessed as signed integer, 32-bit, double Modbus register |

| | | | | |
|---|---|---|---|---|
| 44001-45000 | 404001-405000 | Holding | 4001-5000 | Objects 1-500 accessed as unsigned integer, 32-bit, double Modbus register |
| | | | | |
| 47001-47030 | 407001-407030 | Holding | 7001-7030 | Real time clock/calendar registers |
| 48001-48246 | 408001-408246 | Holding | 8001-8246 | Modbus error codes for slaves 1-246 (8001 if VP4 is slave) |
| 48401 | 408401 | Holding | 8401 | Modbus port baud rate |
| 48402 | 408402 | Holding | 8402 | Modbus port slave address when operating as slave |
| 48403 | 408403 | Holding | 8403 | Modbus port timeout in tenths of seconds when operating as master |
| 48404 | 408404 | Holding | 8404 | Modbus port pre-delay |
| 48405 | 408405 | Holding | 8405 | Modbus port configuration bits (b0=set if big endian; b1=set for slave mode; b2=enable parity; b3=parity is odd; b4=two stop bits |
| | | | | |
| 49901 | 409901 | Holding | 9901 | Configuration mode register |
| 49902 | 409902 | Holding | 9902 | Firmware revision (read only) (nxxyy where n=major, xx=minor, yy=build) |
| 49903 | 409903 | Holding | 9903 | Test mode register (write 9901 to test discrete, 9902 to test analog, 9900 for default I/O config) |
| | | | | |
| --- | 410001-410500 | Holding | 10001-10500 | Poll timers for objects 1-500, 16-bit unsigned integer, single Modbus register |
| | | | | |
| --- | 411001-411500 | Holding | 11001-11500 | New data flags for objects 1-500, 16-bit unsigned integer, single Modbus register |
| | | | | |
| Configuration registers | | | | |
| --- | 420001 | Holding | 20001 | Index register, write 1-500 to index configuration register set for object 1-500 |
| --- | 420002 | Holding | 20002 | Object use configuration bit set (objectUse_t) |

| | | | | |
|---|---|---|---|---|
| --- | 420003 | Holding | 20003 | Poll time in seconds (uint16) |
| --- | 420004 | Holding | 20004 | Default host timeout in seconds (uint16) (unused in VP6-1410) |
| --- | 420005 | Holding | 20005 | Max. quiet time in seconds (uint16) |
| --- | 420006-420007 | Holding | 20006-20007 | Scale factor applied after mask (float) |
| --- | 420008-420009 | Holding | 20008-20009 | Offset applied after scale (float) |
| --- | 420010-420011 | Holding | 20010-20011 | Default value to apply upon read failure, POR, host timeout (u_data) |
| --- | 420012-420013 | Holding | 20012-20013 | Send delta, change in local data needed to resent to remote (u_data) |
| --- | 420014 | Holding | 20014 | Object link reference, link to virtual object |
| | | | | |
| Modbus register definitions apply if 20001 defines this as a Modbus client map | | | | |
| --- | 420015 | Holding | 20015 | Modbus register number/point number (1-based index) (uint16) |
| --- | 420016H | Holding | 20016H | Register type, 0=NONE, 1=REG_0X, 2=REG_1X, 3=REG_3X, 4=REG_4X |
| --- | 420016L | Holding | 20016L | Format for registers |
| --- | 420017H | Holding | 20017H | Remote unit # to query |
| --- | 420017L | Holding | 20017L | Number of fails before calling it a fault (apply default value) |
| --- | 420018-420019 | Holding | 20018-20019 | Bit mask to strip field out of 16-bit register (uint32) |
| --- | 420020-420021 | Holding | 20020-20021 | Bit mask fill bits (uint32) |
| | | | | |
| Hardware register definitions apply if 20001 defines this as a Hardware point | | | | |
| --- | 420015 | Holding | 20015 | Hardware channel code |
| --- | 420016 | Holding | 20016 | Hardware channel qualifier 1 |
| --- | 420017 | Holding | 20017 | Hardware channel qualifier 2 |
| | | | | |
| --- | 420100 | Holding | 20100 | Write value of 20100 to register 20100 to force NV memory write |
| | | | | |
| Data mirror registers, data objects accessed via Modbus function codes other than holding registers | | | | |

| | | | | |
|---|---|---|---|---|
| 00001-00500 | 000001-000500 | Coil | 1-500 | Objects 1-500 accessed as single bit registers |
| 10001-10500 | 100001-100500 | Discrete Input | 1-500 | Objects 1-500 accessed as single bit registers |
| 30001-30500 | 300001-300500 | Input | 1-500 | Objects 1-500 accessed as unsigned integer, 16-bit, single Modbus register |
| 30801 | 300801 | Input | 801 | A/UI inputs accessed as packed register of bits, 1 bit per input, LSB=input 1 (if applicable) |
| 30803 | 300803 | Input | 803 | DO outputs accessed as packed register of bits, 1 bit per output, LSB=input 1 (if applicable) |
| 31001-31500 | 301001-301500 | Input | 1001-1500 | Objects 1-500 accessed as signed integer, 16-bit, single Modbus register |
| 32001-33000 | 302001-303000 | Input | 2001-3000 | Objects 1-500 accessed as unsigned integer, 32-bit, double Modbus register |
| 33001-34000 | 303001-304000 | Input | 3001-4000 | Objects 1-500 accessed as signed integer, 32-bit, double Modbus register |
| 34001-35000 | 304001-305000 | Input | 4001-5000 | Objects 1-500 accessed as IEEE754 floating point, 32-bit, double Modbus register |

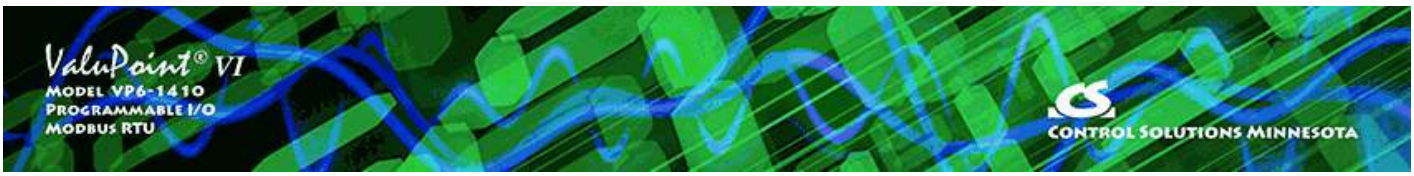
## B.2 Modbus Registers - Real Time Clock Access

The following holding registers are available for access to the battery backed real time clock/calendar in the ValuPoint. Registers 7001-7007 will return the respective element of time as of the register read. The clock could roll over between successive reads, leading to an incorrect overall time stamp. Use the registers in the range of 7001-7007 only if you are basing an algorithm on whether day is the same as previous day, etc. To capture a complete correct timestamp, read registers 7011-7017, and be sure to read 7011 first. Reading register 7011 (year) locks the rest of the time stamp and the remaining registers will return whatever the time/date was when register 7011 was read.

To set the clock/calendar, write all of registers 7011-7017, then write any value to register 7018 to trigger the write. Nothing is done with the content of register 7018 - it is only the trigger to tell ValuPoint to store the content of registers 7011-7017 into the clock/calendar hardware.

| Holding Reg. No. | Writeable | Description |
|---|---|---|

| | | |
|---|---|---|
| 7001 | No | Year |
| 7002 | No | Month |
| 7003 | No | Day of Month |
| 7004 | No | Hour (0..23) |
| 7005 | No | Minute |
| 7006 | No | Second |
| 7007 | No | Day of Week (1=Monday, 2=Tuesday, etc) |
| 7011 | Yes | Year (is also lock trigger for read) |
| 7012 | Yes | Month |
| 7013 | Yes | Day of Month |
| 7014 | Yes | Hour (0..23) |
| 7015 | Yes | Minute |
| 7016 | Yes | Second |
| 7017 | Yes | Day of Week (1=Monday, 2=Tuesday, etc) |
| 7018 | Yes | Lock trigger for write |
| 7021 | No | Minutes since midnight |
| 7022 | No | Day of year (Jan 1 = 1, Dec. 31 = 366 if leap year, else 365) |




# Appendix C - CSV File Format

The easiest way to begin the process of configuring a device by editing a CSV file is to first create one using the configuration tool. The I/O objects will be predefined. Create a couple of additional maps of the type you wish to create many of. Then go to the Object List page. Click the Save button to create a file with place holders for the objects. Then use a spread sheet program to modify the various entries to your liking.

The CSV file must contain columns labeled as follows.

Column A: Object – Numeric object number, 1 to 320, not necessarily the Modbus register number - each object can be accessed via more than one Modbus register number.

Column B: DataFormat - numeric code defining internal data format for this object:
- 0 = null/unused
- 1 = floating point
- 2 = 32-bit unsigned int
- 3 = 32-bit signed int
- 4 = 16-bit unsigned int
- 5 = 16-bit signed int
- 6 = boolean/bit

Column C: IsHardware – T if mapped to physical I/O point, or F

Column D: IsModbus – T if mapped to external Modbus slave register, or F

Column E: IsPacked - T if object is member of multiple object packed Modbus register, or F

Column F: DefPOR – T if default on POR enabled, or F

Column G: DefNOK – T if default on comm. fail enabled, or F

Column H: ReadPoll – T if periodic Read, or F

Column I: WritePoll – T if periodic Write, or F

Column J: WriteDelta – T if Write on Delta, or F

Column K: HighRegFirst - T if most significant part of data is in first register of the pair, or F (applies to data >16 bits)

Column L: EnabMaxQuiet - T if maximum quiet time enabled, or F

Column M: IsPersistent - T if present value is preserved through power outage, or F

Column N: ObjIsLinked - T if object is linked to another object, or F

Column O: PollTime – Integer, Periodic poll time in seconds

Column P: Timeout – Integer, BACnet slave timeout in seconds

Column Q: MaxQuiet – Integer, max quiet time in seconds

Column R: Scale – Real, scale factor

Column S: Offset – Real, offset for scaling

Column T: DefaultValue – Real or Integer as applicable to object type, default value

Column U: SendDelta – Real, delta threshold for Write on Delta

Column V: LinkObj - Object number that this object is linked to as the source of data for this object.

Column W: HwCfg - Hardware configuration code if object is mapped to physical I/O

(0) Unconfigured
(1) 0-10V, 12-bit
(3) 4-20mA, 12-bit with external dropping resistor
(5) discrete, active high
(6) discrete, active low
(7) dry contact, active open
(8) dry contact, active closed
(9) pulse counter
(11) resistance, 12-bit (to 30K)
(13) position pot, 12-bit, 1K to 30K

(100+) thermistor, 12-bit
(101) 10K III Fahrenheit
(102) 10K II Fahrenheit
(103) 3K II Fahrenheit
(104) 20K IV Fahrenheit
(111) 10K III Celsius
(112) 10K II Celsius
(113) 3K II Celsius
(114) 20K IV Celsius

Column X: HwQual1 - Hardware configuration qualifier value #1

Column Y: HwQual2 - Hardware configuration qualifier value #2

Column Z: RemoteRegNum – Integer, Modbus register number

Column AA: RemoteRegType – ASCII string, 2-character code, representing Modbus register type:

- 'NO' – none
- '0X' – coil(s) – uses FC15 to write
- '1X' – discrete input
- '3X' – input register
- '4X' – holding register(s) – uses FC16 to write
- '0S' – coil, use FC5 to write single
- '4S' – holding register, use FC6 to write single

Column AB: RemoteRegFormat – ASCII string, 4-character code, representing Modbus register format:

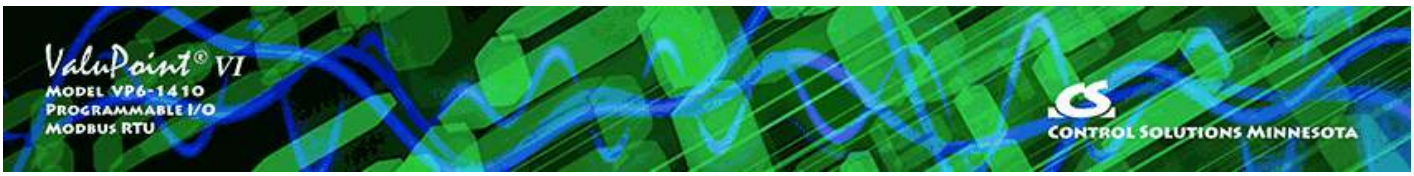
- 'NONE' – none
- 'SIGN' – signed integer (16-bit)
- 'UNSI' – unsigned integer (16-bit)
- 'SDBE' – signed double integer, big endian (32-bit, register pair)
- 'UDBE' – unsigned double integer, big endian (32-bit, register pair)
- 'FPBE' – floating point, big endian (register pair)
- 'BBIT' – bit
- 'SDLE' – signed double integer, little endian (32-bit, register pair)
- 'UDLE' – unsigned double integer, little endian (32-bit, register pair)
- 'FPLE' – floating point, little endian (register pair)

Column AC: SlaveId – Integer, Modbus slave address

Column AD: Mask – Integer, hexadecimal representation

Column AE: Fill – Integer, hexadecimal representation

Column AF: FailCount – Integer, count of comm. fails before Fault indicated




# Appendix D - Trouble Shooting

## D.1 Modbus Trouble Shooting

There are multiple ways of observing errors. One is to check error codes on the Modbus tab of the configuration tool. Another is to check the error codes being reported by the Modbus master that is polling ValuPoint as a slave. The third is to look at the LED indicators inside the ValuPoint (visible through vent slots).

There can be a variety of reasons why you are not getting the data you expect in Modbus communications. No-response errors are probably the toughest because it means no activity is being recognized. CRC errors are marginal progress because it says the devices are at least seeing some bits on the line, even if the bits don't make sense yet. Exception errors are a good sign because it means you are successfully communication with the Modbus device. Receiving an exception error requires receiving a good packet with a good CRC check. This means communication is ok, but configuration is asking for something the Modbus device does not like.

No-response errors:
• Check to see that communication parameters are correct (baud rate, etc).
• Check to see that the slave address matches.
• Check to see that Pre-Delay is at least 50 mS.
• Check wiring and power.
• Check for reversed polarity on RS485 lines. If uncertain, just try swapping them.
• Check to see that slave device is enabled for Modbus communication (many devices default to disabled)

CRC errors:
• Check baud rate and character format.
• Check wiring – if everything else is correct, CRC errors mean noise on the line.
• Check for reversed polarity on RS485 lines. Reversed polarity often looks like just noise.
• Check to see that Pre-Delay is at least 50 mS.

Exception errors:
• Check configuration. You cannot receive an exception error report if you are not successfully communicating with the Modbus device. Wiring, etc, is not a problem. Configuration has something wrong (most often the register number requested is not available).

## D.2 Modbus Exception (error) Codes

When a Modbus slave recognizes a packet, but determines that there is an error in the request, it will return an exception code reply instead of a data reply. The exception reply consists of the slave address or unit number, a copy of the function code with the high bit set, and an exception code. If the function code was 3, for example, the function code in the exception reply will be 0x83. The exception codes will be one of the following:

| Standard Modbus Exception Codes | | |
|---|---|---|
| 1 | Illegal Function | The function code received in the query is not recognized by the slave or is not allowed by the slave. |
| 2 | Illegal Data Address | The data address (register number) received in the query is not an allowed address for the slave, i.e., the register does not exist. If multiple registers were requested, at least one was not permitted. |
| 3 | Illegal Data Value | The value contained in the query's data field is not acceptable to the slave. |
| 4 | Slave Device Failure | An unrecoverable error occurred while the slave was attempting to perform the requested action |
| 6 | Slave Device Busy | The slave is engaged in processing a long-duration command. The master should try again later. |
| 10 | Gateway Path Unavailable | Specialized use in conjunction with gateways, usually means the gateway is misconfigured or overloaded |
| 11 | Gateway Target Device Failed to Respond | Specialized use in conjunction with gateways, indicates no response was received from the target device. |
| ValuPoint Specific Codes (indicating non-exception errors) | | |
| 129 | No Response | Valid only if ValuPoint is Modbus master, indicates the addressed slave has failed to respond 1 or more times. |
| 130 | CRC Error | Valid only if ValuPoint is Modbus master, indicates that a CRC error in slave's reply has been found 1 or more times. |
| 131 | No Response/CRC | Simply indicates both of the above (129, 130) are true. |

## D.3 System Fault Indications

The red LED visible inside the VP6-1410 case, viewed through the vent slots, is the system fault indicator. It will be on during initial power-up boot mode operation, but should otherwise be off except for indicating communications errors.

During normal operation, a watchdog timer is always running to force a soft restart the system in the event of a software hang. Should the system restart as a result of watchdog timeout, the system fault LED will flicker at a very rapid rate for approximately 10 seconds.

System fault indication past the initial few seconds of boot mode followed by possible soft restart indication would consist of the red LED being mostly on, flashing off briefly some number of times, followed by a longer pause remaining on. Count the number of 'off' flashes. This is the fault code.

Fault codes are as follows:

(1-4) Processor abort codes
(5) Resource allocation fault
(6) Bad configuration parameter found in system configuration
(7) EEPROM read/write failed
(8) Flash read/write failed
(9) Application checksum error
(10) Invalid application image
(11) Other unknown error

Report any of these to technical support. There are both a primary copy and backup copy of system configuration information. Both need to fail before the fault code will be indicated. These will generally indicate a hardware failure requiring factory attention. You should really never see any fault codes.

A processor abort will initially be indicated by the red LED on solid, and the yellow LED flashing a code from 1 to 6. However, the watchdog timer will normally restart the system sooner than you can observe this code and normal system fault indication will continue from that point. The abort cause is saved through soft restart. The term 'soft restart' means processor reset and complete reboot of the system as if power cycled. A power cycle is required for hard reset, and results in the same startup sequence except that processor abort codes are not retained.