

This is a printed copy of the compiled help found in the configuration tool.



User Guide

Babel Buster 2

Models BB2-2010, BB2-2011, BB2-6020
LonWorks Modbus Gateway
Rev. 1.0 – March 2014

© 2014 Control Solutions, Inc.

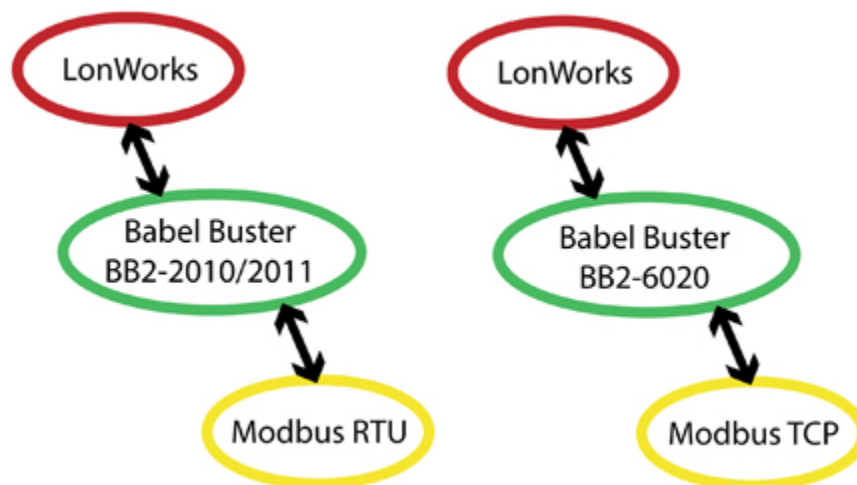
1 Introduction

1.1 How to Use This Guide

The first few sections of this user guide provide background information on how the gateway works, and an overview of the configuration process. The next several sections are guides for each of the tabs found on the screen of the configuration software. The final sections are reference material.

You should at least read the overview sections to gain an understanding of how the gateway functions. You can use the remaining sections as reference material to look up as needed. There is a help icon in the top menu bar of every page in the configuration tool software. Click the help icon (blue button with question mark) at any time to open the section of the user guide that pertains to that page.

1.2 Overview of Gateway Devices



Babel Buster models BB2-2010 and BB2-2011 are LonWorks to Modbus RTU gateways. Each has two processors, an ARM7 and an Echelon FT5000. The FT5000 is running the LonWorks Short Stack microserver, and acts as a LonWorks communications port for the main application running on the ARM7.

Babel Buster model BB2-6020 is a LonWorks to Modbus TCP gateway. It adds a third processor (another ARM7, and more importantly, more memory) that provides the Ethernet support for running Modbus TCP.

All configuration of all LonWorks gateway models is done via a local USB connection to the gateway. Although the BB2-6020 does have a TCP/IP network connection, it does not have the web server common to certain models of Control Solutions gateways. The complexity of configuration of the LonWorks gateway is not well suited to being web based. By using the USB connection for all versions of the LonWorks gateway, the configuration tool and process is consistent throughout.

1.3 Important Safety Notice

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

1.4 Warranty

This 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.

1.5 Required License Information

The BB2-LON configuration tool includes 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.

2 Overview of Gateway Functions

2.1 Object Server Model for a Gateway

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

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

Control Solutions gateways will function as servers, providing a copy of the most recent data found in its data base when a client requests that data. In master/slave terms, the server is a slave while the client is a master. Some applications will treat the gateway as a server from both (all) networks connected. But most applications will want the gateway to be a server on one side, and a client on the other side. The most frequent application of the BB2-2010 or BB2-6020 will have it functioning as a Modbus master (client). The notion of master or slave, however, does not really apply in LonWorks because it is a peer to peer network with no particular master.

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

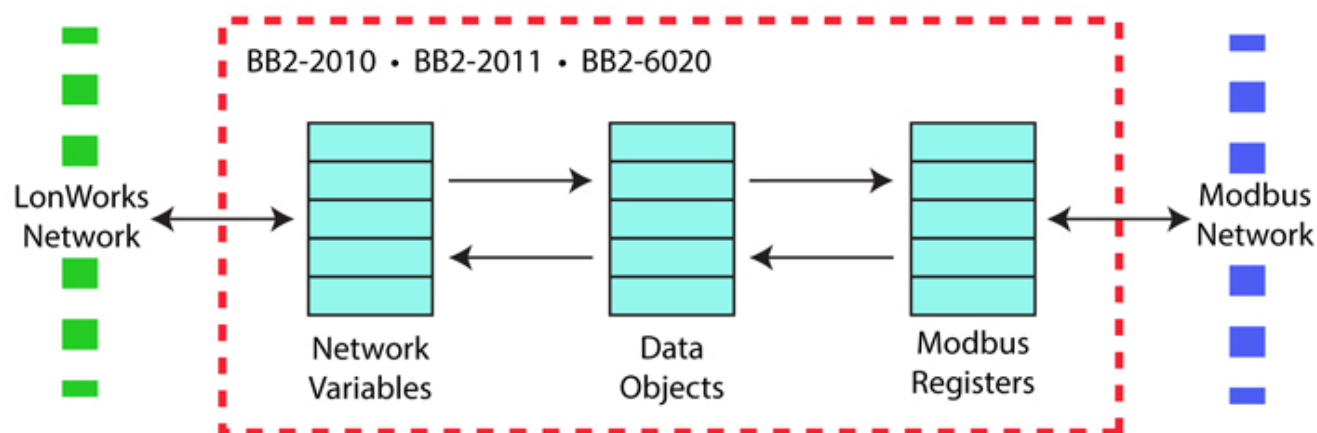
The BB2-2010 can be configured as a Modbus RTU master or slave (client or server), but can never be both at the same time (as specified by Modbus protocol). The BB2-6020 can be master and slave (client and server) at the same time since Modbus TCP supports this possibility.

2.2 Data Flow in the Gateway

The LonWorks network always interacts with Network Variables (NVs) in any LonWorks device, including the Babel Buster 2 LonWorks Gateways. A network management tool such as Echelon's LonMaker is used to "bind" NVs in the gateway with NVs in other devices.

A Network Variable Input (NVI) means other devices will write data to this NV. The NVI is receiving "input" from the LonWorks network. A Network Variable Output (NVO) means data is being written to ther devices. The NVO is sending "output" to the LonWorks network.

There are threeThere different realizations of the same data within the LonWorks gateway. The direct connection to the LonWorks network is the Network Variable. The direct connection to the Modbus network is the Modbus register. The translating connection in between is a gateway data object.



At first, it may not seem clear why there needs to be these different realizations of the "same" data. The fact that the protocols are incompatible is why we need a gateway in the first place. But why not simply send the data "as is"? The following example is a frequently used translation.

The Network Variable in the diagram below is LonMark type SNVT_temp. The LonMark specification for SNVTs (Standard Network Variable Types) provides the scaling for temperature transmitted over the LonWorks network as SNVT_temp. The raw data is a 16-bit binary number, but scaling provides 0.01 degree resolution. There is also a Kelvin offset in SNVT_temp. As a result a temperature of 77°F is a raw value of 2992 in the LonWorks NV. That is probably not what your Modbus PLC wants to see.

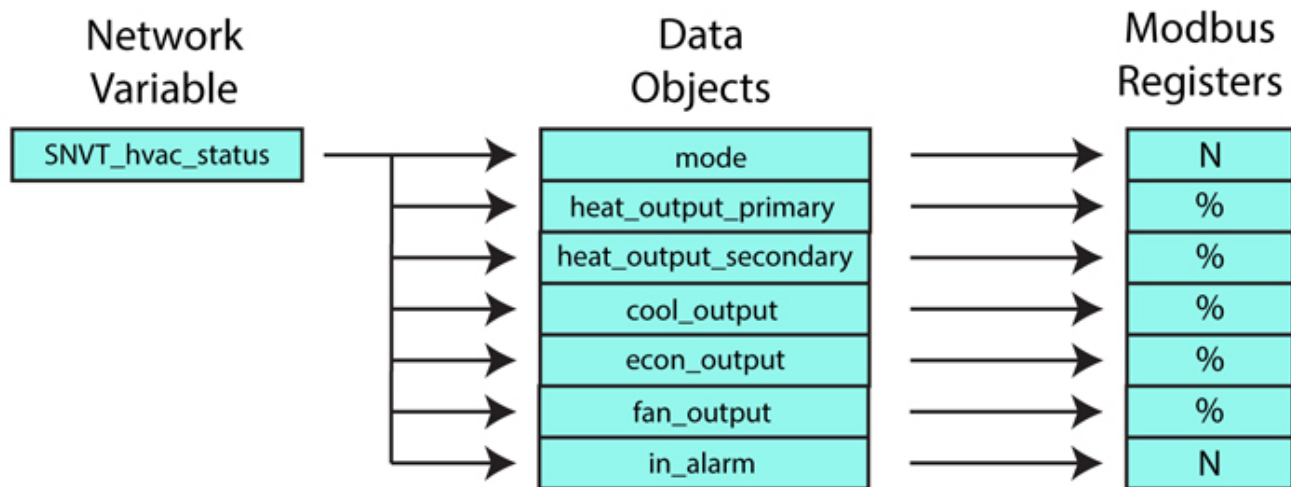
Conversion from LonMark types to standard engineering units is done internally by the LonWorks gateway. When the NV is SNVT_temp, and the internal data object is defined as type 'floating point', the value contained in the internal data object is 25.2°C (when the raw data is 2992). LonWorks values are always metric. The Modbus register mapping provides for further scaling. By applying a scale factor of 1.8 and offset of 32, the Celsius reading now becomes Fahrenheit when provided to Modbus as a holding register.



Next we will discuss an even more compelling reason why data cannot simply be sent "as is". Many LonWorks Network Variables are "structure", meaning a single network variable actually contains multiple pieces of data. A Modbus register can contain only a single value. Therefore, it is not possible to do a direct one-to-one translation of these network variables to Modbus registers. In the case of structured data, a single NV translates into multiple Modbus registers.

A commonly used structured variable is SNVT_hvac_status. It contains seven data fields. These need to be mapped to seven different Modbus registers in order for Modbus to do anything meaningful with the status information. This mapping is partly automated by the configuration software tool provided for the Babel Buster 2 LonWorks Gateways. We say "partly" automated because you do need to select whether or not to include the variable in your mapping, and you also need to enter any applicable scaling on the Modbus side. Scaling for standard NV types will be handled automatically. Scaling for user defined NV types (UNVTs) needs to be entered manually.

The configuration software does not provide any tutorial on what various SNVTs are specified by LonMark. You need to go to the LonMark web site (www.lonmark.org) to obtain a copy of this documentation. If you do not already have a copy, you are strongly encouraged to obtain one since *it is unlikely that you will be able to effectively use any LonWorks gateway without an understanding of the LonWorks Network Variables.*



2.3 LonWorks Objects and Function Blocks

LonWorks devices are required to operate as a collection of LonWorks objects. All LonWorks devices have, at the very minimum, a Node Object. Beyond that, the type and number of objects is entirely up to the manufacturer. The Babel Buster 2 LonWorks Gateways have a total of 241 objects summarized in the following table.

Object Type	LON Object #	NV index	FB array index	Gateway FB #
Node	0	0 .. 2	n/a	n/a
Open Loop Sensor	1 .. 120	3 .. 122	0 .. 119	OLS 1 .. 120
Open Loop Actuator	121 .. 240	123 .. 242	0 .. 119	OLA 1 .. 120

A "LonWorks Object" refers to the function block that encapsulates one or more network variables (NV). In the case of the LonWorks gateways, each function block (FB) contains one network variable. The LonWorks Object number is used when making an object request via the Node Object. Such requests are made by network management tools, especially during network commissioning, and have no use directly related to gateway data translation.

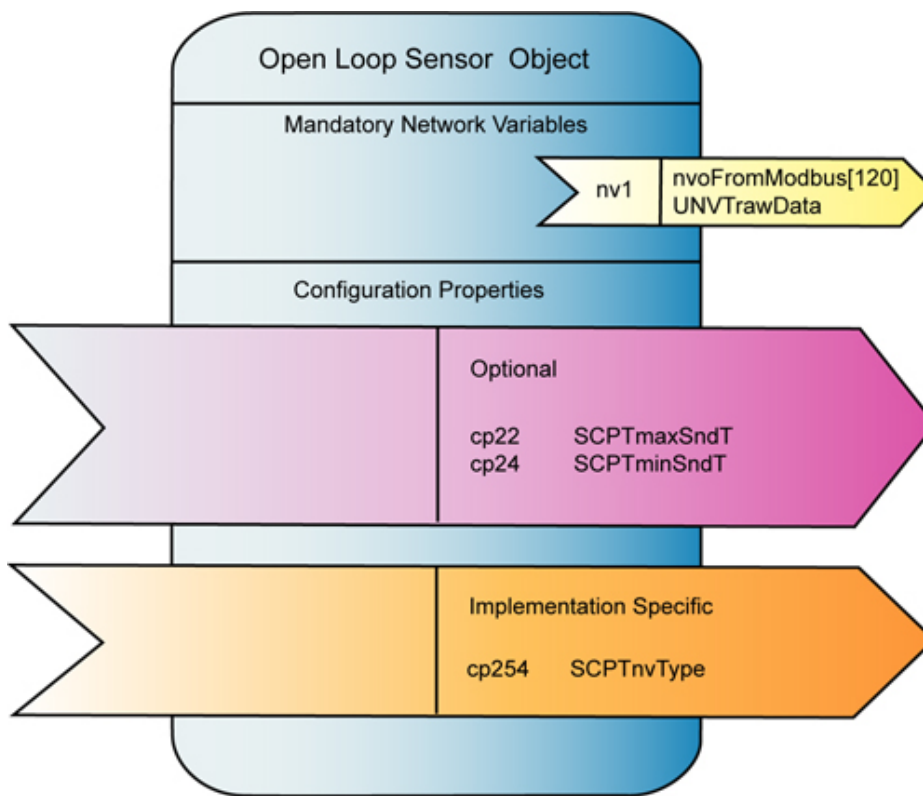
The LonWorks Object and internal gateway "object" are not the same object - they simply use the term "object" to refer to the entity. In the diagrams above, and throughout gateway configuration discussion, the term "Network Variable" is synonymous with LonWorks Object, and "object" refers to the gateway's internal object. Unless the word "object" is preceded by "LonWorks", the reference to "object" will mean internal gateway data object throughout this user guide.

The NV index is used to look up network variables within the LonWorks NV processing. You will usually have little reason to be concerned about NV index, except that this is the number you will see if you are looking at network traffic on the LonScanner protocol analyzer, or using software such as Nodeutil.exe to directly query NVs.

The FB array index is what you will see in the list of variables when using the LonMaker browser to look at variables and configuration properties in the gateway. The gateway FB number is how you will see the FBs referenced in the configuration software for the gateway.

The Open Loop Sensor FB treats Modbus as sensor hardware.

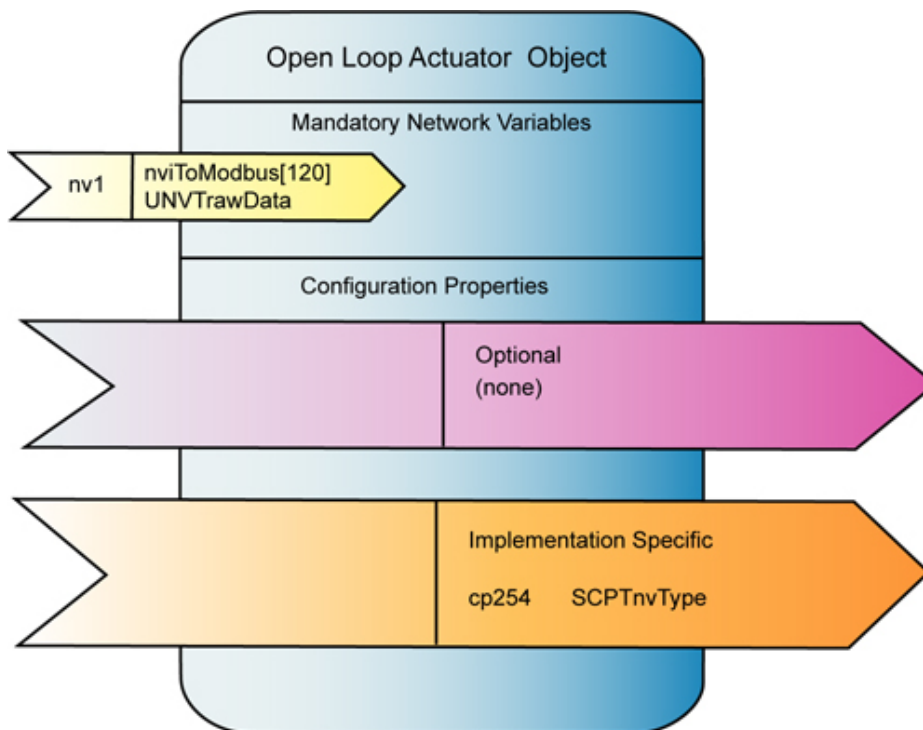
If the gateway is the Modbus



master, the gateway reads Modbus registers from slave devices.

If the gateway is a Modbus slave, it waits for an external master to write to registers in the gateway.

Either way, when new data is obtained via Modbus, it is propagated to the LonWorks network, specifically to any NVs in other nodes that are bound to this NV.



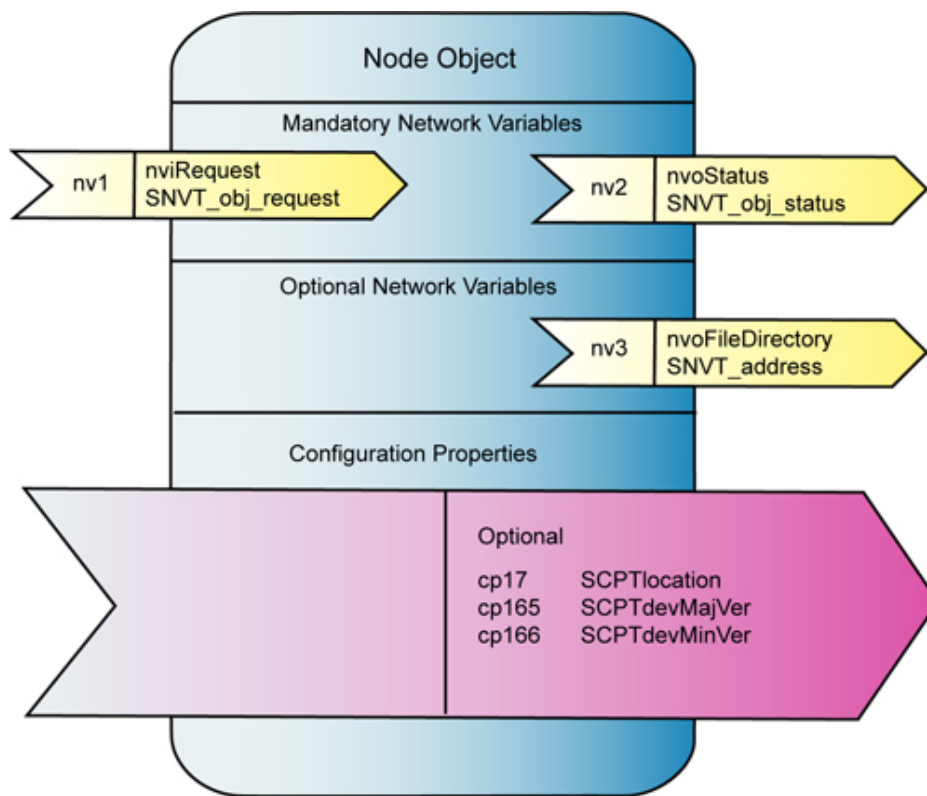
The Open Loop Actuator FB treats Modbus as actuator hardware.

If the gateway is the Modbus master, the gateway writes to Modbus registers in the slave device.

If the gateway is a Modbus slave, it waits for the external master to read registers from the gateway.

Either way, when new data is written to the NV by the LonWorks network (typically by other LonWorks nodes bound to this NV), that data is made available to Modbus.

The Node Object is required primarily as a maintenance feature of the LonWorks device. It is used to enable and disable all other objects in the device, and is used to



check status of other objects. The Node Object also provides the file directory which is necessary in order for the network management tool to access the device's configuration properties.

You normally have no need to be concerned with the Node Object for purposes of configuring the gateway, but this object is critical to network management tools for commissioning the gateway on the network.

3 How to Configure the Gateway

3.1 Pick Your Approach

The configuration process, in summary, is a matter of mapping Network Variables to Modbus registers. There are three basic approaches to configuring the gateway:

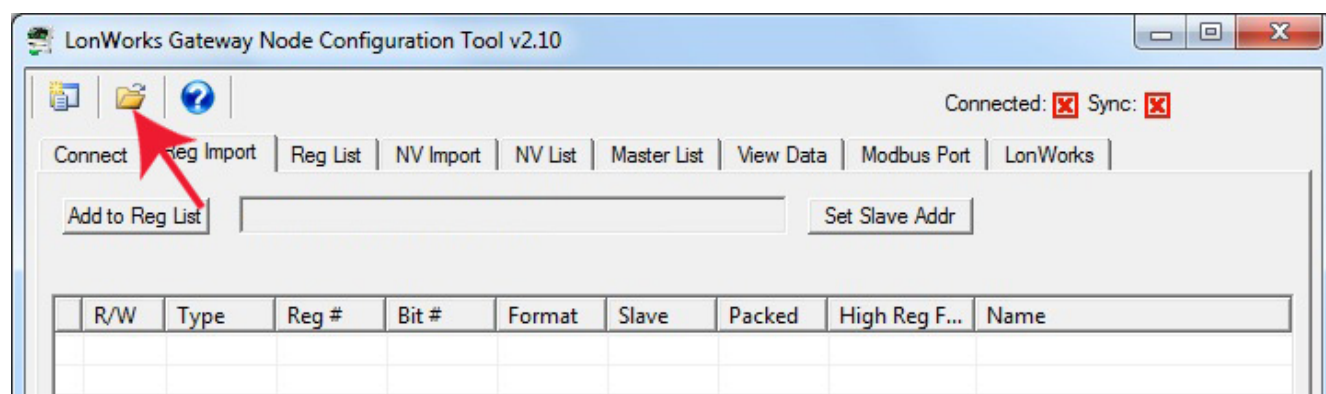
- [Start with spreadsheet of Modbus registers and auto-build the LonWorks network variable set.](#)
- [Start with XIF file for LonWorks device, auto-build the Modbus register set that best matches it.](#)
- [Start from scratch and build configuration entirely manually.](#)

Choose the approach that is most meaningful to you.

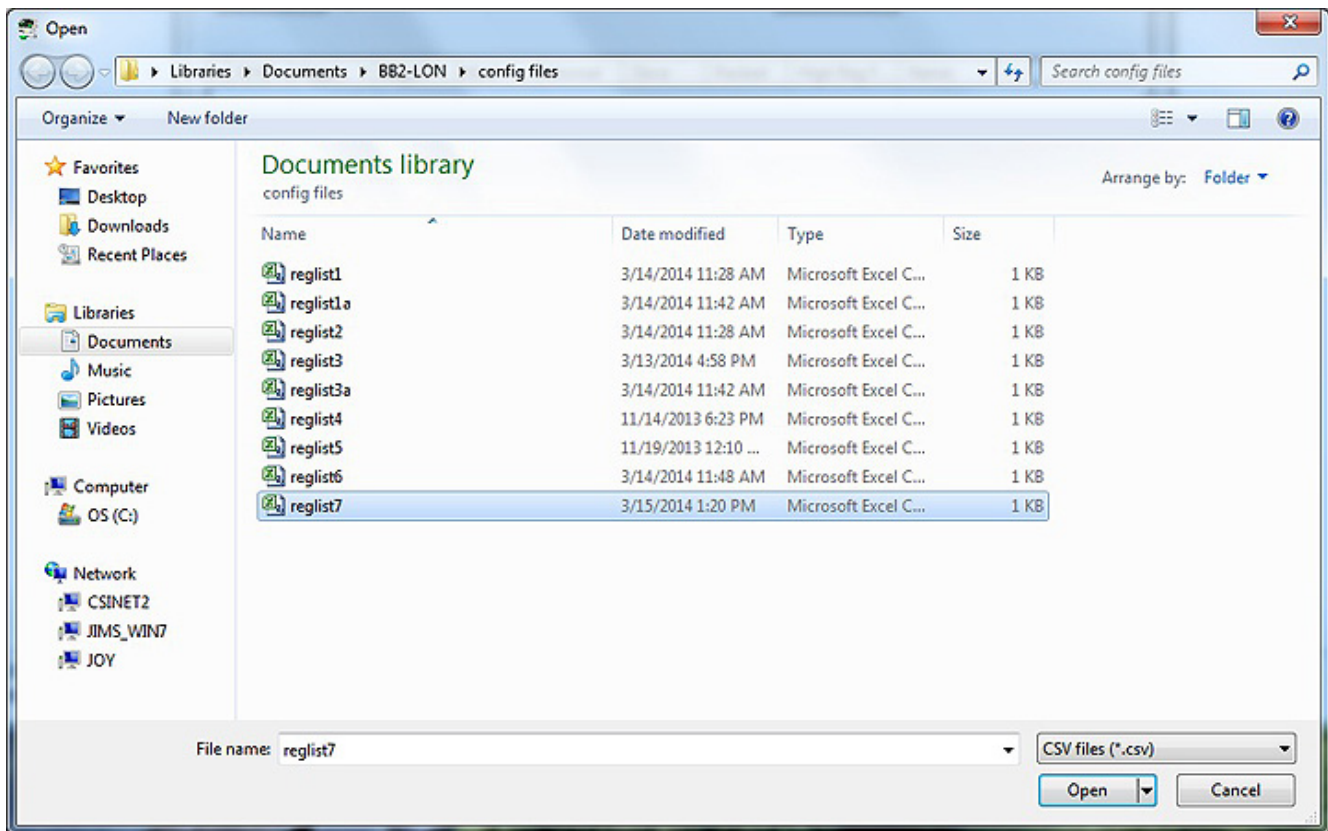
3.2 Build Configuration from CSV List of Modbus Registers

Starting with a Modbus register list for a specific Modbus device, and creating a LonWorks interface for it, is by far the most common application of the LonWorks gateway. You will need to obtain a register list from the manufacturer of the Modbus equipment. If you are lucky, you will receive this in the form of a spreadsheet that you can simply edit. If not, you will need to create a simple spreadsheet and save it in CSV format.

To build your gateway configuration from a CSV file, start by going to the Reg Import page, then click on the file open icon.

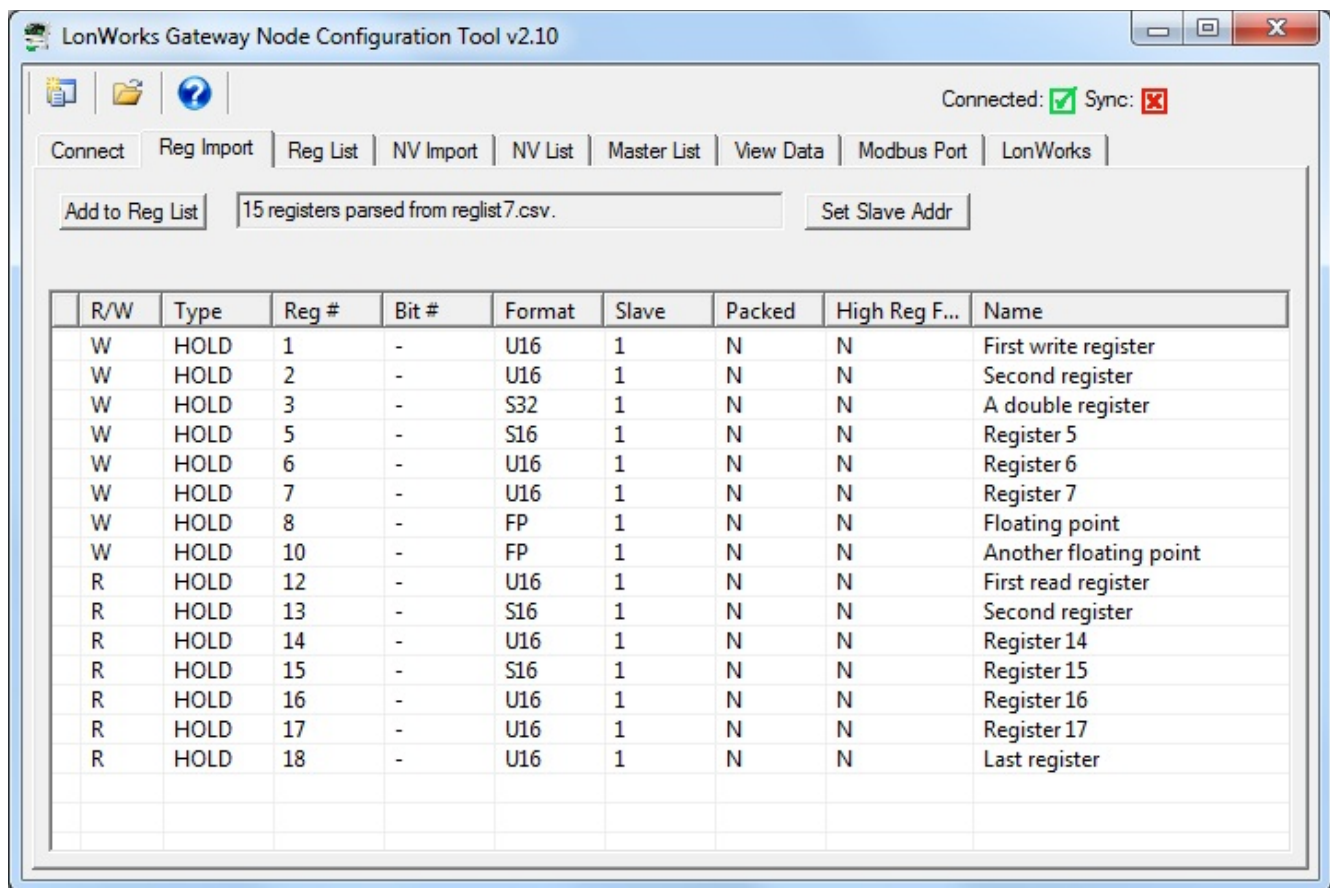


You will see the familiar file open dialog. Select your CSV file.



For reference, the content of the reglist7.csv file loaded here contains the following CSV list, and upon loading, will appear as shown in the next screen shot.

```
RW,MODICON,FORMAT,SLAVE,BITNUMBER,PACKED,HIGHREGFIRST,NAME
W,40001,U16,1,,F,F,First write register
W,40002,U16,1,,F,F,Second register
W,40003,S32,1,,F,F,A double register
W,40005,S16,1,,F,F,Register 5
W,40006,U16,1,,F,F,Register 6
W,40007,U16,1,,F,F,Register 7
W,40008,FP,1,,F,F,Floating point
W,40010,FP,1,,F,F,Another floating point
R,40012,U16,1,,F,F,First read register
R,40013,S16,1,,F,F,Second register
R,40014,U16,1,,F,F,Register 14
R,40015,S16,1,,F,F,Register 15
R,40016,U16,1,,F,F,Register 16
R,40017,U16,1,,F,F,Register 17
R,40018,U16,1,,F,F,Last register
```

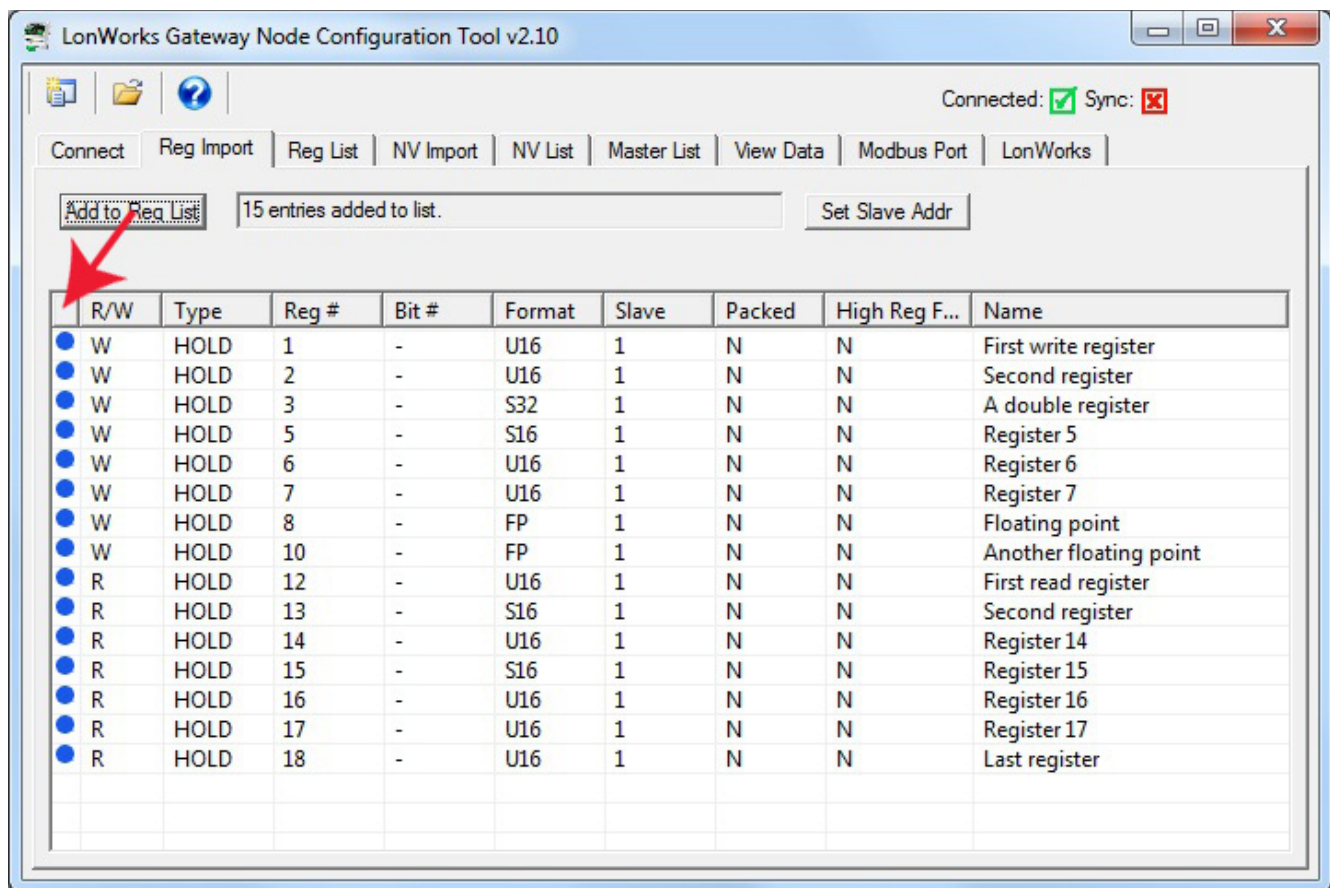



At this point, the register list is contained in what amounts to a "scratch pad". The CSV file is imported into an interim list so that you can choose which registers you want to include in your configuration. If you created the CSV file, you most likely want all of them. But if somebody else sent you a CSV file containing every register in their Modbus device, you will usually want to select only those that are pertinent to your application.

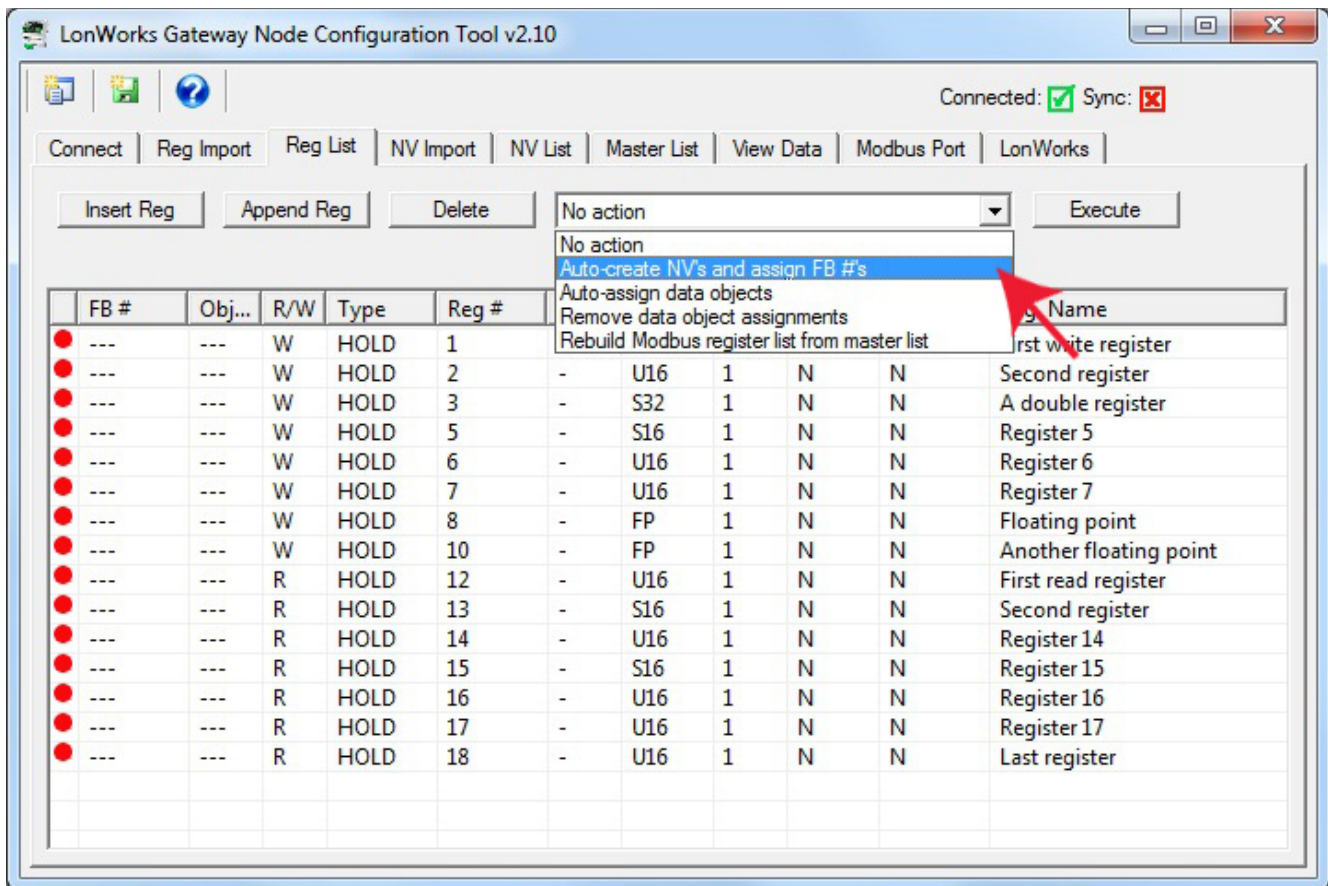
To select all registers, click the icon column header as illustrated by the red arrow below. This will cause the blue dot to appear on each line, which indicates that this register has been selected to be included in the configuration. To select only certain registers, click the icon area of only those rows you wish to include.

Once you have selected the registers, click the "Add to Reg List" button. Until you click this button, you have no registers in the configured register list. You may, at this point, import another CSV and continue to add multiple registers from multiple CSV files.

If you will be connecting two or more of the same type of Modbus device, each having identical register sets, click "Add to Reg List". Then use "Set Slave Addr" to select the next device's slave address. Now click "Add to Reg List" again. This adds the same set of registers a second time, but with a different slave address the second time.

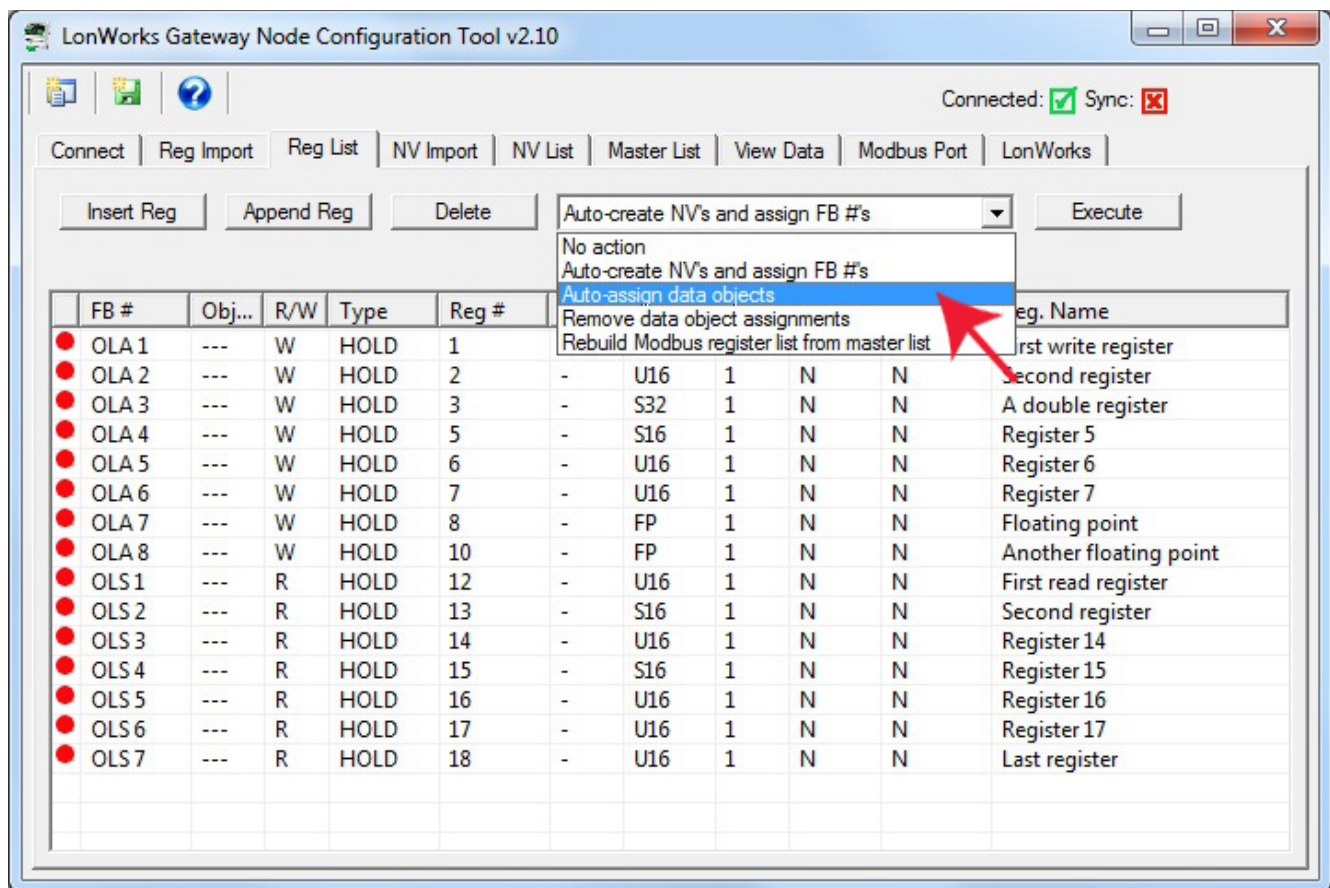


Once you have imported your register list(s), go to the Reg List page. Here is where you begin the auto-build of the rest of your configuration. Start by selecting "Auto-create NV's and assign FB #'s". Then click the Execute button.

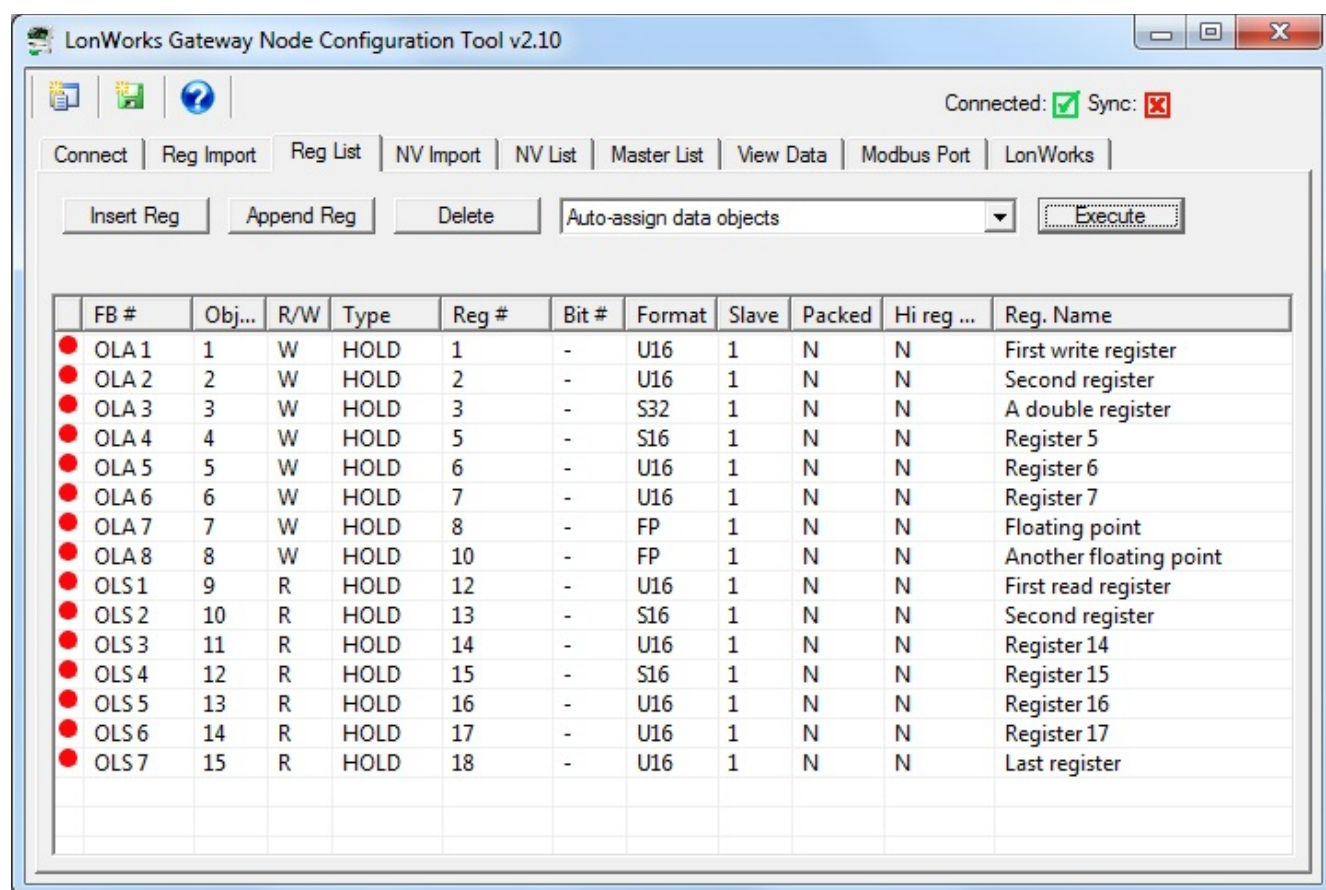


Upon executing the Auto-create NV's, the FB# column will be populated. Modbus registers designated as "W" for write, meaning the gateway will write to these registers in the Modbus device (assuming gateway is master), will be assigned to an Open Loop Actuator function block having a Network Variable Input (NVI). Modbus registers designated as "R" for read, meaning the gateway will read these registers from the Modbus device, will be assigned to an Open Loop Sensor function block having a Network Variable Output (NVO).

Next, select "Auto-assign data objects" from the list and click Execute.



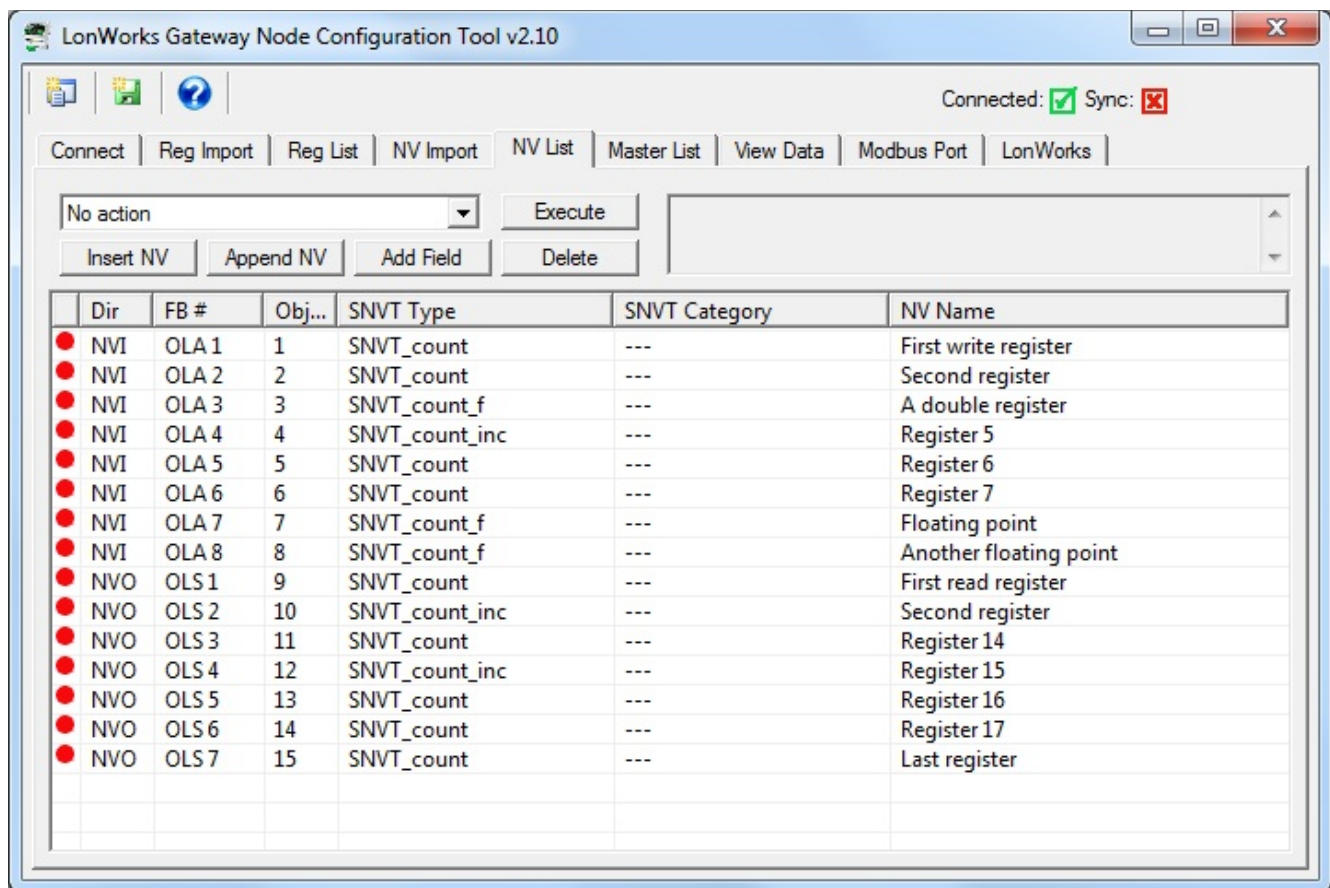
Upon executing Auto-assign data objects, the object numbers allocated will appear in the Obj column.



Creating of the configuration is now largely complete. You can make alterations to things like data scaling at this point if you wish. You can also change network variable types at this point. But for simply putting your Modbus device on the LonWorks network using generic counts as the data type, the configuration is complete. All that remains now is to send the configuration to the gateway device. At this point, the configuration only exists on your PC. It needs to get written into the device before the gateway will be functional.

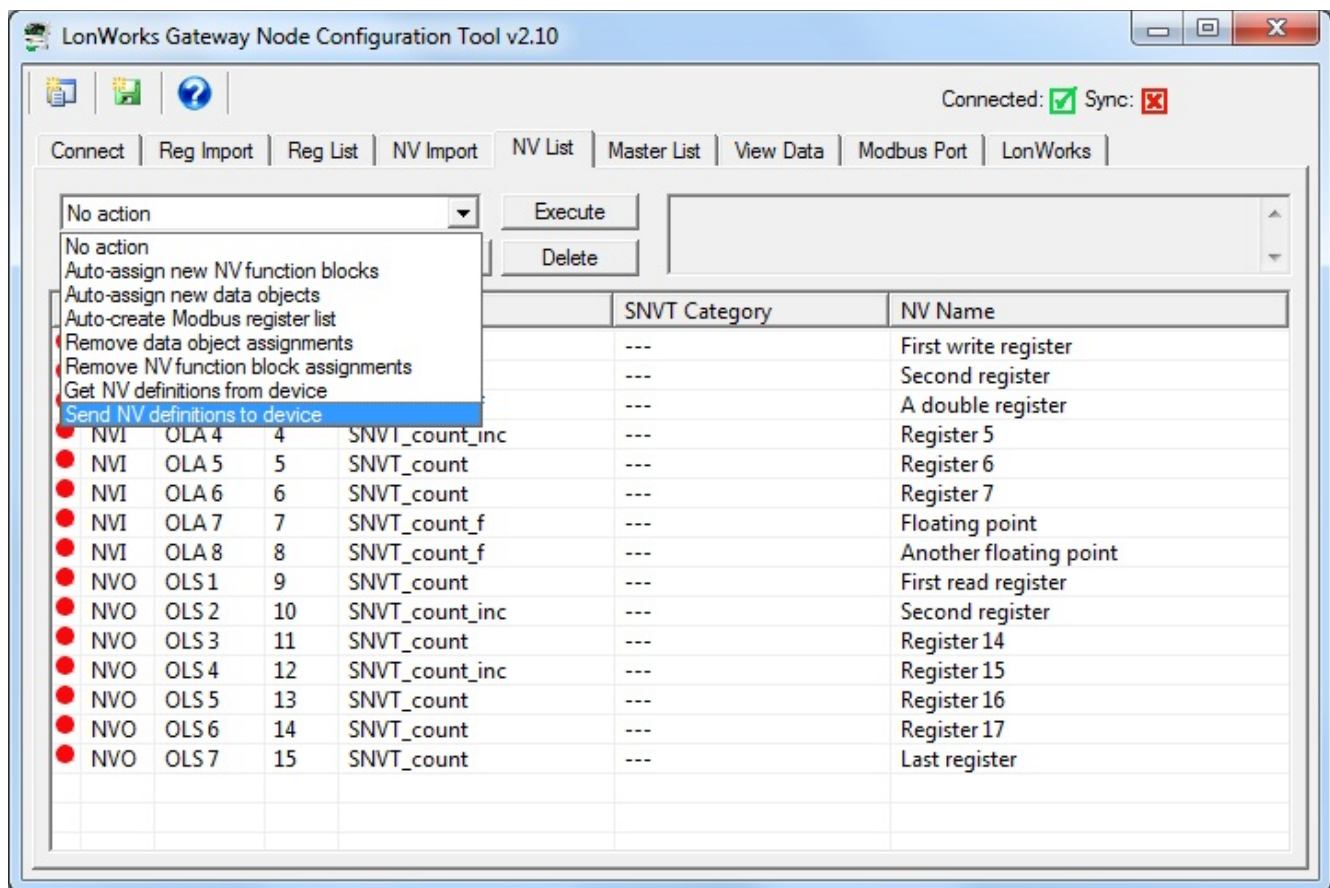
Go to the NV List page. This is the set of network variables that were automatically created for you. If you wish to export an XIF file defining your LonWorks device, you may do so here by clicking the green file icon to create a new file. Some network management tools such as LonMaker can import the XIF directly from the device and you do not need an XIF file. However, some network management tools do require the XIF file, and this is where you create it.

Note: If you want your LonWorks interface to have network variable types other than those that were selected by default, STOP HERE and make those changes. Refer to the NV List page section in this user guide for more detail on changing NV types.

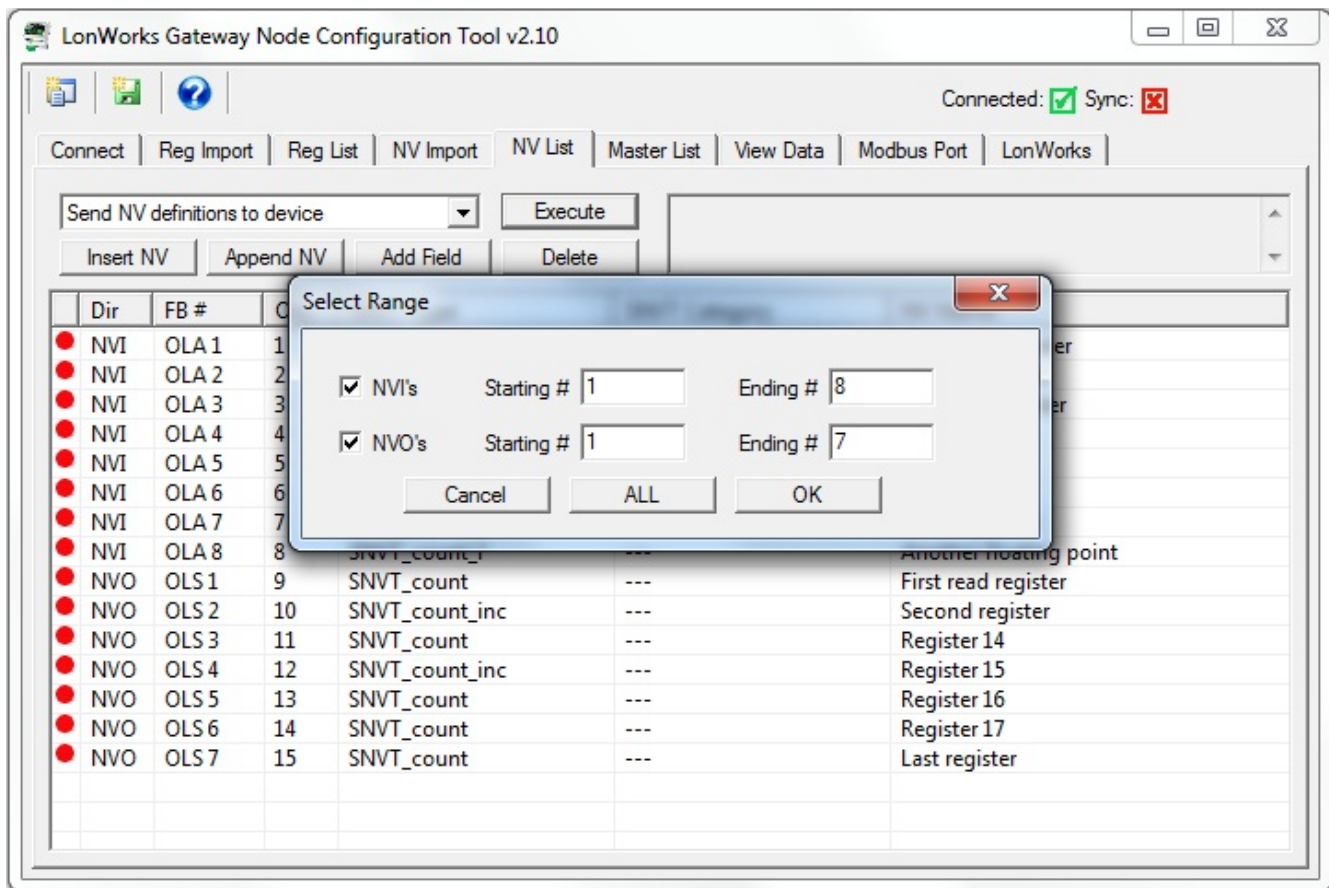


There are two steps in writing the configuration to the gateway device. The first step is to send the network variable (NV) list to the gateway. The second step is to send the data objects (which include the Modbus register mapping) to the gateway.

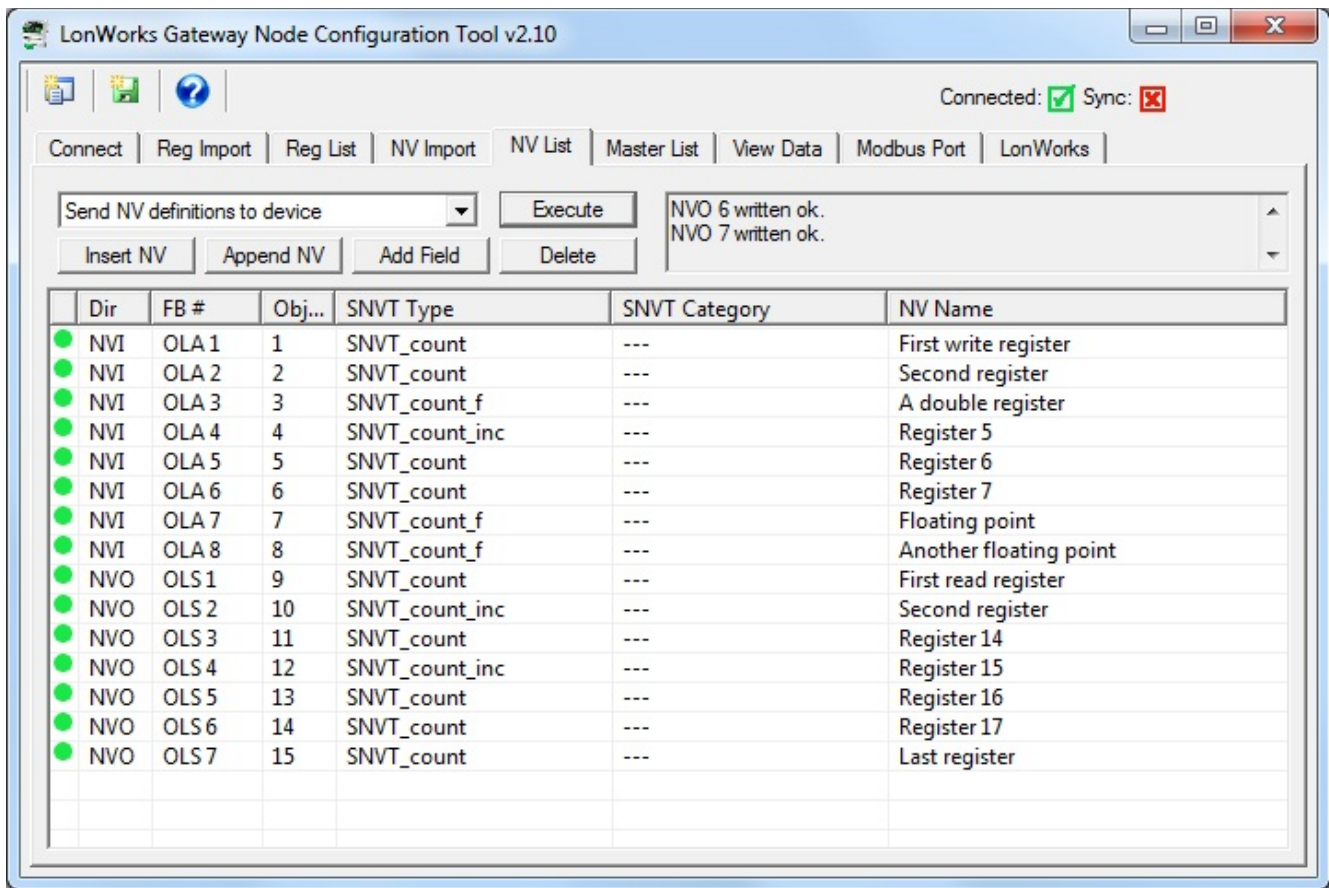
To send the NV list to the gateway, select "Send NV definitions to device" from the list and click Execute.



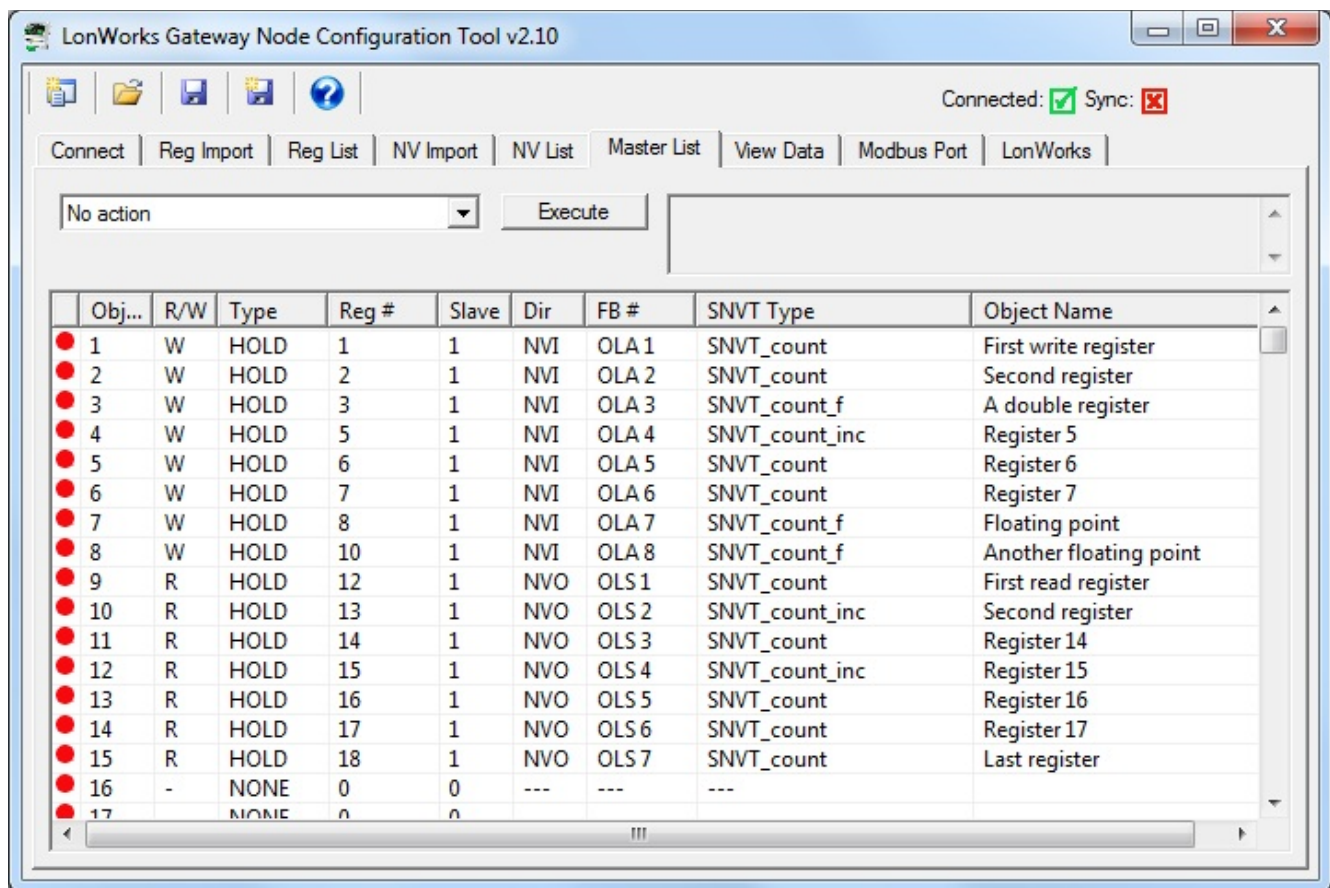
When you click Execute to send the NV list, it will scan the configuration and see how many NV entries are actually used. This will appear in the Select Range dialog. Generally, with a newly created configuration, you would simply click OK. If you later make changes and need to re-send those changes, you can alter the range to send only those items changed.



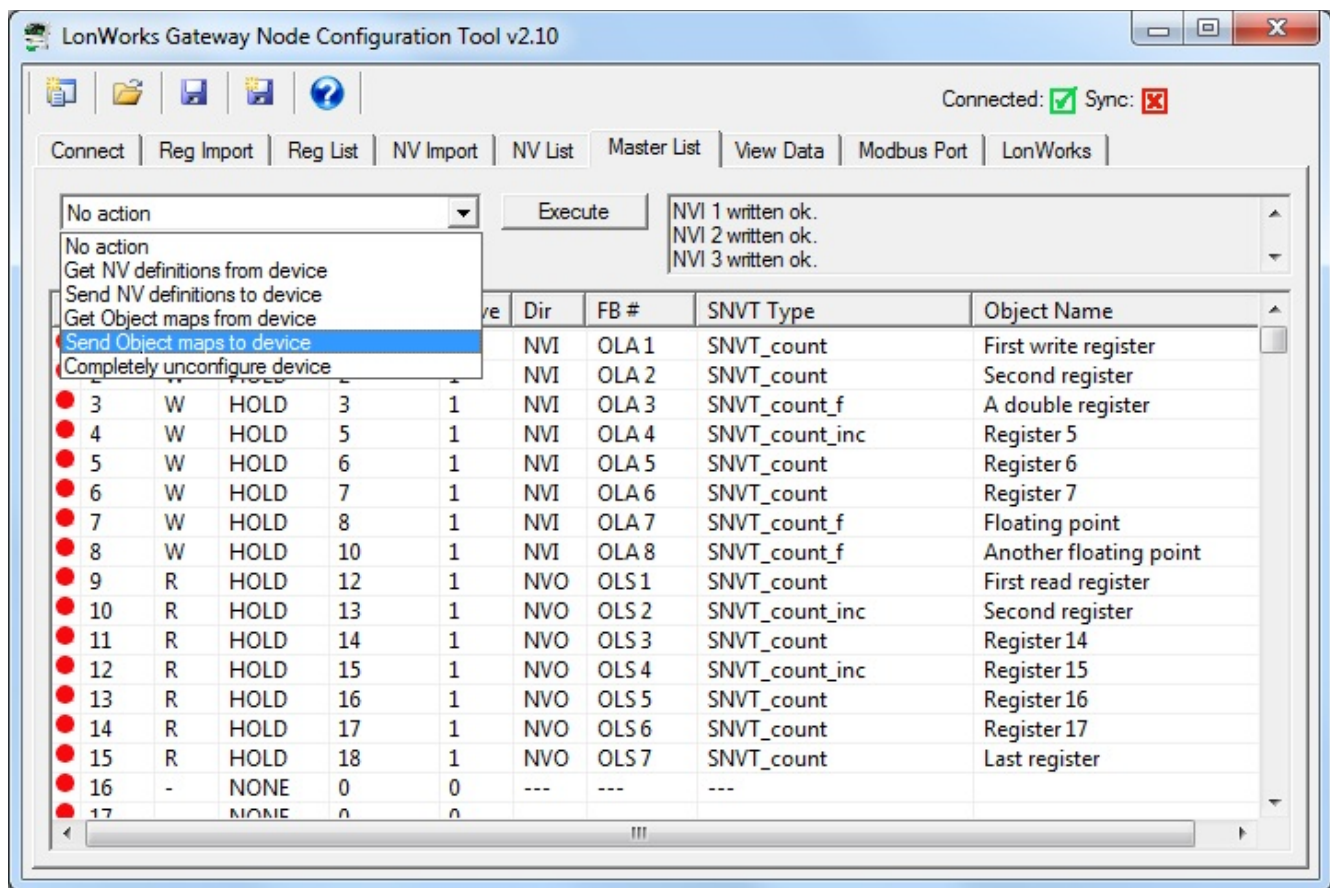
The red dot in the icon column means this line of configuration information has not yet been sent to the gateway device. After executing the Send NV definitions to device, the icons will turn green indicating that the tool software and the device are now in sync and information displayed on the screen is also contained in the device. If you make any changes, the icon will turn red again for the line that was changed. You would then need to re-send.



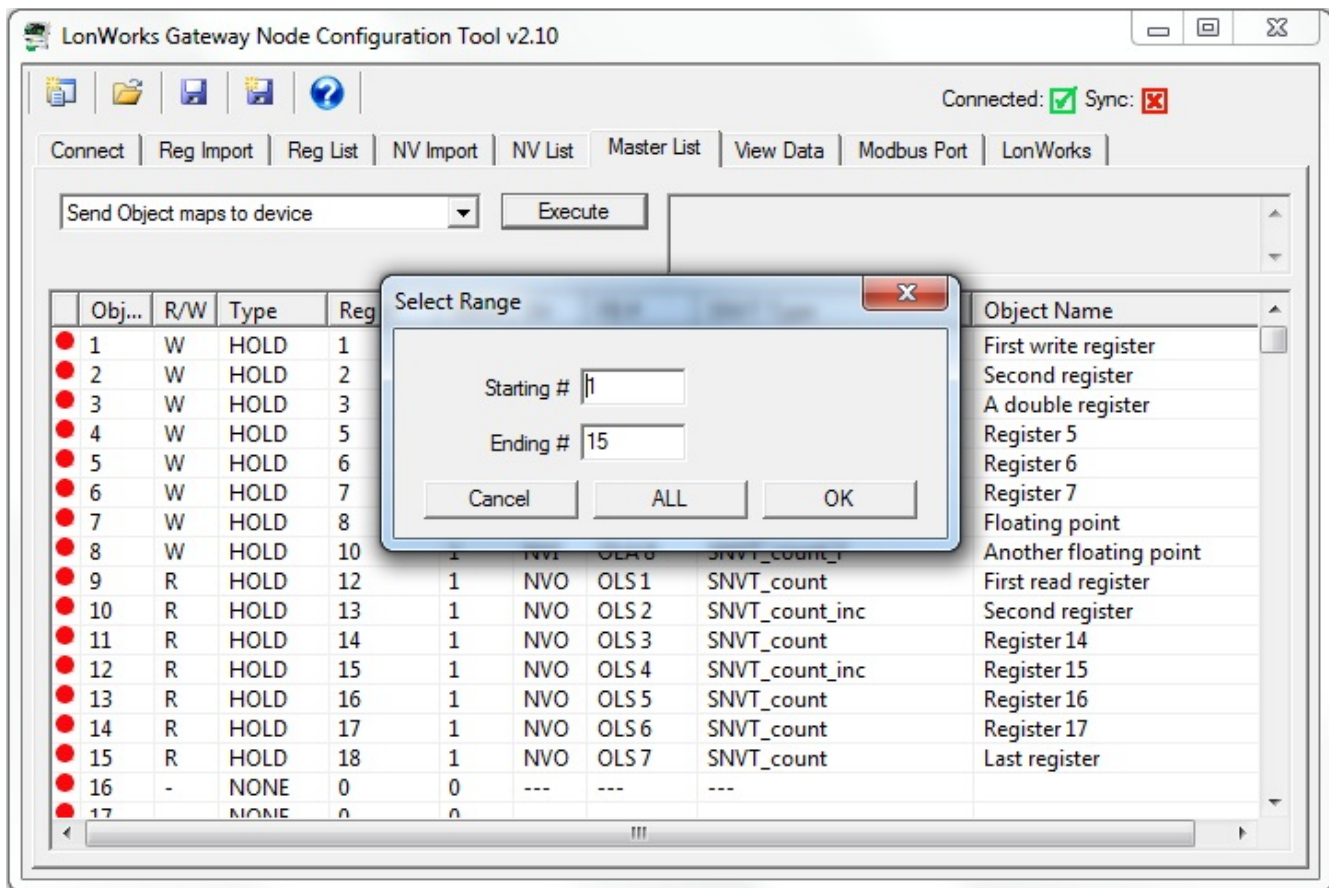
One of two steps have now been completed. Next, go to the Master List page. The Master List is, as the name implies, the master list for your device's configuration. This is where connections from LonWorks network variables to Modbus registers is made. Refer to the Master List section of this user guide for more detail about changes you can make from this page. For purposes of this example, we will assume that you are going to configure the gateway entirely with the default values that were automatically built.



The second step of configuring the gateway device is to send the master list, or object maps, to the device. Select "Send Object maps to device" from the list and click Execute.

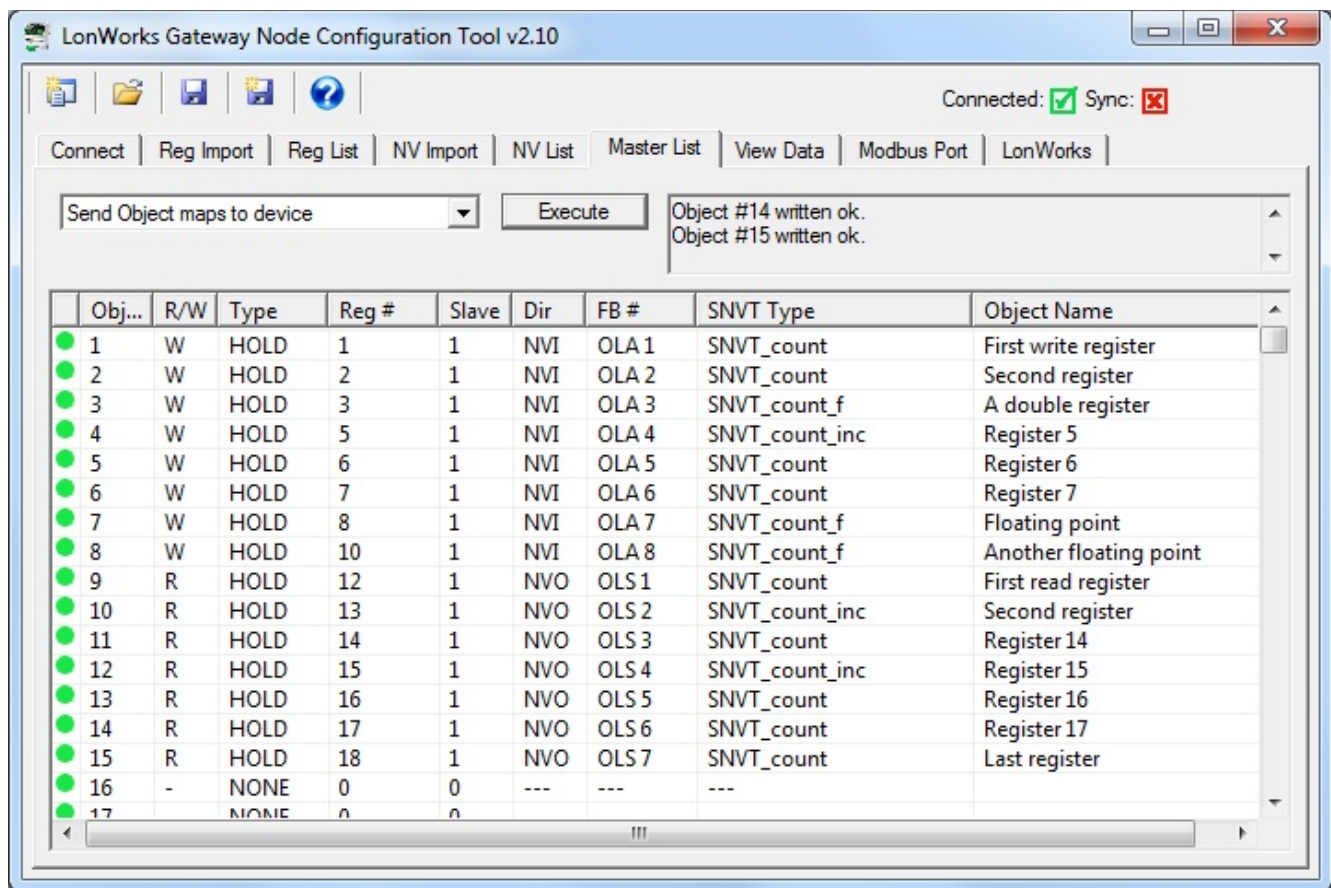


Upon clicking Execute (with Send Objects to device selected), you will see the range dialog again. It will default to sending only those object maps that are valid but not yet sent to the device. Click OK to send the range of object maps shown.



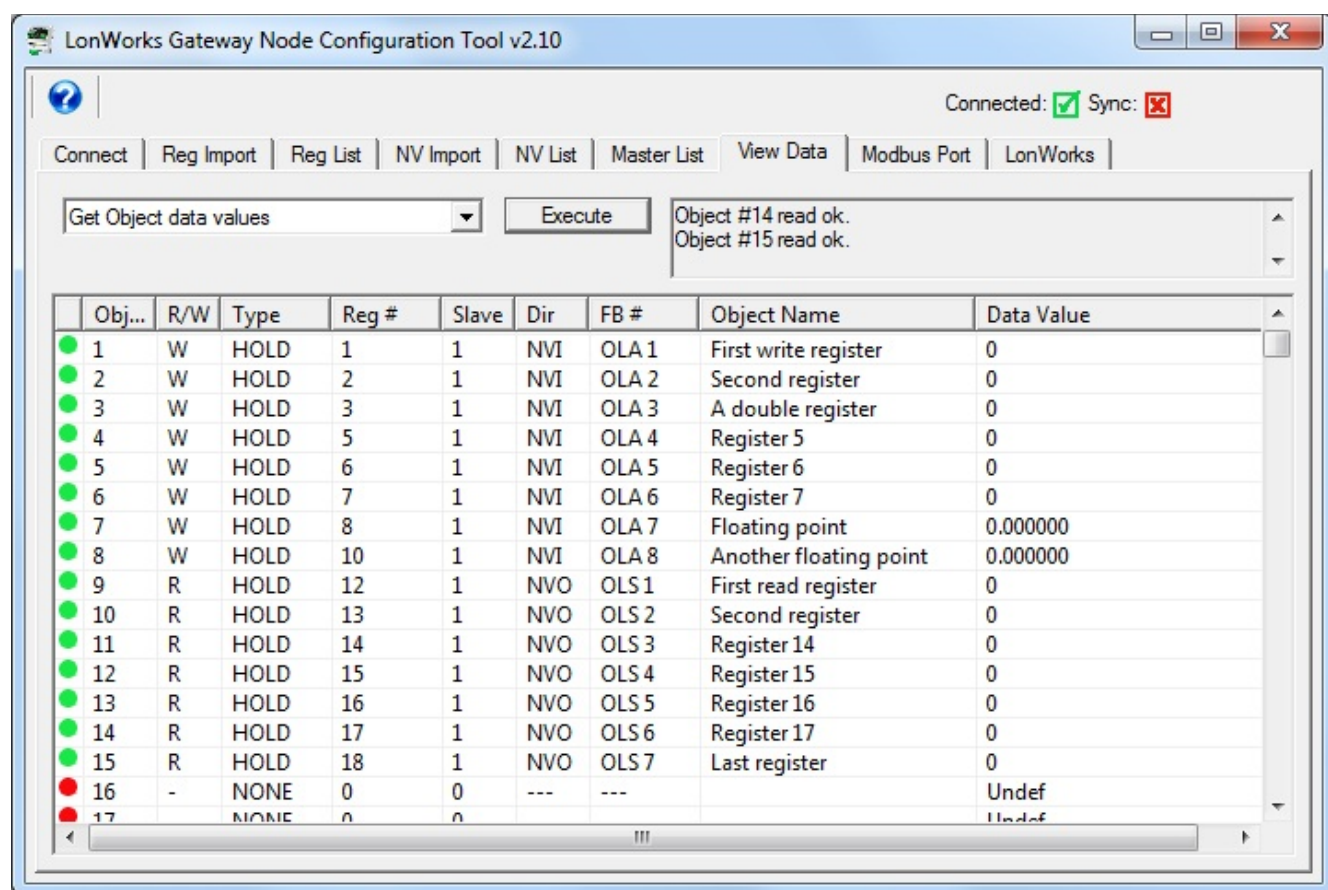
The icon in the first column will be red if the object map has not yet been sent to the gateway device. It will turn green after being sent to the device, and the configuration tool and the device being configured are now in sync.

If you close the tool software but are confident that the gateway device has already been configured, you can connect to the gateway and use "Get Object maps from device" to rebuild the list and confirm the configuration. When doing this, you should get NV definitions first, then get Object definitions. This is because the configuration will be incomplete if the object maps cannot find any network variables in the list to associate with.



The device configuration is largely complete once you have sent both the NV list and Object list to the device. You should also visit the Modbus Port page to set up things like Modbus RTU baud rate, etc. You may also want to visit the LonWorks page to set the node location property.

Once configuration is complete and the gateway is functioning as a gateway, you can retrieve a list of object data by going to the View Data page and selecting Get Object data values.

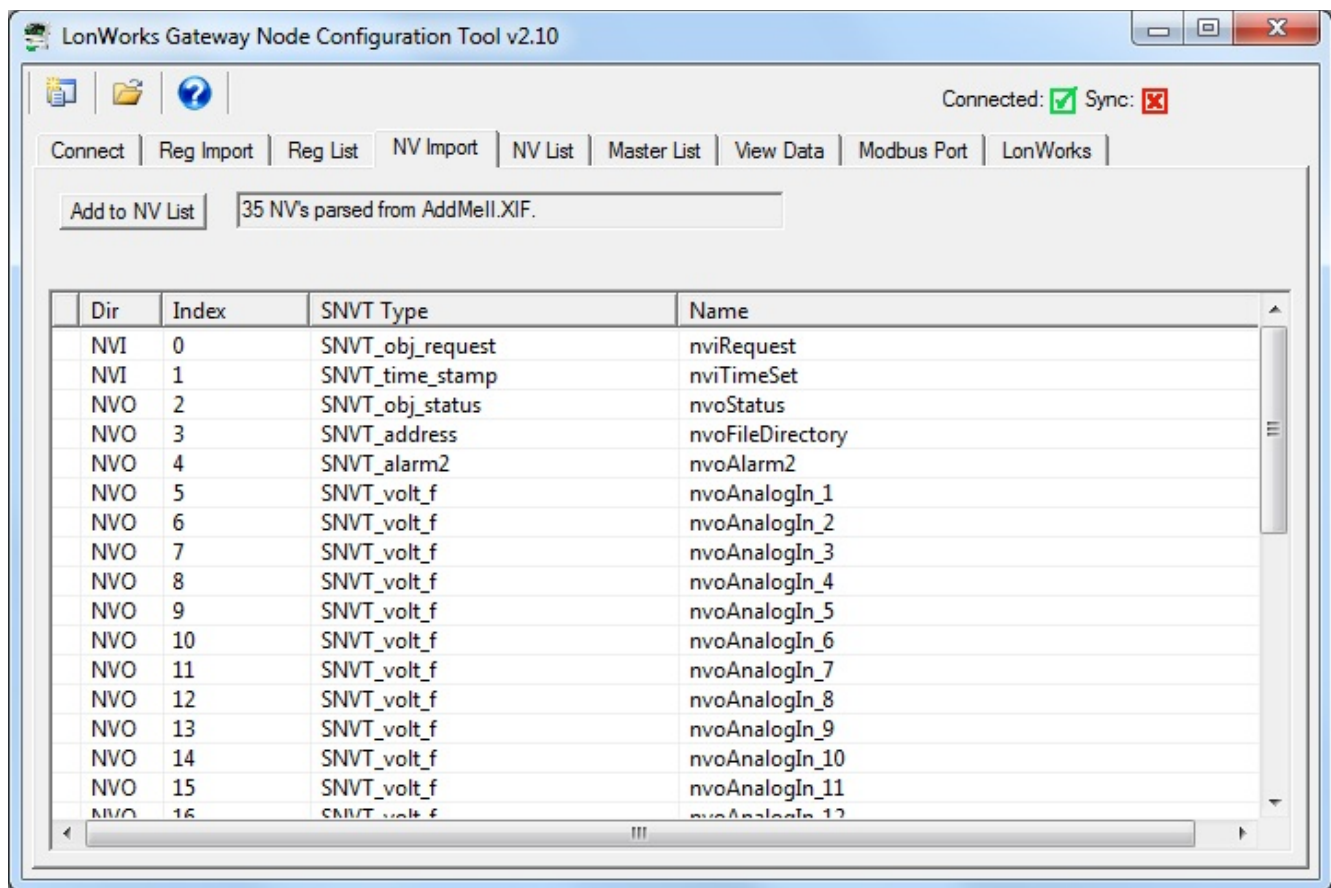


3.3 Build Configuration from XIF File (LonWorks Device)

Starting from an XIF file for a specific LonWorks device, and auto-building the Modbus register list, is the approach you would take if you want to make a Modbus device mimic that particular LonWorks device on the LonWorks network.

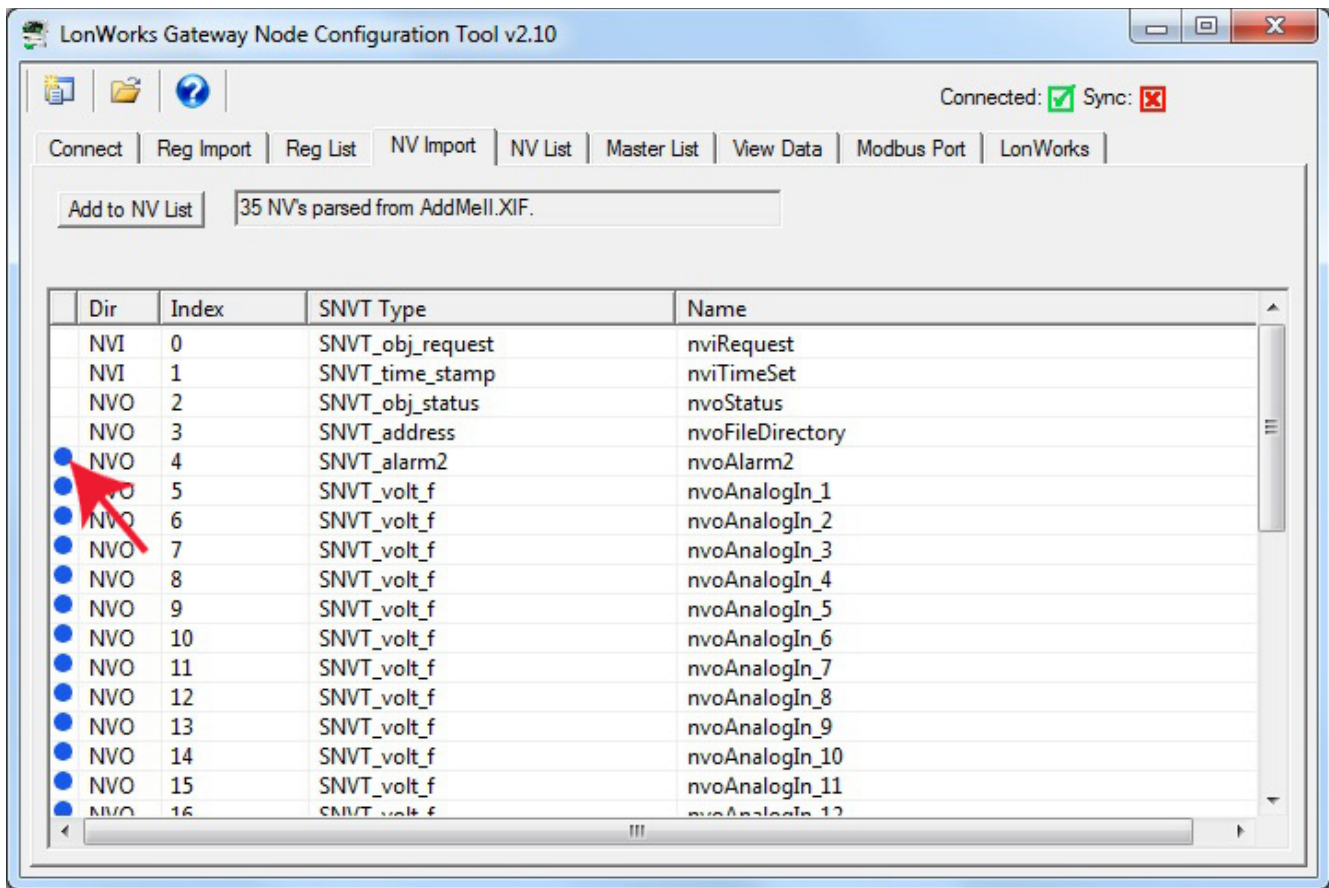
IMPORTANT: If your intent is to put a LonWorks device on a Modbus network, this is not the correct gateway to use. The gateway discussed here only applies where you are interfacing Modbus devices to a LonWorks network, and the LonWorks network is the primary network of the overall system.

To begin building your configuration from an XIF file, skip the Reg Import page and go to the NV Import page. Click on the file icon to open an XIF file. Once the XIF is imported, the list of network variables will be displayed on the NV Import page.

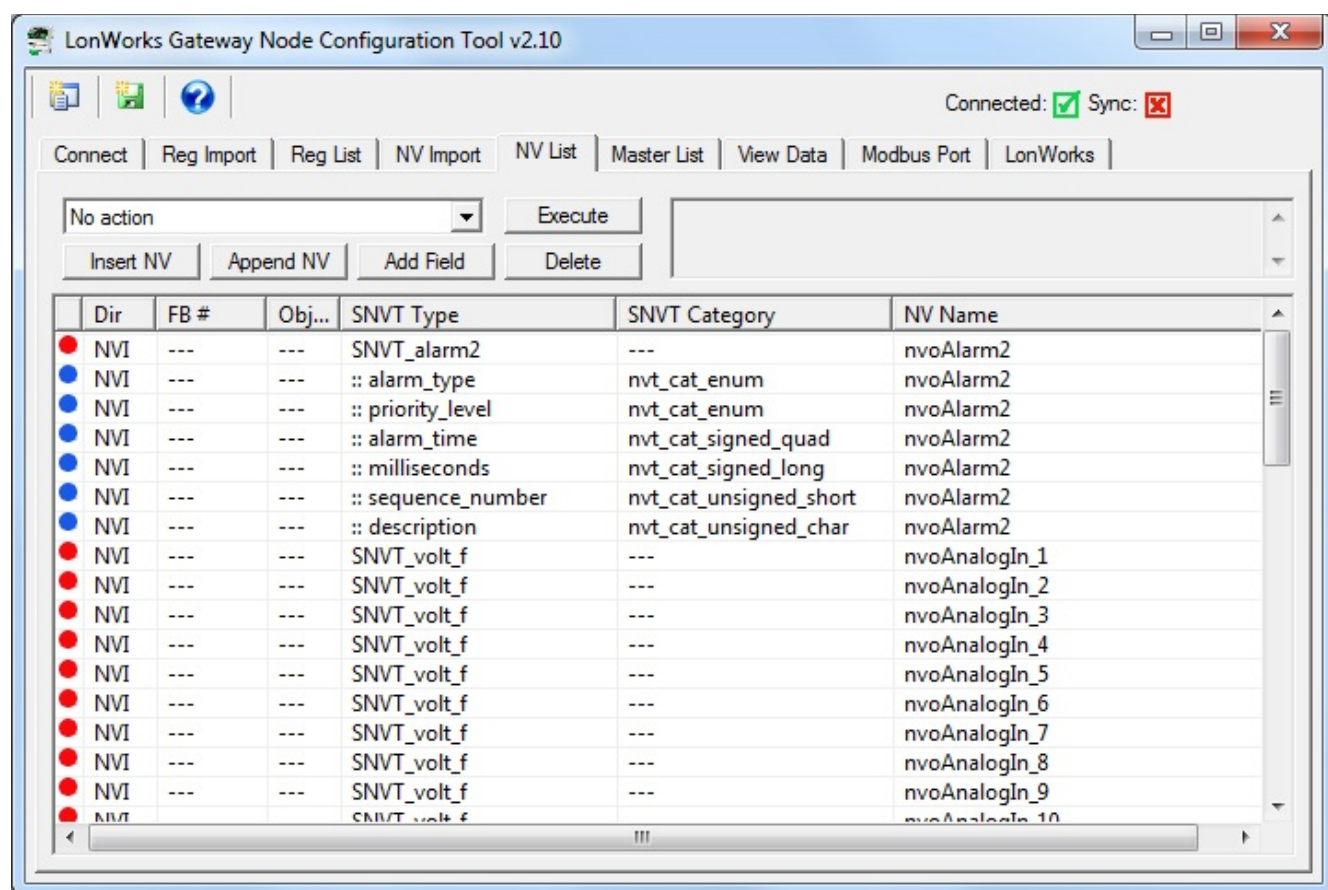


The NV Import page is essentially a scratch pad where you import the content of XIF files, then select which of the available variables you wish to include in your gateway configuration. Click on the icon column header to select all items, or click on the icon column for individual lines to select only those lines. The icon will show a blue dot for those lines that are about to be included.

When you have made your selections, click Add to NV List. The selected variables are now copied to the NV List.



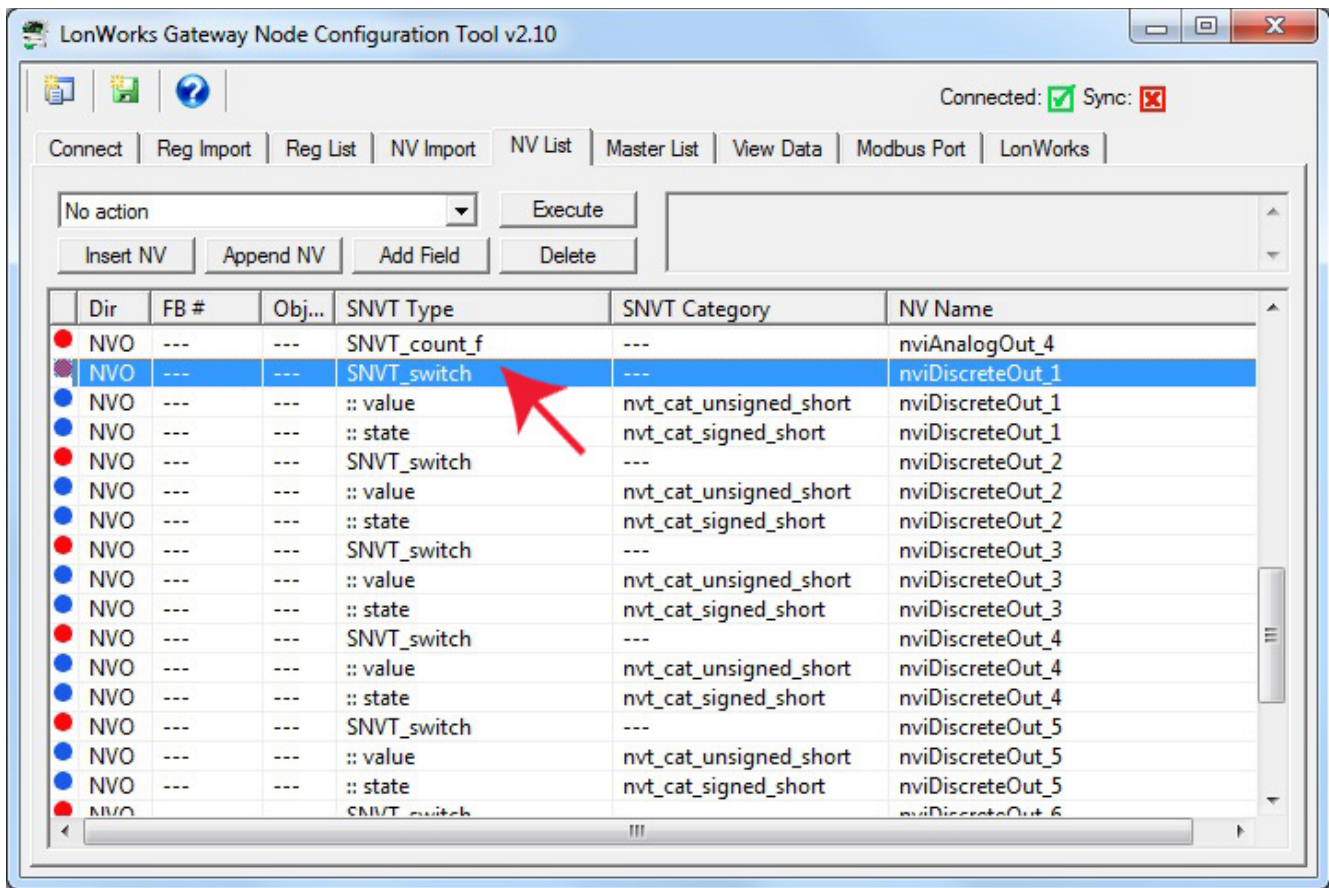
The NV List is the definition of the list of network variables that will become present in your LonWorks device (i.e. this gateway) when viewed as a node on the LonWorks network. The icon in the first column will be red if this NV definition has not yet been written to the gateway, and green if it has been written to the gateway. Blue icons indicate fields of structured network variables. All lines with a blue icon are part of the network variable immediately preceeding the set of blue icons. There will be only one network variable, but multiple Modbus registers, for a structured network variable (refer to previous section in this user guide if you did not already review treatment of structured variables).



If you wish to make changes to the NV List, this is the point where you should do so. Do not proceed to assign function blocks (FB #'s) or data objects until the NV List is finalized.

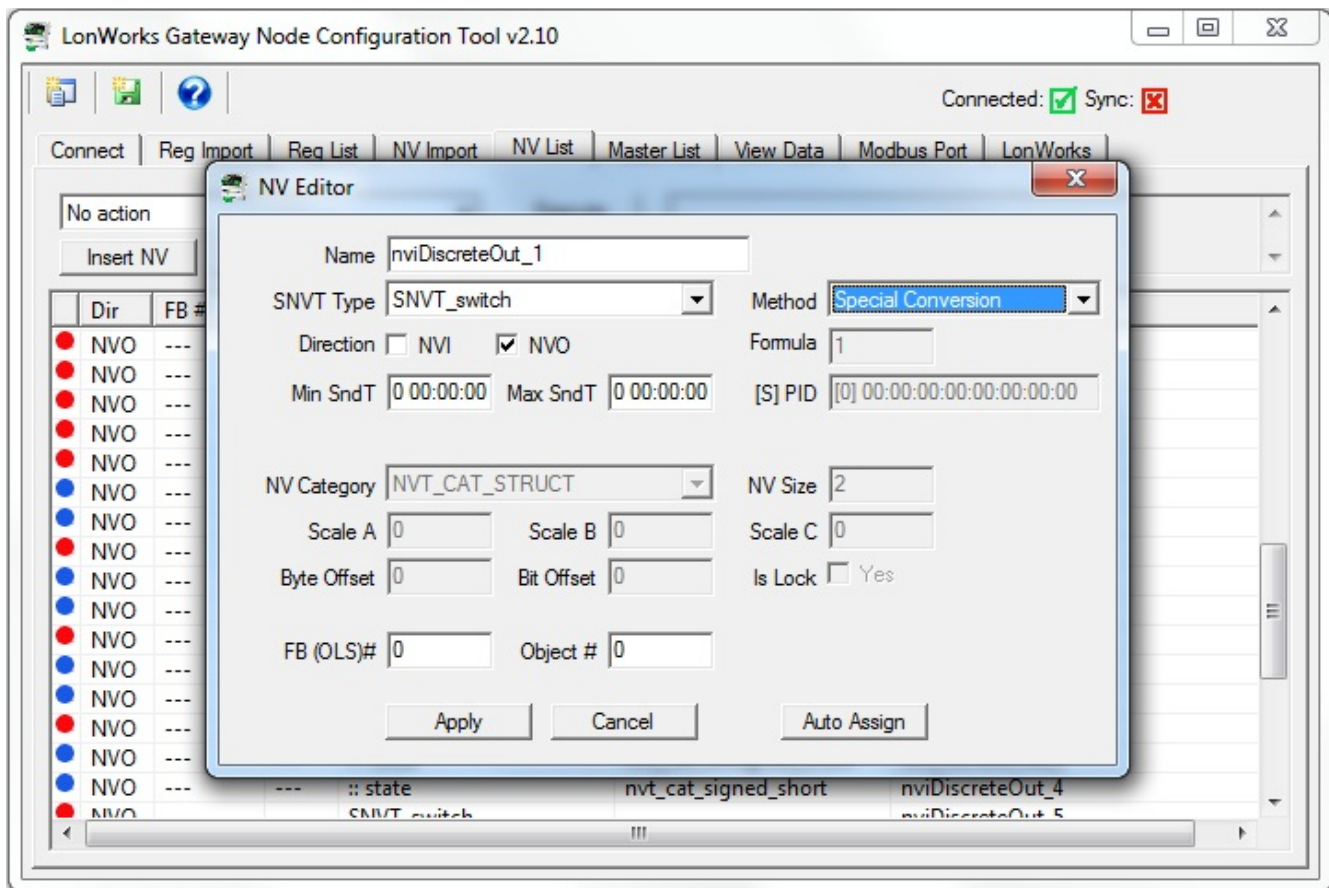
You have a few options here, and these are described in more detail in the NV List section of this user guide. You may add network variables, delete them, or change what type they are. If they are structured, they will be automatically expanded into a list of all of their fields. If you add a structured NV which is not a standard LonMark type, you will need to add fields manually to build up the structure.

Some network variable types provide for special conversions. SNVT_switch is one such NV. To modify the NV definition, double click on the respective line in the NV List.

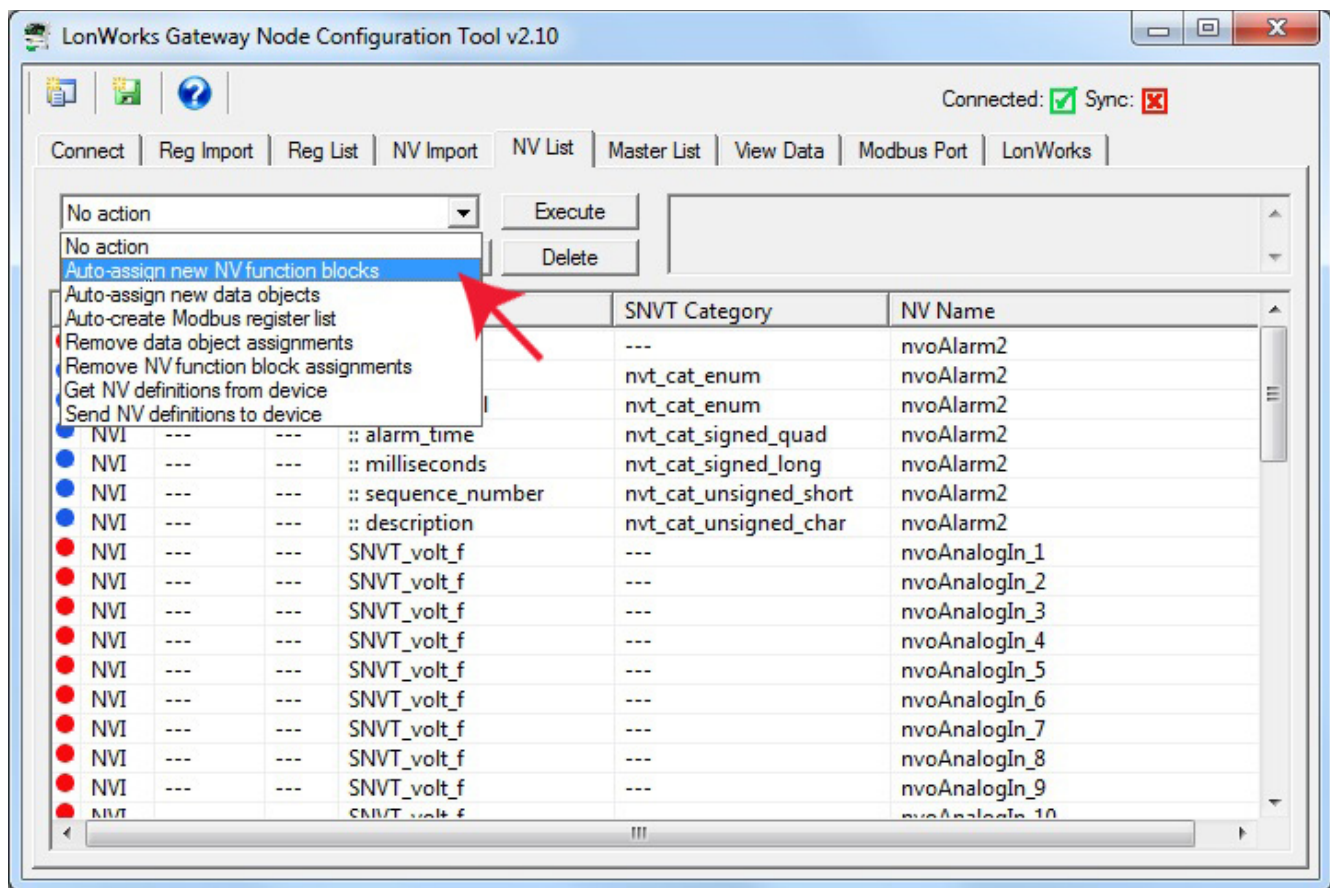


Upon double clicking a network variable in the NV List, the NV Editor dialog will appear. The SNVT_switch example is illustrated here. The LonMark definition of a "switch" actually contains two elements of data, a state and a level (as would be used for a dimmer switch). You cannot correctly control anything via a SNVT_switch without properly dealing with both elements of data. In Modbus terms, this would require two Modbus registers to control one switch, and this is not typically desirable from a Modbus point of view. Therefore, the Babel Buster gateway provides a "special conversion" such that a single Modbus register containing a value from 0 to 100 (implied percent) will result in both parts of the SNVT_switch being set correctly to control on/off (100% or 0%) or any level in between.

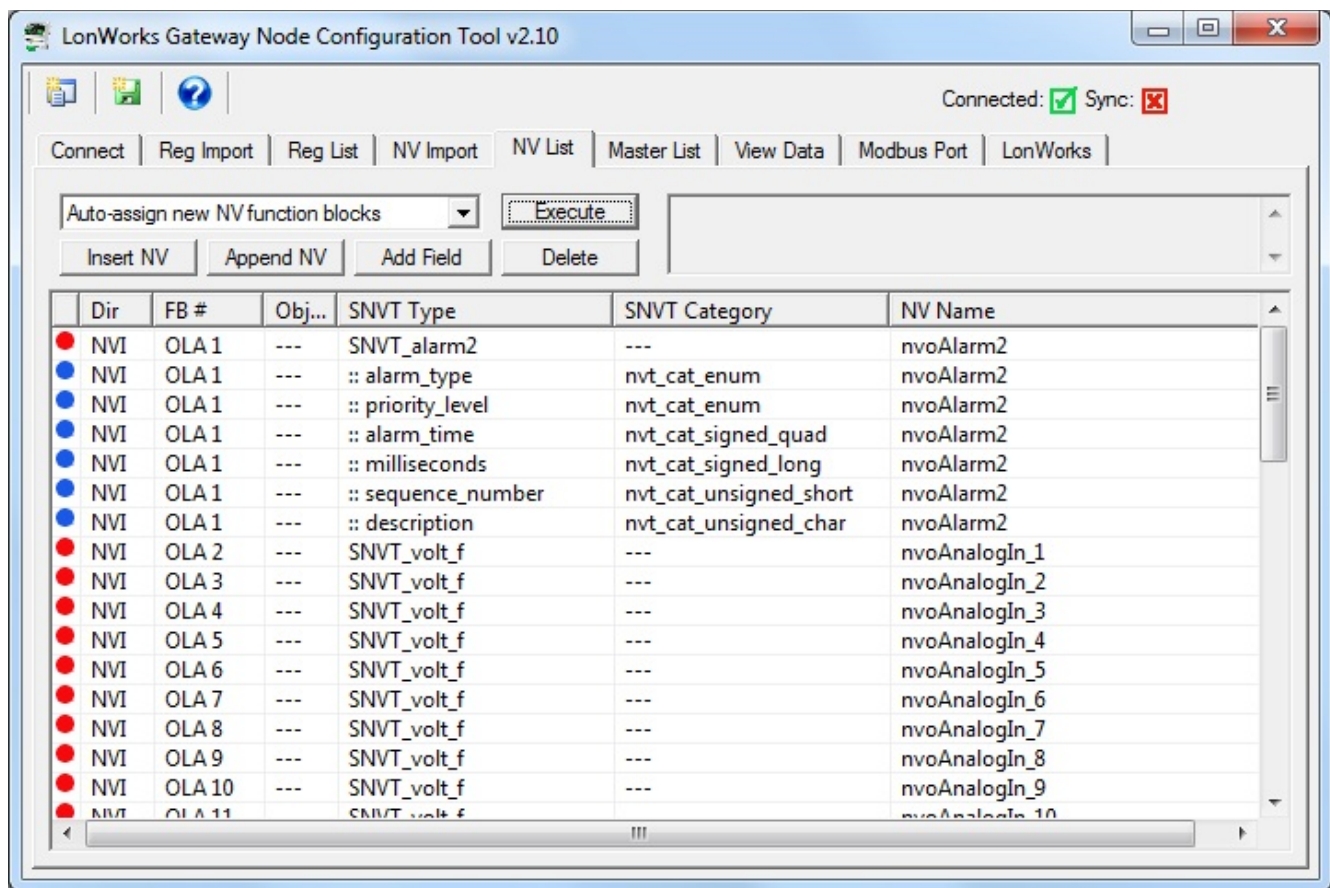
Your selection of whether to use the special conversion will decide whether SNVT_switch maps to one or two Modbus registers, and therefore whether SNVT_switch maps to one or two data objects. For this reason, it is important that you make all of your NV related selections and configurations BEFORE assigning function blocks and data objects.



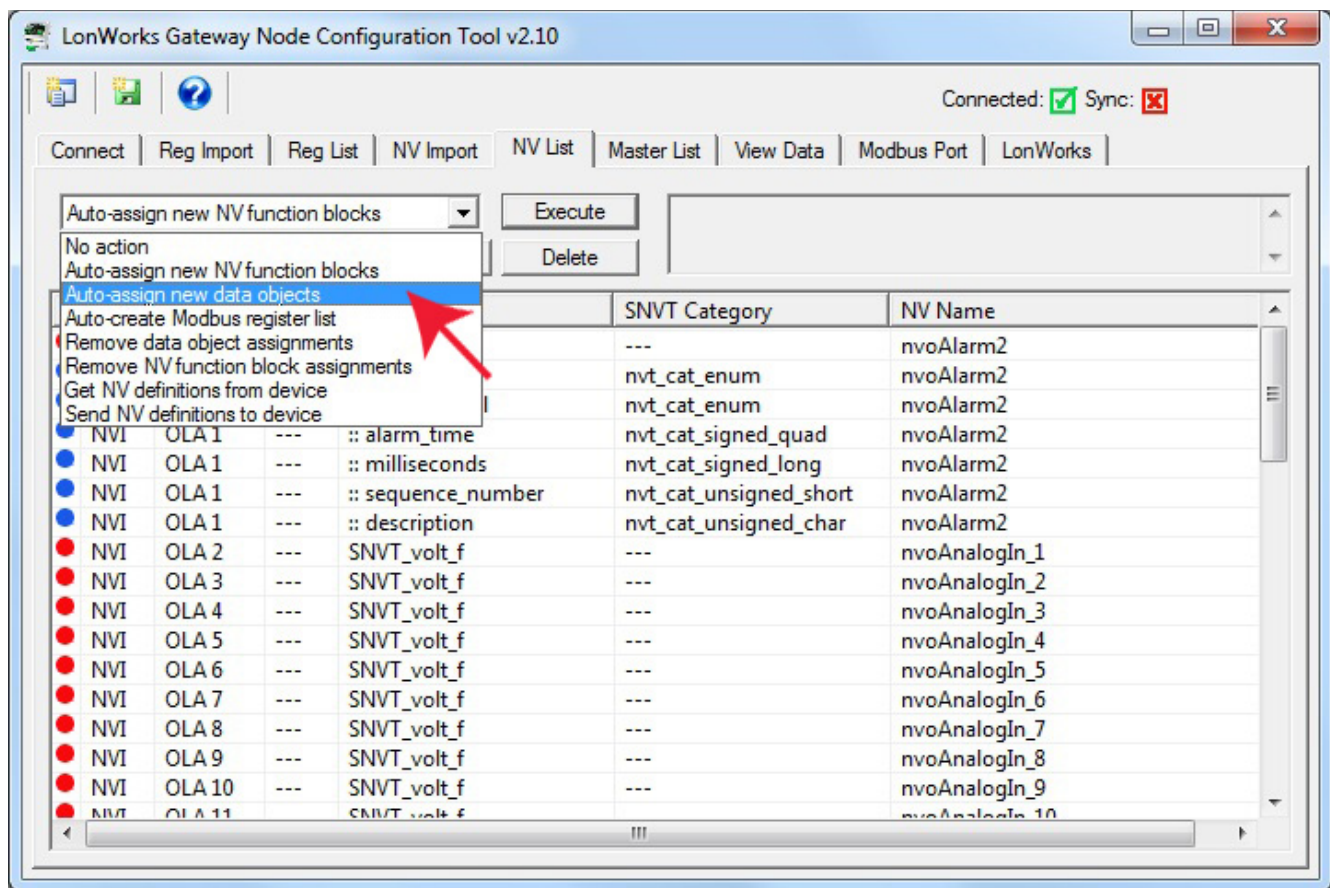
Once you have made all of the desired modifications to network variables in the NV List, proceed to assign function blocks. Select "Auto-assign new NV function blocks" and click Execute.



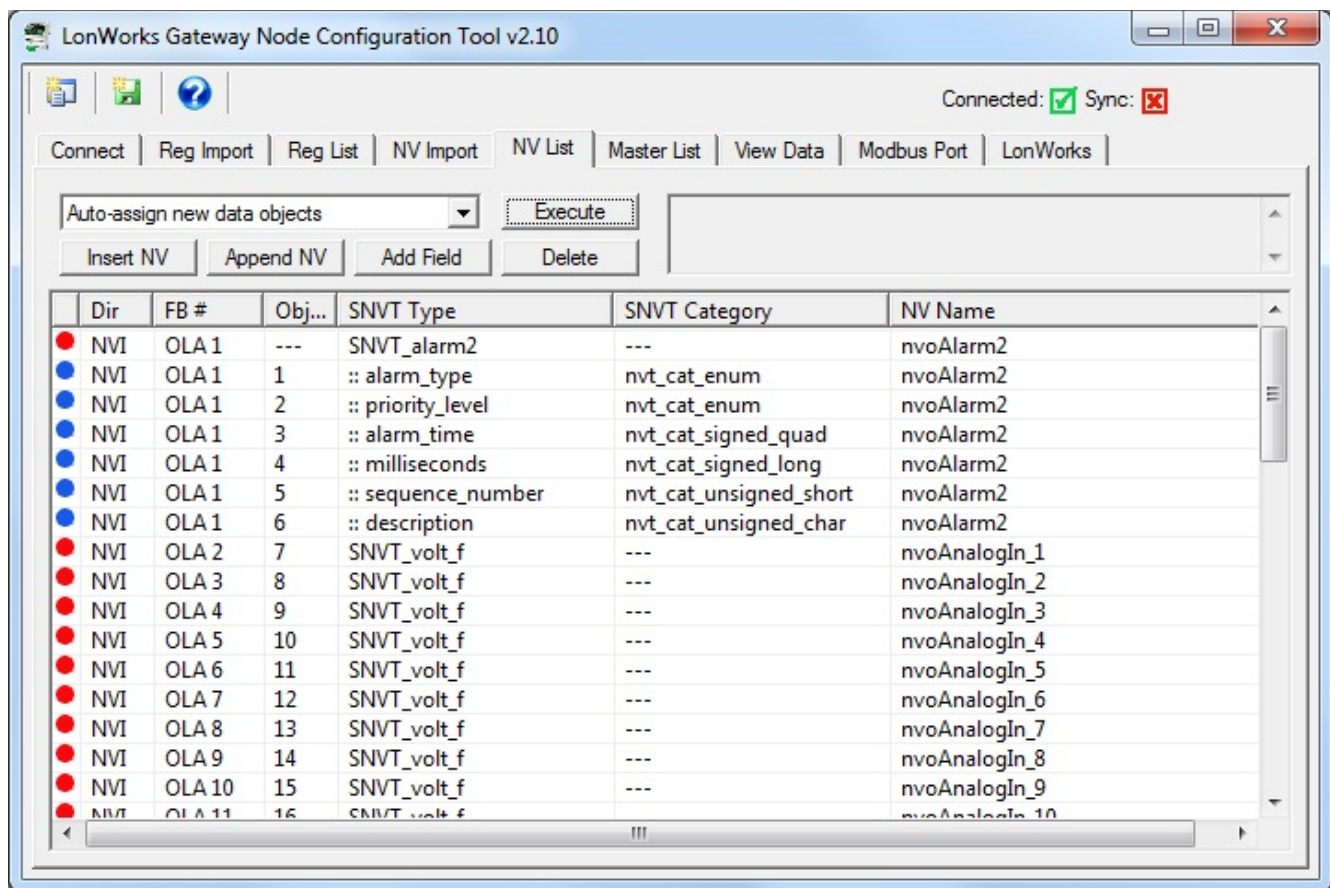
Upon execution of the function block assignments, the FB# column will be populated.



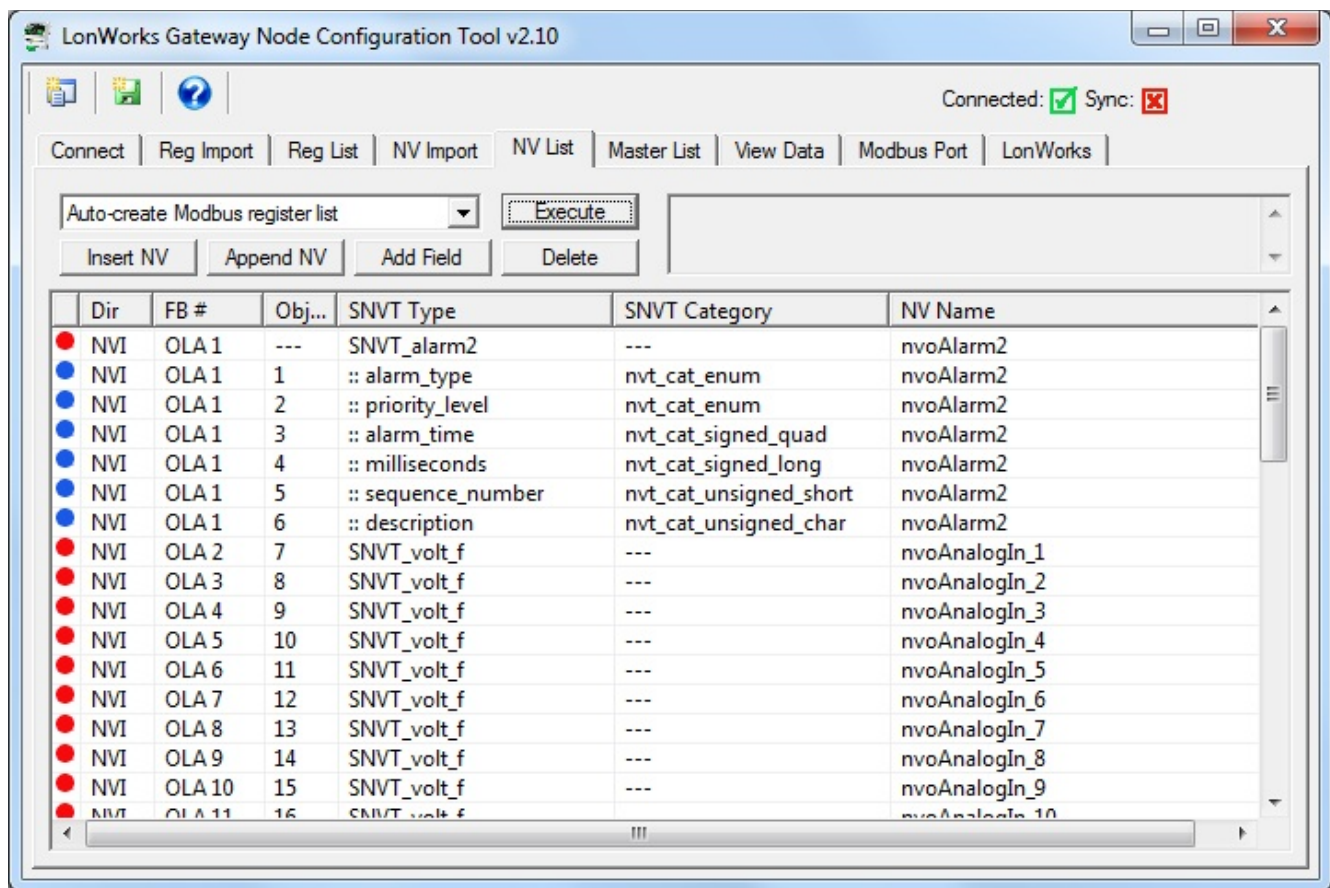
Next, select and execute "Auto-assign new data objects".



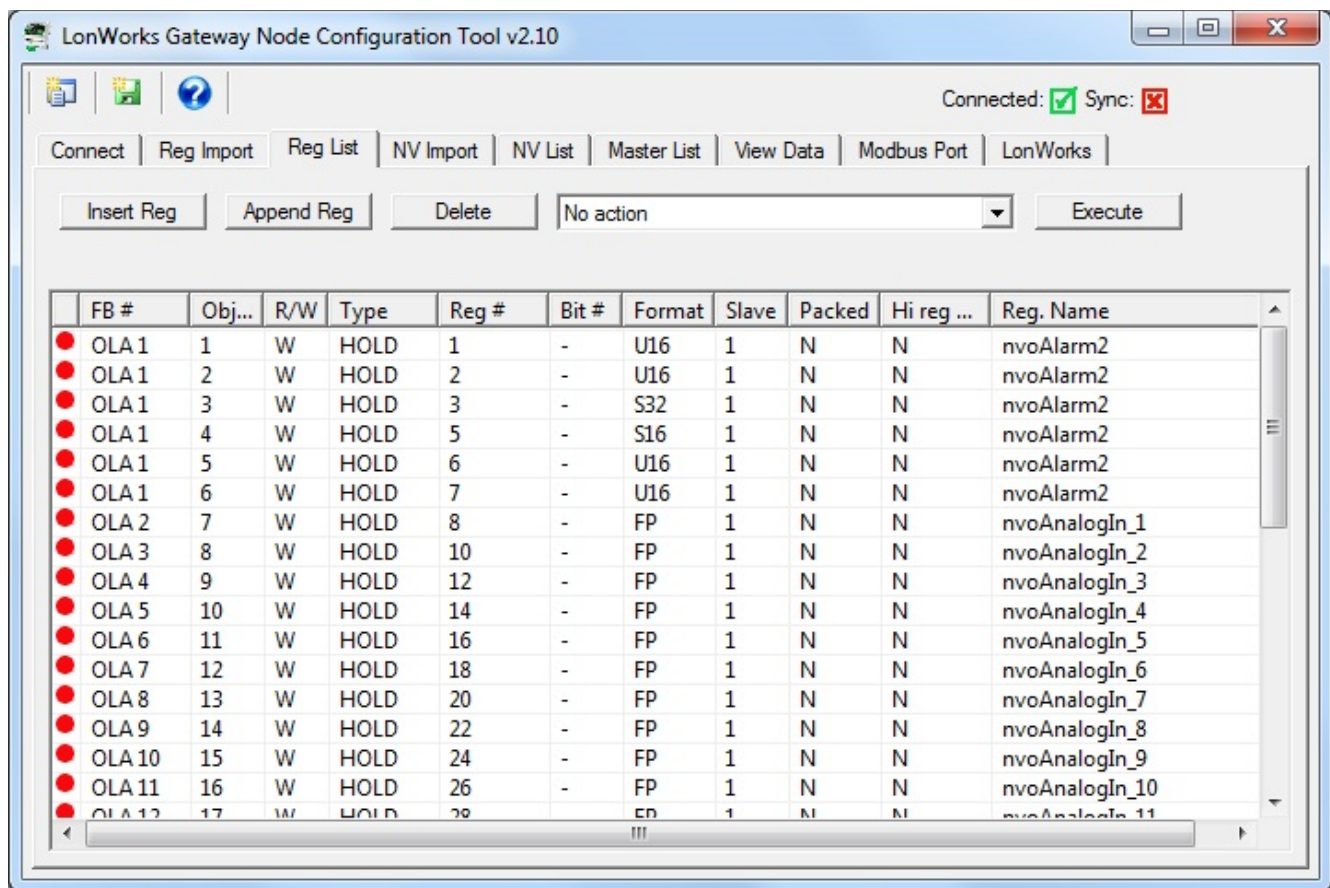
After data object assignment, the Obj... column will be populated.



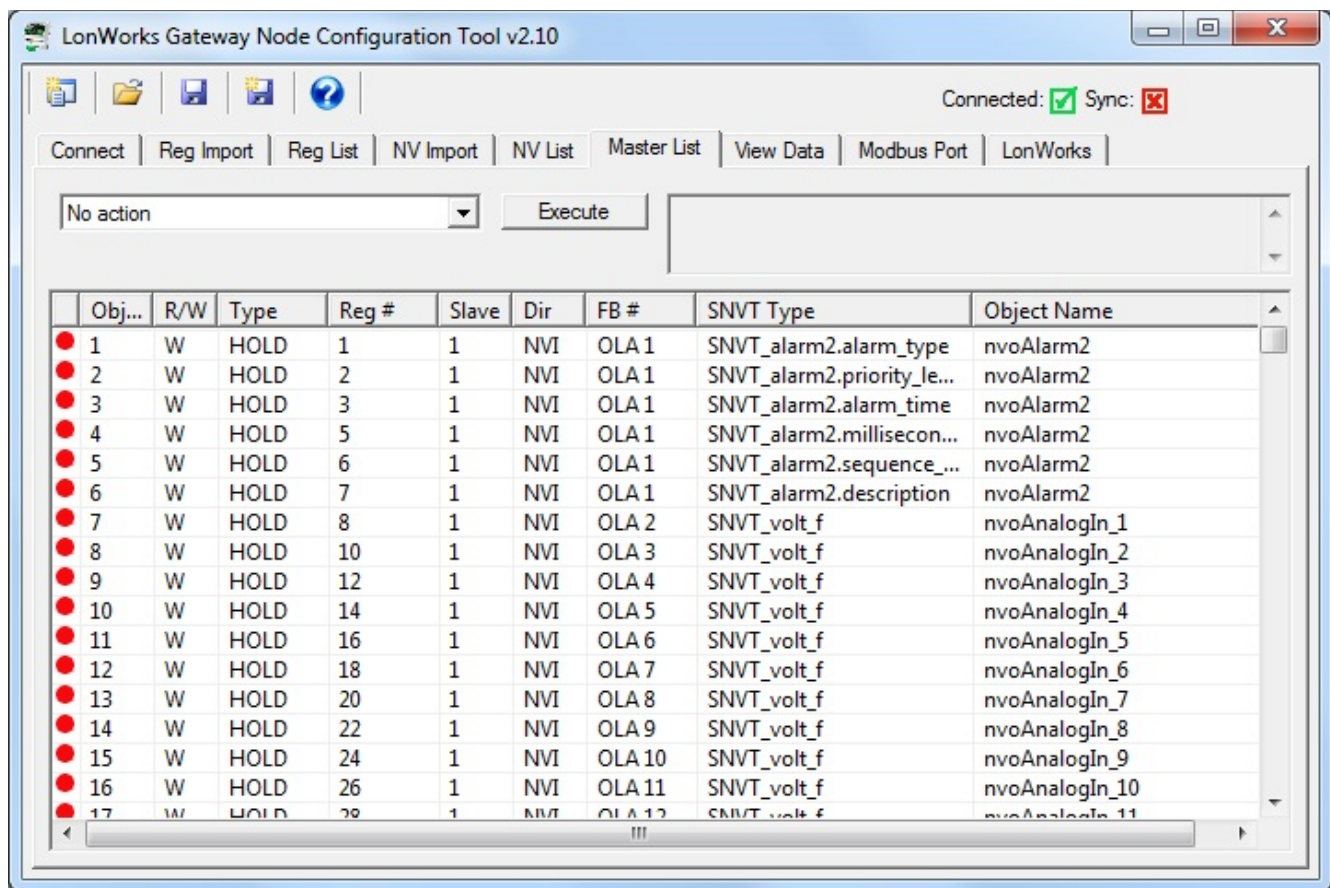
Finally, select and execute "Auto-create Modbus register list". Nothing new will appear on the NV List page, but registers will now appear on the Reg List page.



The Reg List page will become populated after execution of "Auto-create Modbus register list" from the NV List page.



The results of the configuration process are now available on the Master List page. From here, you would proceed to send NV definitions to the device, followed by data object definitions, as illustrated toward the end of the example given above. Once the Master List is complete, the process of sending configuration to the gateway device is the same regardless of approach used to build it.

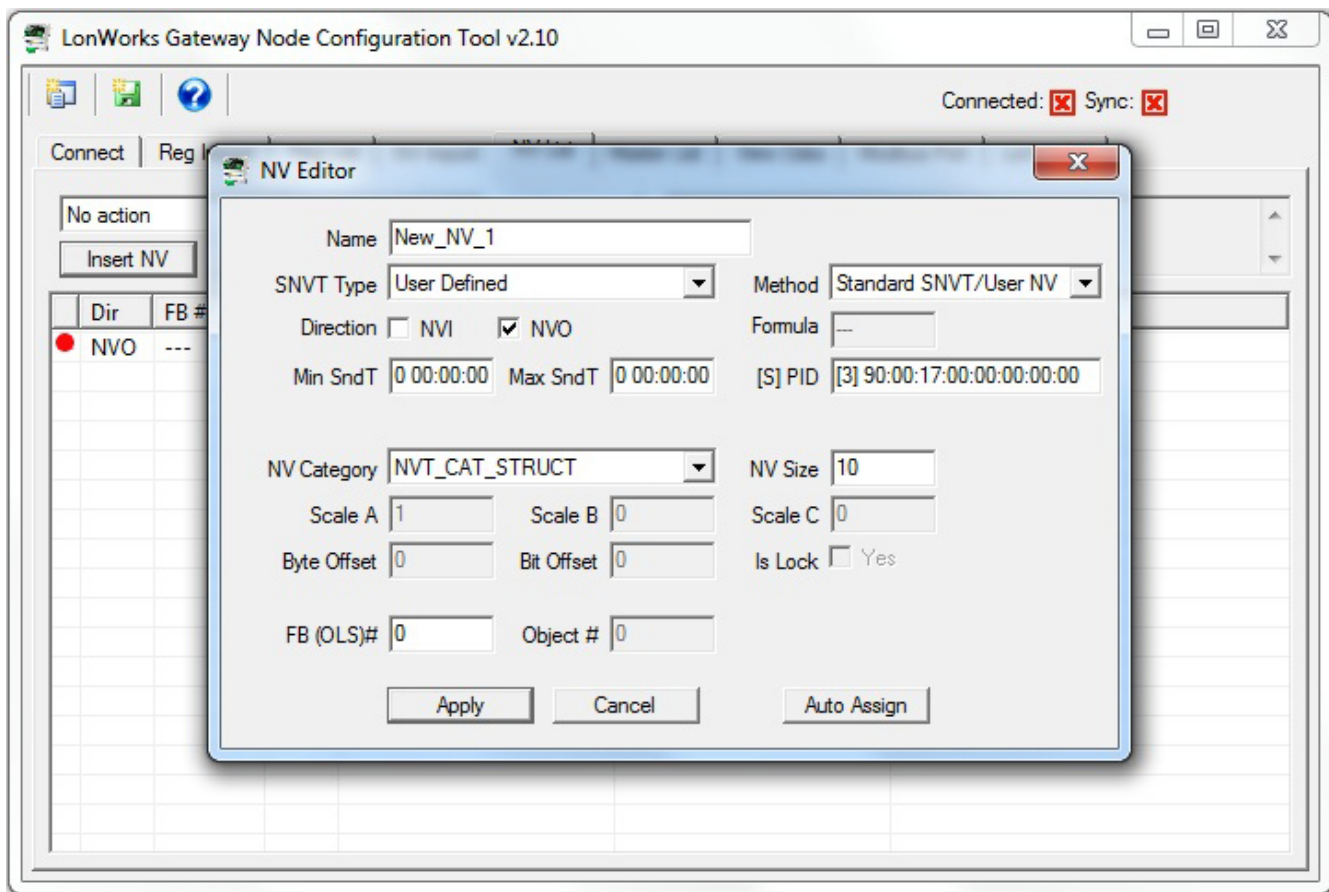


3.4 Build Configuration from Scratch

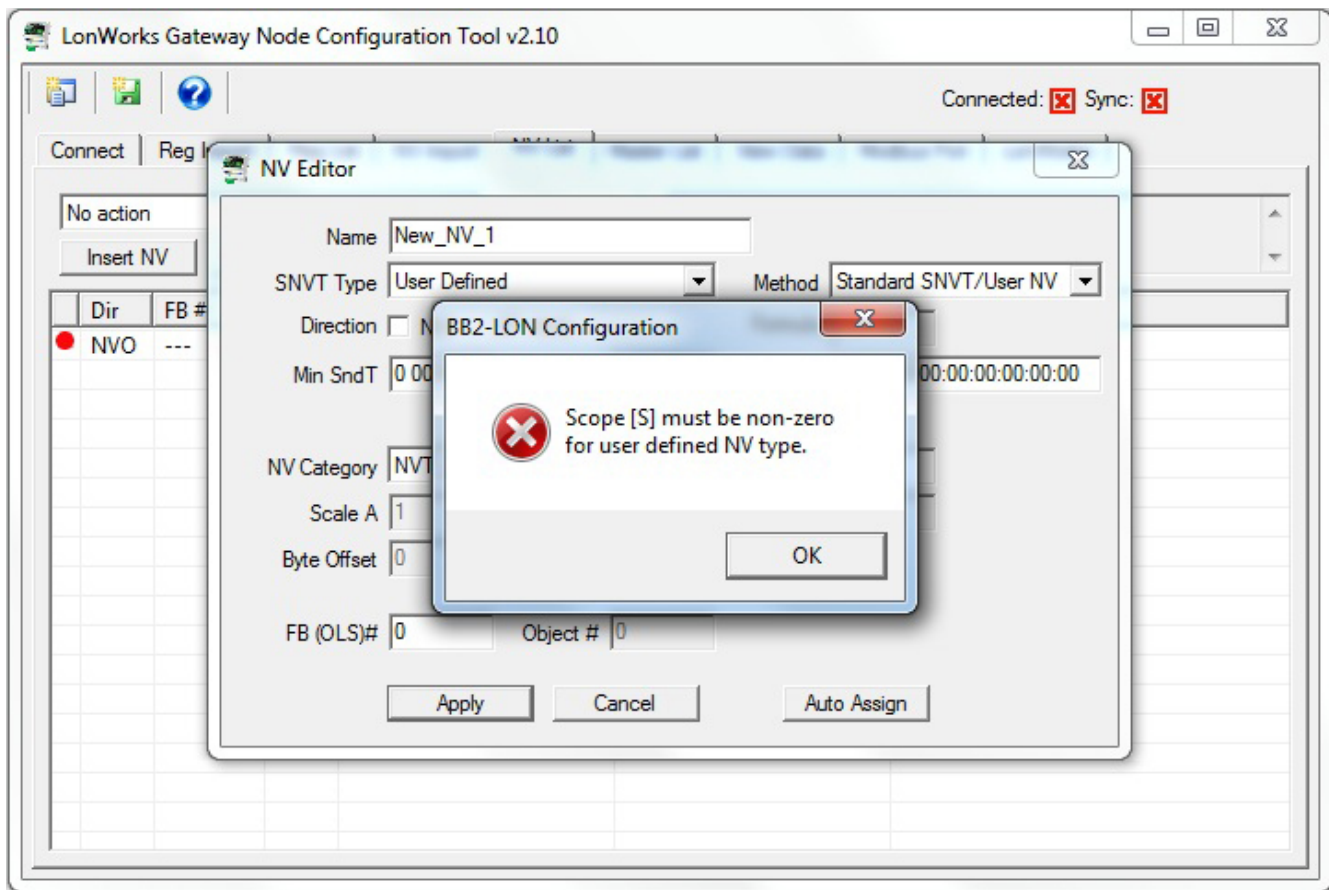
It is not required that you start with either a CSV file or an XIF file. You can use the configuration tool to start with a blank slate and build your configuration from scratch.

It is assumed that you have some familiarity with Modbus, and also an understanding of LonWorks network variables. You will need to obtain a copy of the documentation of SNVT types from LonMark (www.lonmark.org) in order to have any success in creating the LonWorks side of the gateway configuration.

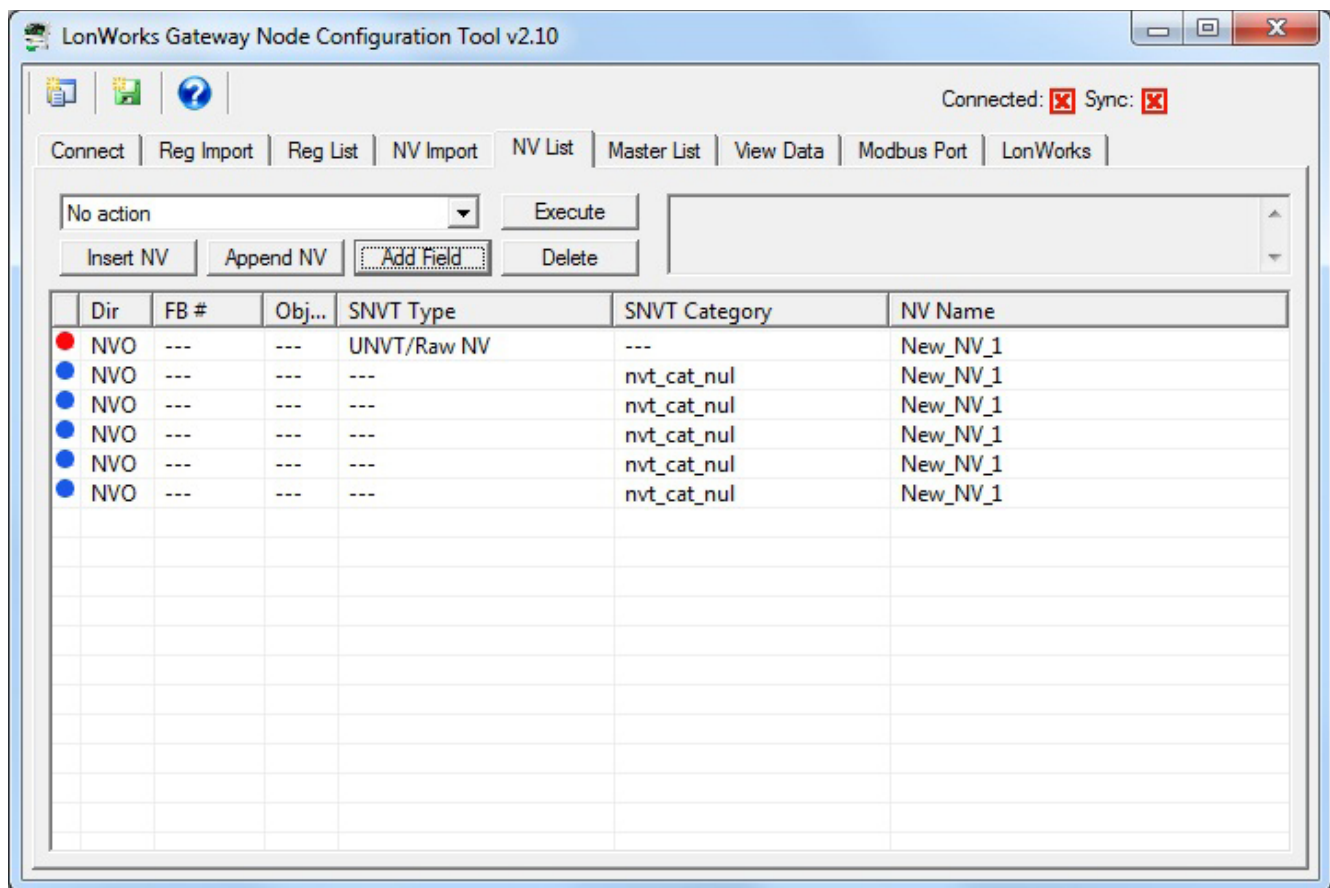
For each network variable you wish to add, click Insert NV.



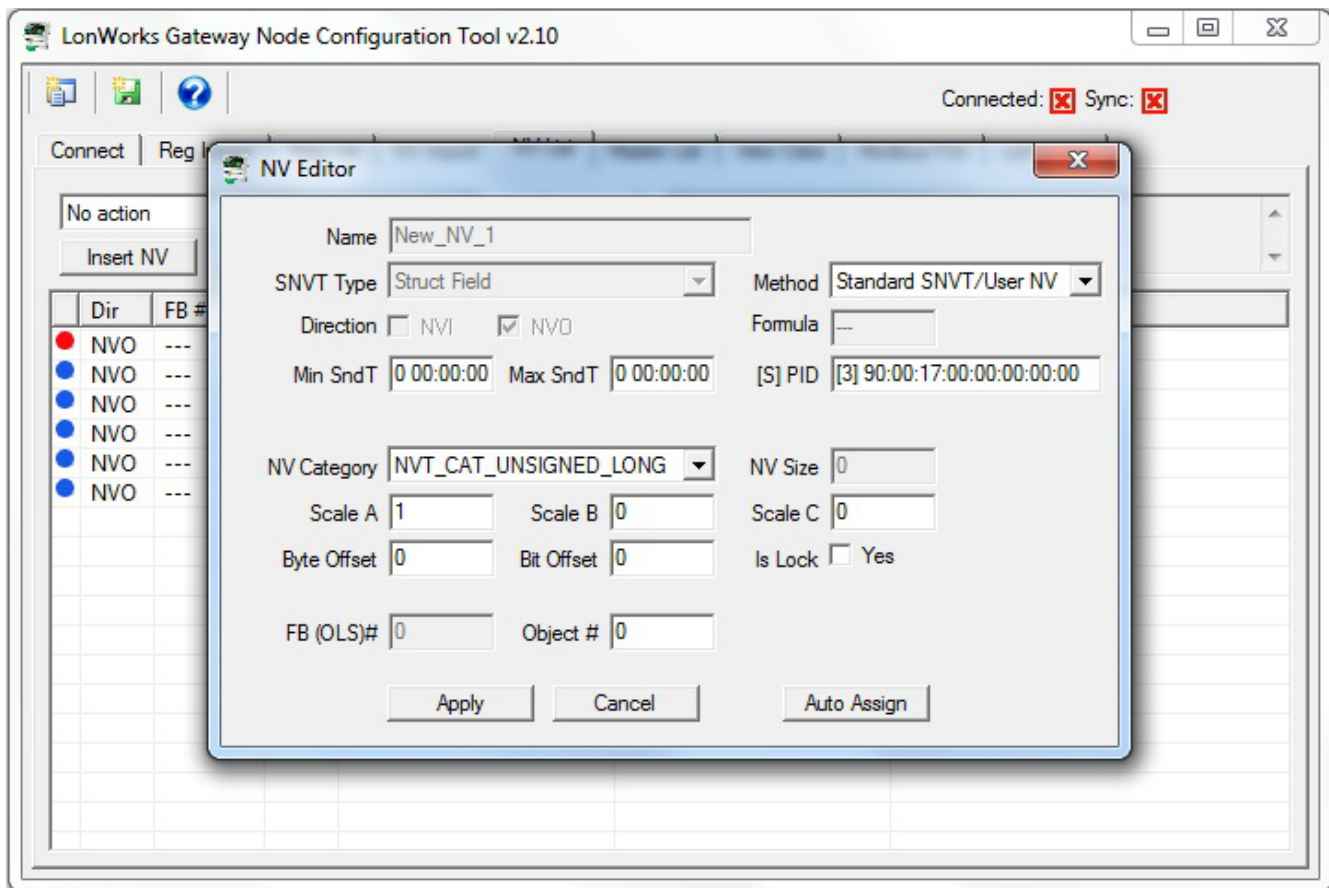
If you do not change the scope, most commonly to 3 for 'manufacturer defined', you will get an error message and not be allowed to proceed. If the scope remains at zero, then the gateway (and all other LonWorks devices on the network) will attempt to interpret your NV as a standard LonMark type and data conversion results will be wrong.



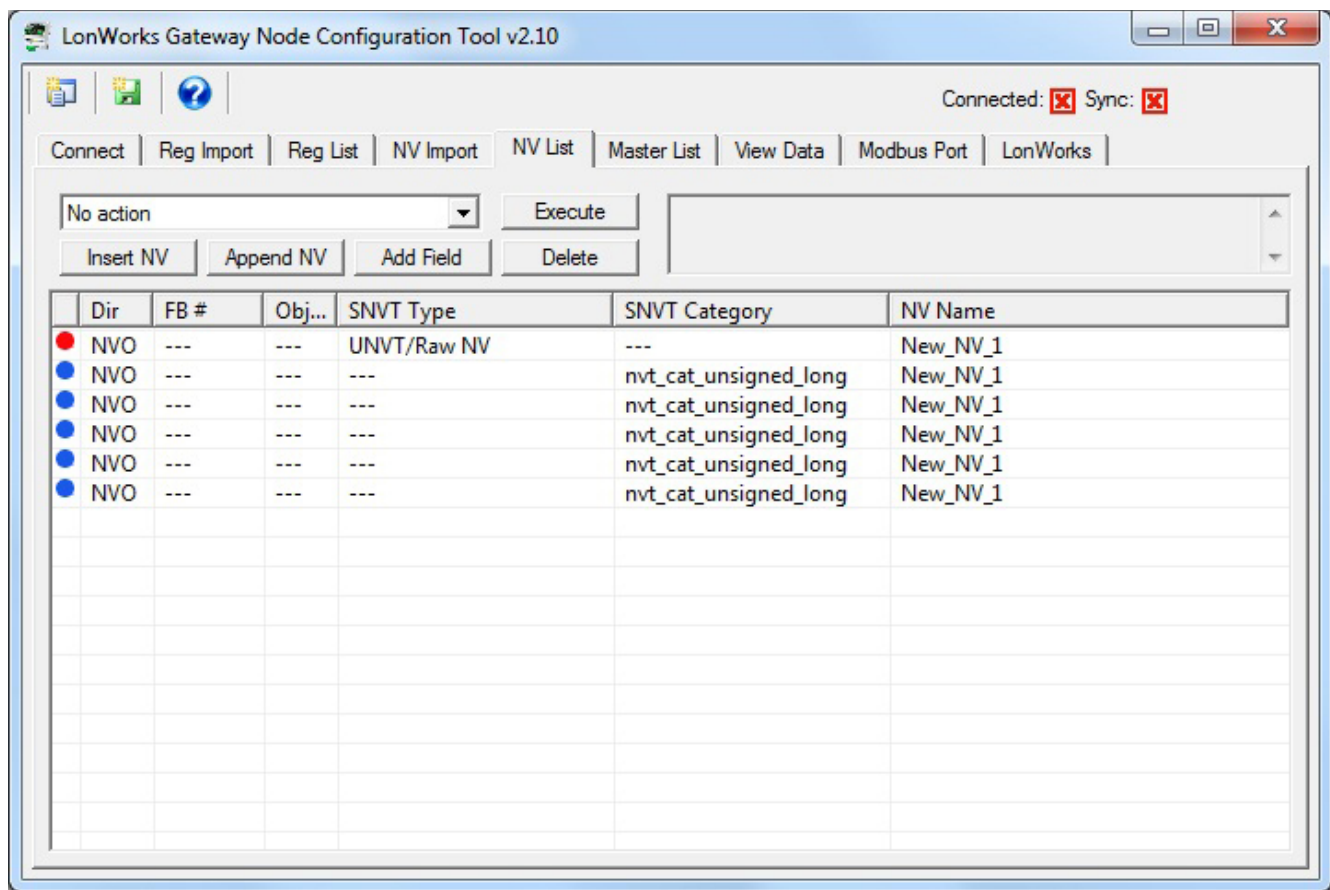
After defining the NV as a structure, you next need to add fields to the structure. Do this by clicking Add Field.



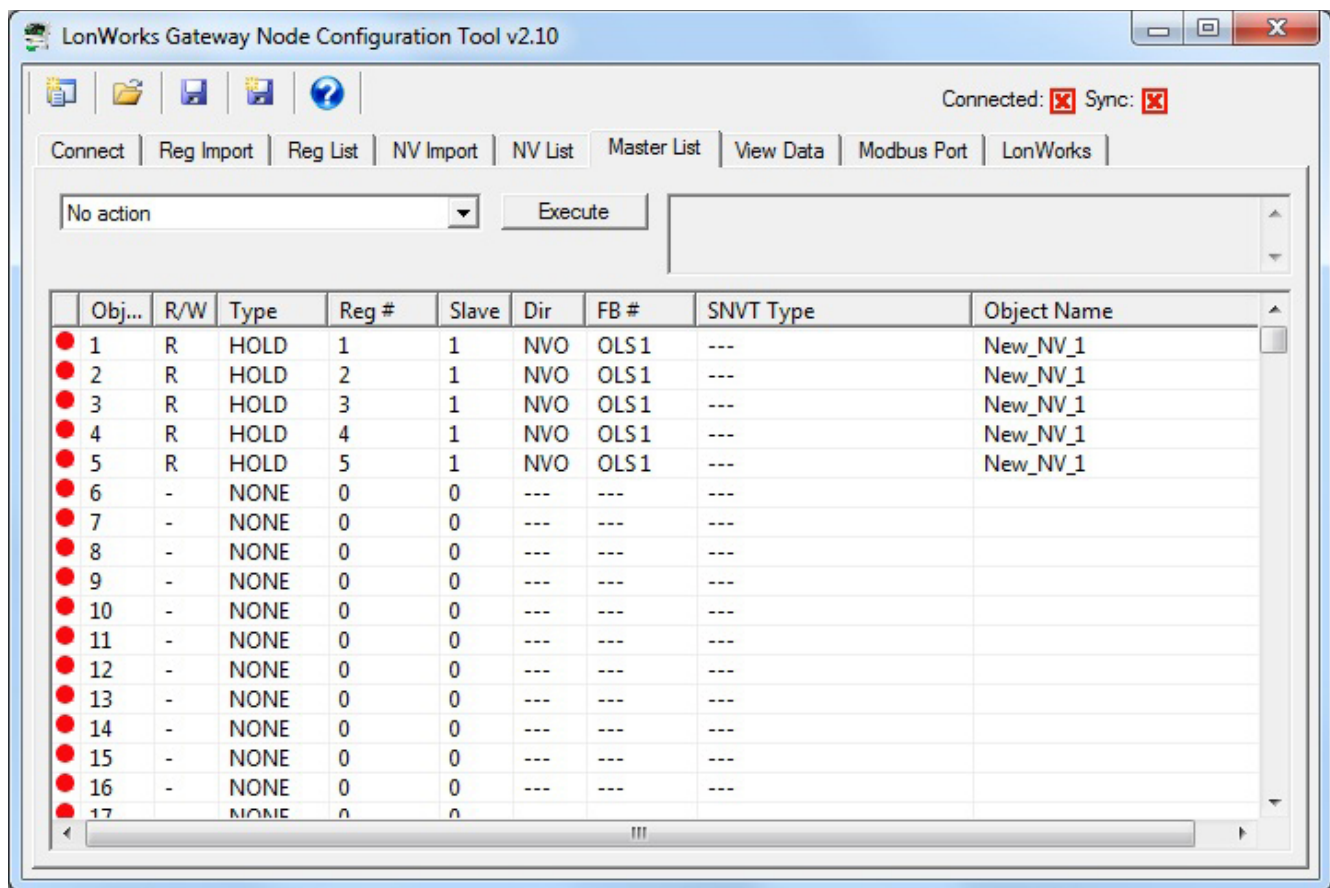
Next, edit each field by double clicking on that line. Specify the data type for the field. It is also important to specify the Byte Offset in the structure. An offset of zero means this field occupies the first byte(s) of the structure. You also need to enter the scale values. These follow the LonMark definition of scale.



Following configuration of the NV and its fields, your NV List will appear as follows for this first single NV that will map to five Modbus registers.



If you have no further network variables to add, proceed with the auto-build of the rest of the configuration. First execute "Auto-assign new NV function blocks". Then execute "Auto-assign new data objects". Finally, execute "Auto-create Modbus register list". Upon completion of these steps, the NV List would appear as follows for this example.



3.5 Commissioning Gateway as Node on Network

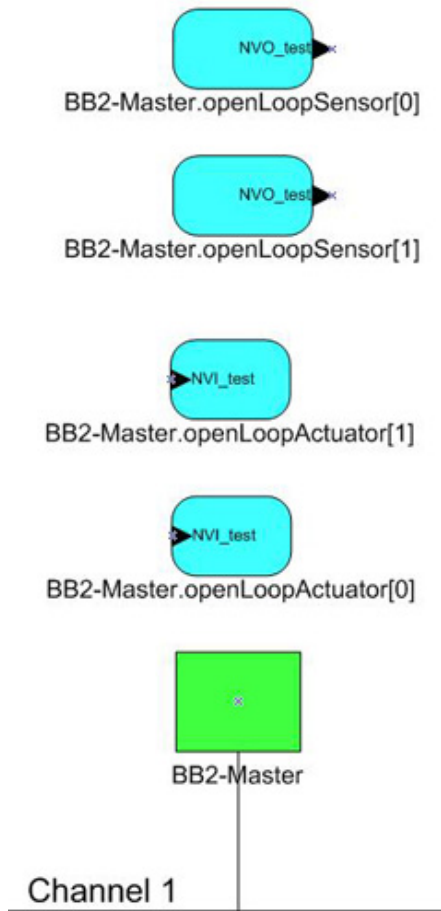
Even without any configuration, the Babel Buster LonWorks gateway will appear as a valid LonWorks node on the network. Without any configuration, all of the network variables default to a UNVTrawData type of variable. The only purpose of this variable type is to reserve the necessary memory space to provide storage for the largest NV type possible (31 bytes). Network variables that still have this default UNVT type will be regarded as 'unused' by the configuration tool. But they will still appear and be available to any network management tool on the LonWorks side.

Configuration will generally include changing the NV types to something more meaningful to your application. You will generally want to do as much pre-configuring of the gateway as possible before commissioning it on the LonWorks network. When you are ready to commission, follow the same procedure you would for commissioning any other LonWorks node using your favorite network management tool.

If you are commissioning the gateway using LonMaker, start by placing a device on the drawing and commissioning it following the standard practices. You can import the XIF from the device as well as import configuration properties from the device.

Once the device is on your drawing, start placing function blocks as needed to expose the various network variables. Then draw connectors to bind network

variables as applicable.



4 Tool 'Connect' Page

4.1 Connecting Configuration Tool to Gateway Device

The Babel Buster 2 LonWorks Gateway includes a USB port for configuration of the gateway. This eliminates any conflicts with the communications on either the Modbus or LonWorks ports and does not even require either network to be functional in order to configure the gateway. The only interface required is a USB cable. You do need to install the USB driver the first time you connect a Babel Buster via USB. After that, the gateway simply shows up as a COM port since USB is simulating a serial port for purposes of configuring the gateway.



Use your PC's device manager (found in the control panel) to locate the COM port that the gateway appears on once connected via USB. Select this COM port on the Connect page of the tool, and click Connect.

Most of the configuration will be identical for BB2-2010, BB2-2011, and BB2-6020. The only significant difference is that the TCP settings will show up when BB2-6020 is selected as Device Model.

If you are configuring a new gateway for the first time, select the check box indicating that. If you are updating a gateway already commissioned (installed on LonWorks network), then check that box. Doing so will prevent you from making changes that will cause your LNS database to become out of sync with the device.

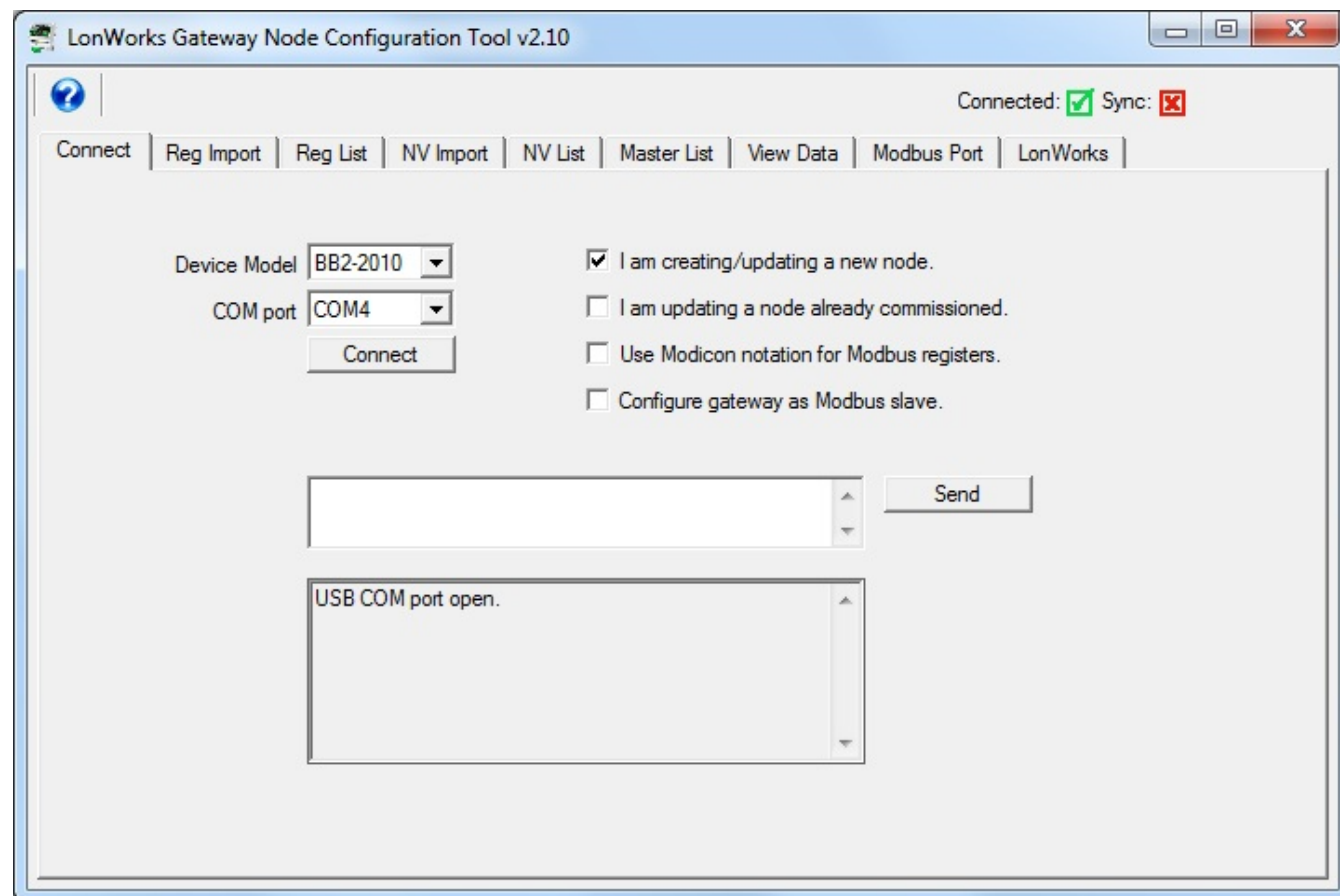
Check the 'Use Modicon' box if you prefer to see the 40001 style register numbers. Keep in mind that 40001 is not technically the register number - this is simply short hand for saying "holding register 1".

Check the 'configure as slave' box if you will be setting up the Babel Buster gateway as being the Modbus slave device, expecting an external Modbus master to query it. Some of the register related configuration changes based on whether the gateway will be Modbus master or slave.

IMPORTANT: If you have connected USB to the gateway, and then power cycle the gateway, the USB

connection will be lost. You will need to unplug the USB cable from the gateway and reconnect, then reconnect the configuration tool. It may also sometimes be necessary to close and restart the configuration tool software to re-establish a lost USB connection.

IMPORTANT: If you select a COM port and 'connect' but do not get any response in the configuration tool, it is possible you opened a COM port that is a valid port but is not the USB port for the gateway. When this happens, the tool will get hung up waiting for a response. At the same time, Windows gets confused and doesn't let the configuration tool time out either. If you are stuck in this state, close the configuration tool, forcefully via the system task manager if necessary, and restart - including disconnecting and reconnecting the USB cable.



The "Connected" box will turn green if the COM port is simply able to be opened. It does not actually mean communications are successful. To test that, type the command "cver" in the command window, and click Send. This will send a request for firmware version to the device. If successful, you will see something comparable to the example below.

The command window is also used for a number of other diagnostic commands. Refer to the section on "Diagnostics via the USB Console" for more details. Normally, these diagnostic commands will not be required, but can be helpful in some instances of trouble shooting.

LonWorks Gateway Node Configuration Tool v2.10

Connected: ☒ Sync: ☒

Connect | Reg Import | Reg List | NV Import | NV List | Master List | View Data | Modbus Port | LonWorks

Device Model: BB2-2010
COM port: COM4
Connect

☒ I am creating/updating a new node.
☐ I am updating a node already commissioned.
☐ Use Modicon notation for Modbus registers.
☐ Configure gateway as Modbus slave.

cver

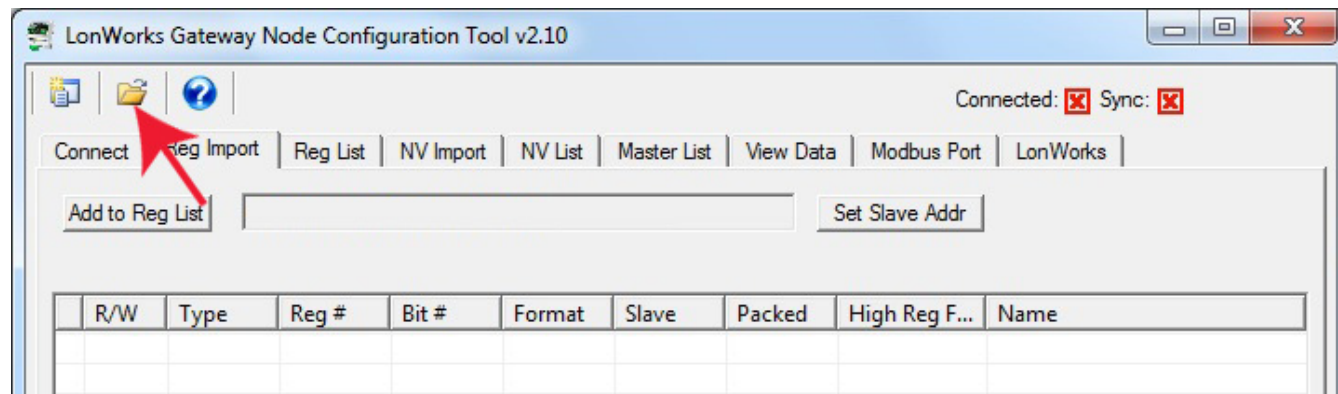
Send

BB2-2010 v3.10.3
:->

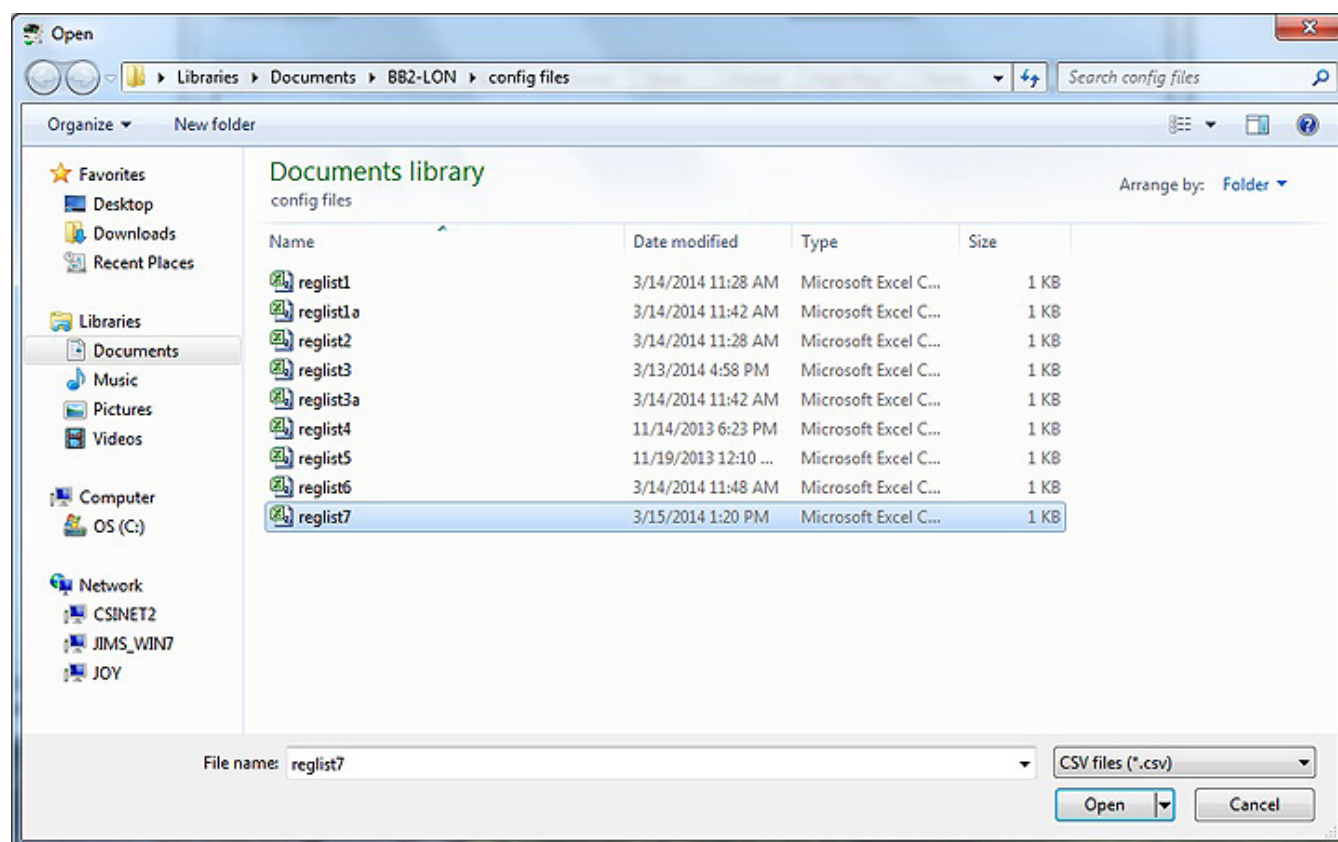
5 Tool 'Reg Import' Page

5.1 Importing a CSV Register List

To import a Modbus register list from a CSV file, start by going to the Reg Import page, then click on the file open icon.



You will see the familiar file open dialog. Select your CSV file.



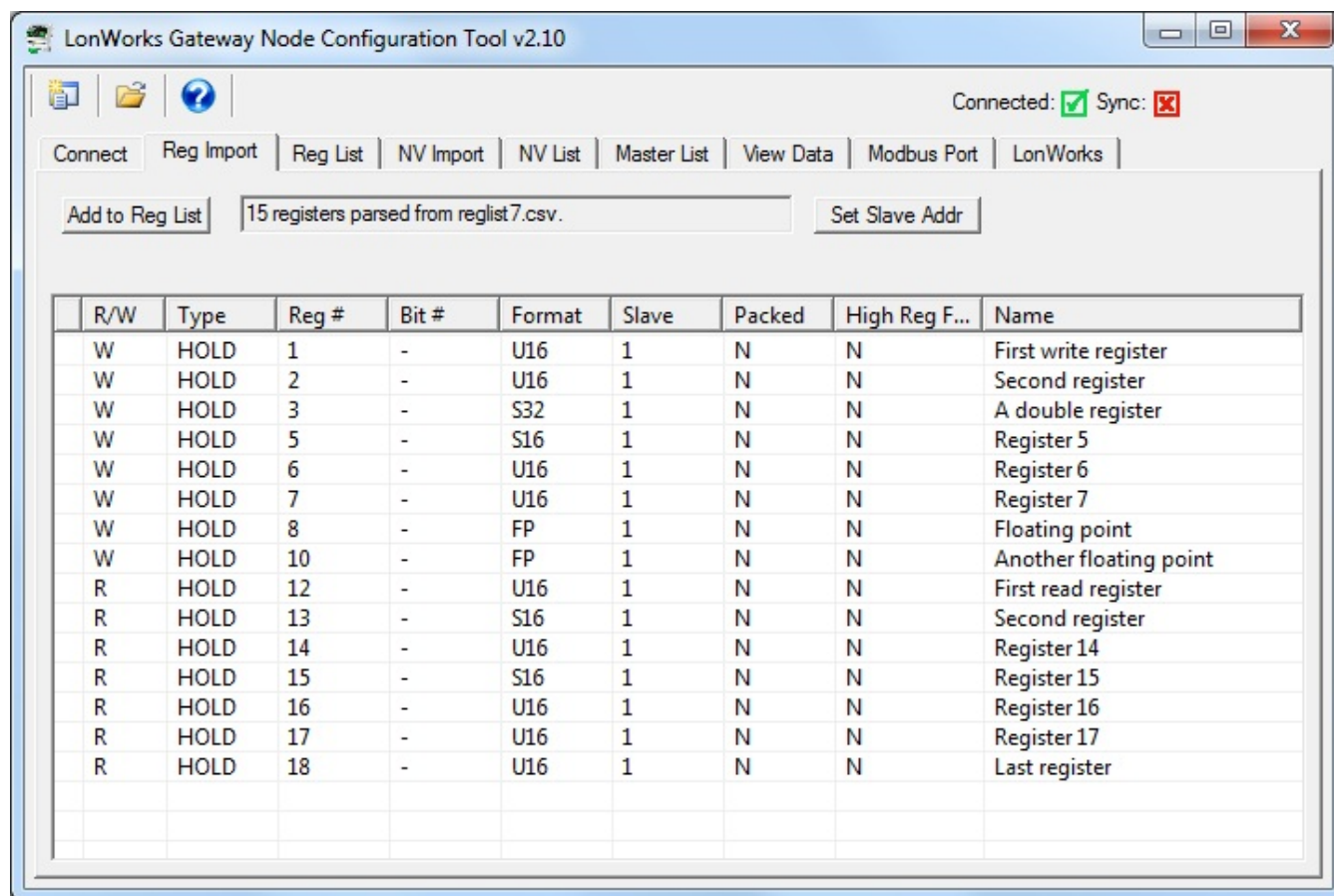
The format required for the CSV file is described in Appendix D of this user guide. For reference, the content of the reglist7.csv file loaded here contains the following CSV list, and upon loading, will appear as shown in the next screen shot.

```
RW,MODICON,FORMAT,SLAVE,BITNUMBER,PACKED,HIGHREGFIRST,NAME
```

```

W,40001,U16,1,,F,F,First write register
W,40002,U16,1,,F,F,Second register
W,40003,S32,1,,F,F,A double register
W,40005,S16,1,,F,F,Register 5
W,40006,U16,1,,F,F,Register 6
W,40007,U16,1,,F,F,Register 7
W,40008,FP,1,,F,F,Floating point
W,40010,FP,1,,F,F,Another floating point
R,40012,U16,1,,F,F,First read register
R,40013,S16,1,,F,F,Second register
R,40014,U16,1,,F,F,Register 14
R,40015,S16,1,,F,F,Register 15
R,40016,U16,1,,F,F,Register 16
R,40017,U16,1,,F,F,Register 17
R,40018,U16,1,,F,F,Last register

```

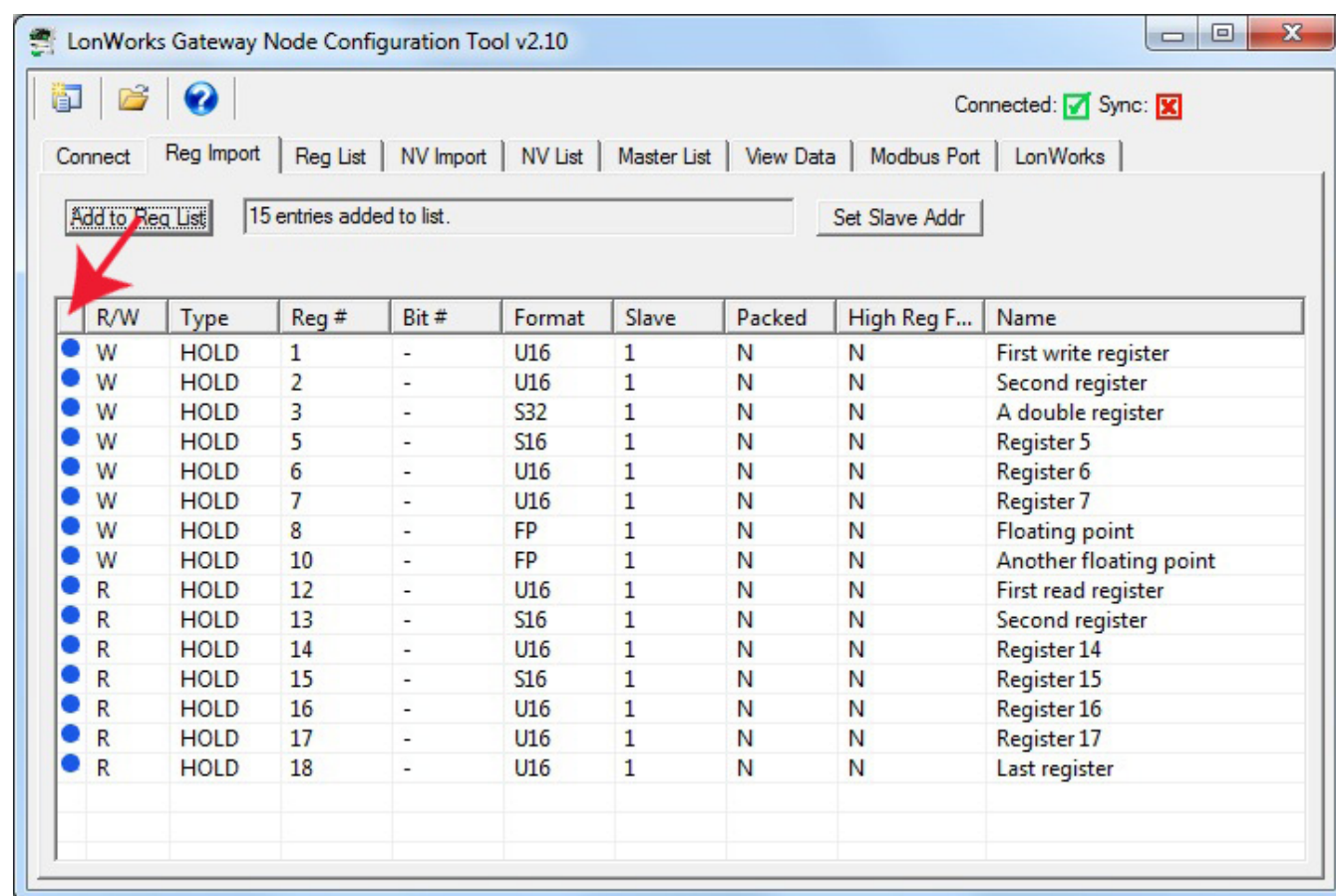


At this point, the register list is contained in what amounts to a "scratch pad". The CSV file is imported into an interim list so that you can choose which registers you want to include in your configuration. If you created the CSV file, you most likely want all of them. But if somebody else sent you a CSV file containing every register in their Modbus device, you will usually want to select only those that are pertinent to your application.

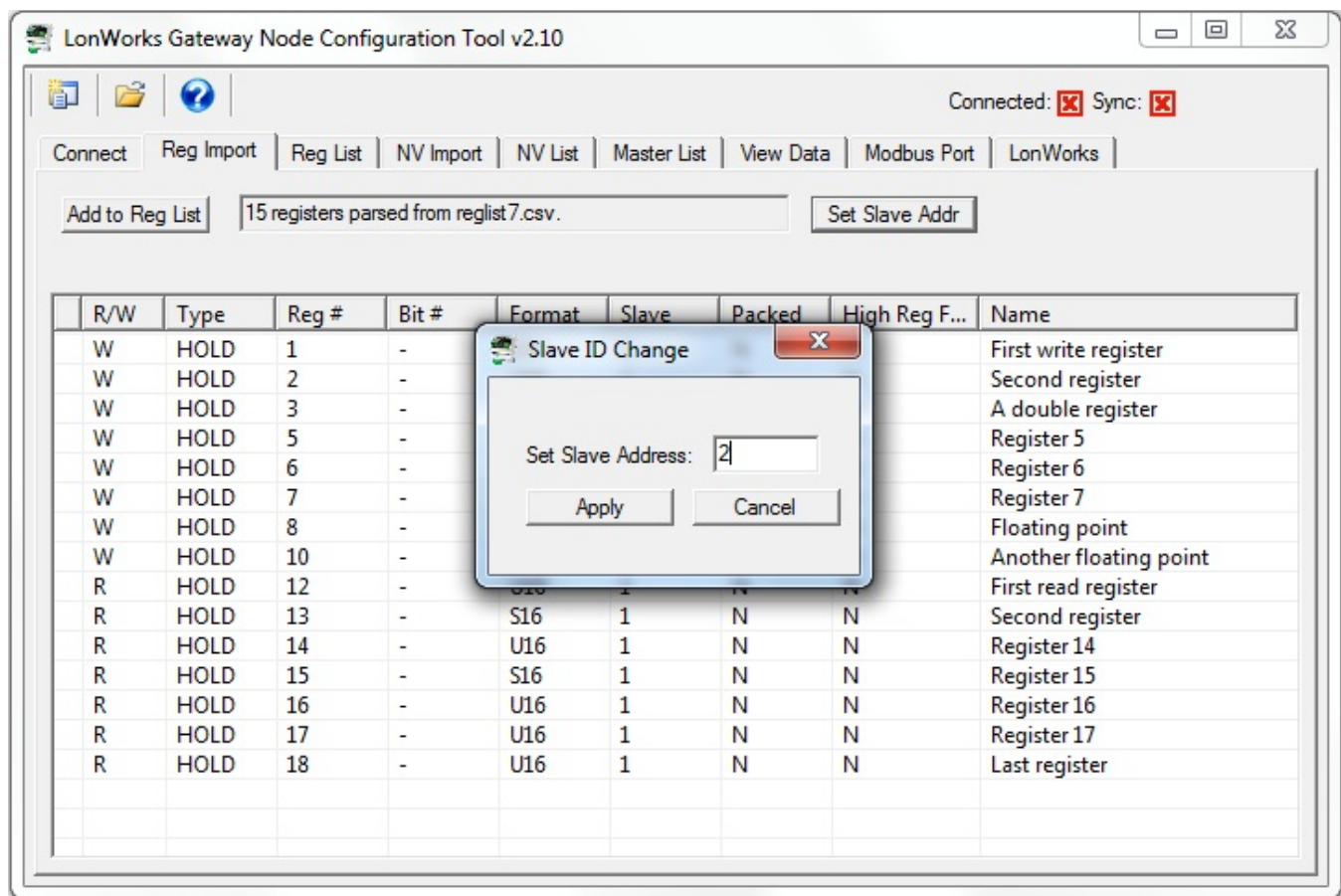
To select all registers, click the icon column header as illustrated by the red arrow below. This will cause the blue dot to appear on each line, which indicates that this register has been selected to be included in the configuration. To select only certain registers, click the icon area of only those rows you wish to include.

Once you have selected the registers, click the "Add to Reg List" button. Until you click this button, you

have no registers in the configured register list. You may, at this point, import another CSV and continue to add multiple registers from multiple CSV files.

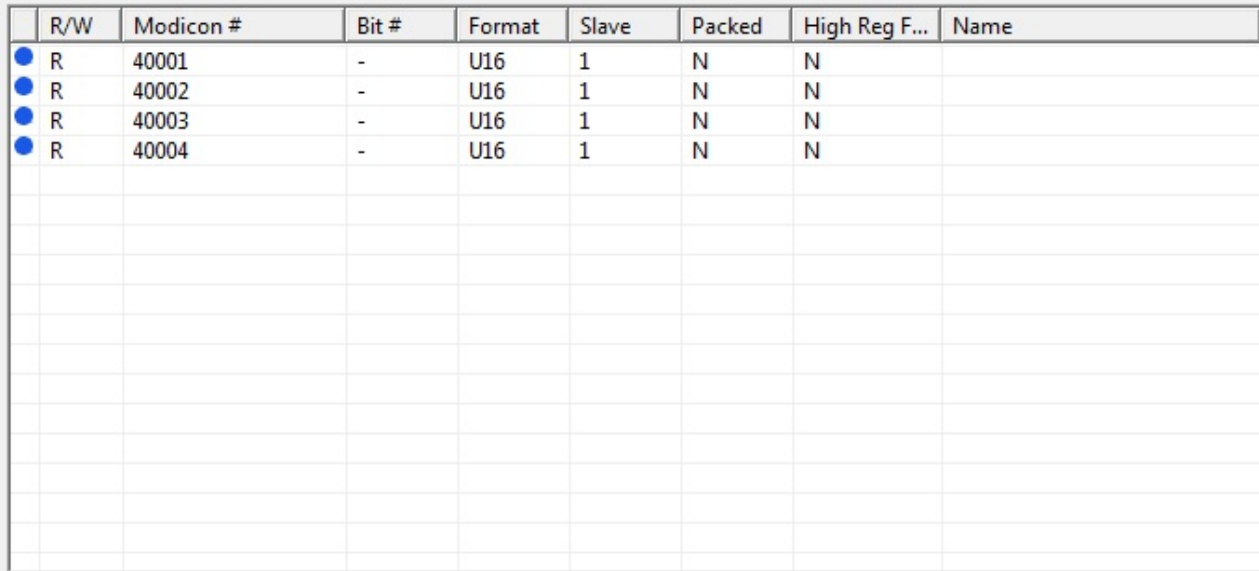


If you will be connecting two or more of the same type of Modbus device, each having identical register sets, click "Add to Reg List". Then use "Set Slave Addr" to select the next device's slave address. Now click "Add to Reg List" again. This adds the same set of registers a second time, but with a different slave address the second time.



The use of the file open icon on the toolbar at the top of the screen is mentioned above. The first icon on the toolbar is the 'new' button. This completely clears the register import list. The other icon is the help button. Click that button any time you want to open a copy of this page.

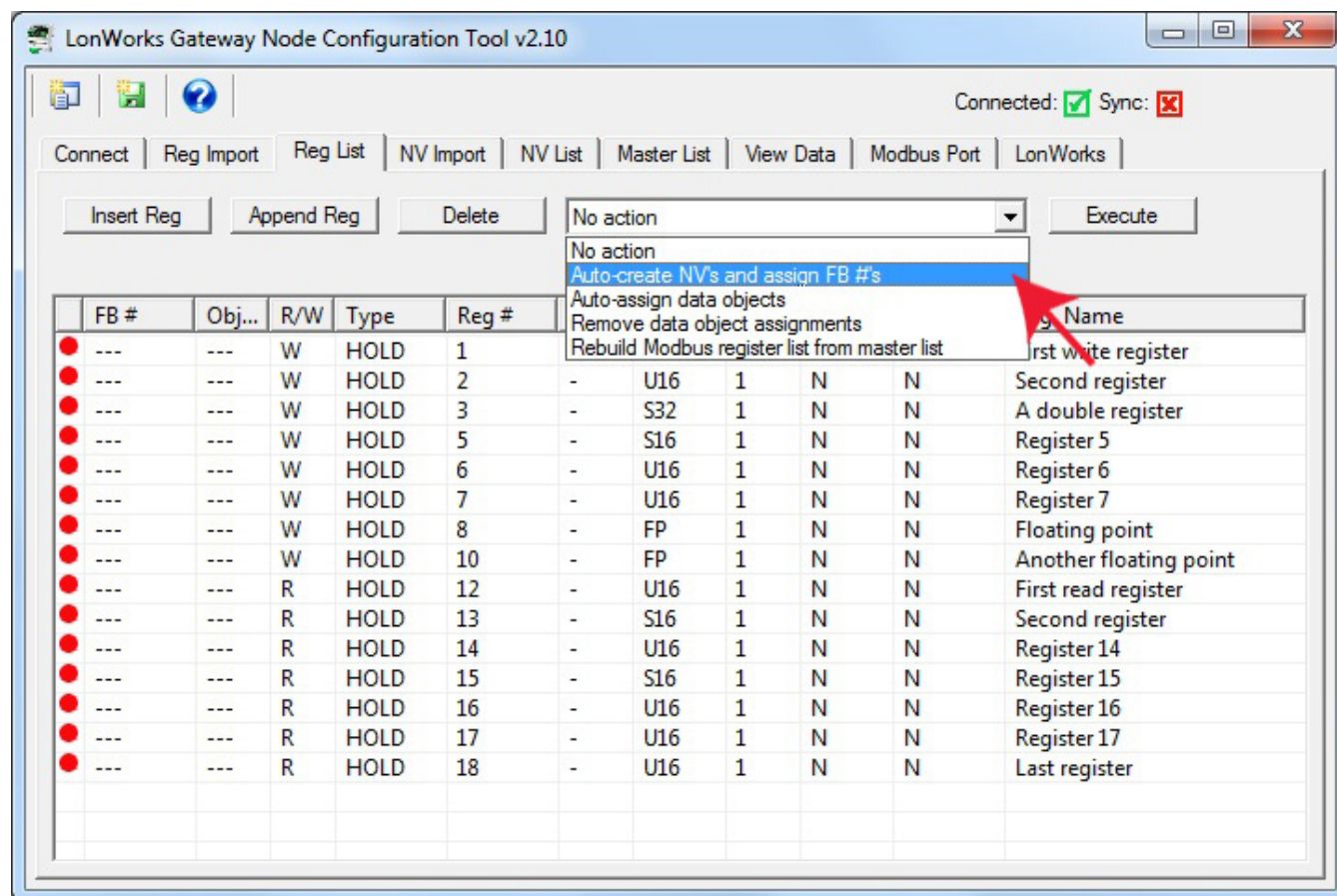
When Modicon display of register numbers has been selected on the Connect page, the Type and Reg # columns are replaced with a single Modicon # column. All other aspects of the Reg Import page remain the same regardless of how register numbers are displayed.



6 Tool 'Reg List' Page

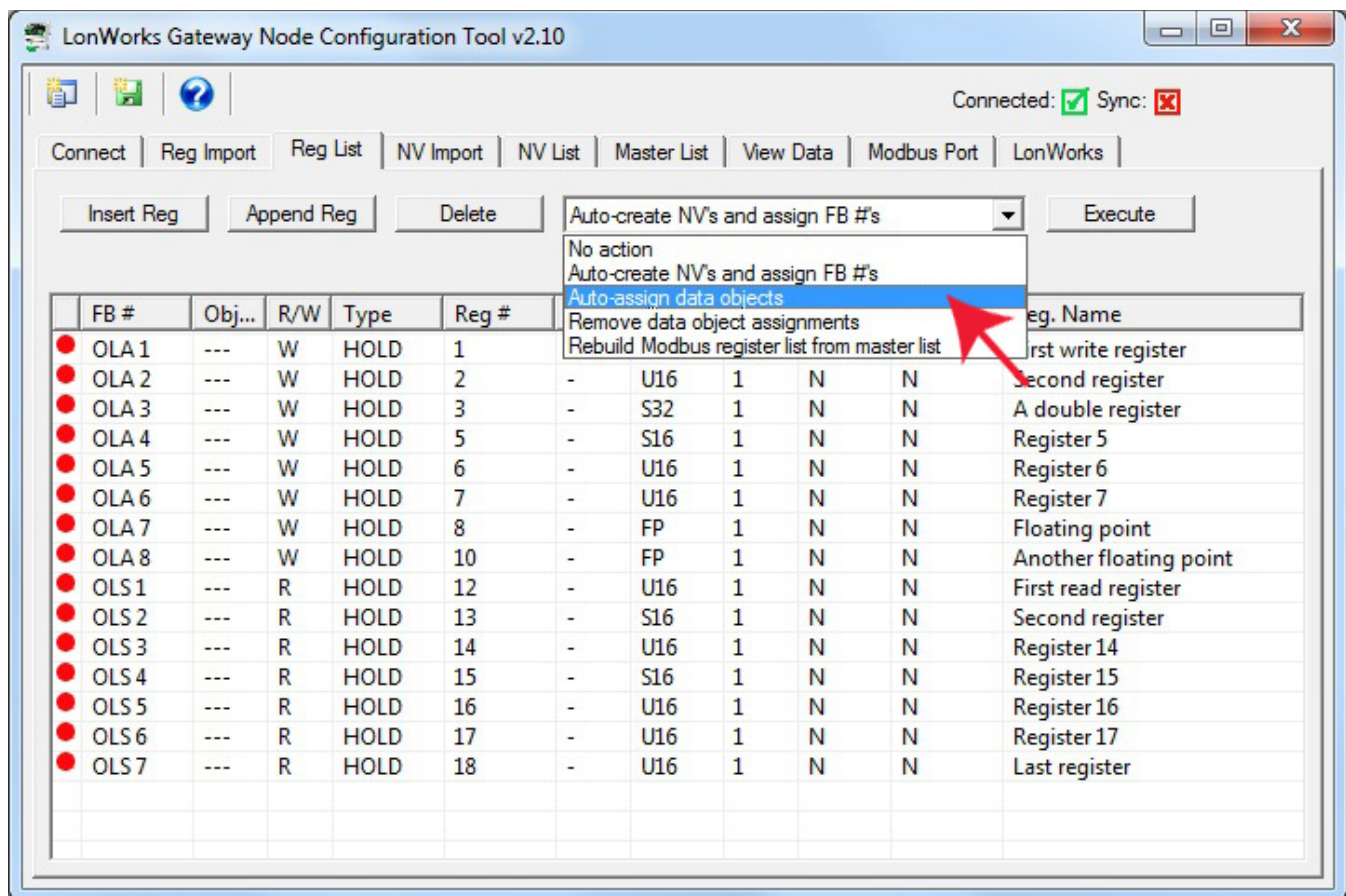
6.1 Auto-Building the Configuration

Once you have imported your register list(s) on the Reg Import page, go to the Reg List page. Here is where you begin the auto-build of the rest of your configuration. Start by selecting "Auto-create NV's and assign FB #'s". Then click the Execute button.

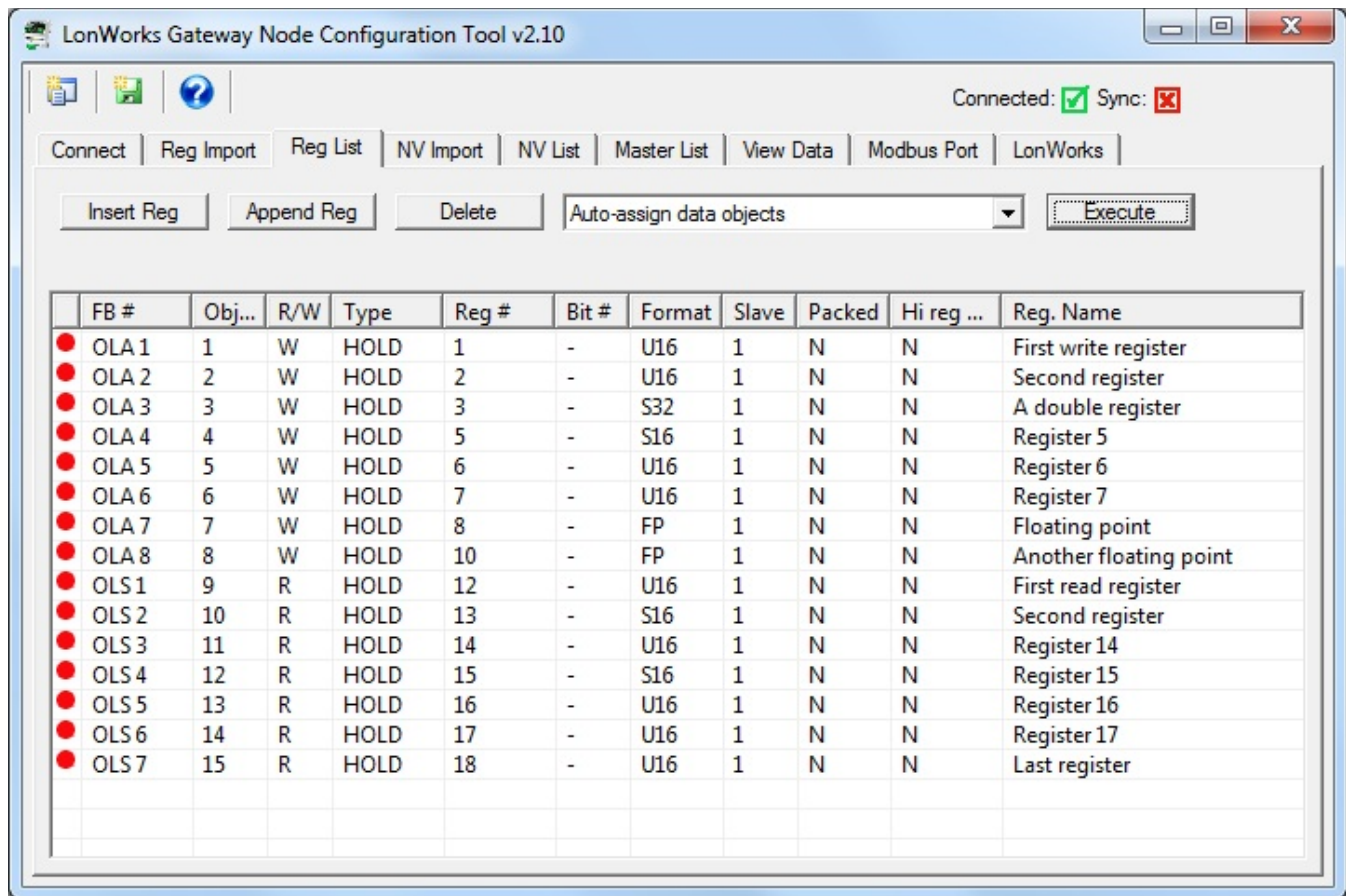


Upon executing the Auto-create NV's, the FB# column will be populated. Modbus registers designated as "W" for write, meaning the gateway will write to these registers in the Modbus device (assuming gateway is master), will be assigned to an Open Loop Actuator function block having a Network Variable Input (NVI). Modbus registers designated as "R" for read, meaning the gateway will read these registers from the Modbus device, will be assigned to an Open Loop Sensor function block having a Network Variable Output (NVO).

Next, select "Auto-assign data objects" from the list and click Execute.



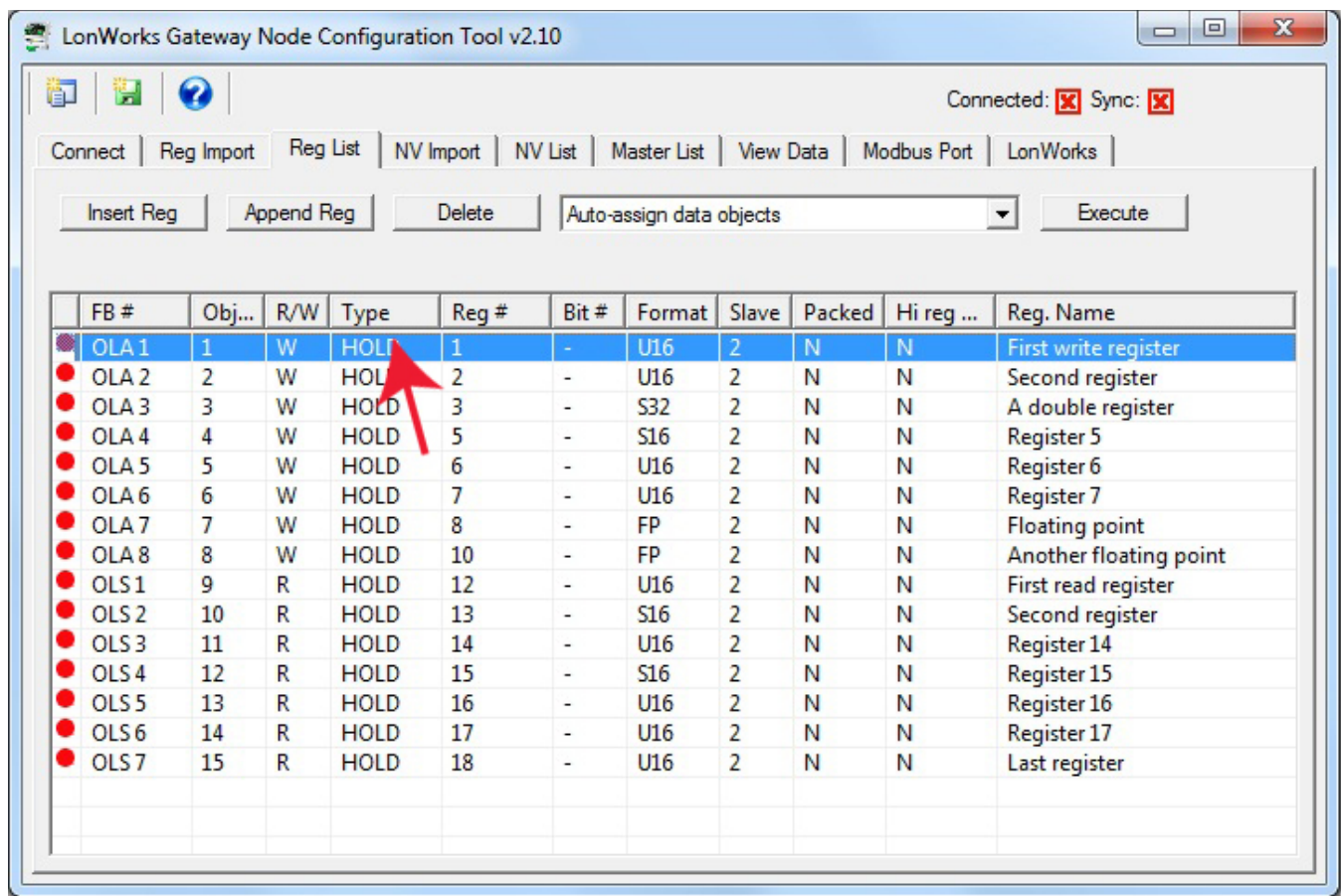
Upon executing Auto-assign data objects, the object numbers allocated will appear in the Obj column.



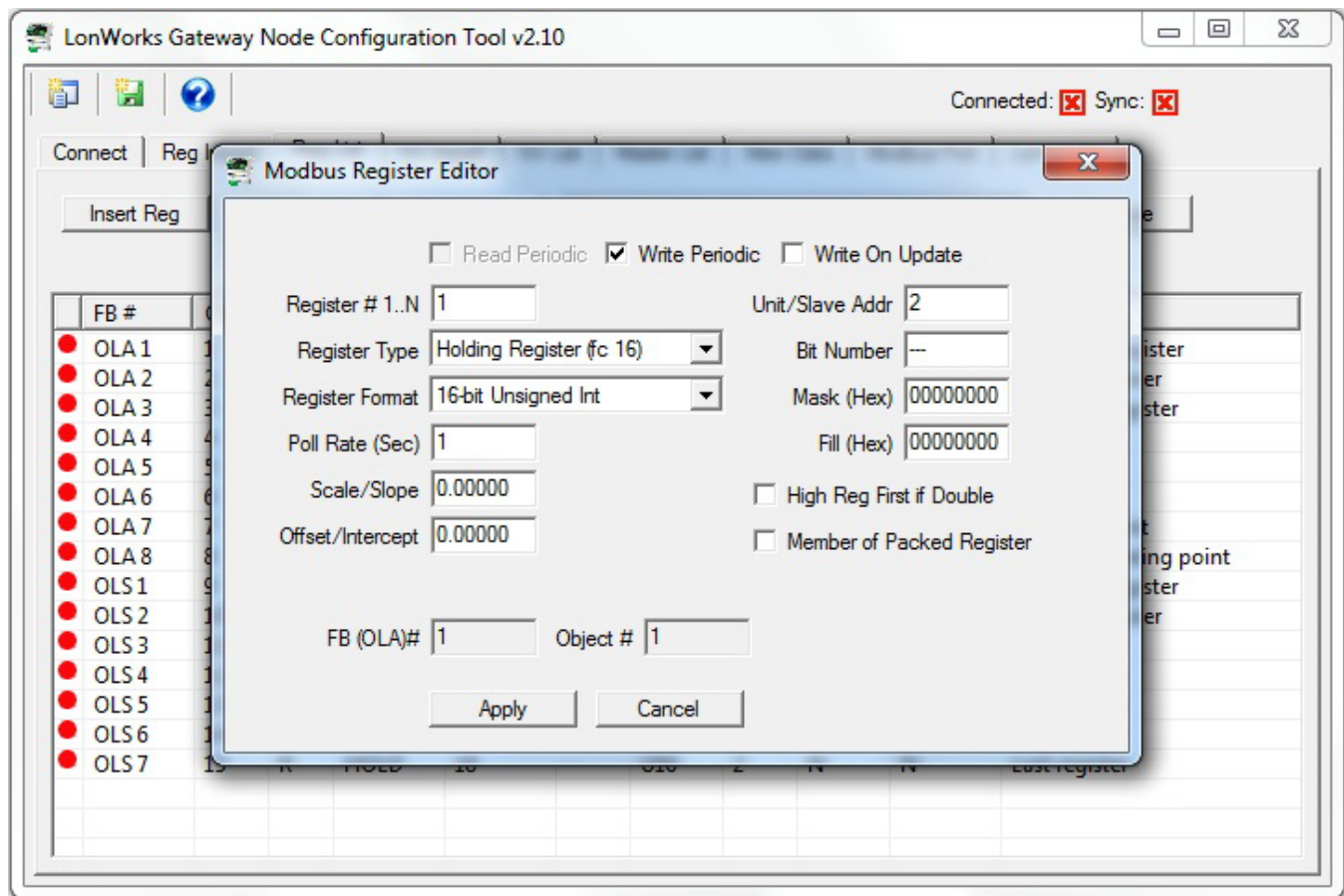
Creating of the configuration is now largely complete. You can make alterations to things like data scaling at this point if you wish. You can also change network variable types at this point. But for simply putting your Modbus device on the LonWorks network using generic counts as the data type, the configuration is complete. All that remains now is to send the configuration to the gateway device. At this point, the configuration only exists on your PC. It needs to get written into the device before the gateway will be functional.

6.2 Editing the Register List

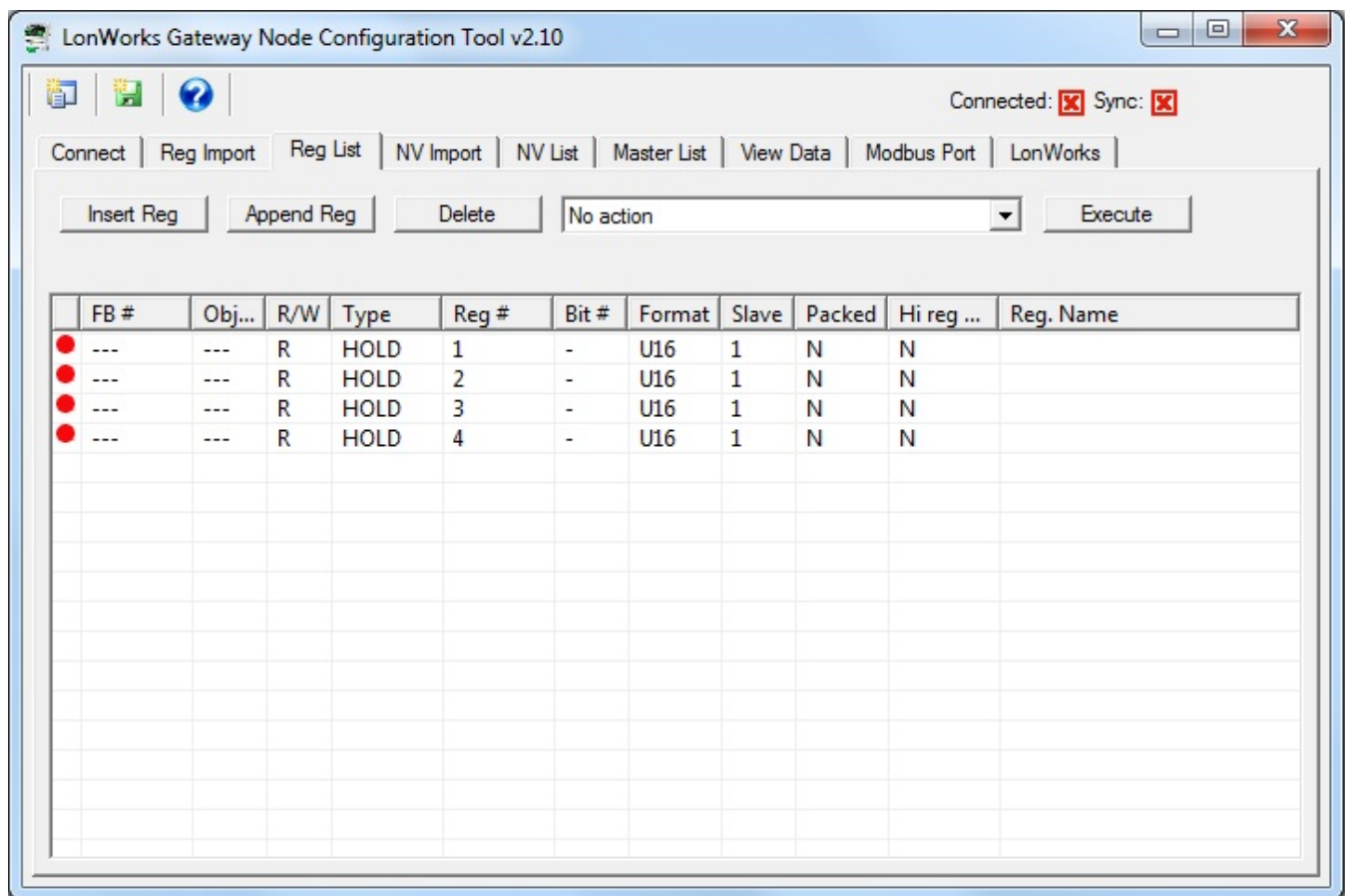
To edit the configuration of an existing Modbus register, double click on that line on the Reg List.



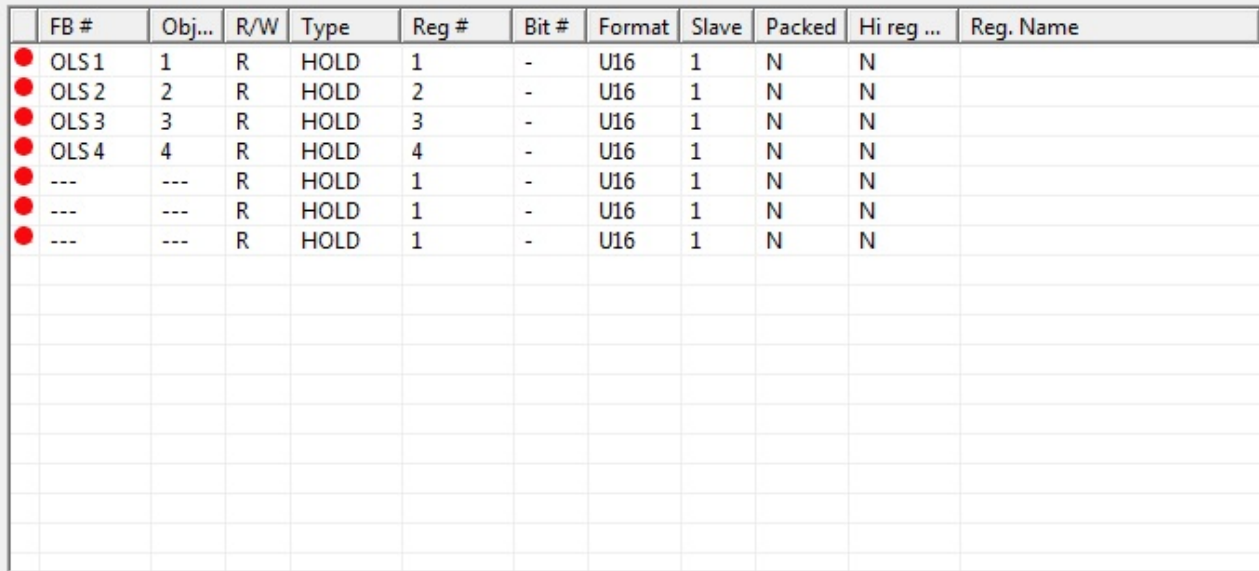
Upon double clicking a line on the Reg List, the dialog shown below will pop up for editing that entry.



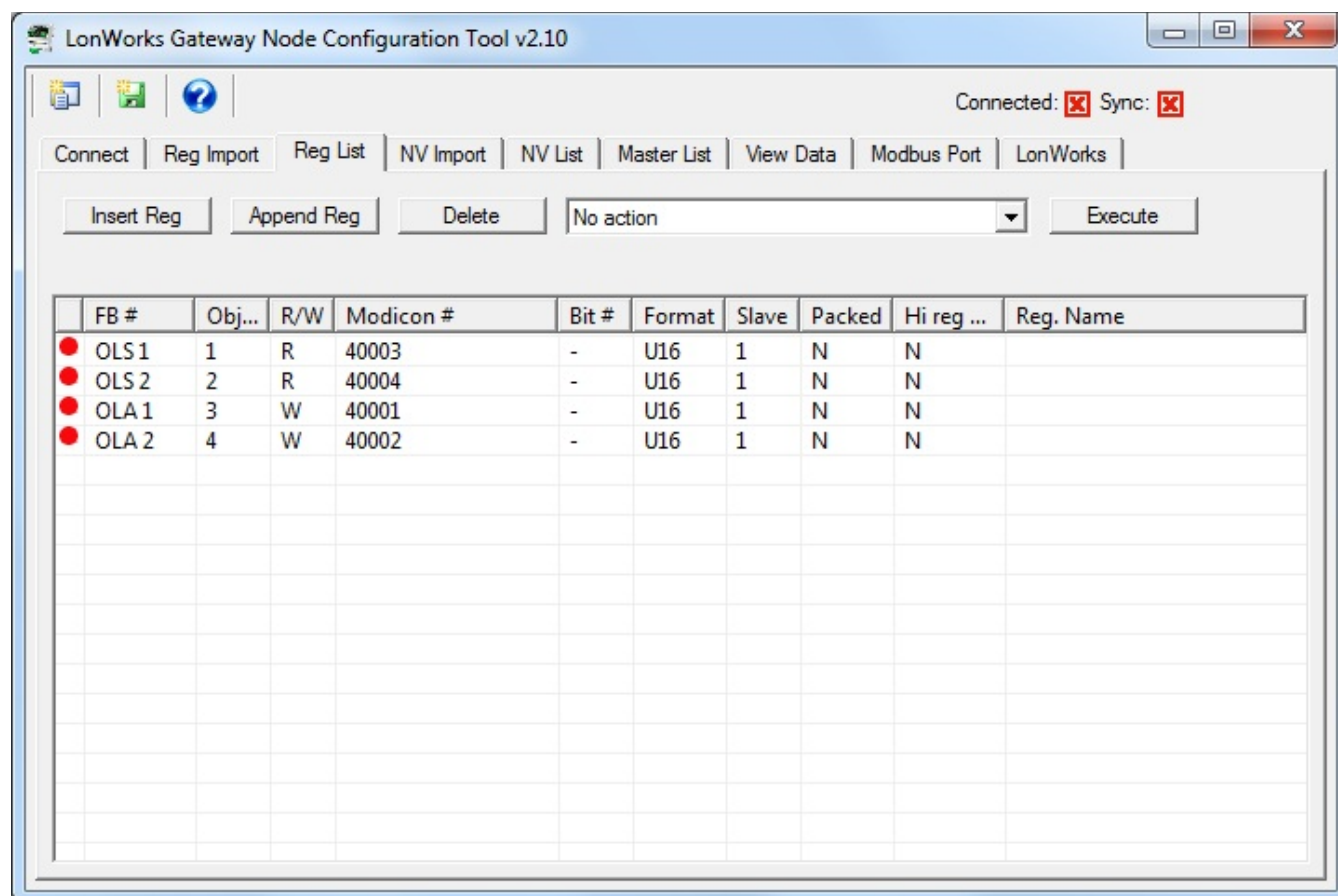
When only a list of registers is contained in the Reg List, and no function blocks (FB) or data objects have been allocated, you can use the Insert Reg, Append Reg, and Delete to alter the register list size. To insert a line in the middle of the list, click (single click) on a line on the Reg List page, then click Insert to insert a new register before the selected line. Append will always add another register to the end of the list.



Once function blocks and/or data objects have been assigned to the register list, appending or inserting new registers will be meaningless because you will be unable to allocate function blocks or data objects to those registers added. To add registers to a partially completed configuration, start by appending network variables on the NV List page, then go to the Master List to edit the modbus registers associated with the newly added NV's. Once that is completed, you can come back to the Reg List page and rebuild the list from the master list (execute action selected from drop down list).

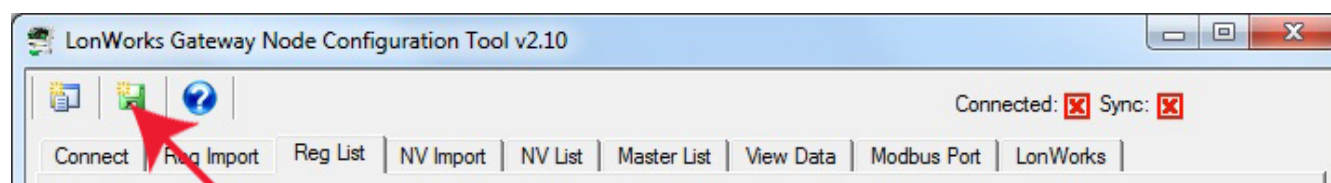


When Modicon display of register numbers has been selected on the Connect page, the Type and Reg # columns are replaced with a single Modicon # column. All other aspects of the Reg List page remain the same regardless of how register numbers are displayed.



6.3 Register List Export

Once you have created a Modbus register list, you can export your register list.



To export, click the new file icon on the toolbar. The CSV file that is exported will be in the file format described in Appendix D of this user guide. This means you can import the same file back into the configuration tool (on Reg Import page) at a later time, or open the file using a spreadsheet program to create documentation of your Modbus register set.

6.4 Definition of Modbus Register Configuration Parameters

The Modbus Register Editor dialog is accessed by clicking on a line on the Reg List page, or by clicking the Modbus portion of a line on the Master List page. The dialog will vary slightly depending on whether the register is associated with an NVI or NVO, whether the gateway is being configured as master or slave, and whether or not you are using Modicon display mode.

A Modbus register

associated with a Network Variable Output, or Open Loop Sensor function block, will read from a Modbus slave when the gateway is functioning as Modbus master.

Network Variable Output means this object provides data to the LonWorks network and expects to get it from the Modbus device.

The Modbus Register Editor dialog box is shown with the following settings:

- ☒ Read Periodic ☐ Write Periodic ☐ Write On Update
- Register # 1..N: 1
- Unit/Slave Addr: 1
- Register Type: Holding Register (fc 16)
- Bit Number: ---
- Register Format: 16-bit Unsigned Int
- Mask (Hex): 00000000
- Poll Rate (Sec): 1
- Fill (Hex): 00000000
- Scale/Slope: 0.00000
- ☐ High Reg First if Double
- Offset/Intercept: 0.00000
- ☐ Member of Packed Register
- FB (OLS)#: 1
- Object #: 0
- Buttons: Apply, Cancel

A Modbus register associated with a Network Variable Input, or Open Loop Actuator function block, will write to a Modbus slave when the gateway is functioning as Modbus master.

Network Variable Input means this object receives data from the LonWorks network and wants to provide it to the Modbus device.

The Modbus Register Editor dialog box is shown with the following settings:

- ☐ Read Periodic ☒ Write Periodic ☐ Write On Update
- Register # 1..N: 2
- Unit/Slave Addr: 1
- Register Type: Holding Register (fc 16)
- Bit Number: ---
- Register Format: 16-bit Unsigned Int
- Mask (Hex): 00000000
- Poll Rate (Sec): 1
- Fill (Hex): 00000000
- Scale/Slope: 1.00000
- ☐ High Reg First if Double
- Offset/Intercept: 1.00000
- ☐ Member of Packed Register
- FB (OLA)#: 2
- Object #: 4
- Buttons: Apply, Cancel

The read versus write selection is disabled when the gateway will be functioning as a Modbus slave. The gateway does not initiate any reads or writes in this scenario.

As a slave, the gateway expects the external master to write to registers associated with Open Loop

Sensors, and read from registers associated with Open Loop Actuators.

The Modbus Register Editor dialog box is shown with the following settings:

- ☒ Read Periodic ☐ Write Periodic ☐ Write On Update
- Register # 1..N: 1
- Unit/Slave Addr: 1
- Register Type: Holding Register (fc 16)
- Bit Number: ---
- Register Format: 16-bit Unsigned Int
- Mask (Hex): 00000000
- Host Timeout (Sec): 1
- Fill (Hex): 00000000
- Scale/Slope: 0.00000
- ☐ High Reg First if Double
- Offset/Intercept: 0.00000
- ☐ Member of Packed Register
- FB (OLS)#: 1
- Object #: 0
- Buttons: Apply, Cancel

The Register # and Register Type will be replaced with a single Modicon # window when the configuration tool has been set to use Modicon display mode (selected on Connect page). The configuration contained in the gateway itself is the same either way. The Modicon setting only changes how the information is displayed in the tool.

The Modbus Register Editor dialog box is shown with the following settings:

- ☒ Read Periodic ☐ Write Periodic ☐ Write On Update
- Modicon #: 40003
- Unit/Slave Addr: 1
- Register Format: 16-bit Unsigned Int
- Bit Number: ---
- Mask (Hex): 00000000
- Poll Rate (Sec): 1
- Fill (Hex): 00000000
- Scale/Slope: 1.00000
- ☐ High Reg First if Double
- Offset/Intercept: 1.00000
- ☐ Member of Packed Register
- FB (OLS)#: 1
- Object #: 1
- Buttons: Apply, Cancel

Read Periodic	Tells the gateway (master) to read this register at the rate set by the Poll Rate. Only valid for Open Loop Sensor function blocks.
Write Periodic	Tells the gateway (master) to write this register at the rate set by the Poll Rate. Only valid for Open Loop Actuator function blocks.
Write On Update	Tells the gateway to write this register when new data is received from LonWorks. One or the other of the 'write' options can be selected, but not both. Only valid for Open Loop Actuator function blocks.
Register #	Register number 1..N using standard register numbering (not Modicon). Register 1 is also address zero on the wire. If registers are documented by the manufacturer as starting at zero, add 1 to all numbers.
	Valid register types are Coil, Discrete Input, Input Register, and Holding Register. Coil and holding register have two options, each indicated by a

Register Type	different "fc" number in parenthesis. This number is the function code for writing to these registers. FC 5 and 6 will write only a single register, while FC 15 and 16 will write multiple registers. Codes 15 and 16 will also be used by default to send a single register. Some Modbus slaves are particular about which function code is used to write, and you need to make the appropriate selection here.
Modicon #	If Modicon notation is selected on the Connect page, then Register # and Register Type are replaced by a single Modicon # window. When enabled, Modicon numbers as described below are used.
Register Format	<p>Register format options include 16-bit or 32-bit, unsigned or signed integer, or floating point. When 32-bit integer or floating point are selected, you are really reading or writing two consecutive registers. Modbus protocol defines a holding register (or input register) as strictly 16 bits. Any larger data element is spread across multiple registers. When using register pairs, the order in which the registers are interpreted is not standardized. You have the option of selecting either order by use of the High Reg First if Double check box.</p> <p>There is also a Bit format. This ONLY applies to coil and discrete input. If you are attempting to read a single bit from a holding register, you must read the register as unsigned 16-bit (or 32-bit if applicable) and then apply a mask. The format 'Bit' refers to the format of data in the Modbus slave; it does not refer to your desired end result.</p> <p>Support for "Mod10" in 2, 3, and 4-register formats is also included. This format is often found in power meters or analyzers, and requires specific interpretation to convert Mod10 to a floating point value useful to LonWorks.</p>
Poll Rate (Sec)	When reading or writing periodically, set this to the number of seconds between polls. Poll rate applies when the gateway is Modbus master.
Host Timeout (Sec)	Host timeout only applies when the gateway is a Modbus slave. If the external master does not write new data to the gateway within this amount of time, the register will be flagged as in default, and if the data object is configured to do so, it will then assume a default value that you configure from the Master List page.
Scale/Slope	Data is multiplied by this value when read from Modbus, before storing to the internal gateway data object (and hence LonWorks NV). The process is reversed when writing to Modbus. A scale value of zero will mean 'no scaling' which would be mathematically equivalent to scale=1.0. This scale does not apply when the gateway is functioning as a Modbus slave.
Offset/Intercept	This value is added to data when read from Modbus, before storing to the internal gateway data object (and hence LonWorks NV). The process is reversed when writing to Modbus. This offset does not apply when the gateway is functioning as a Modbus slave.
Unit/Slave Addr	This is the slave device address that the gateway will query when the gateway is Modbus master. If the TCP version of gateway is used, the unit is also used to look up the IP address of the slave. This value is not used when the gateway itself is the slave.
	Bit Number is used to create a mask for use when wanting to extract a single bit from a holding register or input register. It is common for Modbus devices to pack up to 16 different status bits in a single holding register. But to be meaningful as a LonWorks status indicator, you may wish to extract a single bit from the register. The Mask value is logically And-ed

Bit Number	<p>with the data read from the Modbus device, and then shifted right until the masked position is in the least significant bit position of the result.</p> <p>The Bit Number is simply a short cut method for creating a value in the Mask window. Note that when you enter a number, the Mask value also changes to some hexadecimal value. This is your mask.</p>
Mask (Hex)	You can enter the Mask directly as a hexadecimal value, or use the Bit Number window to create it for you. The use of the Mask value is noted above in the Bit Number discussion.
Fill (Hex)	Fill is also a hexadecimal mask, but used in a different way. It is not used when reading; it is only used when writing to Modbus registers. The Fill value will be logically Or-ed with data received from LonWorks before writing it to Modbus. Some Modbus devices require that certain bits always be written as a logic 1 (one) and therefore the Fill is required to cause this to happen regardless of what the variable data is.
High Reg First if Double	<p>Floating point and 32-bit integer values occupy two consecutive Modbus registers. You will need to consult manufacturer's documentation to determine whether the most significant half of the data is in the first or second register. Check this box if the most significant half of the data ('high reg') is the first or lower numbered register.</p> <p>If you are getting seemingly random results when reading double registers, try swapping them to see if the order is reversed.</p>
Member of Packed Reg	You have the option of mapping multiple data objects to a single Modbus register. If this register entry is one of several pointing to the same Modbus register number, but with different Mask values, check this box to force the gateway to go looking for all pieces of data before writing it to the Modbus device. It is only a matter of improved efficiency when reading, but can be critical when writing.
FB #	This is the LonWorks function block number assigned to this Modbus register.
Object #	This is the gateway data object number assigned to this Modbus register.

6.5 Modicon Register Numbers Explained

There is a great deal of confusion around Modbus register numbers such as 40001. This type of numbering is NOT officially recognized by the Modbus protocol standard. Virtually all Modbus devices whose documentation reference 40001 do not actually have a register 40001 - this is short hand for holding register #1, which is actually register address 0 (zero).

Modbus started out as a proprietary communication protocol created by Modicon, which became Gould Modicon, which became part of Schneider Electric (possibly after some other interim name changes as well). The number 40000 originally referred to the memory addressing scheme within the PLC's (programmable logic controllers) that Modicon was producing at the time. As use of the protocol became more wide spread, and with the protocol eventually becoming an open protocol, some of the historical artifacts of the original implementation hung around. One of those is the idea that 40001 is an actual register number.

The advantage in using the Modicon notation is that two pieces of information are included in a single number: (a) The register type; (b) The register number. A register number offset defines the type.

The types of registers referenced in Modbus devices, and supported by Babel Buster gateways, include the following:

- Coil (Discrete Output)
- Discrete Input
- Input Register
- Holding Register

Valid address ranges as originally defined for Modbus were 0 to 9999 for each of the above register types. Valid ranges allowed in the current specification are 0 to 65,535. The address range originally supported by Babel Buster gateways was 0 to 9999. The extended range addressing was later added to all new Babel Buster products that use this notation. (The BB2-2010, BB2-2011, and BB2-6020 are user switchable between Modicon display mode and standard display mode where the first register is #1.)

The address range applies to each type of register, and one needs to look at the function code in the Modbus message packet to determine what register type is being referenced. The Modicon convention uses the first digit of a register reference to identify the register type.

Register types and reference ranges recognized by the gateway in Modicon display mode are as follows:

0x = Coil = 00001-09999
 1x = Discrete Input = 10001-19999
 3x = Input Register = 30001-39999
 4x = Holding Register = 40001-49999

Translating references to addresses, reference 40001 selects the holding register at address 0000 (also referred to as register number 1). The reference 40001 will appear in some manufacturer's documentation and is used to define the Modbus register when the gateway is set to use Modicon display mode. The address 0000 will be transmitted in the message packet.

On occasion, it is necessary to access more than 10,000 of a register type. Based on the original convention, there is another defacto standard that looks very similar. Additional register ranges recognized by "Extended Modicon Notation" are as follows:

0x = Coil = 000001-065535
 1x = Discrete Input = 100001-165535
 3x = Input Register = 300001-365535
 4x = Holding Register = 400001-465535

When using the extended register referencing, it is mandatory that all register references be exactly six digits. This is the only way Babel Buster will know the difference between holding register 40001 and coil 40001. If coil 40001 is the target, it must appear as 040001.

6.6 Deciphering Modbus Documentation

Documentation for Modbus is not well standardized. Actually there is a standard, but not well followed when it comes to documentation. You will have to do one or more of the following to decipher which register a manufacturer is really referring to:

- Look for the register description, such as holding register, coil, etc. If the documentation says #1, and tells you they are holding registers, then you have holding register #1. You also have user friendly documentation.
- Look at the numbers themselves. If you see the first register on the list having a number 40001, that really tells you register #1, and it is a holding register. This form of notation is referred to as Modicon.
- Look for a definition of function codes to be used. If you see a register #1, along with notation telling you to use function codes 3 and 16, that also tells you it is holding register #1.

IMPORTANT: Register 1 is address 0. Read on...

d) Do the numbers in your documentation refer to the register number or address? Register #1 is address zero. If it is not clear whether your documentation refers to register or address, and you are not getting the expected result, try plus or minus one for register number. All Control Solutions products refer to register numbers in configuration software or web pages. However, some manufacturers document their devices showing address, not register numbers. When you have addresses, you must add one when entering that register into configuration software from Control Solutions.

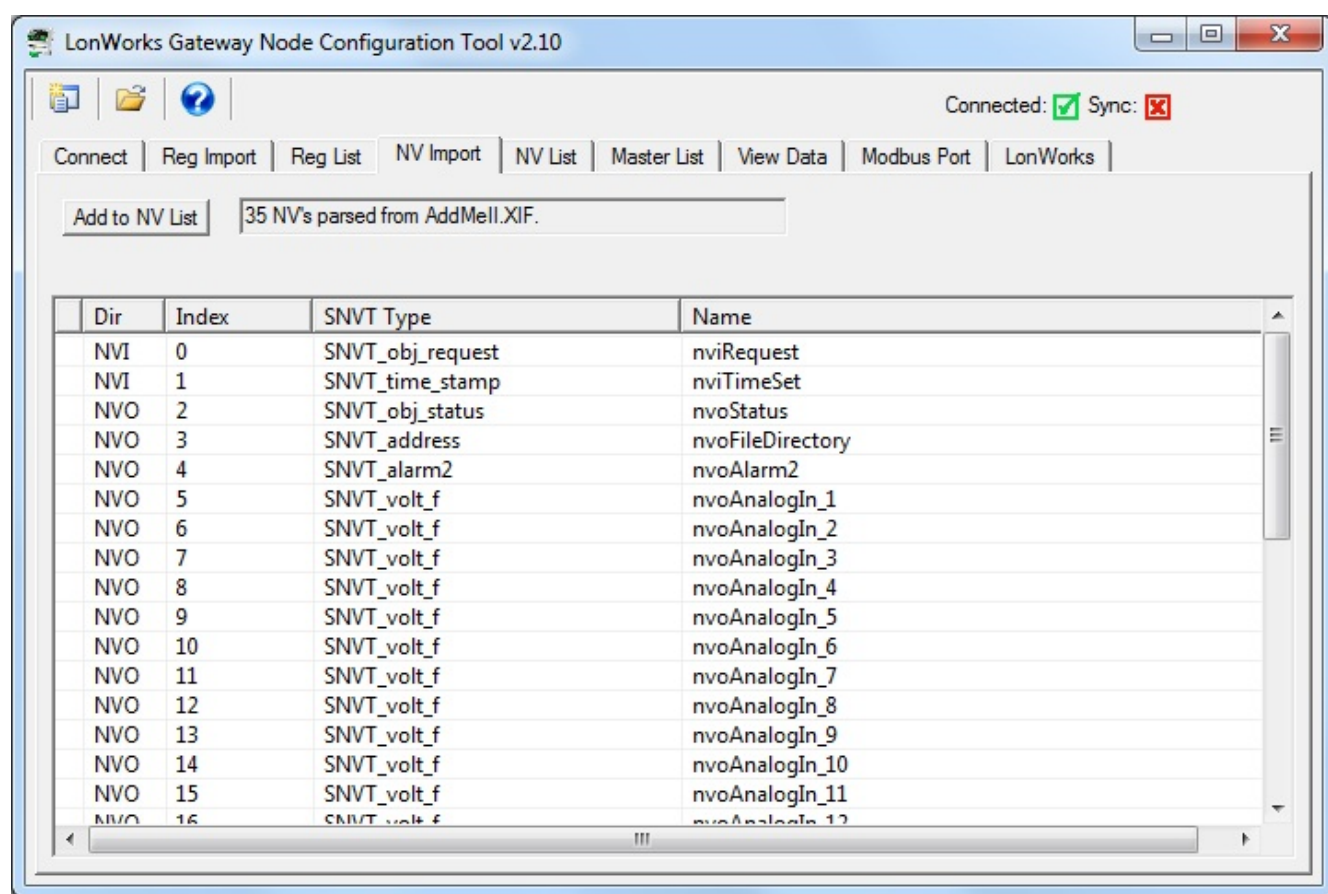
7 Tool 'NV Import' Page

7.1 Importing an XIF File

Starting from an XIF file for a specific LonWorks device, and auto-building the Modbus register list, is the approach you would take if you want to make a Modbus device mimic that particular LonWorks device on the LonWorks network.

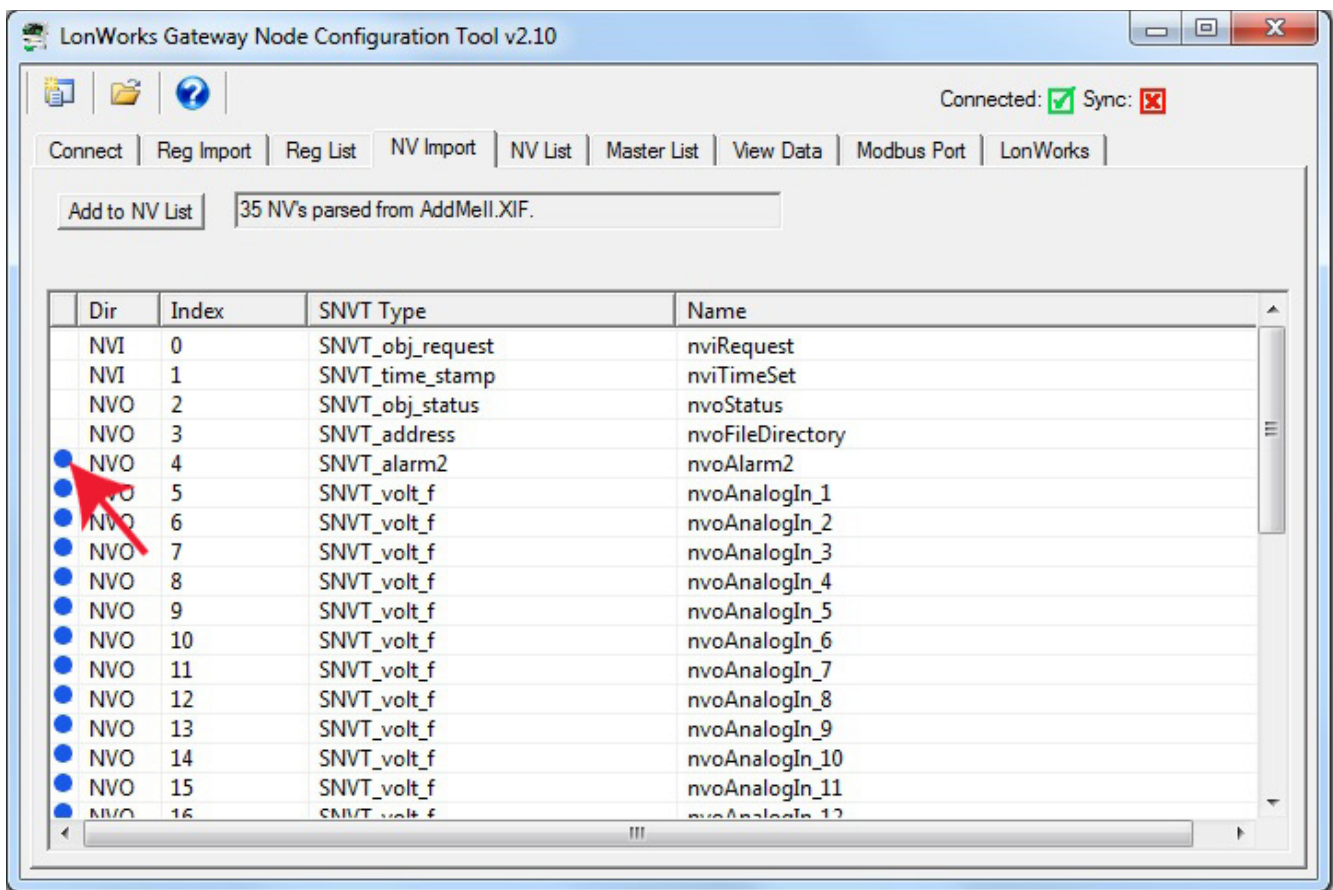
IMPORTANT: If your intent is to put a LonWorks device on a Modbus network, this is not the correct gateway to use. The gateway discussed here only applies where you are interfacing Modbus devices to a LonWorks network, and the LonWorks network is the primary network of the overall system.

To begin building your configuration from an XIF file, go to the NV Import page. Click on the file icon to open an XIF file. Once the XIF is imported, the list of network variables will be displayed on the NV Import page.



The NV Import page is essentially a scratch pad where you import the content of XIF files, then select which of the available variables you wish to include in your gateway configuration. Click on the icon column header to select all items, or click on the icon column for individual lines to select only those lines. The icon will show a blue dot for those lines that are about to be included.

When you have made your selections, click Add to NV List. The selected variables are now copied to the NV List.



If you need multiple copies of the same set of network variables, you can click the Add to NV List more than once. The same set of selected NV's will be added again each time you click the button.

You may also import different XIF files and continue to add to the NV List by opening a file, selecting NV's, clicking Add, and repeating the process.

The NV Import list will hold up to 1000 network variables at one time. Most LonWorks devices will have a list much shorter than that, and can potentially be added multiple times. It should be noted however, that a single NV on the NV Import list will turn into multiple entries in the NV List if it is a structured NV. The NV List has a capacity for 1000 entries - which may or may not equate to 1000 network variables depending on whether any of them are structured.

Although the maximum list size is set at 1000, the NV List is will never actually contain 1000 network variables. The LonWorks FT5000 microserver provides 120 network variable inputs, and 120 network variable outputs, for a total of 240 network variables which can be structured variables. If structured, the NV will map to multiple data objects. There is a pool of 400 data objects to work with. Each data object, in turn, maps to a Modbus register.

Therefore, the actual limits will be the lesser of 240 LonWorks network variables or 400 Modbus registers. Furthermore, there is a limit of 120 network variables in each direction, and the direction cannot be changed. This restriction is imposed by the LonWorks architecture.

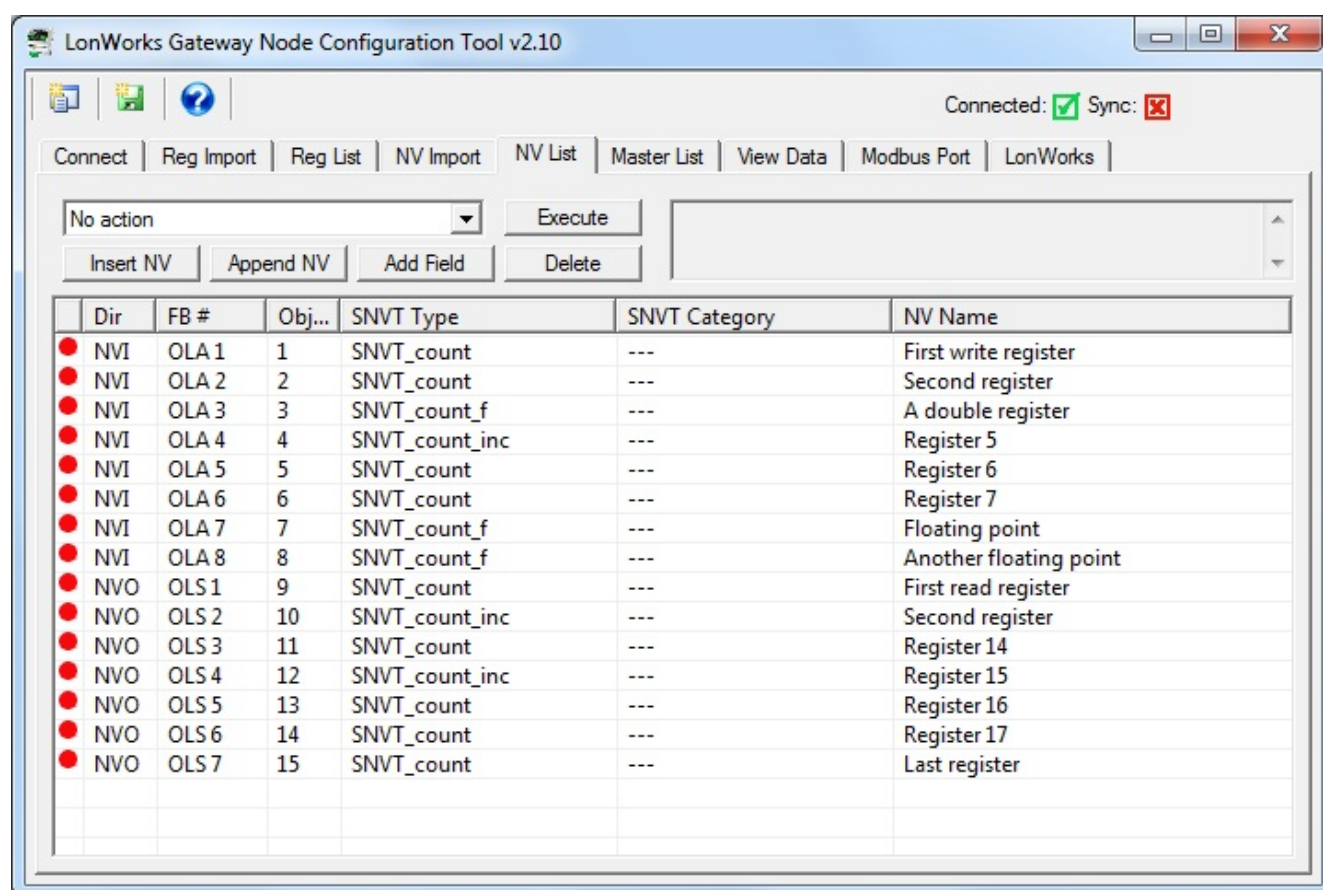
8 Tool 'NV List' Page

8.1 Configuration from Register List

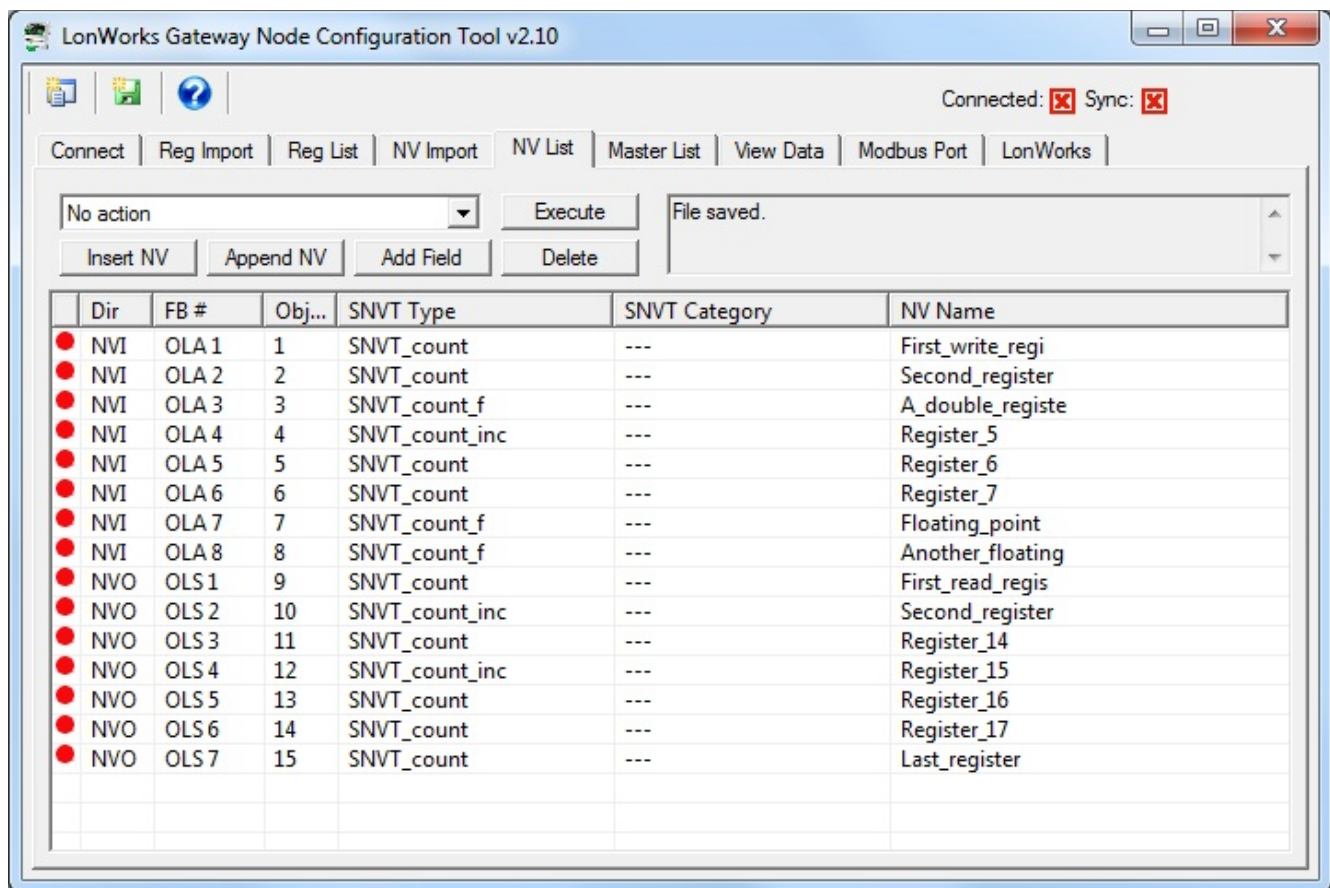
Creating of the configuration is completed primarily on the Reg Import and Reg List pages. Once registers are imported, you come here to the NV List page.

This is the set of network variables that were automatically created for you. If you wish to export an XIF file defining your LonWorks device, you may do so here by clicking the green file icon to create a new file. Some network management tools such as LonMaker can import the XIF directly from the device and you do not need an XIF file. However, some network management tools do require the XIF file, and this is where you create it.

If you want your LonWorks interface to have network variable types other than those that were selected by default, this is where you make those changes.

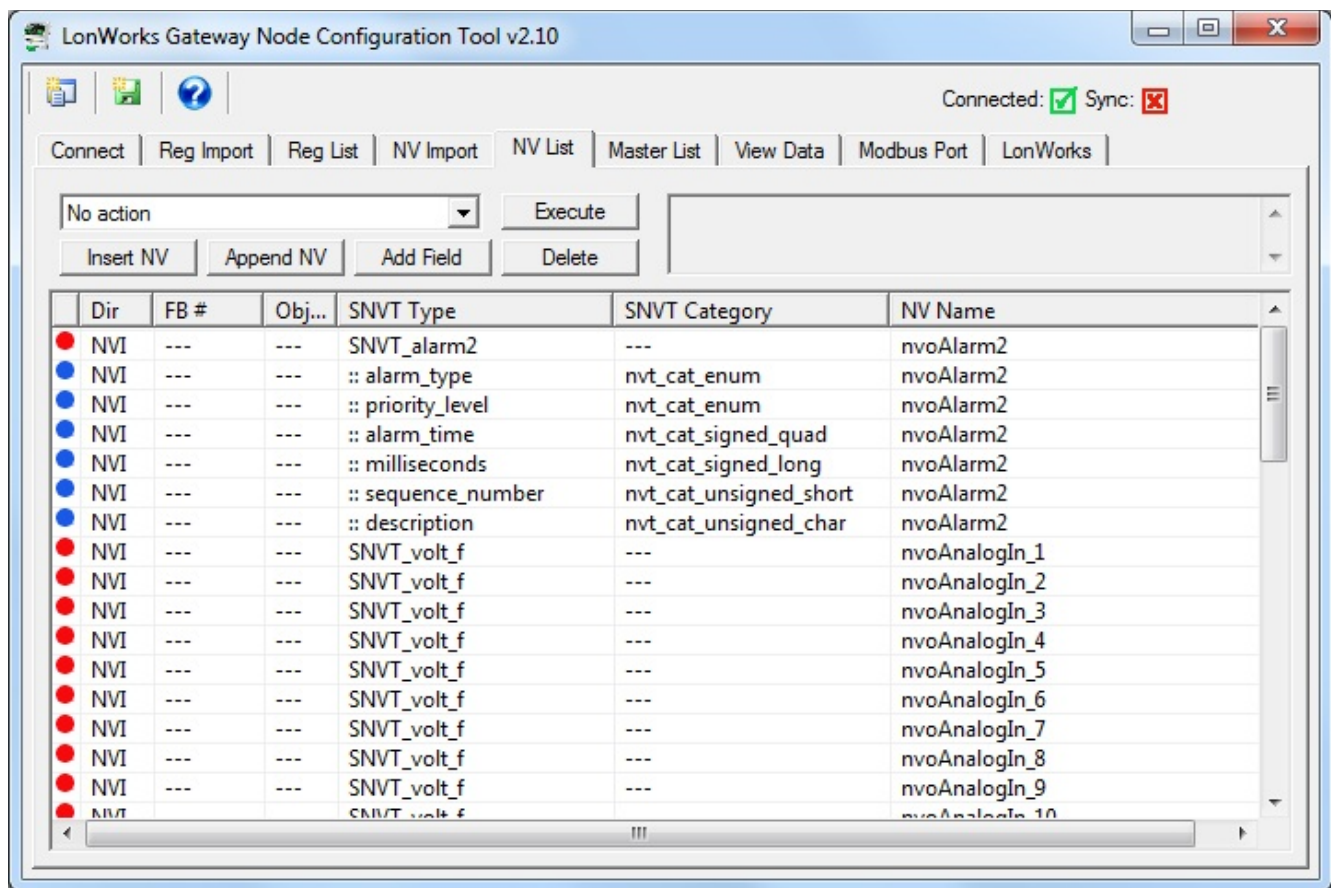


The names provided in the CSV file you imported might not necessarily qualify as legal network variable names. A "legal" network variable name is limited to 16 characters and no spaces. Therefore, spaces will be replaced with underscores and names will be truncated in the process of exporting the XIF file. Following XIF export (file save), the NV List above will appear as shown below instead. You may wish to edit the names *before* creating an XIF file.



8.2 Configuration from XIF File

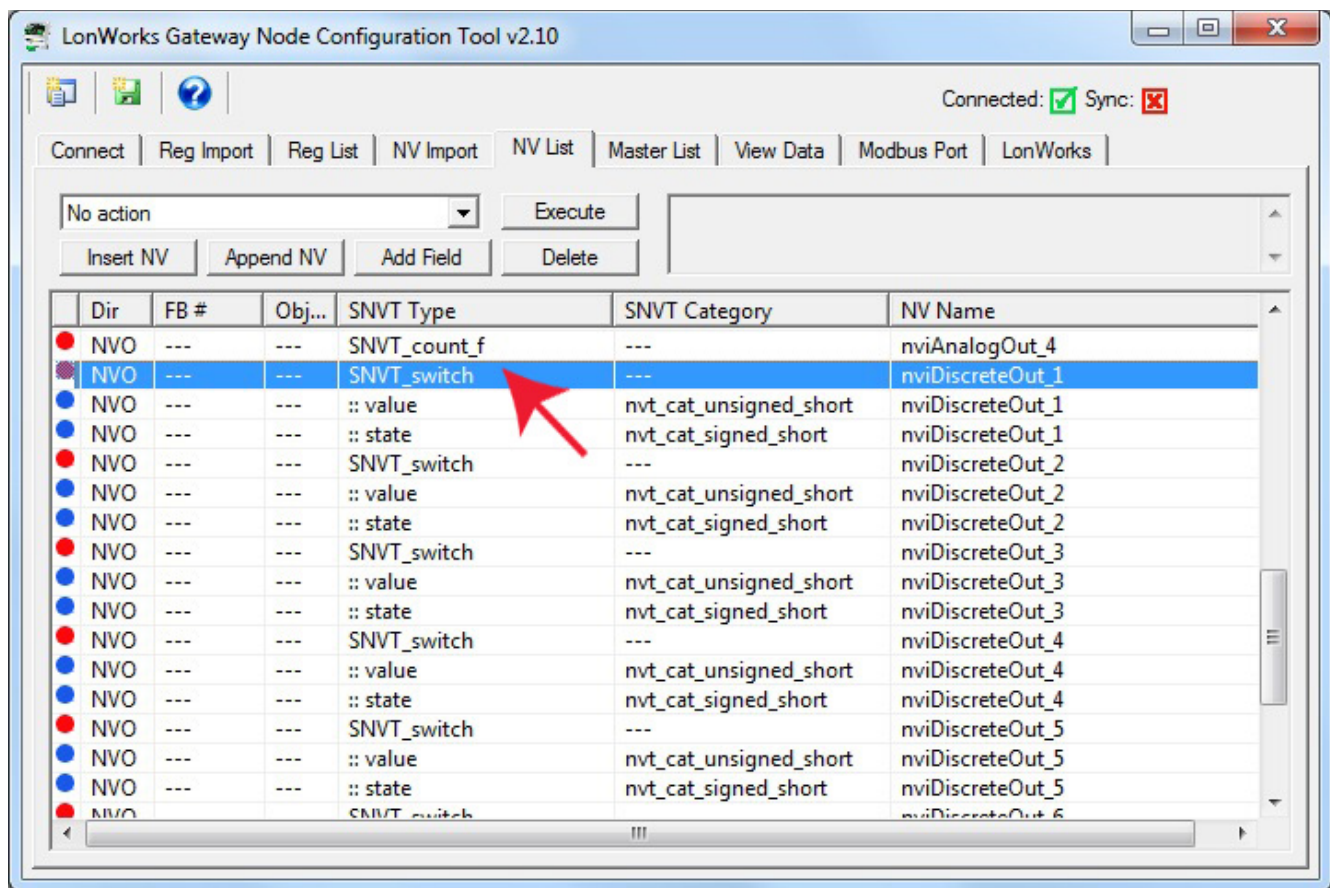
The NV List is the definition of the list of network variables that will become present in your LonWorks device (i.e. this gateway) when viewed as a node on the LonWorks network. The icon in the first column will be red if this NV definition has not yet been written to the gateway, and green if it has been written to the gateway. Blue icons indicate fields of structured network variables. All lines with a blue icon are part of the network variable immediately preceeding the set of blue icons. There will be only one network variable, but multiple Modbus registers, for a structured network variable (refer to previous section in this user guide if you did not already review treatment of structured variables).



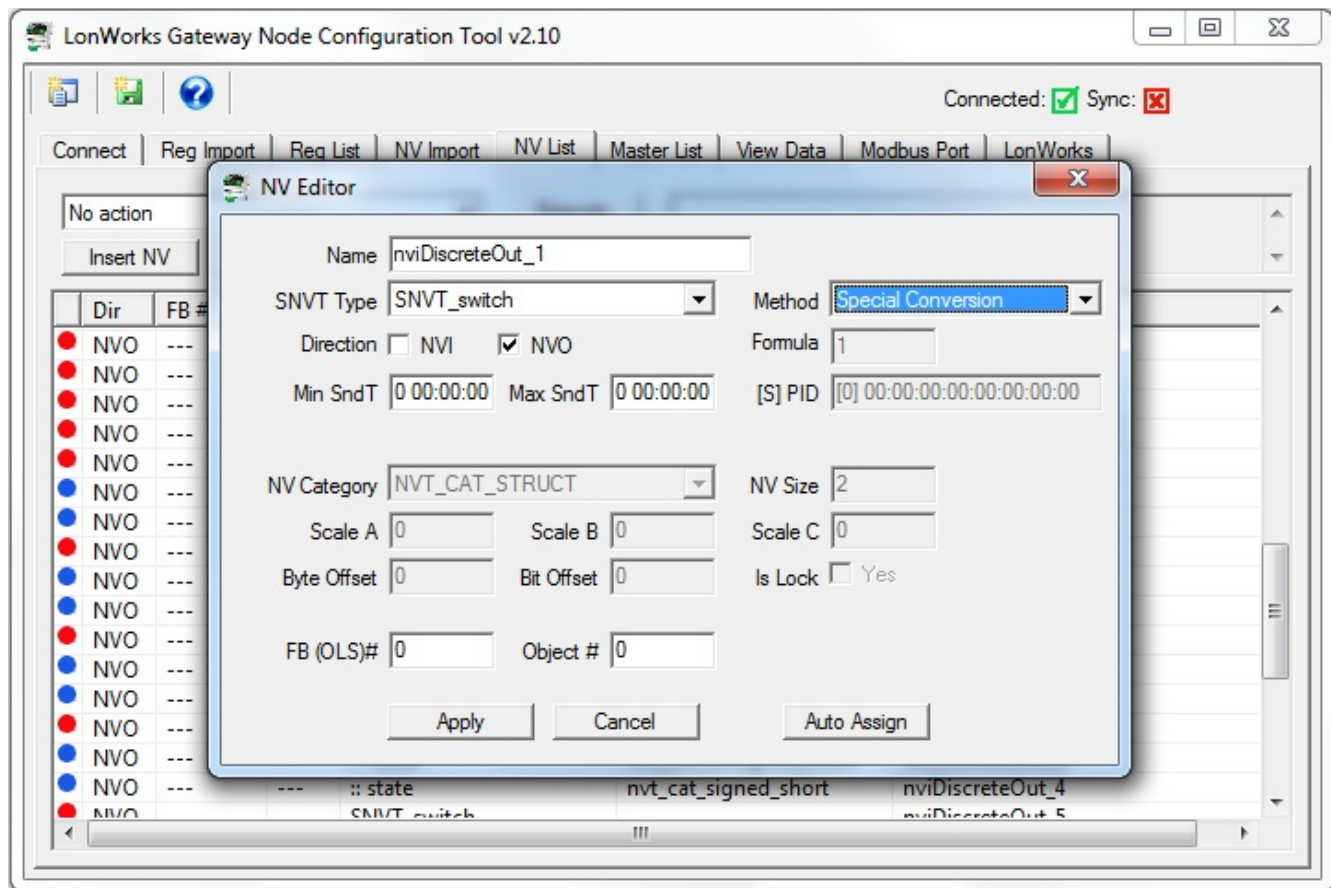
If you wish to make changes to the NV List, this is the point where you should do so. Do not proceed to assign function blocks (FB #'s) or data objects until the NV List is finalized.

You have a few options here, and these are described in more detail below. You may add network variables, delete them, or change what type they are. If they are structured, they will be automatically expanded into a list of all of their fields. If you add a structured NV which is not a standard LonMark type, you will need to add fields manually to build up the structure.

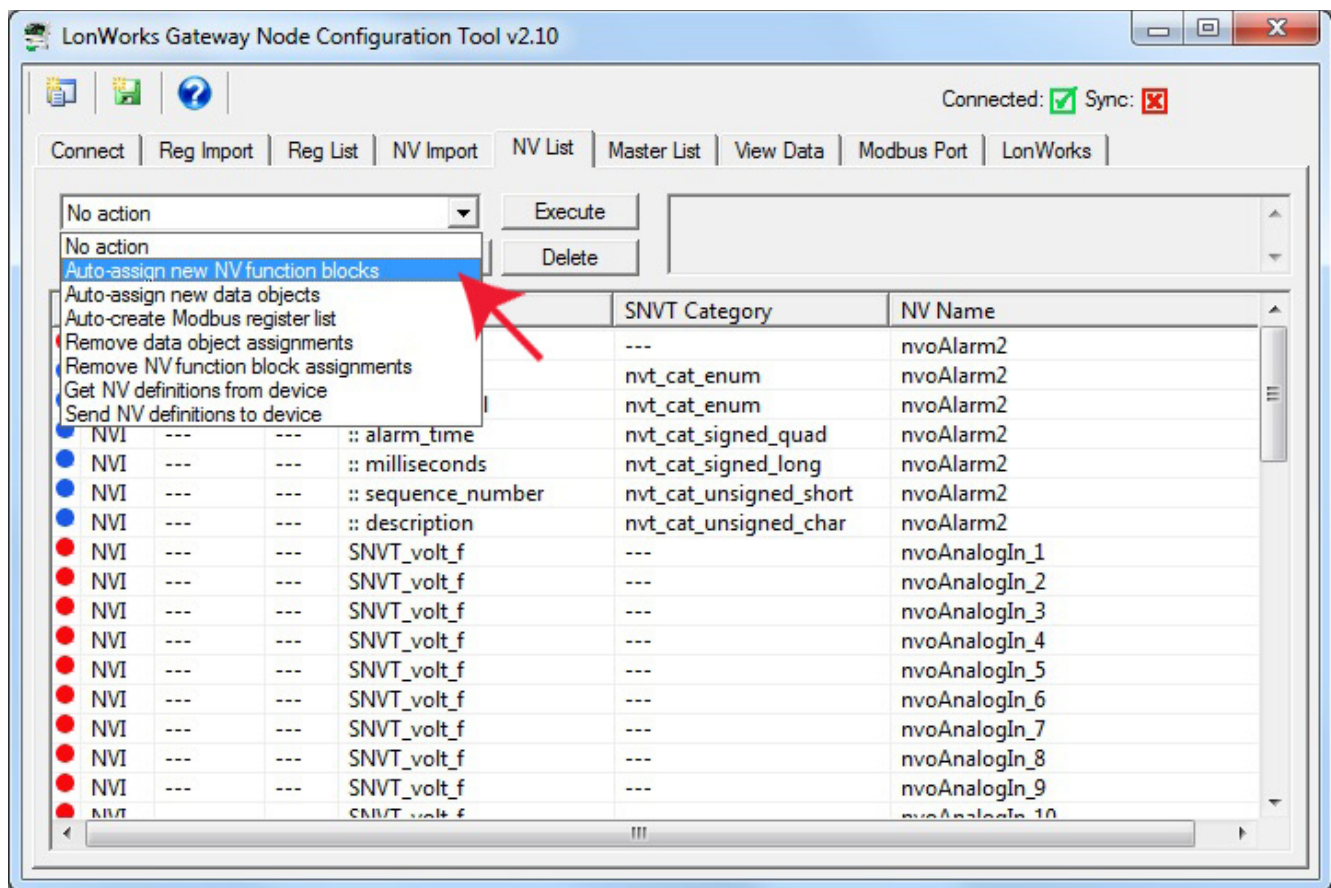
Some network variable types provide for special conversions. SNVT_switch is one such NV. To modify the NV definition, double click on the respective line in the NV List.



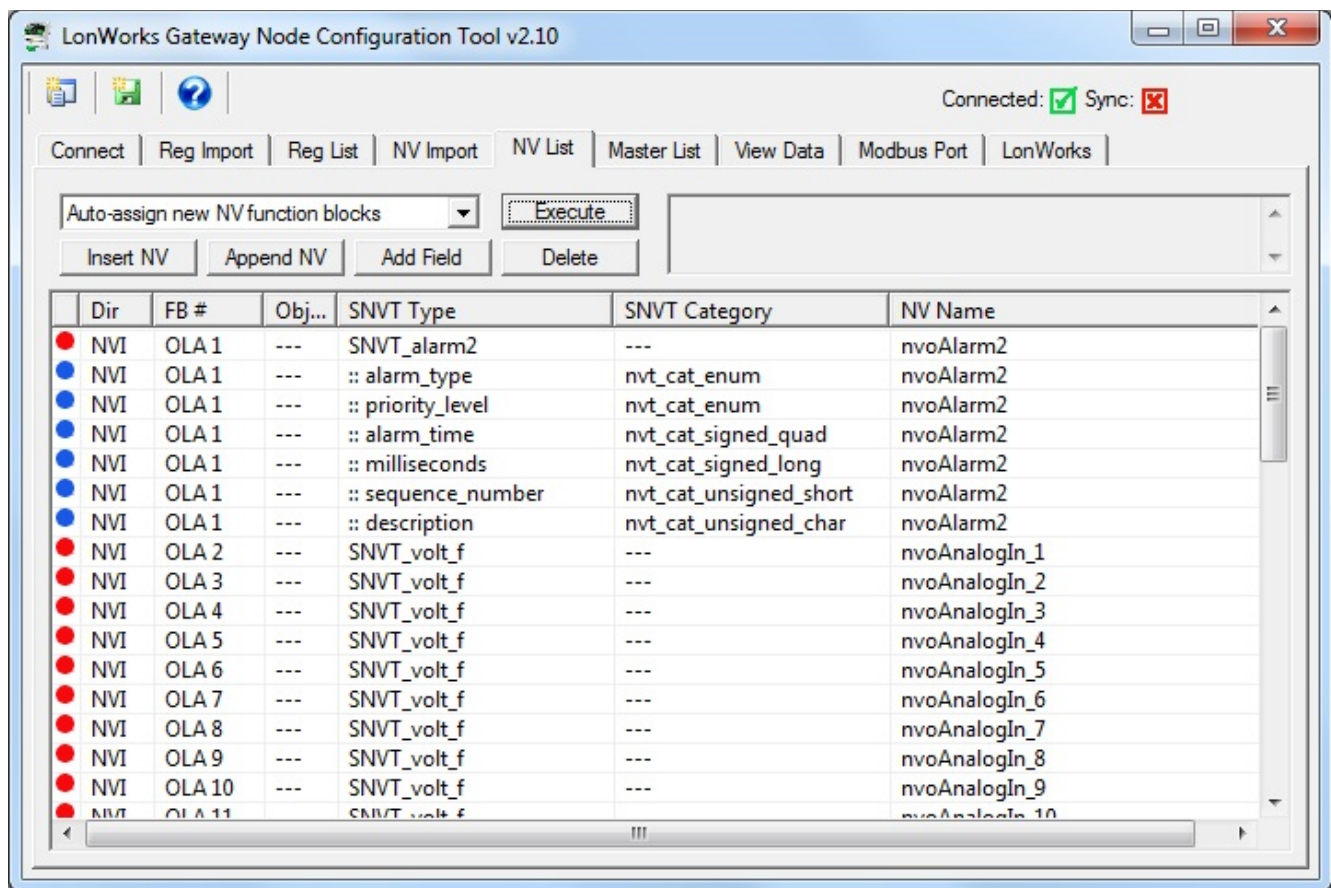
Upon double clicking a network variable in the NV List, the NV Editor dialog will appear. The SNVT_switch example is illustrated here. Your selection of whether to use the special conversion will decide whether SNVT_switch maps to one or two Modbus registers, and therefore whether SNVT_switch maps to one or two data objects. For this reason, it is important that you make all of your NV related selections and configurations BEFORE assigning function blocks and data objects.



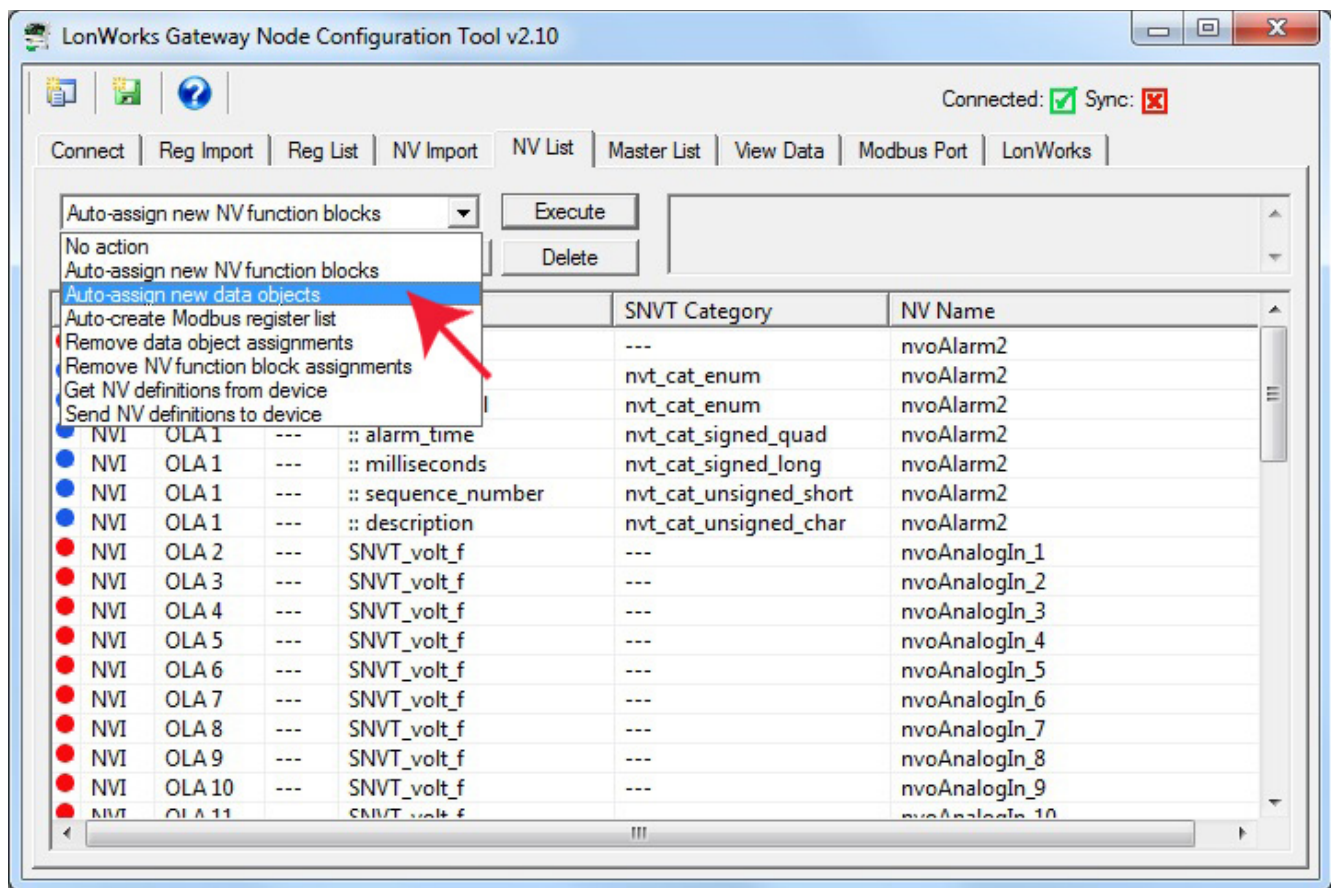
Once you have made all of the desired modifications to network variables in the NV List, proceed to assign function blocks. Select "Auto-assign new NV function blocks" and click Execute.



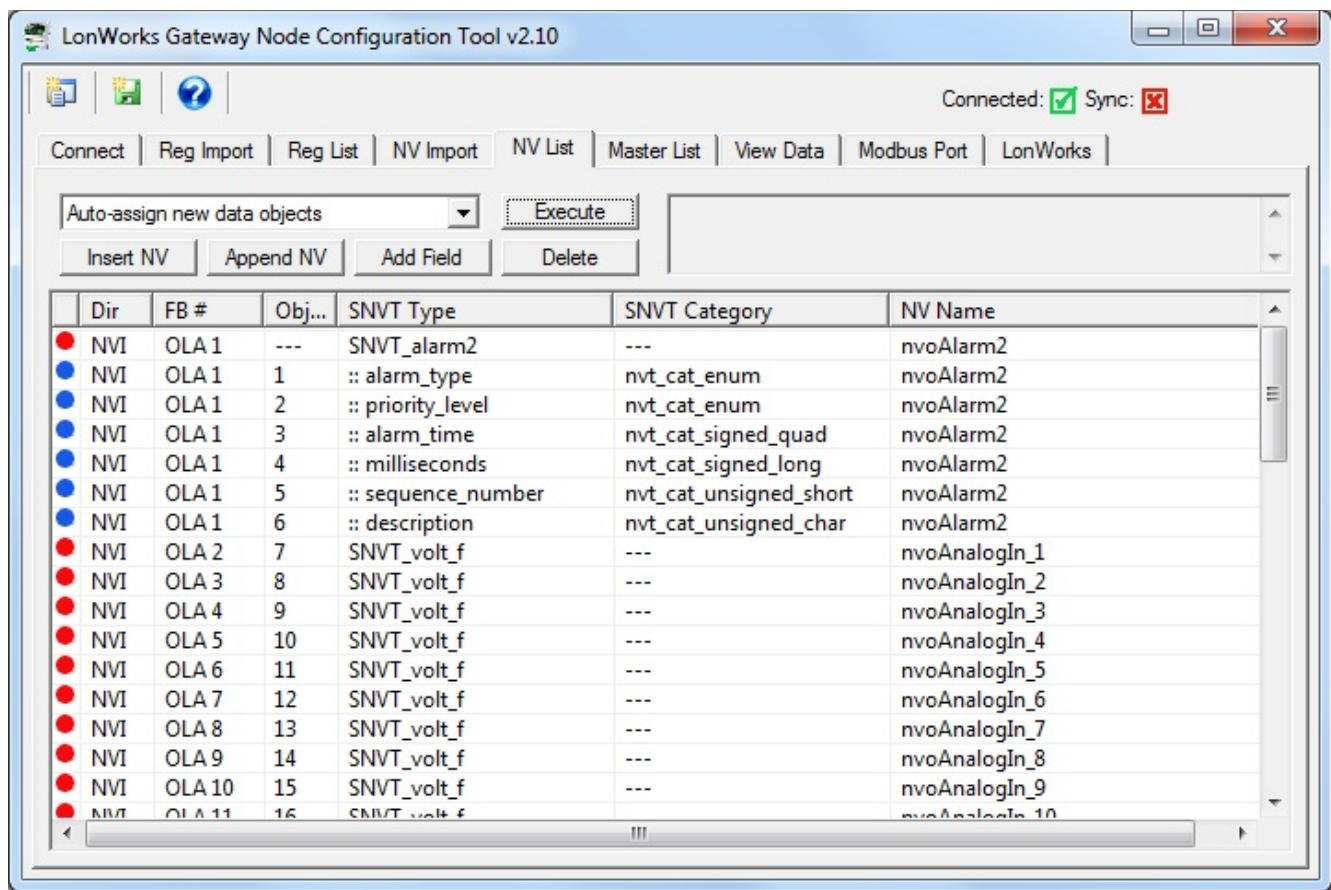
Upon execution of the function block assignments, the FB# column will be populated.



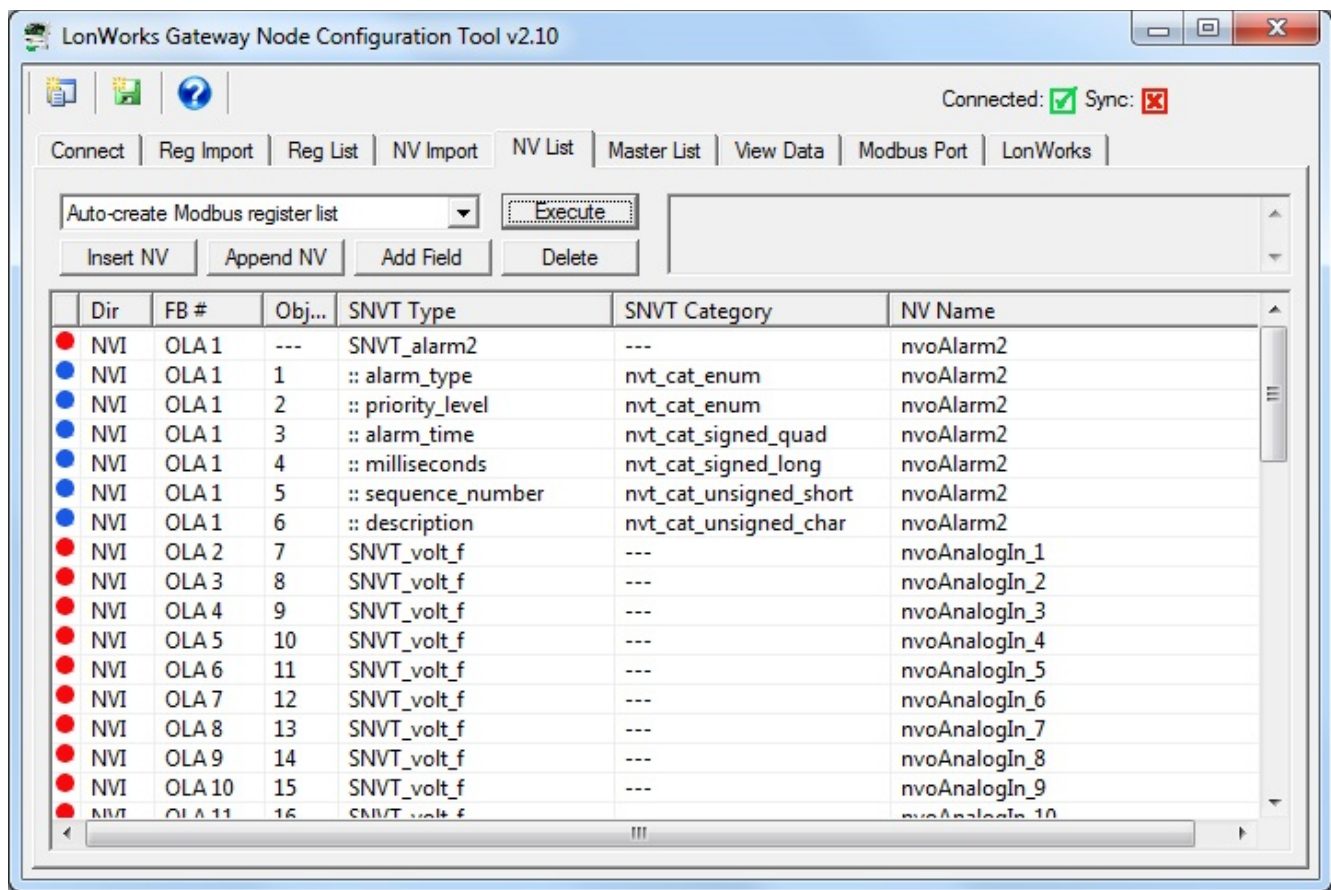
Next, select and execute "Auto-assign new data objects".



After data object assignment, the Obj... column will be populated.



Finally, select and execute "Auto-create Modbus register list". Nothing new will appear on the NV List page, but registers will now appear on the Reg List page.

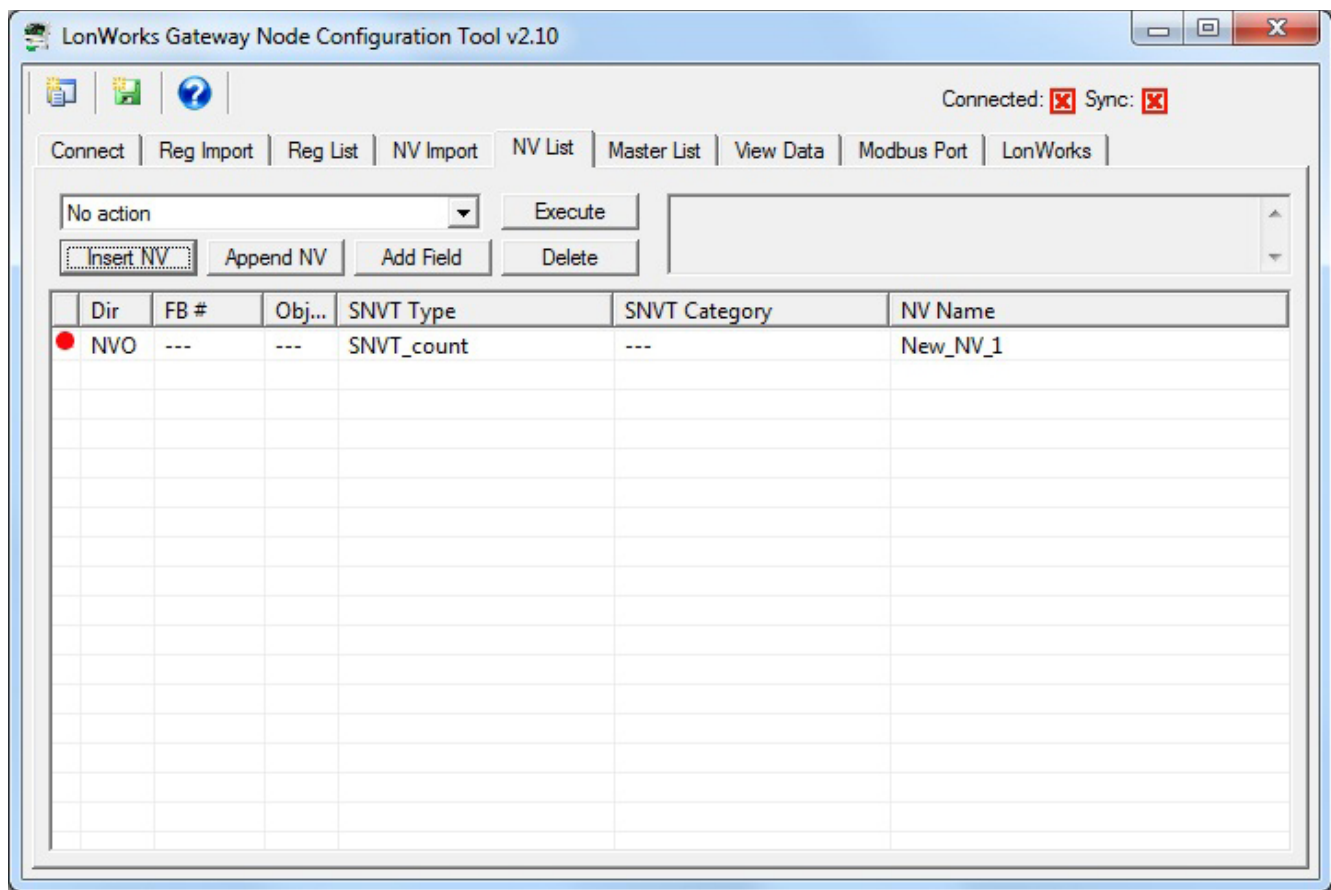


8.3 Building Configuration Manually

It is not required that you start with either a CSV file or an XIF file. You can use the configuration tool to start with a blank slate and build your configuration from scratch.

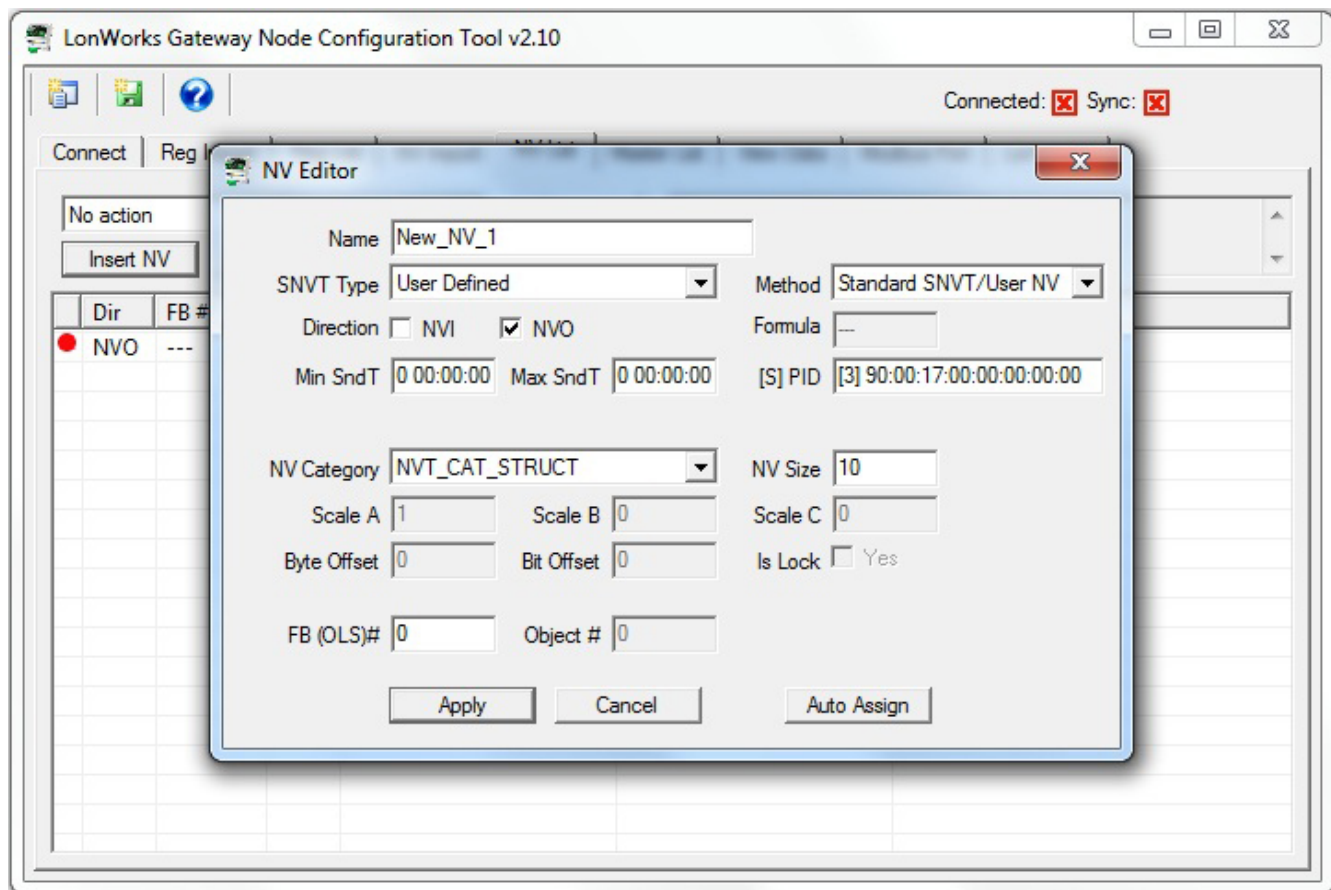
It is assumed that you have some familiarity with Modbus, and also an understanding of LonWorks network variables. You will need to obtain a copy of the documentation of SNVT types from LonMark (www.lonmark.org) in order to have any success in creating the LonWorks side of the gateway configuration.

For each network variable you wish to add, click Append NV to add one at the end of the list, or Insert NV to add one immediately before the selected line on the list (if any).

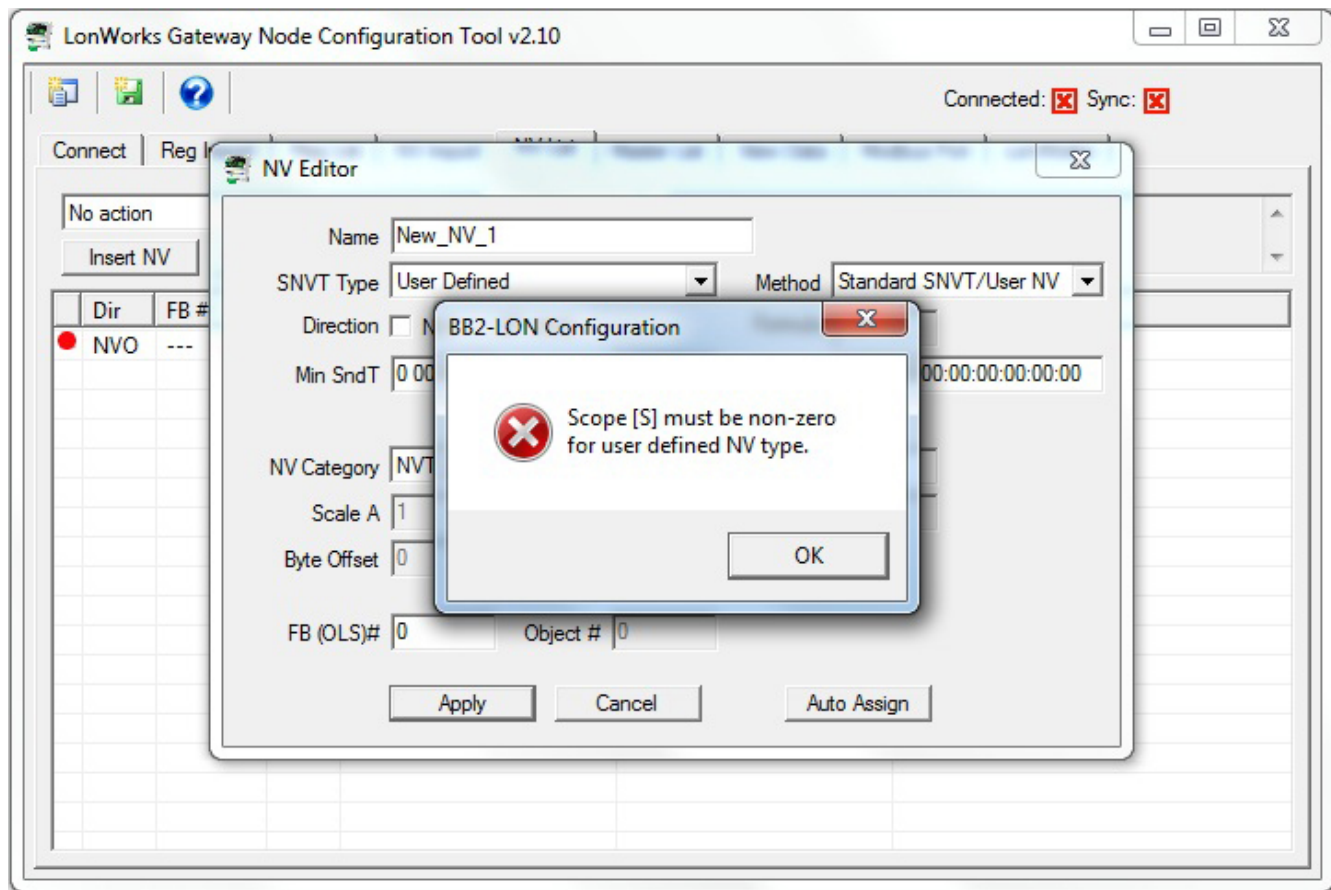


The newly inserted NV will default to SNVT_count. If you wish to change it, double click on the line to be changed, and the NV Editor dialog will appear. In the example that follows, we will complicate things to the maximum by creating a user defined structured network variable.

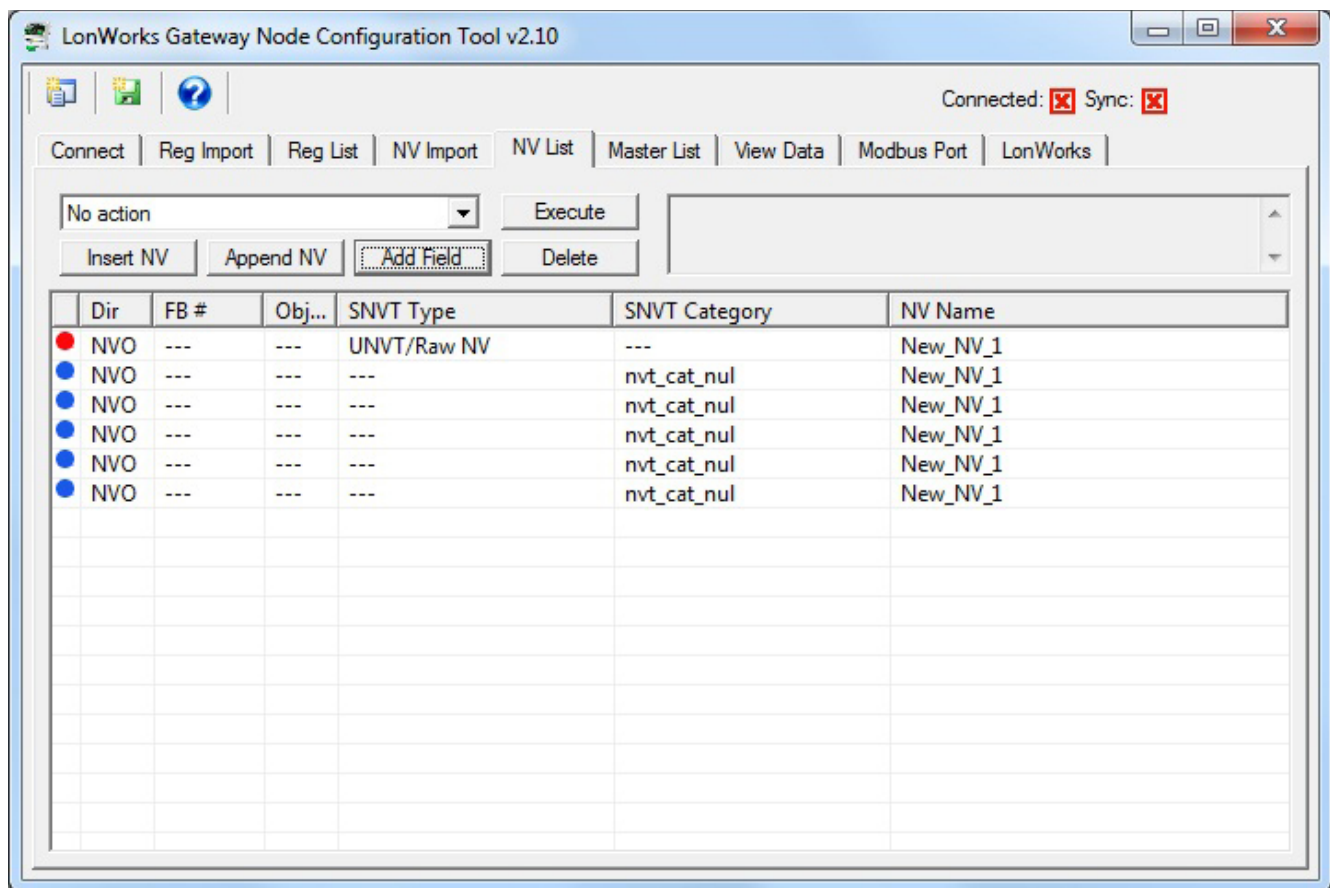
When creating a structure, the NV needs to be set to NV Category NVT_CAT_STRUCT and the NV Size needs to be set to the number of bytes that make up this variable in the LonWorks device. You also need to change the Scope [S] and PID (program ID) to something unique that defines the manufacturer of the device you are conforming to.



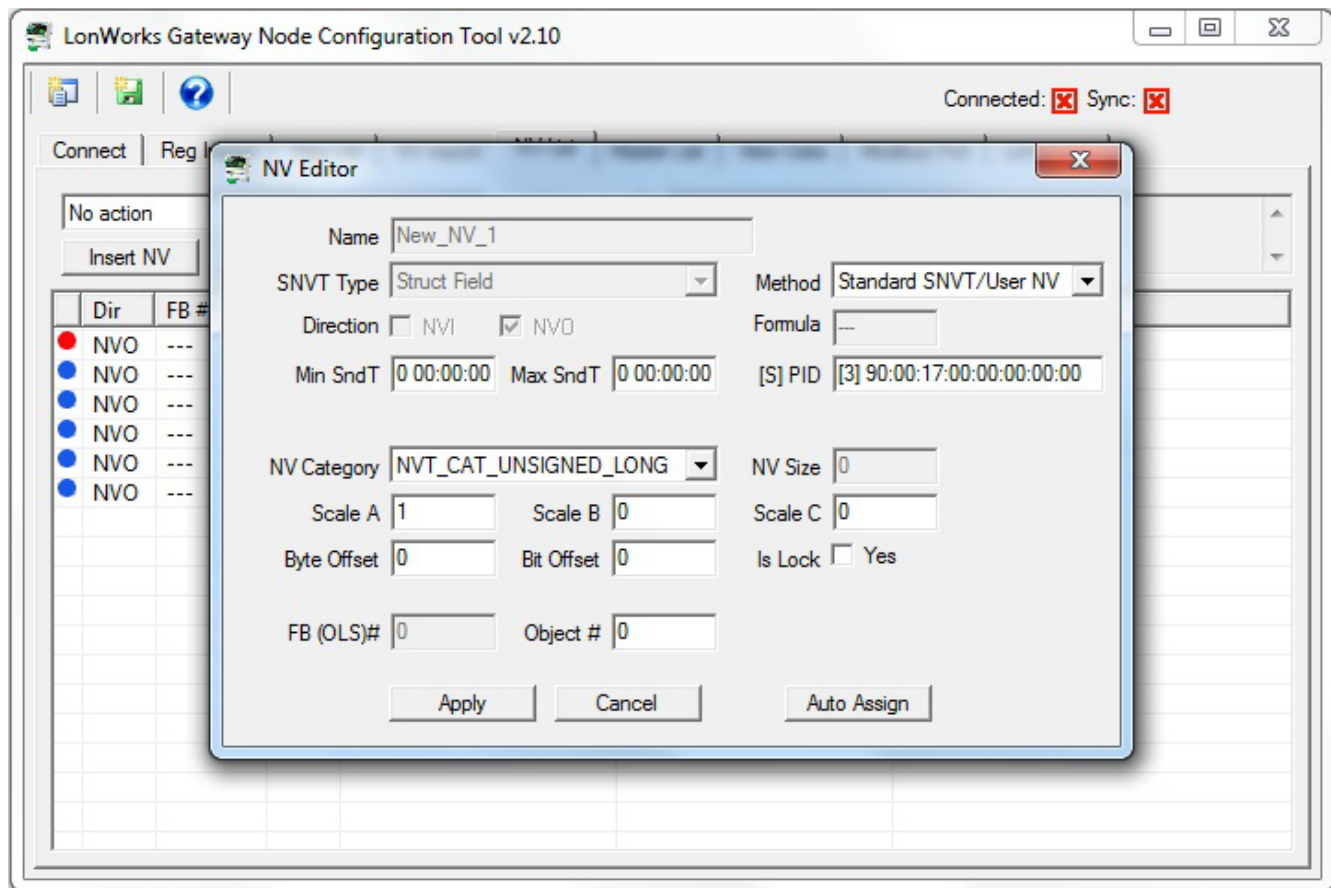
If you do not change the scope, most commonly to 3 for 'manufacturer defined', you will get an error message and not be allowed to proceed. If the scope remains at zero, then the gateway (and all other LonWorks devices on the network) will attempt to interpret your NV as a standard LonMark type and data conversion results will be wrong.



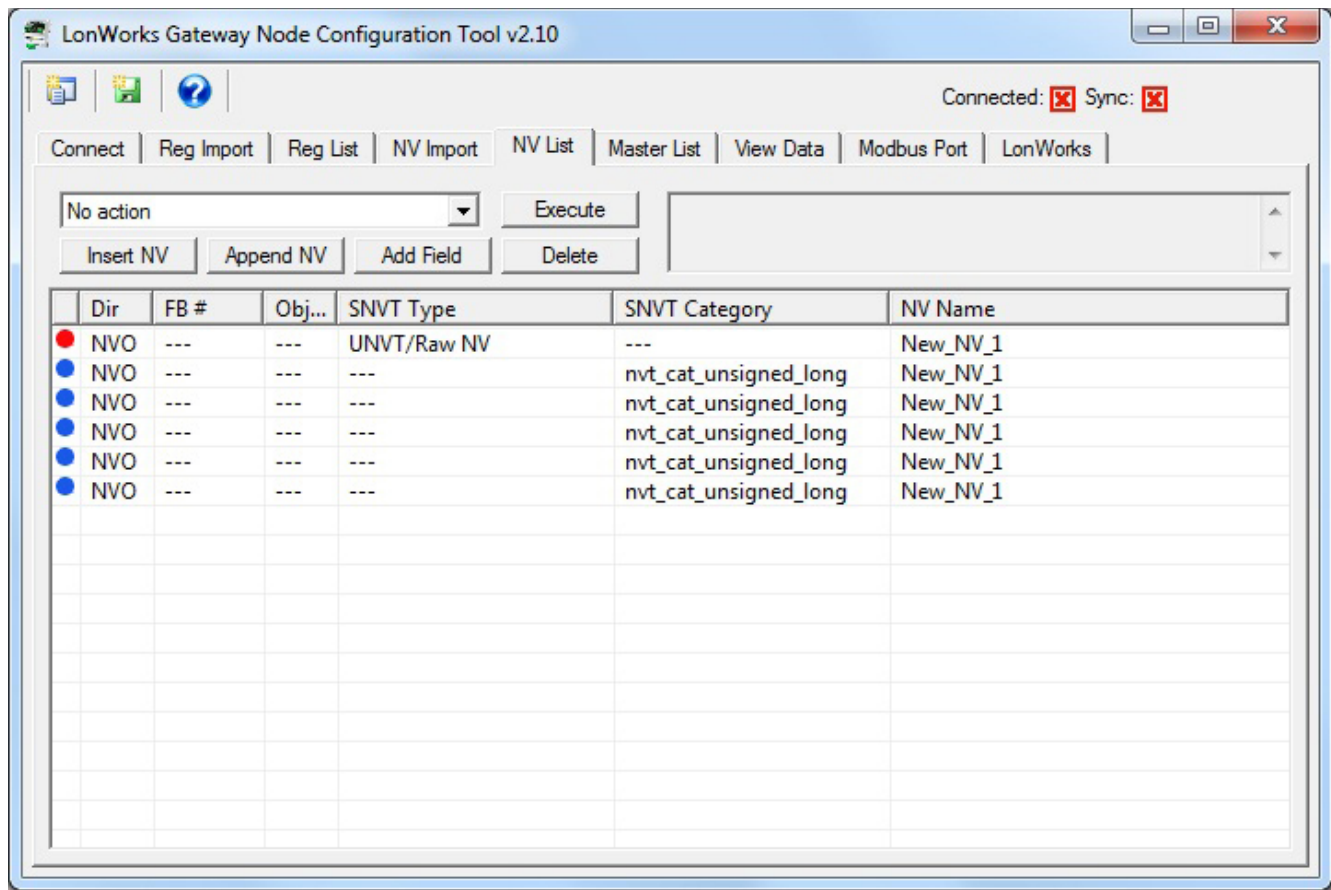
After defining the NV as a structure, you next need to add fields to the structure. Do this by clicking Add Field.



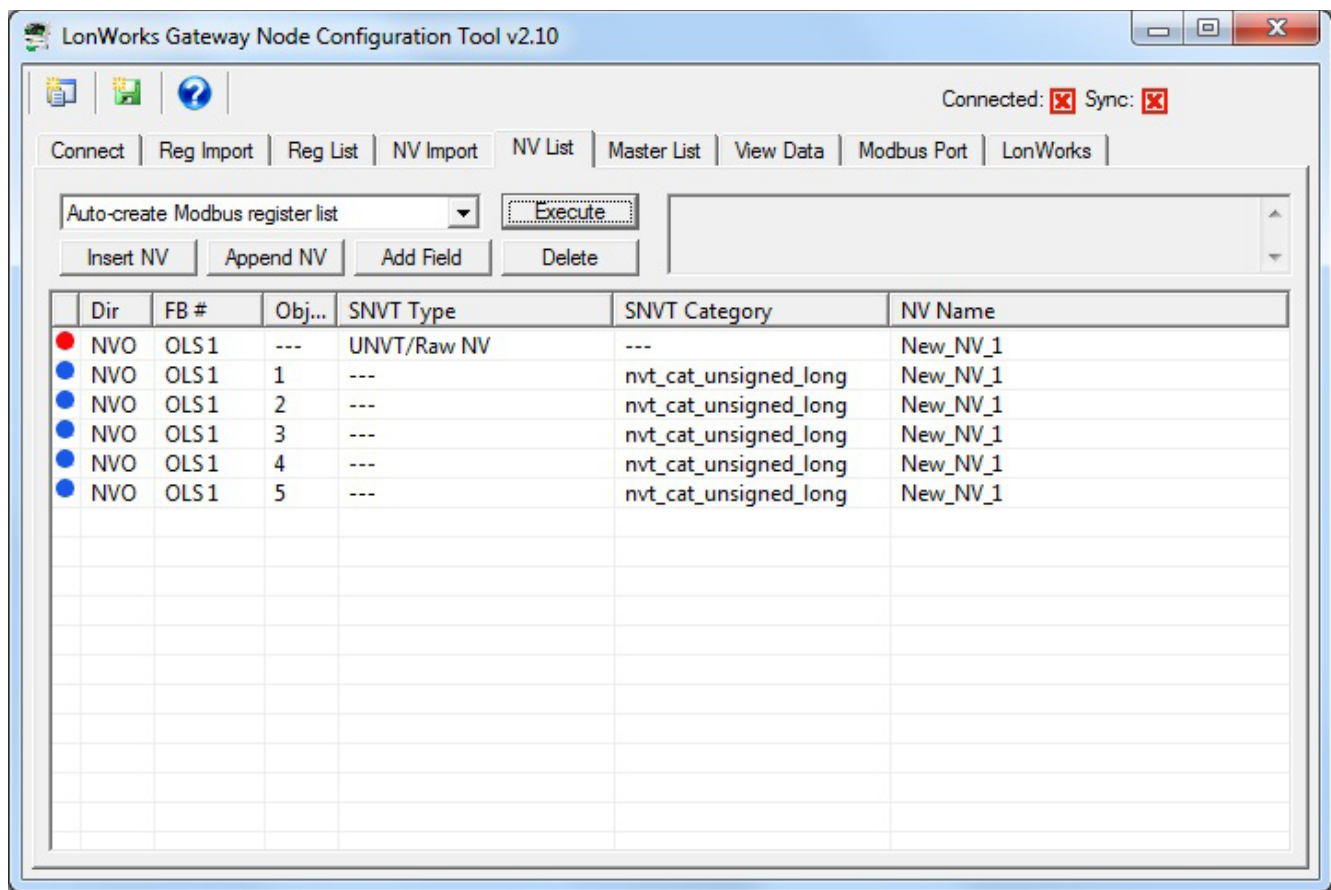
Next, edit each field by double clicking on that line. Specify the data type for the field. It is also important to specify the Byte Offset in the structure. An offset of zero means this field occupies the first byte(s) of the structure. You also need to enter the scale values. These follow the LonMark definition of scale.



Following configuration of the NV and its fields, your NV List will appear as follows for this first single NV that will map to five Modbus registers.



If you have no further network variables to add, proceed with the auto-build of the rest of the configuration. First execute "Auto-assign new NV function blocks". Then execute "Auto-assign new data objects". Finally, execute "Auto-create Modbus register list". Upon completion of these steps, the NV List would appear as follows for this example.



8.4 Using the NV Editor

The layout of the NV Editor will be the same in all cases, but certain windows in the dialog will be set to 'read-only' in some cases depending on what is being configured.

The following example illustrates the most simple of the NV Editor operations, namely just selecting a different standard SNVT type from the list of SNVT Types. When a standard LonMark SNVT type is selected, most of the configuration parameters will be fixed by LonMark and set to read-only in the editor dialog. Click Apply to accept the changes, or Cancel to discard.

Do not click Auto Assign if you are setting up your NV list for the first time. Create and configure all of your network variables (NV's) first. Then from the NV List, execute the auto-assign functions noted above. You should only use the Auto Assign button to assign function block and object numbers when adding a new NV to an existing configuration where most objects have already been assigned.

The NV Editor dialog box is shown with the following settings:

- Name: New_NV_1
- SNVT Type: SNVT_temp
- Method: Standard SNVT/User NV
- Direction: ☐ NVI, ☒ NVO
- Formula: ---
- Min SndT: 0 00:00:00, Max SndT: 0 00:00:00
- [S] PID: [0] 00:00:00:00:00:00:00
- NV Category: NVT_CAT_SIGNED_LONG
- NV Size: 2
- Scale A: 1, Scale B: -1, Scale C: -2740
- Byte Offset: 0, Bit Offset: 0
- Is Lock: ☐ Yes
- FB (OLS)#: 0, Object #: 0
- Buttons: Apply, Cancel, Auto Assign

If you were to create a 'user defined' NV that provided the exact same results as SNVT_temp, the NV Editor dialog would appear as follows:

The NV Editor dialog box is shown with the following settings:

- Name: New_NV_1
- SNVT Type: User Defined
- Method: Standard SNVT/User NV
- Direction: ☐ NVI, ☒ NVO
- Formula: ---
- Min SndT: 0 00:00:00, Max SndT: 0 00:00:00
- [S] PID: [3] 90:00:17:00:00:00:00
- NV Category: NVT_CAT_SIGNED_LONG
- NV Size: 2
- Scale A: 1, Scale B: -1, Scale C: -2740
- Byte Offset: 0, Bit Offset: 0
- Is Lock: ☐ Yes
- FB (OLS)#: 0, Object #: 0
- Buttons: Apply, Cancel, Auto Assign

The SNVT_switch example is illustrated below. The LonMark definition of a "switch" actually contains two elements of data, a state and a level (as would be used for a dimmer switch). You cannot correctly control anything via a SNVT_switch without properly dealing with both elements of data. In Modbus terms, this would require two Modbus registers to control one switch, and this is not typically desirable from a Modbus point of view. Therefore, the Babel Buster gateway provides a "special conversion" such that a single Modbus register containing a value from 0 to 100 (implied percent) will result in both parts of the SNVT_switch being set correctly to control on/off (100% or 0%) or any level in between.

Your selection of whether to use the special conversion will decide whether SNVT_switch maps to one or two Modbus registers, and therefore whether SNVT_switch maps to one or two data objects. For this reason, it is important that you make all of your NV related selections and configurations BEFORE assigning function blocks and data objects.

The NV Editor dialog box is shown with the following settings:

- Name: nviDiscreteOut_1
- SNVT Type: SNVT_switch
- Method: Special Conversion
- Direction: ☐ NVI, ☒ NVO
- Formula: 1
- Min SndT: 0 00:00:00, Max SndT: 0 00:00:00
- [S] PID: [0] 00:00:00:00:00:00:00
- NV Category: NVT_CAT_STRUCT
- NV Size: 2
- Scale A: 0, Scale B: 0, Scale C: 0
- Byte Offset: 0, Bit Offset: 0
- Is Lock: ☐ Yes
- FB (OLS)#: 0, Object #: 0

Buttons at the bottom: Apply, Cancel, Auto Assign.

Creating a user defined structured network variable is illustrated below. The parent NV must have SNVT type set to User Defined. The Scope [S] and Program ID PID must be non-zero. Scope of 3 means manufacturer defined. Refer to LonMark standards for additional information about scope values.

The NV Category must be NVT_CAT_STRUCT and the NV Size must be set to the total number of bytes of data that this NV will occupy, including all fields of the structure.

The NV Editor dialog box is shown with the following settings:

- Name: New_NV_1
- SNVT Type: User Defined
- Method: Standard SNVT/User NV
- Direction: ☐ NVI, ☒ NVO
- Formula: --
- Min SndT: 0 00:00:00, Max SndT: 0 00:00:00
- [S] PID: [3] 90:00:17:00:00:00:00
- NV Category: NVT_CAT_STRUCT
- NV Size: 10
- Scale A: 1, Scale B: 0, Scale C: 0
- Byte Offset: 0, Bit Offset: 0
- Is Lock: ☐ Yes
- FB (OLS)#: 0, Object #: 0

Buttons at the bottom: Apply, Cancel, Auto Assign.

For each field of the structured NV, select a standard data type for NV Category. The first field will have a byte offset of zero. But all subsequent fields will need to have a non-zero offset so that the data is taken from the right place in the structure. When the network variable (NV) is read from the LonWorks device, or sent to the LonWorks device, it is simply a string of up to 31 bytes of data. Interpreting the bytes correctly requires both a data type, and Byte Offset. Offset of zero is the first byte in the data stream. As an example, if you have two consecutive NVT_CAT_UNSIGNED_LONG fields, the offset for the second one must be 2 since each 'unsigned long' (in LonWorks terms) is 2 bytes. Note that the definition of 'unsigned long' in LonWorks is different than the definition of 'unsigned long' in most computer programming languages such as C.

Special conversions available for certain SNVT's are listed below. The conversion formula is determined automatically based on the SNVT type to which 'Special Conversion' is applied. The table below lists conversion from NV to internal data object value. The process is reversed for converting internal data object value to a standard NV. The conversion is initially made to floating point, but if the internal data object is configured as integer, then the value will be truncated and retained as an integer. The internal data object value is also the value that will be accessed as a Modbus register.

Special conversion formula 1	SNVT_switch	The structure consisting of state and value is converted to a single floating point value in the range of 0.00% to 100.00%.
Special conversion formula 2	SNVT_elapsed_tm	The structure containing 5 fields from days to milliseconds is converted to a single floating point value of seconds.

NV Editor fields are used as follows:

Name	NV name that will be exported to XIF file when saved. Name is limited (LonWorks requirement) to 16 characters and may contain no spaces.
SNVT Type	SNVT type is any of the LonMark standard network variable types. This field may also be set to "User Defined" to work with non-standard variable types including UNVT's or User Network Variable Types.
Direction	Sets Input or Output direction of the Network Variable. This direction setting is only in the NV List, and your selection will be assigned to the appropriate function block type when they are assigned. If an FB# has already been allocated, you cannot change the direction.
Min Snd T	The minimum send time is the minimum amount of time that must elapse between updates of a network variable output (NVO). It does not apply to an NVI. Minimum send time defines the maximum update rate. If set to zero, the feature is disabled. The LonMark standard default is 15 seconds.
Max Snd T	The maximum send time is the maximum amount of time that can elapse between updates. If there has been no data value change to report, the NVO is transmitted anyway when the maximum send time expires. This does not apply to an NVI. If set to zero, the feature is disabled.

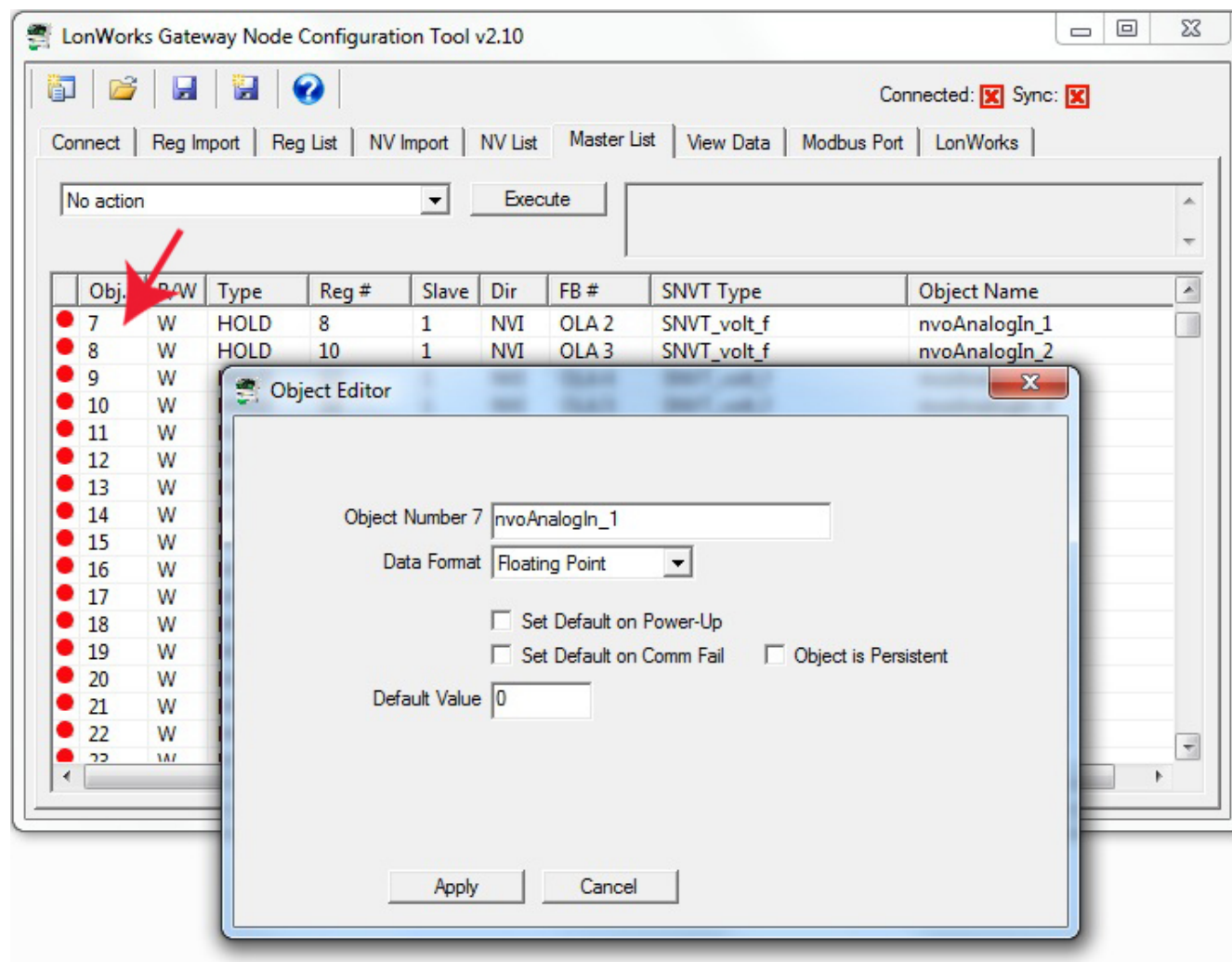
Method	Method will most often be "Standard SNVT/User NV". Other options include "Copy Raw Data" and "Special Conversion". The available special conversions are noted above. "Copy raw" means exactly that - copy bytes of data as they are without any conversion or other treatment.
Formula	Identifies which conversion formula applies when "Special Conversion" is selected as Method.
[S] PID	<p>The [S] is network variable scope. A scope of 0 indicates standard LonMark NV types. A scope of 3 through 6 indicates manufacturer defined NV types.</p> <p>0 – Standard – applies to all devices. 1 – Reserved for future use. 2 – Reserved for future use. 3 – Manufacturer – applies to all devices from the manufacturer specified in the program ID template. 4 – Manufacturer and Device Class – applies to all devices from the manufacturer with the device class specified in the program ID template. 5 – Manufacturer, Device Class, and Device Subclass – applies to all devices from the manufacturer with the device class and device subclass specified in the program ID template. 6 – Manufacturer, Device Class, Device Subclass, and Device Model – applies to all devices of the specified type and manufacturer specified in the program ID template.</p> <p>The PID, consisting of 8 hexadecimal values, is the Program ID of the device or device class for which the UNVT (User Network Variable Type) applies.</p>
NV Category	NV Category defines the data type for non-structured network variables, or declares the NV to be a structure, or enumeration. When used in a field of a structure, "bitfield" is also applicable. Although "array" and "union" are defined by LonMark, they are not generally used in the definitions of standard NV types, and are not processed by the gateway.
NV Size	NV size is the number of bytes of data provided by this NV. If the NV is structured, this is the number of bytes in the entire structure when looking at the parent NV. This is the number of bytes in just the field if looking at a field in a structure.
Scale A, B, C	<p>The scale values are used to convert raw binary data as transmitted over the wire to engineering units as would be displayed in a user interface. The scale values are fixed and defined by LonMark for Standard Network Variable Types. You need to enter them for user defined types.</p> <p>The scaling formula is $S = a \cdot 10^b \cdot (R + c)$ where R is raw data, and S is scaled data.</p>
Byte Offset	Byte offset is only used in structured network variables, and defines, for each field of the structure, where in the string of data bytes this field begins.
Bit Offset	Bit offset is used to select a specific bit when the NV Category is "bitfield". Byte offset is used to specify which byte within a structure the bit should be taken from. Bit offset is 0 to 7, with bit offset of 0 being the most significant bit in the byte. Note that this is backwards from most interpretations of "bit 0". LonWorks interpretation of bit is bit offset from the start of the byte, and start of byte is interpreted as most significant bit.
Is Lock	The Lock applies only to structured variables. When a structured NVO maps to multiple Modbus registers, the NVO will be updated (transmitted on the LonWorks network) any time any part of the structure is updated from the Modbus side. If it is desired that the NVO should only be transmitted as a result of update to a specific field (allowing all parts of the structure to be updated before transmitted), then you want to specify one field of the structure as the "lock". The NVO will only

	be transmitted on the LonWorks network when this field is updated from the Modbus side.
FB #	The function block number that has been assigned will be indicated here, or will be zero if not yet assigned. The direction (NVO or NVI) will determine whether the FB is indicated as OLS (Open Loop Sensor) or OLA (Open Loop Actuator).
Object #	The internal data object number that has been assigned will be indicated here, or will be zero if not yet assigned. This object number is where you will find mapping for the Modbus register associated with the displayed NV entry.

9 Tool 'Master List' Page

9.1 Editing Configuration from Master List Page

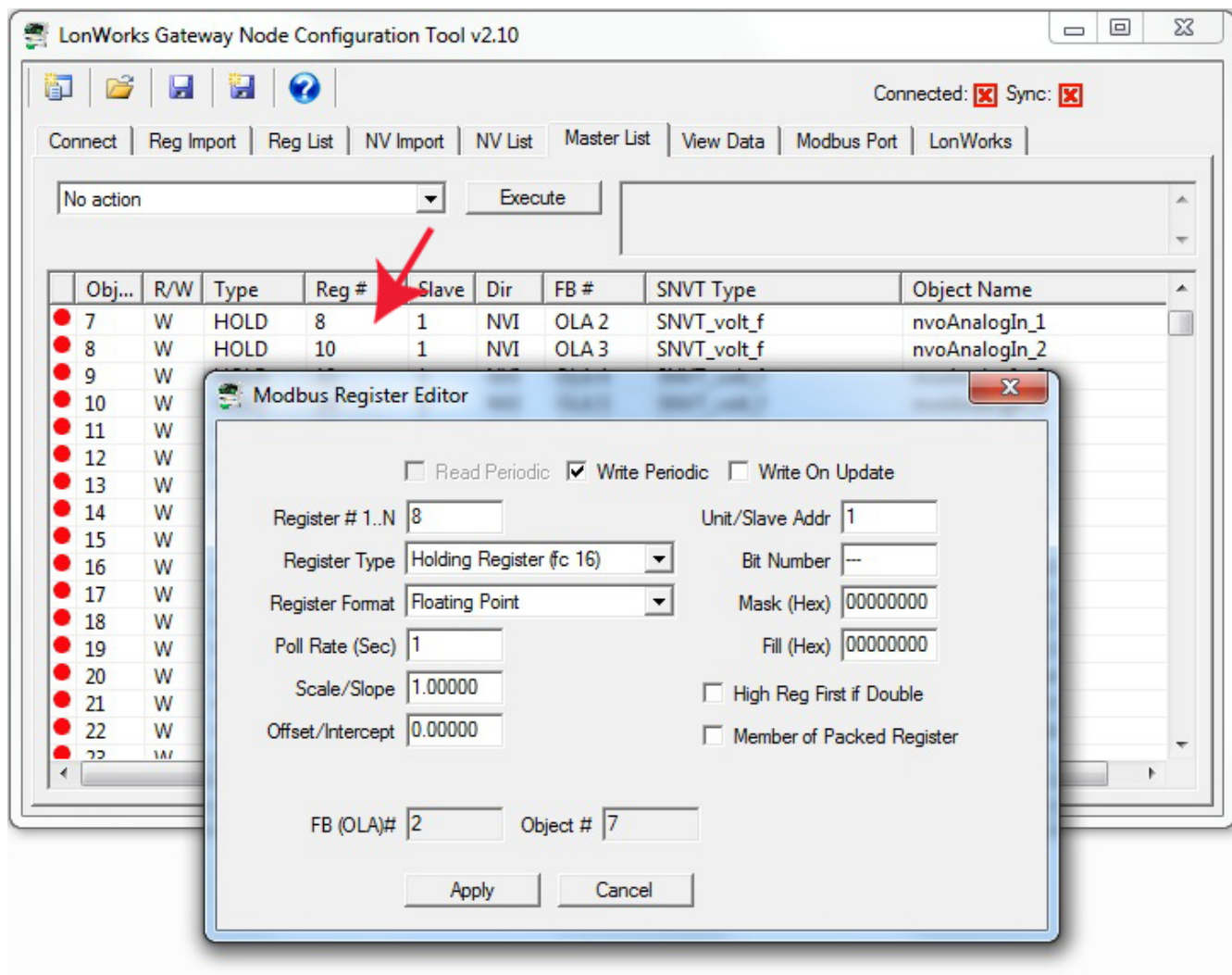
Clicking on the Object number column will open the Object Editor dialog as illustrated below.



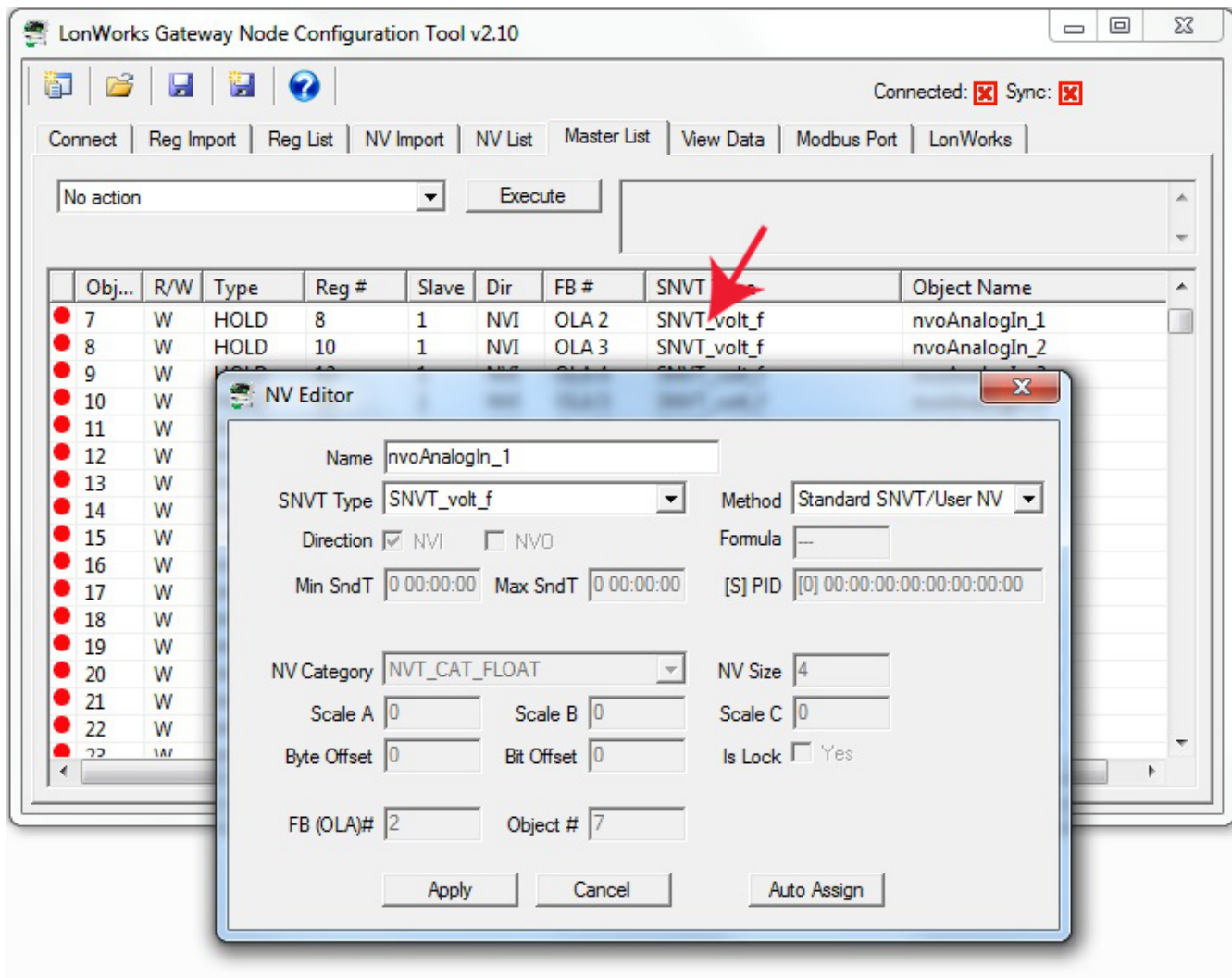
Object Number/Name	The object number you clicked on will be displayed, along with a window where you may enter an object name. This name is primarily for documentation purposes since Modbus cannot read names, and LonWorks can only use the Network Variable name.
Data Format	The internal data format of the object is selected here. If the object is auto-assigned, it will default to a format assumed to be appropriate for the network variable associated with it. Data will be converted as necessary, but if you are reading a floating point value from Modbus and store it as an integer, you will lose some resolution or accuracy in the process.
Default on Power-Up	Select this option if you want the default value indicated on this page to be written into the object at power-up. Depending on how the object is used, this may also result in writing to a Modbus register in an external slave and/or updating a Network Variable Output on the LonWorks side.
	Select this option if you want the object to assume the default value upon

Default on Comm Fail	failure to communicate with the associated Modbus slave device, or upon failure of the external Modbus master to update this object within the host timeout period. The comm fail does not apply to the LonWorks side of the gateway, only to the Modbus side.
Object is Persistent	Select this option if you want the object to retain its most recent data value through power cycles or restarts.
Default Value	This is the default data value that will be used if either of the "Set Default" options are selected above.

Clicking on the Modbus register area of a line on the Master List will open the Modbus Register Editor dialog as illustrated below. This is the same editor described in Section 6 of this user guide. Refer to that section for details about the Modbus Register Editor.



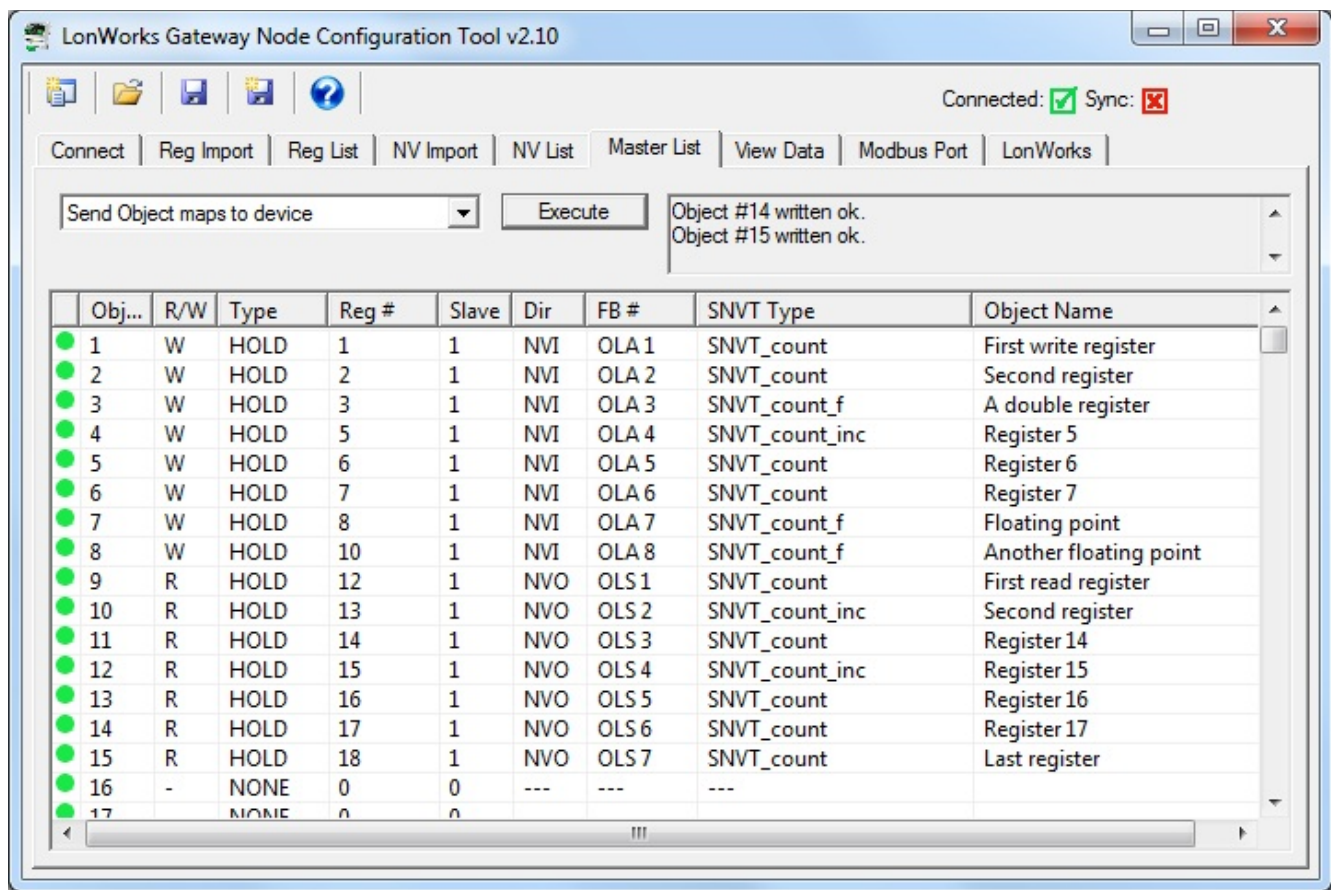
Clicking on the Network Variable area of a line on the Master List will open the NV Editor dialog as illustrated below. This is the same editor described in Section 8 of this user guide. Refer to that section for details about the NV Editor.



9.2 Sending Configuration To Device

Select "Send Object maps to device" to send Object configuration to the gateway. You must first "Send NV definitions to device". There are a total of four pages containing information that must be sent to the gateway to fully configure it.

- NV List (or Master List): Send NV definitions to device
- Master List: Send Object maps to device
- Modbus Port: Write Device
- LonWorks: Set Location



9.3 Getting Configuration From Device

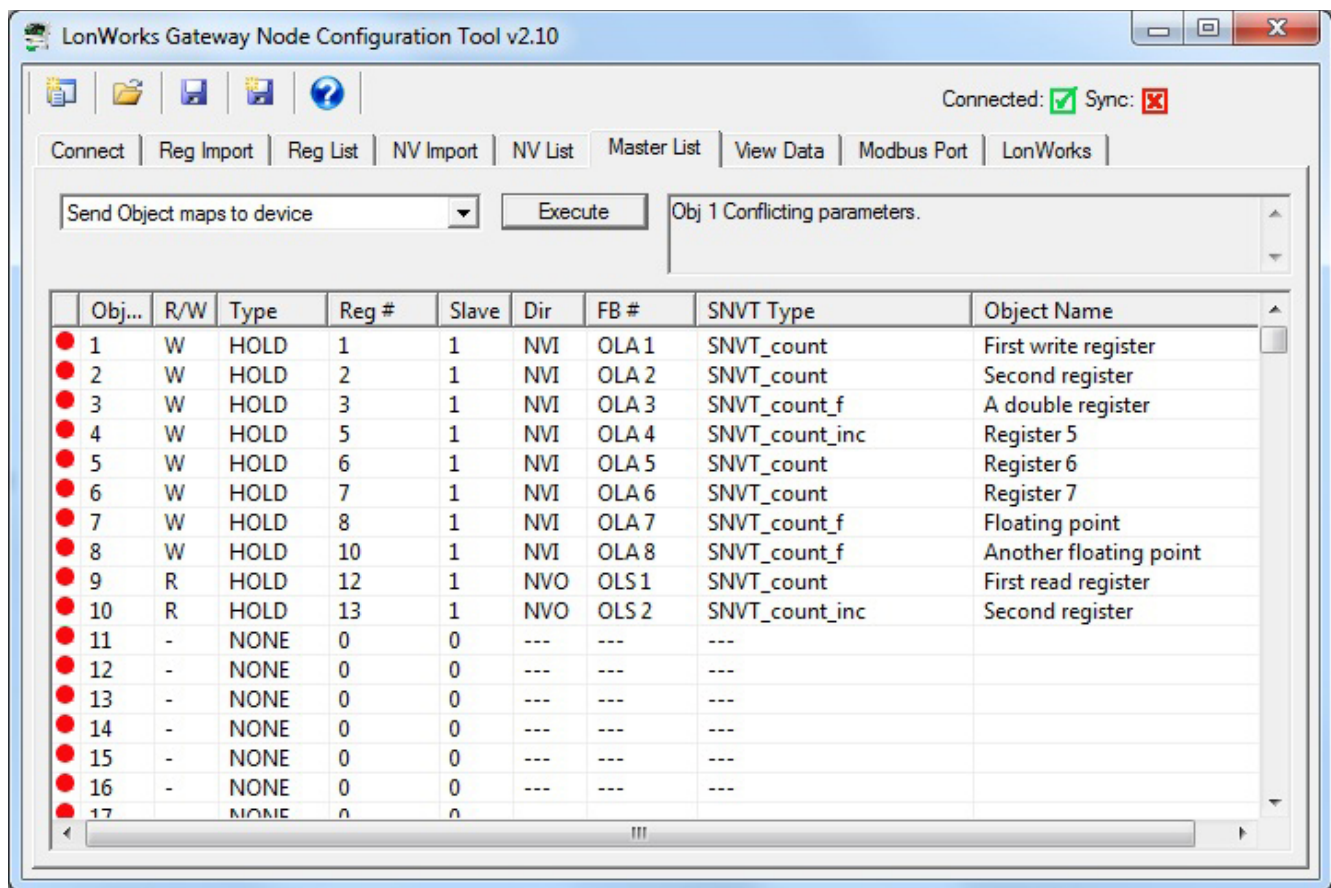
The configuration tool can be used to retrieve all necessary information from a previously configured device. Once retrieved, you can use that information to replicate the same configuration in another device, or simply save the configuration to a file for later use.

There are a total of four pages containing information that must be read from the gateway to obtain the complete configuration.

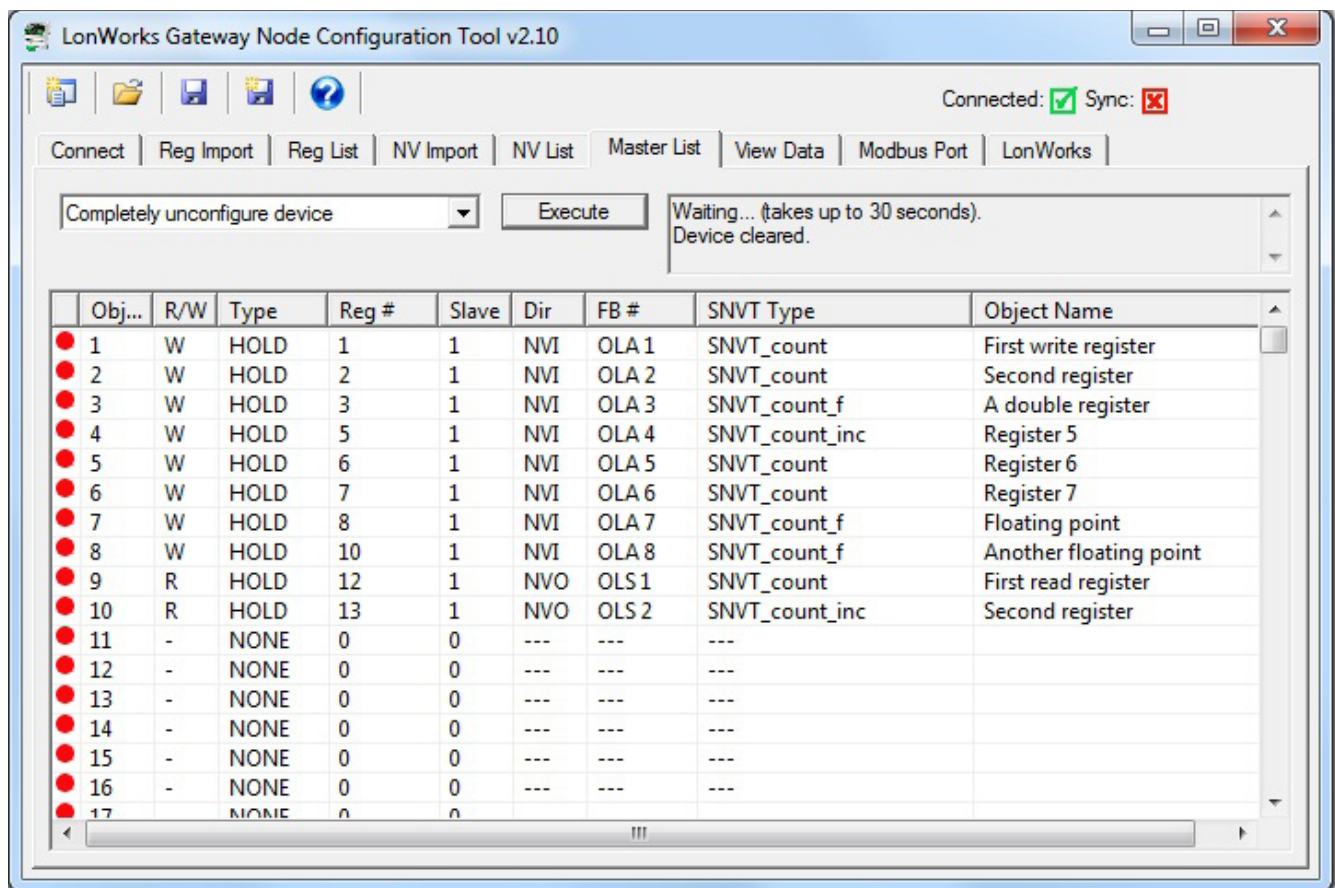
- NV List (or Master List): Get NV definitions from device
- Master List: Get Object maps from device
- Modbus Port: Read Device
- LonWorks: Get Location

9.4 Fixing Conflicts

You may run into the error message illustrated below if making configuration changes to a gateway previously configured. The "Conflicting parameters" means that the attempted object configuration references a network variable that is not configured the way the object is set to expect.



The easiest way to resolve configuration conflicts is to completely clear the device configuration and resend network variables, followed by object maps. When you execute "Completely unconfigure device", there will be a delay of typically 30 seconds while the device reprograms all of its internal non-volatile configuration memory, restoring it to an 'unused' state. Once complete, proceed with sending configuration to the device as noted above.

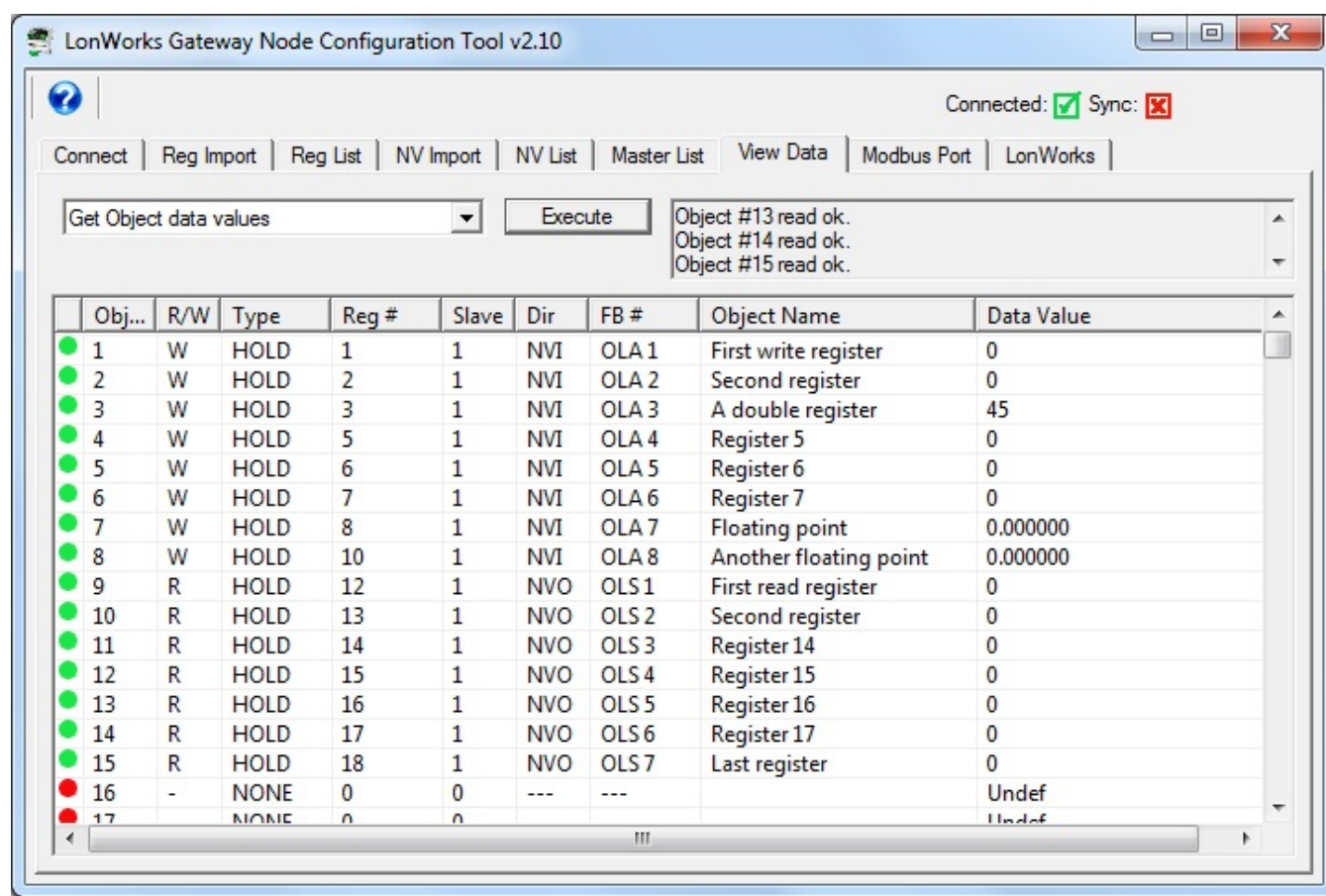


10 Tool 'View Data' Page

10.1 Viewing Object Data

Once the gateway is fully configured and operational, you can look at object values on the View Data page. This page displays the contents of the internal data objects. In most cases, this will also represent what is found in the respective Modbus registers. If there is a one-to-one mapping of object to non-structured network variable, then this page will generally reflect what is found in the respective network variable except that NV scaling to raw data on the LonWorks network will often be different than data displayed here. What is displayed here is most often shown in human readable engineering units.

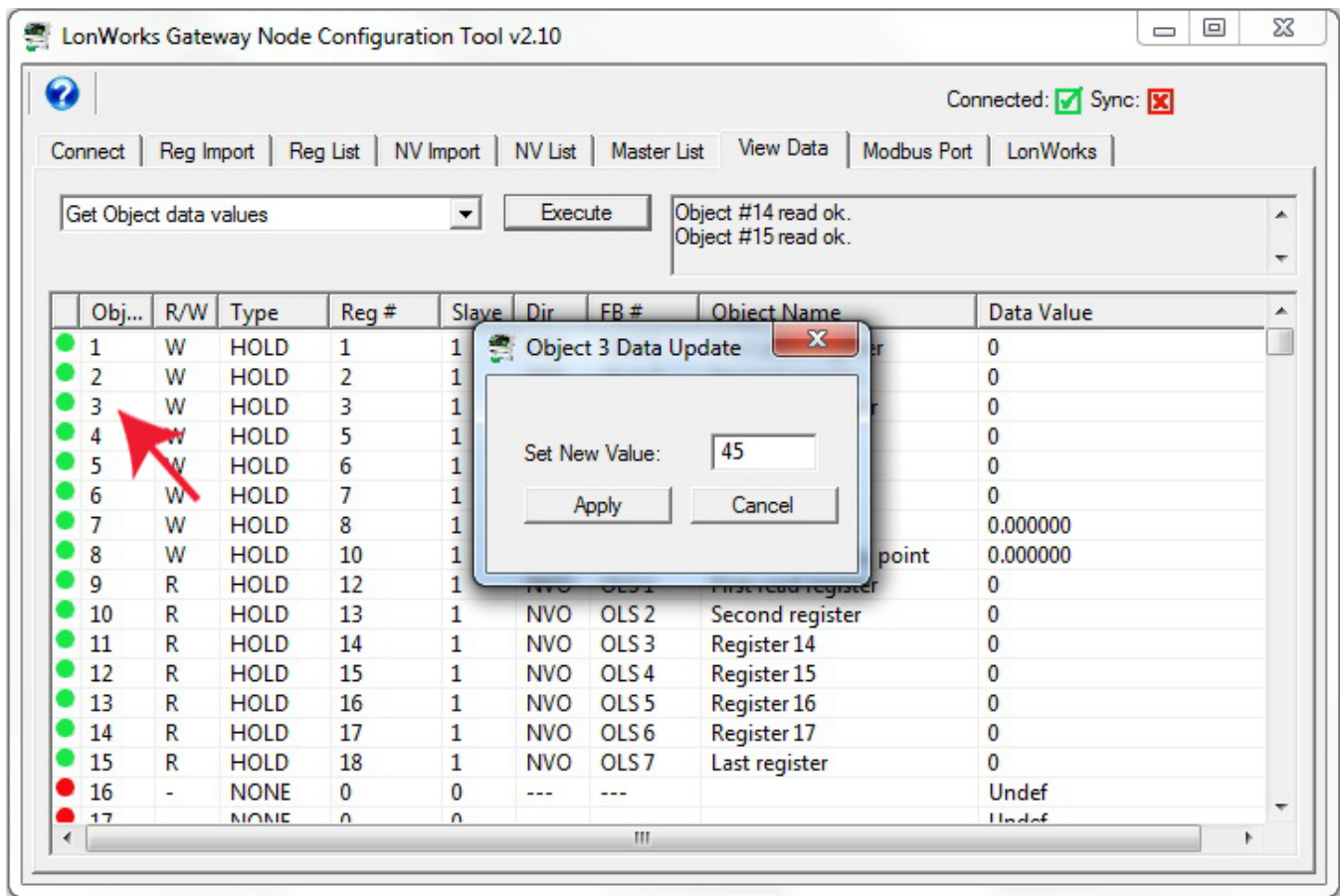
The most user friendly way to view data from the LonWorks side is to use the network management tool's NV browser, as found in Echelon's LonMaker. You can also use Nodeutil to view NV data in raw form via the LonWorks network. The third option is that you can view NV data in raw form via the USB diagnostic console (Appendix A).



10.2 Changing Object Data

Double click on any line on the View Data page to bring up the Data Update dialog. Enter a new value, and click Apply. This will write the new data value into the internal data object. Depending on your configuration, this may result in writing to a Modbus register in an external slave device, and/or may result in writing to a LonWorks Network Variable Output.

You should also be aware that, depending on your configuration, your newly set data value could get immediately overwritten. For example, if the data object is mapped to a Modbus register that is being read every 2 seconds, your value will remain in effect for only up to 2 seconds, until the next time the Modbus data is read and placed into this object.



LonWorks Gateway Node Configuration Tool v2.10

Connected: ☒ Sync: ☐

Connect | Reg Import | Reg List | NV Import | NV List | Master List | View Data | Modbus Port | LonWorks

Get Object data values

Object #15 read ok.
Object #3 written ok.

Obj...	R/W	Type	Reg #	Slave	Dir	FB #	Object Name	Data Value
1	W	HOLD	1	1	NVI	OLA 1	First write register	0
2	W	HOLD	2	1	NVI	OLA 2	Second register	0
3	W	HOLD	3	1	NVI	OLA 3	A double register	45
4	W	HOLD	5	1	NVI	OLA 4	Register 5	0
5	W	HOLD	6	1	NVI	OLA 5	Register 6	0
6	W	HOLD	7	1	NVI	OLA 6	Register 7	0
7	W	HOLD	8	1	NVI	OLA 7	Floating point	0.000000
8	W	HOLD	10	1	NVI	OLA 8	Another floating point	0.000000
9	R	HOLD	12	1	NVO	OLS 1	First read register	0
10	R	HOLD	13	1	NVO	OLS 2	Second register	0
11	R	HOLD	14	1	NVO	OLS 3	Register 14	0
12	R	HOLD	15	1	NVO	OLS 4	Register 15	0
13	R	HOLD	16	1	NVO	OLS 5	Register 16	0
14	R	HOLD	17	1	NVO	OLS 6	Register 17	0
15	R	HOLD	18	1	NVO	OLS 7	Last register	0
16	-	NONE	0	0	---	---		Undef
17	-	NONE	0	0	---	---		Undef

11 Tool 'Modbus Port' Page

11.1 Modbus RTU Port Settings

The Modbus Port page will change in appearance based on whether you are configuring Modbus RTU or Modbus TCP. When configuring TCP, additional information is displayed regarding IP addresses.

Modbus registers that will be queried by the gateway when it is functioning as master are those listed on the Reg List page. Modbus registers available to an external master when the gateway is functioning as slave are listed in Appendix E.

The Modbus port parameters that must be set for Modbus RTU operation are as follows.

Master/Slave Mode	Check the applicable box to select desired mode of operation. In most cases, the Babel Buster gateway will be Modbus Master. The mode will change when Write Device is clicked. (It is suggested that you make all applicable changes first, then click Write Device.)
Baud Rate	Standard baud rates up to 38400 are supported.
Character Format	Modbus RTU is always required to be 8 data bits. The parity and number of stop bits may be selected here.
Timeout	Timeout applies only when the gateway is operating as Modbus master. This is the amount of time (in seconds, fractions to tenths supported) that the gateway will wait for the slave device to respond. If the gateway has not received a response within this time, it is counted as a timeout and the no-response error status is indicated.

	Note that if Timeout when Master is zero, you will get nothing but 'no response' errors because the master is not waiting at all for any response from the slave device.
Address	Address applies only when the gateway is operation as Modbus slave. This is the address to which the gateway will respond on the RTU network.
Pre-Delay	Pre-Delay is the amount of time that the RS485 transmitter will be online before the start of the first character. It also provides a delay between queries in the event the gateway is too fast for other Modbus devices on the network. In most cases, some pre-delay is required, and experience has shown that 50 mS is a universal value that usually works.
High Reg First	<p>Registers containing data longer than 16 bits are actually multiple registers treated as a single data value. In the case of 32-bit data values (32-bit integer or floating point), two consecutive registers are used. However, the order in which the registers appear in the register map is not standardized. Selection of the register ordering is done in the register maps (see Section 6) when the gateway is operating as Modbus master. When the gateway is a Modbus slave, then the register ordering is set here and it applies to all slave registers in the gateway.</p> <p>Select "high reg first" if the most significant data is to be provided in the first register, or lowest numbered register. If there is any confusion about this, the data will be scrambled into a result that is usually so far off, it will be obvious. If your data does not make any sense, try changing this "high reg first" setting and see if you get better results. Remember that this only applies to 32-bit integer or floating point values. If your data is wrong for a 16-bit integer, your problem lies elsewhere.</p>

11.2 Modbus TCP Port Settings

There only RTU settings that also apply to TCP are the Timeout when Master, and "Slave has high reg first" selection. Modbus TCP is both master and slave at the same time, so it is not necessary to select one or the other. Modbus TCP in the BB2-6020 gateway will respond to any and all unit numbers, so Address when Slave is not needed.

Note that as a slave, the BB2-6020 will respond to any unit number. However, as master, the unit number is used to look up the IP address of the slave. Unit number is listed on the Reg List page as slave address.

Configuring the BB2-6020 for Modbus TCP requires that you provide an IP address for the gateway itself, along with the applicable subnet mask as illustrated below. The default TCP port for Modbus TCP is 502.

Gateway can be 0.0.0.0 if all slave devices are on the same subnet. If you will be accessing Modbus TCP devices on another network, you need to provide a gateway IP address. This IP address is expected to be a NAT router if you will be attempting to access devices on another network.

IMPORTANT: The TCP gateway will default to having an IP address of 10.0.0.101 as shipped from the factory. After changing the IP address to whatever works on your network, you will need to reboot the server before the new IP address will take effect.

LonWorks Gateway Node Configuration Tool v2.10

Connected: ☒ Sync: ☒

Connect | Reg Import | Reg List | NV Import | NV List | Master List | View Data | Modbus Port | LonWorks

This device is Modbus ☐ Slave ☒ Master

Read Device Write Device ☒

Modbus Port Baud Rate: 19200
 Modbus Character: 8 Data, No Parity, 1 Stop
 Timeout when Master (S): 0.5
 Address when Slave: 0
 Pre-Delay (mS): 50
☐ Slave has high reg first

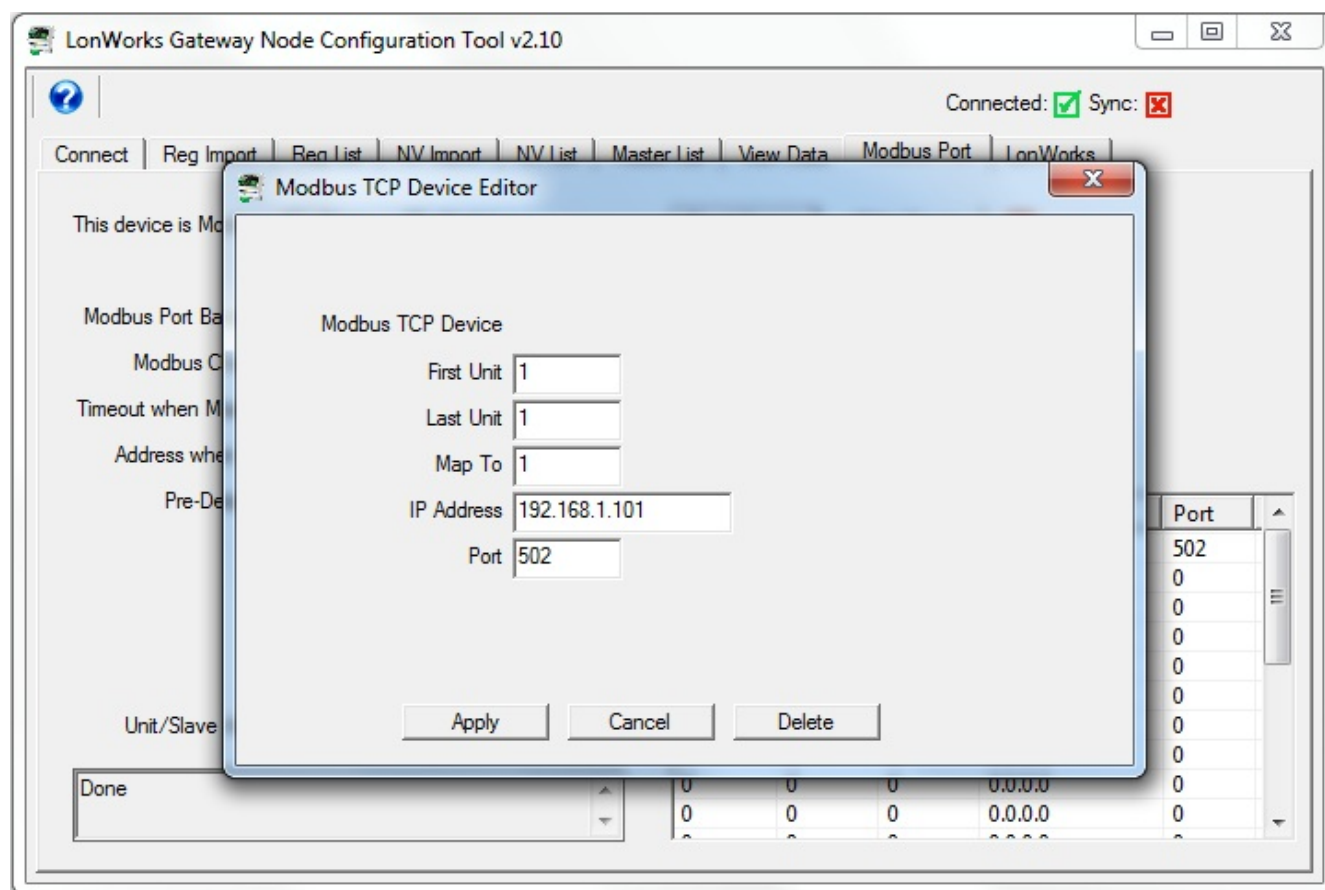
IP Address: 192.168.1.148
 Subnet Mask: 255.255.255.0
 Gateway: 192.168.1.1
 TCP Port: 502

First U...	Last U...	Map To	IP Address	Port
1	1	1	192.168.1.101	502
0	0	0	0.0.0.0	0
0	0	0	0.0.0.0	0
0	0	0	0.0.0.0	0
0	0	0	0.0.0.0	0
0	0	0	0.0.0.0	0
0	0	0	0.0.0.0	0
0	0	0	0.0.0.0	0
0	0	0	0.0.0.0	0
0	0	0	0.0.0.0	0
0	0	0	0.0.0.0	0

Unit/Slave Address: 1 Get Error Info

Done

In addition to setting the IP address of the gateway itself, you need to provide an IP address of at least one Modbus TCP slave if the BB2-6020 will be functioning as a master. Double click on a line in the IP address list, and enter unit numbers and IP address as illustrated below. If you have only one TCP device, and will refer to it as unit 1, then enter just 1 for all three unit number entries.



11.3 Modbus TCP Device Mapping

You have the option of mapping multiple TCP devices (up to 20), and you can remap the unit numbers that will be used. The unit number given as 'First Unit' will be translated to the 'Map To' number in the query sent to the IP address on that line. If the Last Unit is greater than First Unit, then this denotes the range of units that will be translated. If First is 2 and Last is 5, then units (slave addresses) 2 through 5 as shown on the Reg List will be translated to units 1 through 4 in queries sent to that IP address.

It is common for power meters to have multiple unit numbers at the same IP address. However, most Modbus TCP devices will only have a single unit number, and in many cases will disregard unit number altogether since use of unit was once considered optional in Modbus TCP. Even if the Modbus TCP slave you are querying does not care about unit number, you still need to use a unit number (slave address) to look up its IP address.

LonWorks Gateway Node Configuration Tool v2.10

Connected: ☒ Sync: ☒

Connect | Reg Import | Reg List | NV Import | NV List | Master List | View Data | Modbus Port | LonWorks

This device is Modbus ☐ Slave ☒ Master

Read Device Write Device ☒

Modbus Port Baud Rate: 19200
 Modbus Character: 8 Data, No Parity, 1 Stop
 Timeout when Master (S): 0.5
 Address when Slave: 0
 Pre-Delay (mS): 50
☐ Slave has high reg first

IP Address: 192.168.1.148
 Subnet Mask: 255.255.255.0
 Gateway: 192.168.1.1
 TCP Port: 502

First U...	Last U...	Map To	IP Address	Port
1	1	1	192.168.1.101	502
2	5	1	173.11.32.89	502
6	9	1	173.11.32.91	502
0	0	0	0.0.0.0	0
0	0	0	0.0.0.0	0
0	0	0	0.0.0.0	0
0	0	0	0.0.0.0	0
0	0	0	0.0.0.0	0
0	0	0	0.0.0.0	0
0	0	0	0.0.0.0	0
0	0	0	0.0.0.0	0
0	0	0	0.0.0.0	0

Unit/Slave Address: 1 Get Error Info

Done

11.4 Get Error Info

The Babel Buster gateway keeps error counts for each slave device. The counts increment for CRC and no-response (timeout) errors. The most recent exception code is retained for exceptions. Click Get Error Info to retrieve error information, which will be displayed in the log window just below the button. Errors are self-resetting. If errors are indicated, that indication will go away when the error is resolved.

12 Tool 'LonWorks' Page

12.1 Viewing ID Information

Click the Get ID's button to read the Babel Buster gateway's program ID and Neuron ID. The program ID should always come back as 90:00:17:47:1E:84:04:01 where the only potential variation is that the last field, '01', may have been altered. If you get any other result, confirm that you are connected to a BB2-2010, BB2-2011, or BB2-6020.

There is no restriction on what the Neuron ID might be. It is displayed here for reference just in case you are trying to correlate traffic in the LonScanner protocol analyzer.

LonWorks Gateway Node Configuration Tool v2.10

Connected: ☒ Sync: ☐

Connect | Reg Import | Reg List | NV Import | NV List | Master List | View Data | Modbus Port | LonWorks

Program ID: 90:00:17:47:1E:84:04:01 | Alter Prog ID: 1

Neuron ID: 07:00:09:51:19:00 | Get ID's

Node Location: Subsystem 1 | Get Location | Set Location ☒

	Subnet	Node	Length	Domain ID	
Domain 0	1	3	1	80	Set Domain 0
Domain 1	---	---	0		Set Domain 1

Get Domains

Domain 0 read
Domain 1 read

12.2 Node Location

The location string provided by the device's Node Object may be read using Get Location, or written using Set Location. The location string will often get overwritten by the network management tool during network installation.

12.3 Node Domains

You have the option of viewing current domain settings using the Get Domains button, or setting the domains with the respective Set Domain buttons. In most cases, it is not necessary that you set the domains ahead of time. The network management tool will usually overwrite them anyway during installation.

Important note about being able to discover nodes on the network:

Being able to discover the gateway on the network, and the network management tool's ability to install it, depends on the device having at least one domain table entry set for a zero length domain. Without a zero length domain, the network management tool cannot query the device (in our case, the Babel Buster gateway). If your network management tool is unable to communicate with the Babel Buster, and you see here that both domains have a length greater than zero, change one of them to be zero length again.

12.4 Changing Program ID

You should normally have no reason to change the program ID. However, if you are pre-configuring multiple Babel Buster gateways with network variables that are not the same from one gateway to the next, this will look like different "program interfaces" to the network management tool. Another way of stating this is that the XIF files produced for the gateways will not be the same.

When the XIF file or program interface does not define the exact same set of network variables from one gateway to the next, you will need to create additional program ID's to allow the network management tool to see them as unique different devices. Enter a number between 21 and 255 in the ID alter window, and click Alter Prog ID. You will need to restart the gateway (power cycle) before the new ID will take effect.

Note that ID numbers 1 through 20 are reserved for Control Solutions standard gateway ID numbers. User defined program ID numbers may be 21 through 255. Note that the input window accepts a decimal number even though the complete program ID is displayed as hexadecimal.

Appendix A Diagnostics via USB Console

A.1 Connecting to Console

The USB connection to the Babel Buster gateway emulates a serial port. You can access the USB console via the Connect page. Enter commands in the command window and click Send. The result will be displayed in the log window below the command window.

If you will be using the USB console more extensively, it may be more convenient to connect via a terminal program like PuTTY (free download) or HyperTerminal (included with Windows until Win7, now you need to download it). Find the correct COM port number using your PC's device manager, and configure your terminal program to talk to that port.

Note: If the gateway is power cycled or restarted, you will most often need to disconnect USB, close the terminal program (or Control Solutions configuration software), then reconnect USB and restart your program.

A.2 Commands

Commands that you may type in using the USB console are as follows:

pr [n] - read point data, n=1..400 (e.g. *pr 1*)
prv [n] - read virtual point, n=register per Appendix E (e.g. *prv 8001*)
pw [n v] - write point data, n=1..400, v=data value (e.g. *pw 1 44.5*)
pwv [n v] - write virtual point, n=register per Appendix E (e.g. *pwv 8001 0*)
pc [n] - view point configuration for object 'n' (e.g. *pc 1*)
pclist [n1 n2] - view point configuration for list in range (e.g. *pclist 3 10*)
ps [n] - show point status for object t'n' (e.g. *ps 1*)
plist - list points - one line summary per defined point
p - alias for 'plist'

cver - list firmware version

mod set reg [n] - set Modbus target register number
mod set type [0X | 1X | 3X | 4X | 0XS | 4XS] - set Modbus target register type
mod set format [FP | U32 | S32 | U16 | S16 | BIT] - set Modbus target data format
mod set slave [n] - set Modbus slave address that Master will poll
mod set ? - show Modbus target settings

mparm - show Modbus port settings (use tool to set configuration)

merr [n] - show error codes for Modbus slave ID n

mod read - read Modbus target and display reply (refer to mod set commands above)

mod write [v] - write data to Modbus target, data 'v' is single float or integer value

mod raw 01 03 00 01 00 01 - send raw Modbus packet, pause, and show reply raw

nvr [n] - read NV at index 'n' (decimal), size determined automatically.

nvw [n b1 b2 ...] - write NV at index 'n' (decimal) with hex bytes given.

Sensor NVO 1 is NV index 3, Actuator NVI 1 is NV index 123.

nvot [n] - displays index and SCPTnvType for NVO at sensor FB 'n', n=1..120.

nvit [n] - displays index and SCPTnvType for NVI at actuator FB 'n', n=1..120.

nid - displays Neuron ID

npid - displays device's Program ID

nstate - displays state of node, single value as follows:

0: Invalid, Echelon use only

1: Invalid, Echelon use only

2: Has application, unconfigured

- 3: Applicationless, unconfigured
- 4: Configured, online (normal operating state)
- 5: Invalid, Echelon use only
- 6: Hard offline
- 7: Invalid, Echelon use only
- 0x0C: Configured, soft-offline
- 0x8C: Configured, in bypass mode

The following commands are available via USB console only with the Modbus TCP version:

sver - displays version of firmware running in TCP server

smap [n] - displays TCP device map #n

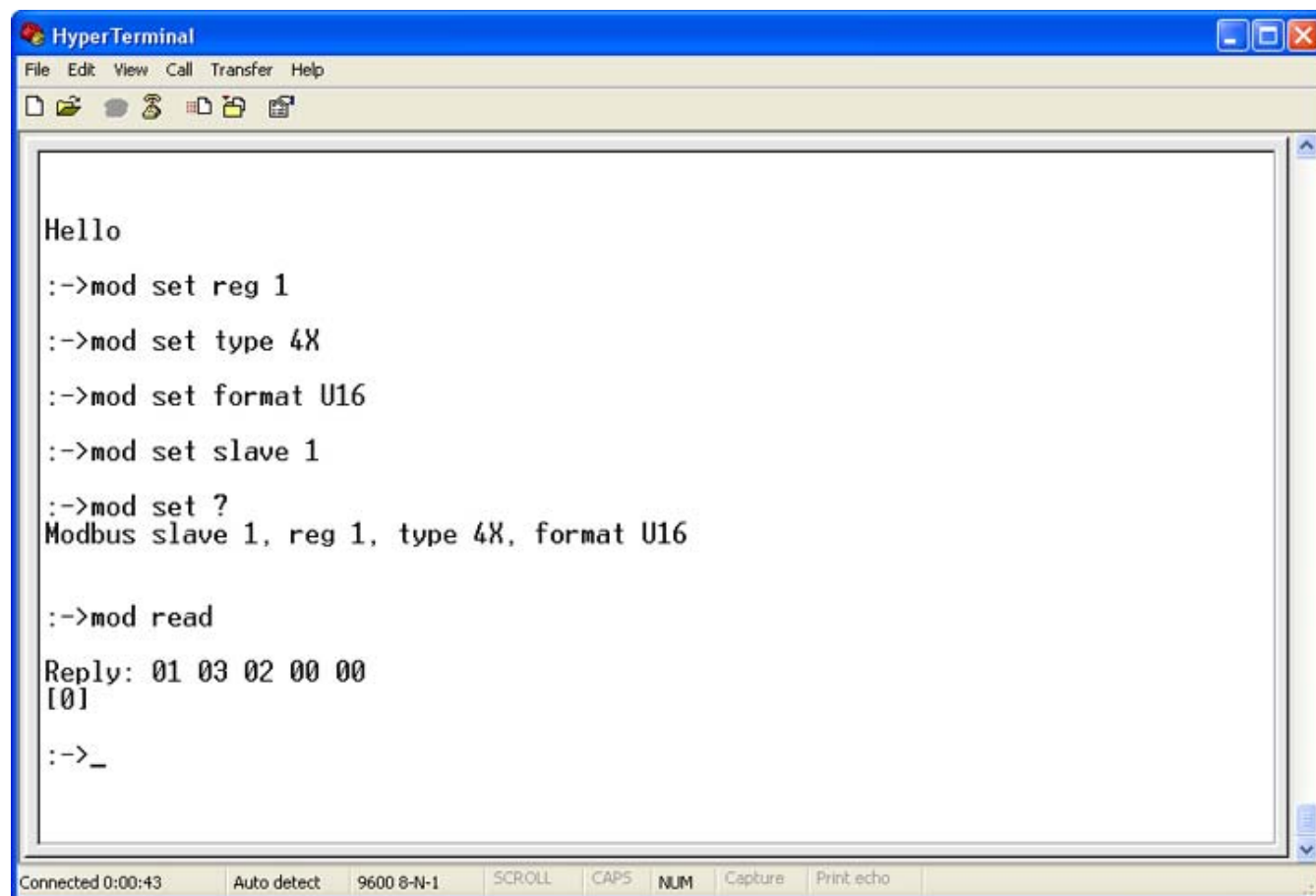
saddr - shows IP address currently in use by TCP server

smac - displays MAC address of TCP server

A.3 Modbus Diagnostic Example

One of the more useful aspects of the diagnostic console is being able to "test" Modbus registers without going through the trouble of configuring objects in the gateway. This is especially useful if you have configured an object but it is not returning Modbus data as expected. Using the Modbus diagnostics illustrated here, you can directly access the Modbus device via the Modbus network and begin to diagnose the problem.

The following console session illustrates setting up the Modbus target, and reading the register. The resulting Modbus reply is displayed in raw form along with the interpreted data value "[0]".



```
HyperTerminal
File Edit View Call Transfer Help

Hello

:->mod set reg 1
:->mod set type 4X
:->mod set format U16
:->mod set slave 1
:->mod set ?
Modbus slave 1, reg 1, type 4X, format U16

:->mod read
Reply: 01 03 02 00 00
[0]

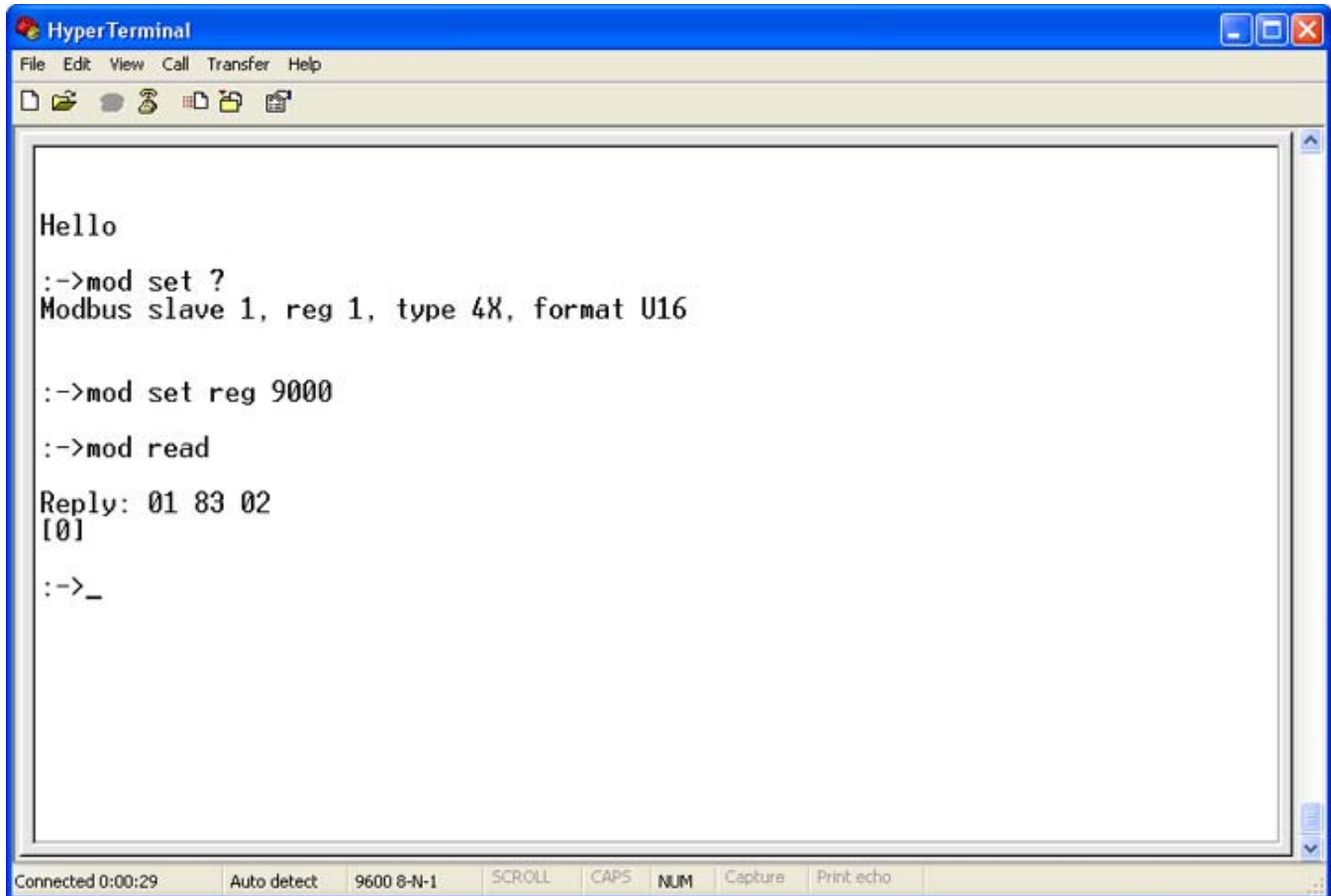
:->_

Connected 0:00:43  Auto detect  9600 8-N-1  SCROLL  CAPS  NUM  Capture  Print echo
```

The following example shows setting a register number that is deliberately wrong for the slave in use. It illustrates what the reply looks like when the slave reports an "illegal data address" exception. The first

byte is the slave address or unit number. The second byte is the function code. When the MSB in the function code byte is set (visually looks like adding 80), this means the next byte is the exception code. In this case, the exception code is 2, which means illegal data address.

If you would like more detailed information about what the raw content of a Modbus packet looks like, you can obtain a copy of the Modbus protocol specification at www.modbus.org. The Modbus protocol specification is surprisingly easy to read compared to some protocol specifications.



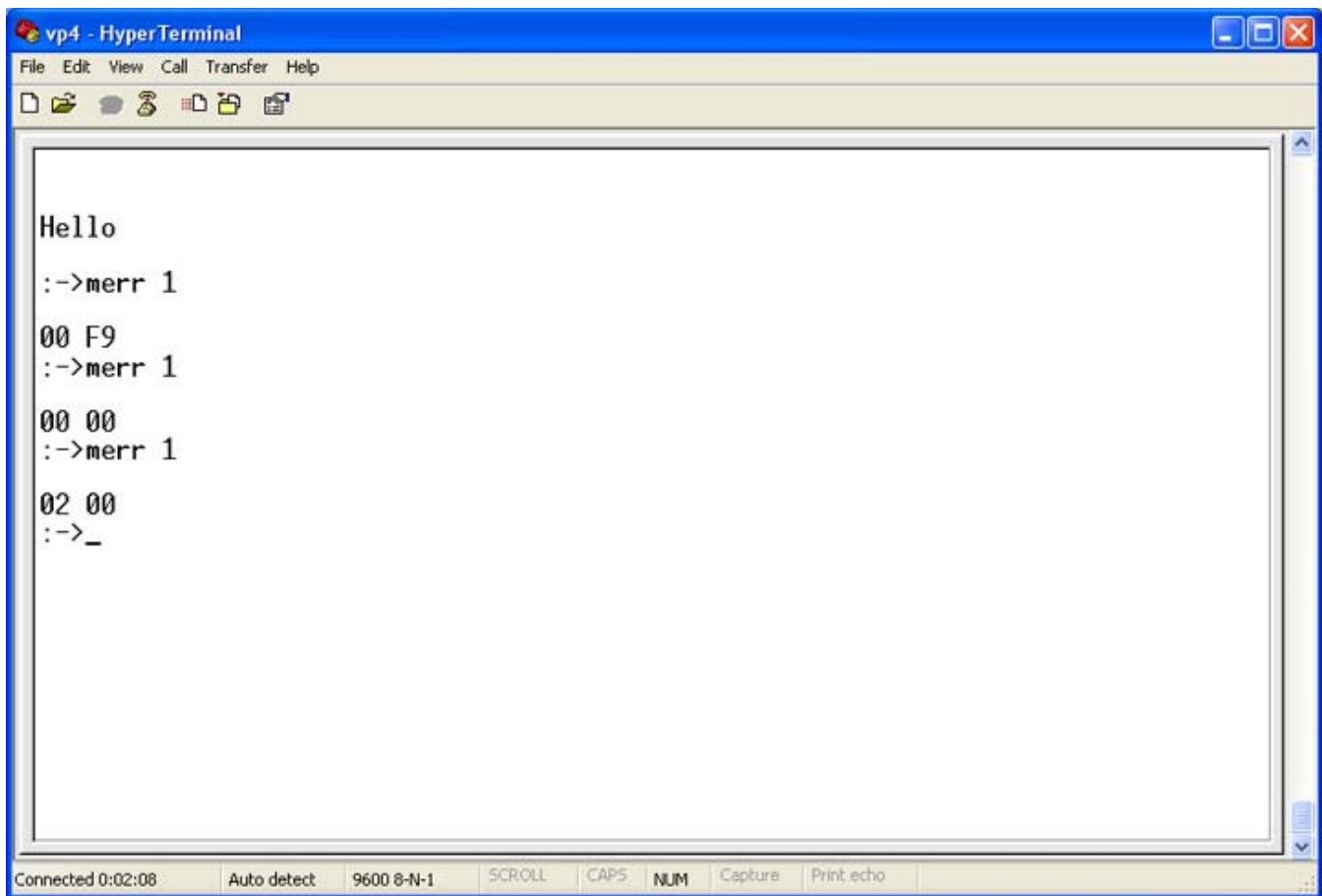
```
HyperTerminal
File Edit View Call Transfer Help
[Icons]
Hello
: ->mod set ?
Modbus slave 1, reg 1, type 4X, format U16

: ->mod set reg 9000
: ->mod read
Reply: 01 83 02
[0]
: ->_

Connected 0:00:29  Auto detect  9600 8-N-1  SCROLL  CAPS  NUM  Capture  Print echo
```

The screen shot below shows use of the merrr command to check Modbus error reporting. The first example shows both no-response errors and CRC errors. The second example shows no errors. The third example shows an exception error, in this case "illegal data address".

The format of the second byte is AB where A is the hexadecimal count of no-response errors, and B is the hexadecimal count of CRC errors. The counts will stop incrementing at hexadecimal F, which is decimal 15. The counts clear upon the first successful query to the respective slave address.



```
vp4 - HyperTerminal
File Edit View Call Transfer Help

Hello
:->merr 1
00 F9
:->merr 1
00 00
:->merr 1
02 00
:->_

Connected 0:02:08  Auto detect  9600 8-N-1  SCROLL  CAPS  NUM  Capture  Print echo
```

Appendix B LonWorks Trouble Shooting

B.1 General Practice, LED Indicators

The BB2-2010, BB2-2011, and BB2-6020 are most often used to interface a Modbus device to a LonWorks network. Diagnostics favor LonWorks learning of problems with the Modbus device. Because of the peer to peer nature of LonWorks and the network variable binding, there is little the Modbus device can do to determine if there is a problem with the LonWorks device. Problems with the Modbus device are indicated to LonWorks via the Object Status.

You can exercise the LonWorks side of the gateway without commissioning on a live network by using Echelon's Nodeutil tool. This is available from the Downloads page at www.echelon.com. Using Nodeutil, you can query network variables, and also update network variable inputs in the gateway. When accessing network variable data with Nodeutil, be aware that data is provided as a series of bytes displayed in hexadecimal form. Therefore, if the variable is SNVT_temp and the data represents 25.2°C, the raw data displayed by Nodeutil will be 0B B0. It will be very useful to have a hexadecimal calculator pulled up on your PC, and have the LonMark SNVT.pdf document handy.



The LonWorks indicator LEDs display LonWorks network activity. Note, however, that activity of these LEDs is not only affected by configuration of the device, but by whether the device is commissioned on the network and whether other devices are also communicating.

Mode	Data LED	Status LED
Wink	Alternates between yellow & green 10 times, then resume normal mode.	Alternates between red & green 10 times, then resume normal mode.
Normal	<p>Yellow flash indicates NV update was sent by gateway.</p> <p>Green flash indicates NV update was received by gateway.</p>	<p>Red indicates Neuron chip is not running.</p> <p>Green indicates gateway's host processor is communicating with LonWorks Neuron chip.</p>

The "wink" behavior is invoked by sending a wink command to the BB2 gateway via the LonWorks network. This is generally just a diagnostic to see if you are successfully communicating with the device via LonWorks. Other than the few seconds it takes to execute the wink, the device will always be in "normal" mode as far as LED indications are concerned in the table above.

B.2 Object Status Bits

Object status bits that explicitly report Modbus device problems are as follows:

open_circuit

This status bit is set when the Modbus slave device does not respond. The slave timeout is set on the Modbus Port page. If the slave does not respond within this time, the query is aborted and an error reported as open_circuit (for lack of any more appropriate definition in Object Status).

unable_to_measure

This status bit only applies to Modbus TCP, and means that there is no available path to the requested slave unit number. This generally means the TCP device table has not been initialized, or the slave unit number given is not found in the table. Check the Modbus TCP IP address list on the Modbus Port page.

electrical_fault

This status bit indicates that there have been numerous CRC errors attempting to contact the Modbus slave device. If this occurs, it will most often occur universally with all slaves on the network since this usually indicates a wiring problem with the RS485 line for Modbus RTU.

The TCP stack takes care of CRC checks and retries for Modbus TCP, and therefore this status bit is effectively unused with Modbus TCP. Any CRC error that occurs with Modbus TCP will never be seen at the application level.

comm_failure

This status bit indicates that the Modbus slave device has returned an exception code, which is the Modbus term for error code returned by a slave. If you get 'comm_failure' it actually means that you are communicating with the slave device, but it is rejecting your request.

Any of the above status bits can be regarded as a 'comm failure'. The status bit in the Object Status that is labeled comm_failure is simply selected to represent exception errors in an effort to differentiate between different types of comm failures to aid in diagnostics.

The most common exception code is 'illegal data address' which means you have requested a register number that the slave does not recognize. You simply need to check the register number and correct it or try a different one.

Important: When the comm_failure bit is set indicating an exception error, it means that a register associated with that network variable has reported the problem. If the NV is structured and there are multiple Modbus registers involved, there is no way of knowing which of the several registers has reported the exception.

The gateway is unable to keep track of specific exception codes on a register by register basis. However, you can use the USB console to manually read the register(s) in question to see specifically what exception error is being returned. Refer to Appendix A for more information about the use of the USB diagnostic console.

disabled

This status bit will indicate if the function block, or object, is currently disabled. Objects are expected to be disabled while configuration properties are changed via the LonWorks network, including NV type, and minimum/maximum send times. This disabling of the object is normally handled by the network management tool (e.g. LonMaker) and the object should be re-enabled by the same tool upon

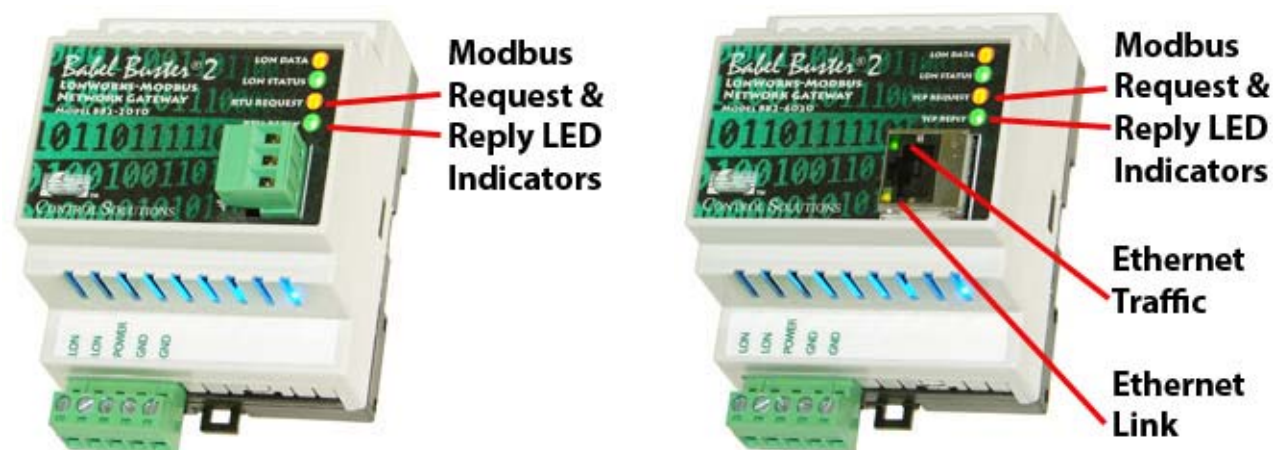
completion of changes.

Appendix C Modbus Trouble Shooting

C.1 Observing Modbus Errors, LED Indicators

Modbus errors can be observed via Object Status bits as described in Appendix B. Modbus errors can be displayed by slave address using the Get Error Info button on the Modbus Port page (see Section 11). The information displayed when using the Get Error Info button will indicate timeouts or no-response errors as well as Modbus exception codes returned by the slave.

The LED indicators on the front of the gateway also indicate Modbus errors. These are global indicators that do not tell you which device or which register is having trouble, but these indicators are a very quick way to observe whether there are problems, and also whether there is any activity at all.



The request and reply LEDs will indicate Modbus traffic as indicated in the table below.

The Ethernet traffic LED will indicate any traffic on the Ethernet network, and does not necessarily indicate Modbus TCP traffic. The traffic LED will typically be off more than on, flashing on each time traffic is indicated. If the traffic LED is on completely solid, the server is not running (normal for a half minute or so during startup).

The Ethernet link LED will be on any time there is a connection to the network. If the Ethernet cable is unplugged, this light will go out. If Modbus TCP is failing and this light is out, check Ethernet cables.

Mode	Request LED	Reply LED
Gateway is Master	Flash yellow each time master (gateway) sends a request to a remote slave.	Flash green when master receives a good response.
		Flash red when master receives exception message from slave, or if timed out with no response from slave.
Gateway is Slave	Flash yellow each time slave (gateway) receives a request	Flash green when slave recognizes request as good/valid and sends a good reply.
		Flash red when slave

	from external master.	receives a request that results in replying with an exception, or there was a CRC error (RTU only) in the request.
Gateway is Master	TCP only: If TCP is unable to make a connection with the IP address given for the TCP slave, the request LED will not flash yellow (because no request was sent yet), but the reply LED will flash red each time the connection attempt times out or fails.	

C.2 Modbus Reference Information

Modbus Register Types

The types of registers referenced in Modbus devices include the following:

- Coil (Discrete Output)
- Discrete Input
- Input Register
- Holding Register

Whether a particular device includes all of these register types is up to the manufacturer. It is very common to find all I/O mapped to holding registers only. Coils are 1-bit registers, are used to control discrete outputs, and may be read or written. Discrete Inputs are 1-bit registers used as inputs, and may only be read. Input registers are 16-bit registers used for input, and may only be read. Holding registers are the most universal 16-bit register, may be read or written, and may be used for a variety of things including inputs, outputs, configuration data, or any requirement for "holding" data.

Modbus Function Codes

Modbus protocol defines several function codes for accessing Modbus registers. There are four different data blocks defined by Modbus, and the addresses or register numbers in each of those overlap. Therefore, a complete definition of where to find a piece of data requires both the address (or register number) and function code (or register type).

The function codes most commonly recognized by Modbus devices are indicated in the table below. This is only a subset of the codes available - several of the codes have special applications that most often do not apply.

Function Code	Register Type
1	Read Coil
2	Read Discrete Input
3	Read Holding Registers
4	Read Input Registers
5	Write Single Coil
6	Write Single Holding Register
15	Write Multiple Coils
16	Write Multiple Holding Registers

Modbus Exception (error) Codes

When a Modbus slave recognizes a packet, but determines that there is an error in the request, it will

return an exception code reply instead of a data reply. The exception reply consists of the slave address or unit number, a copy of the function code with the high bit set, and an exception code. If the function code was 3, for example, the function code in the exception reply will be 0x83. The exception codes will be one of the following:

1	Illegal Function	The function code received in the query is not recognized by the slave or is not allowed by the slave.
2	Illegal Data Address	The data address (register number) received in the query is not an allowed address for the slave, i.e., the register does not exist. If multiple registers were requested, at least one was not permitted.
3	Illegal Data Value	The value contained in the query's data field is not acceptable to the slave.
4	Slave Device Failure	An unrecoverable error occurred 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.

Modicon convention notation for Modbus registers

Modbus was originally developed by Gould-Modicon, which is presently Schneider Electric. The notation originally used by Modicon is still often used today, even though considered obsolete by present Modbus standards. The advantage in using the Modicon notation is that two pieces of information are included in a single number: (a) The register type; (b) The register number. A register number offset defines the type.

The types of registers referenced in Modbus devices, and supported by Babel Buster gateways, include the following:

- Coil (Discrete Output)
- Discrete Input
- Input Register
- Holding Register

Valid address ranges as originally defined for Modbus were 0 to 9999 for each of the above register types. Valid ranges allowed in the current specification are 0 to 65,535. The address range originally supported by Babel Buster gateways was 0 to 9999. The extended range addressing was later added to all new Babel Buster products that use this notation.

The address range applies to each type of register, and one needs to look at the function code in the Modbus message packet to determine what register type is being referenced. The Modicon convention uses the first digit of a register reference to identify the register type.

Register types and reference ranges recognized by Babel Buster (LonWorks) gateways are as follows:

0x = Coil = 00001-09999
 1x = Discrete Input = 10001-19999
 3x = Input Register = 30001-39999
 4x = Holding Register = 40001-49999

Translating references to addresses, reference 40001 selects the holding register at address 0000 (also referred to as register number 1). The reference 40001 will appear in documentation and is used to define the Modbus register in the location property of the functional block in a LonWorks gateway. The address 0000 will be transmitted in the message packet. Addresses are often not directly used by the application or the user.

On occasion, it is necessary to access more than 10,000 of a register type. Based on the original convention, there is another defacto standard that looks very similar. Additional register types and reference ranges recognized by Babel Buster (LonWorks) gateways are as follows:

0x = Coil = 000001-065535
1x = Discrete Input = 100001-165535
3x = Input Register = 300001-365535
4x = Holding Register = 400001-465535

When using the extended register referencing, it is mandatory that all register references be exactly six digits. This is the only way Babel Buster will know the difference between holding register 40001 and coil 40001. If coil 40001 is the target, it must appear as 040001.

If registers are 16-bits, how does one read Floating Point or 32-bit data?

Modbus protocol defines a holding register as 16 bits wide; however, there is a widely used defacto standard for reading and writing data wider than 16 bits. The most common are IEEE 754 floating point, and 32-bit integer. The convention may also be extended to double precision floating point and 64-bit integer data.

The wide data simply consists of two consecutive "registers" treated as a single wide register. Floating point in 32-bit IEEE 754 standard, and 32-bit integer data, are widely used. Although the convention of register pairs is widely recognized, agreement on whether the high order or low order register should come first is not standardized. For this reason, many devices, including all Control Solutions gateways, support register "swapping". This means you simply check the "swapped" option (aka "High reg first" in some devices) if the other device treats wide data in the opposite order relative to Control Solutions default order.

Control Solutions Modbus products all default to placing the high order register first, or in the lower numbered register. This is known as "big endian", and is consistent with Modbus protocol which is by definition big endian.

What does notation like 40001:7 mean?

This is a commonly used notation for referencing individual bits in a register. This particular example references register 40001, bit 7. Bits are generally numbered starting at bit 0, which is the least significant or right most bit in the field of 16 bits found in a Modbus register. (Note that bit numbering in most Modbus devices is opposite the order defined for bit fields in LonWorks network variables.)

How does one read individual bits in a register?

The bit mask shown in the expanded form of the RTU read map is a 4 digit hexadecimal (16 bit) value used to mask out one or more bits in a register. The selected bits will be right justified, so a single bit regardless of where positioned in the source register will be stored locally as 0 or 1. The notation of register number followed by a colon and number from 0 to 15 indicates a single bit picked from that register. The hex bit mask values would be as follows, assuming a register number of 40001.

40001:0 mask: 0001
40001:1 mask: 0002
40001:2 mask: 0004
40001:3 mask: 0008

40001:4 mask: 0010
40001:5 mask: 0020
40001:6 mask: 0040
40001:7 mask: 0080
40001:8 mask: 0100
40001:9 mask: 0200
40001:10 mask: 0400
40001:11 mask: 0800
40001:12 mask: 1000
40001:13 mask: 2000
40001:14 mask: 4000
40001:15 mask: 8000

Sometimes a 16-bit register is used to hold two 8-bit values. To strip bytes using the bit mask, you would enter the following:

Low byte mask: 00FF
High byte mask: FF00

Deciphering Modbus Documentation

Documentation for Modbus is not well standardized. Actually there is a standard, but not well followed when it comes to documentation. You will have to do one or more of the following to decipher which register a manufacturer is really referring to:

- a) Look for the register description, such as holding register, coil, etc. If the documentation says #1, and tells you they are holding registers, then you have holding register #1. You also have user friendly documentation.
- b) Look at the numbers themselves. If you see the first register on the list having a number 40001, that really tells you register #1, and it is a holding register. This form of notation is often referred to as the old Modicon convention.
- c) Look for a definition of function codes to be used. If you see a register #1, along with notation telling you to use function codes 3 and 16, that also tells you it is holding register #1.

IMPORTANT: Register 1 is address 0. Read on...

- d) Do the numbers in your documentation refer to the register number or address? Register #1 is address zero. If it is not clear whether your documentation refers to register or address, and you are not getting the expected result, try plus or minus one for register number. All Control Solutions products refer to register numbers in configuration software or web pages. However, some manufacturers document their devices showing address, not register numbers. When you have addresses, you must add one when entering that register into configuration software from Control Solutions.

Can I put 2 gateways on the same Modbus network?

You can not have more than one Master on a Modbus RTU (RS-485) network. Therefore, if the gateway is to be configured as the Master, you can only have 1 gateway. You cannot use multiple gateways to read more points from the same Modbus slave device.

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

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

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

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

Appendix D Modbus CSV Register List Format

D.1 Data Labels on Header Line

The required format for importing Modbus register lists into the Babel Buster LonWorks gateway from a CSV file is intended to be as forgiving as possible. The first line of the CSV file must be a header line containing labels for the columns of data if there is more than just a list of numbers in the file. If only a list of numbers, with one entry per line, is found, then some assumptions are made about the implied header line. Otherwise, the header line must provide the column labels.

The available column labels are outlined in the table below. The minimum requirement is the use of REG and TYPE, or alternatively use of MODICON which implies both register and type. The remaining labels are optional, although in most cases at least some additional labels will be desirable. The order in which the labels appear does not matter, so long as the data on subsequent lines follows the same order as the header line.

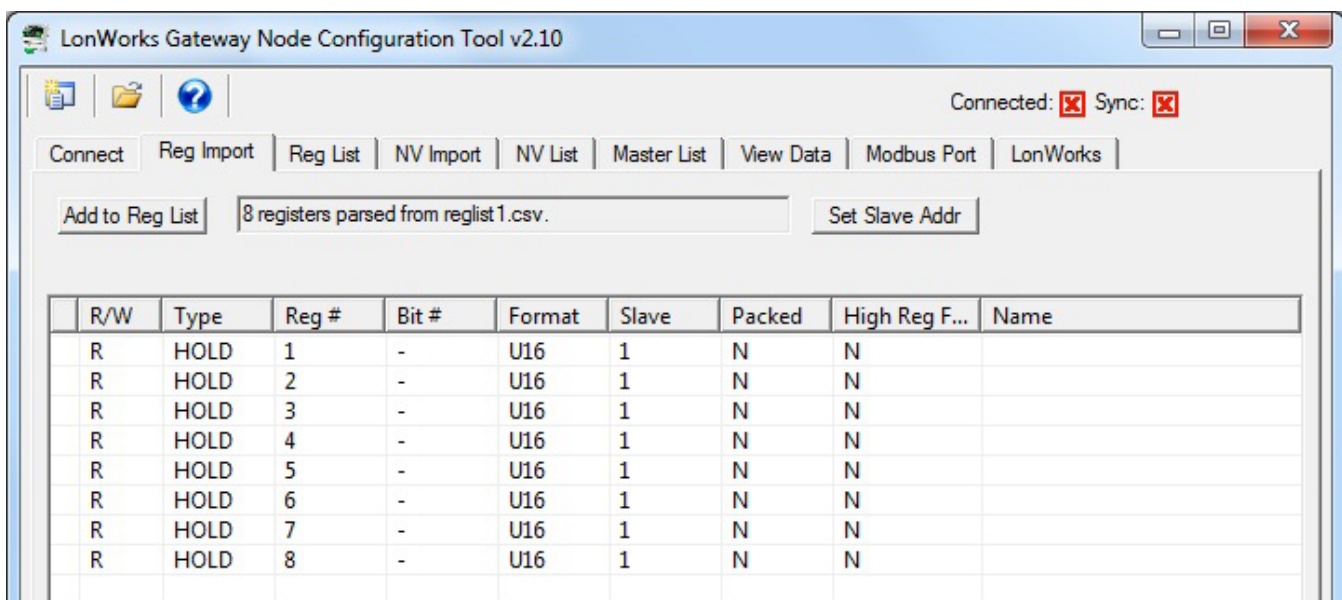
Label	Content	What it means
MODICON	Register number e.g. 40001	If MODICON is used, then do not include REG or TYPE. Register numbers should use Modicon notation, such as 40001 for holding register number 1.
REG	Register number e.g. 1	REG and TYPE are used together instead of MODICON when register numbers will be given as starting at 1.
TYPE	One of: COIL, DISC, INPUT, HOLD, COIL1, HOLD1	TYPE indicates what register type is numbered in the REG column. If omitted, will default to HOLD for holding register. The types COIL1 and HOLD1 will force function codes 5 and 6 for writes rather than the default 15 and 16.
RW	Either R or W	Enter R to read the register, or W to write it. If omitted, defaults to R. Use W+ for write on Update.
FORMAT	One of S16, U16, S32, U32, S64, FP, BIT, M102, M103, M104	Specify S or U for Signed or Unsigned, 16 or 32 bit integer. Specify FP for IEEE754 floating point. Specify BIT when TYPE is COIL or DISC. Specify S64 for Signed 64-bit integer. Specify M102, M103, M104 for Mod10 format 2, 3, and 4-register values respectively.
SLAVE	Number from 1 to 247	Modbus slave number that should be read or written.
BITNUMBER	Number from 0 to 15 for 16-bit register or 0 to 31 for 32-bit register	Bit number when masking out a single bit. When using the mask, FORMAT must be U16 or U32. Do not use BIT - the format refers to the format of data at the source, not the end result. Important: Leave field empty if bit mask should not be used. The number 0 will result in a mask for bit position zero.
		When consecutive register maps (lines on spreadsheet) refer to same register number using different bit numbers, the

PACKED	Either T or F	PACKED field should be T to cause the register to be read once and results distributed to multiple objects. Defaults to F if not specified.
HIGHREGFIRST	Either T or F	Applies only to 32-bit (including floating point) registers, T specifies that the first register (lowest numbered register) in the register pair will contain the most significant half of the data. F specifies the opposite order.
NAME	Any character string up to 40 characters	Any character string of up to 40 characters, used for documentation reference only.

D.2 Example CSV Files and Imports

The following are examples of rather simple CSV files with Modbus registers and a screen shot of the resulting import. While brief, these examples are intended to show some of the possible variations in format.

<p>REG</p> <p>1</p> <p>2</p> <p>3</p> <p>4</p> <p>5</p> <p>6</p> <p>7</p> <p>8</p>	<p>The simplest CSV file for register import is just a list of numbers, optionally having a header line with just REG.</p> <p>If the label REG appears on the first line, or the first number encountered appears to be a standard register number, it will be assumed to be a holding register using standard notation (not Modicon).</p> <p>Importing this CSV results in the following screen shot.</p>
--	--



```

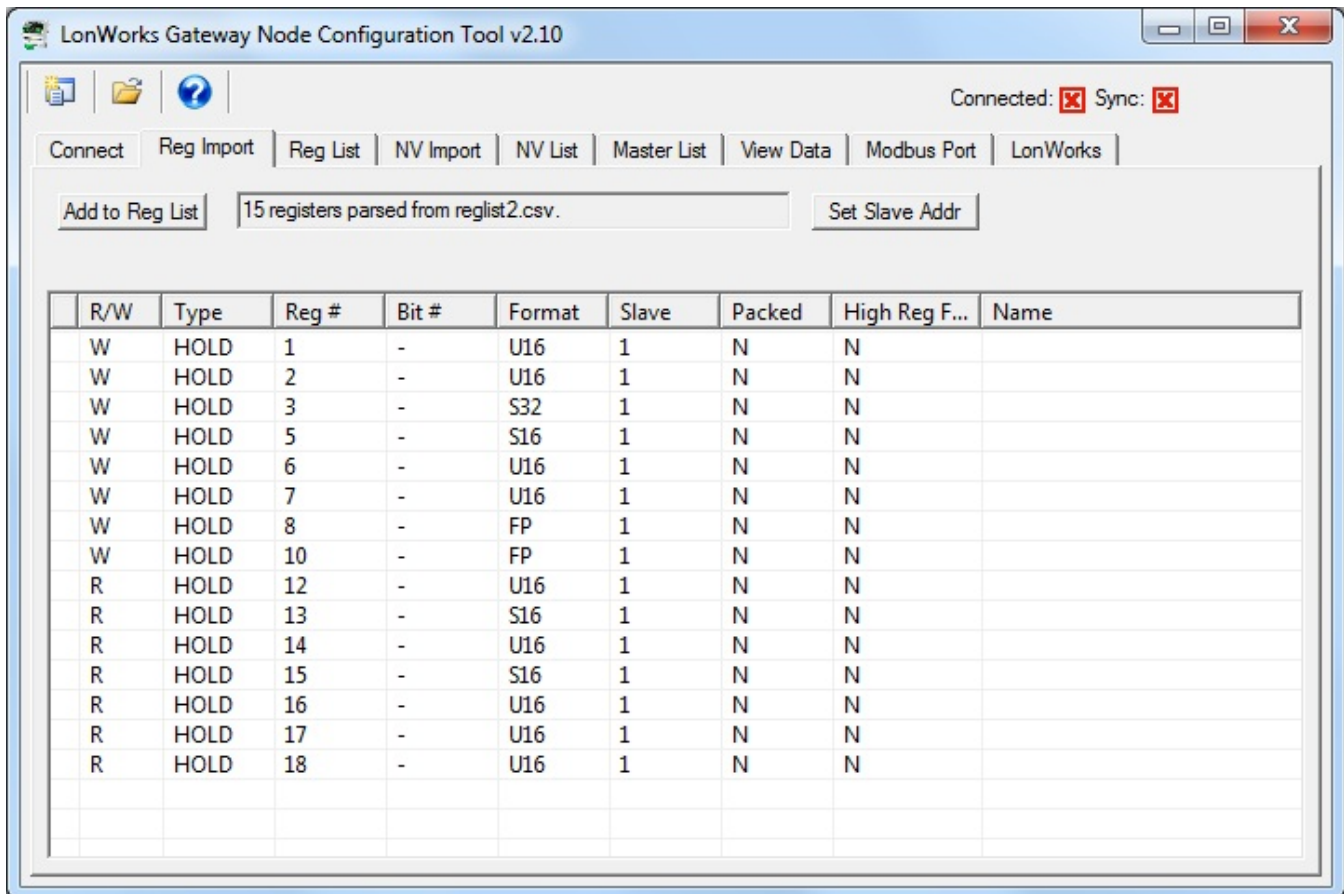
RW,REG,TYPE,FORMAT,SLAVE,BITNUMBER,PACKED,HIGHREGFIRST
W,1,HOLD,U16,1,,F,F
W,2,HOLD,U16,1,,F,F
W,3,HOLD,S32,1,,F,F
W,5,HOLD,S16,1,,F,F
W,6,HOLD,U16,1,,F,F
W,7,HOLD,U16,1,,F,F
W,8,HOLD,FP,1,,F,F
W,10,HOLD,FP,1,,F,F
R,12,HOLD,U16,1,,F,F
R,13,HOLD,S16,1,,F,F
R,14,HOLD,U16,1,,F,F
R,15,HOLD,S16,1,,F,F
R,16,HOLD,U16,1,,F,F
R,17,HOLD,U16,1,,F,F
R,18,HOLD,U16,1,,F,F

```

A more complex register list using multiple labels in the header line, and standard register numbering, is illustrated here.

The header line is required when more than just a list of register numbers is given. The order of labels is not important, as long as they match the columns of data that follow.

The following screen shot is the result of importing this CSV file.



```

MODICON
40001
40002
40003
40004

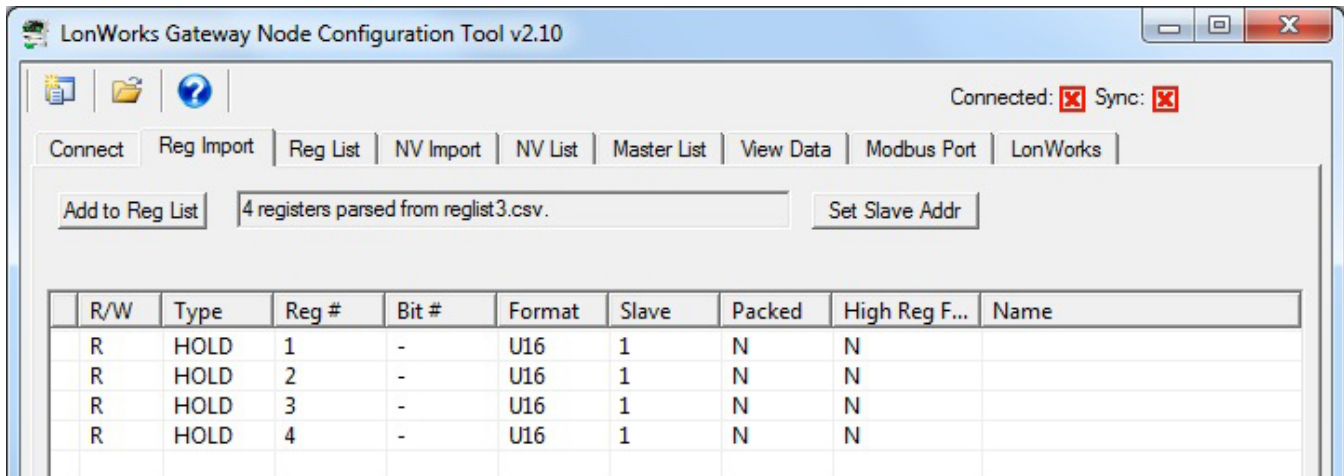
```

The other form of simplest CSV file for register import is just a list of numbers, optionally having a header line with just MODICON.

If the label MODICON appears on the first line, or the first number encountered appears to be a Modicon holding register number, all numbers will be interpreted as Modicon. The test is whether the first number encountered is between 40001 and

49999.

Importing this CSV results in the following screen shot.

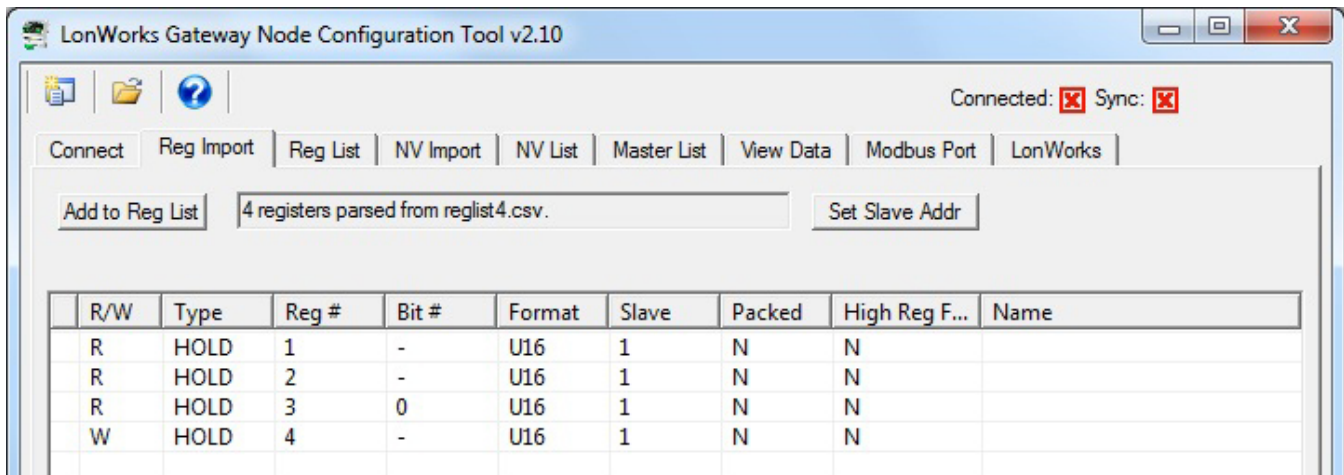


```
MODICON,BITNUMBER,RW
40001,,R
40002,,R
40003,0,R
40004,,W
```

This is the same simple CSV with a couple of additional columns.

Holding register 3 will be masked to use only bit 0 in the value placed in the object to be mapped to this register. The empty bit fields for other registers mean masking will not be used for those registers.

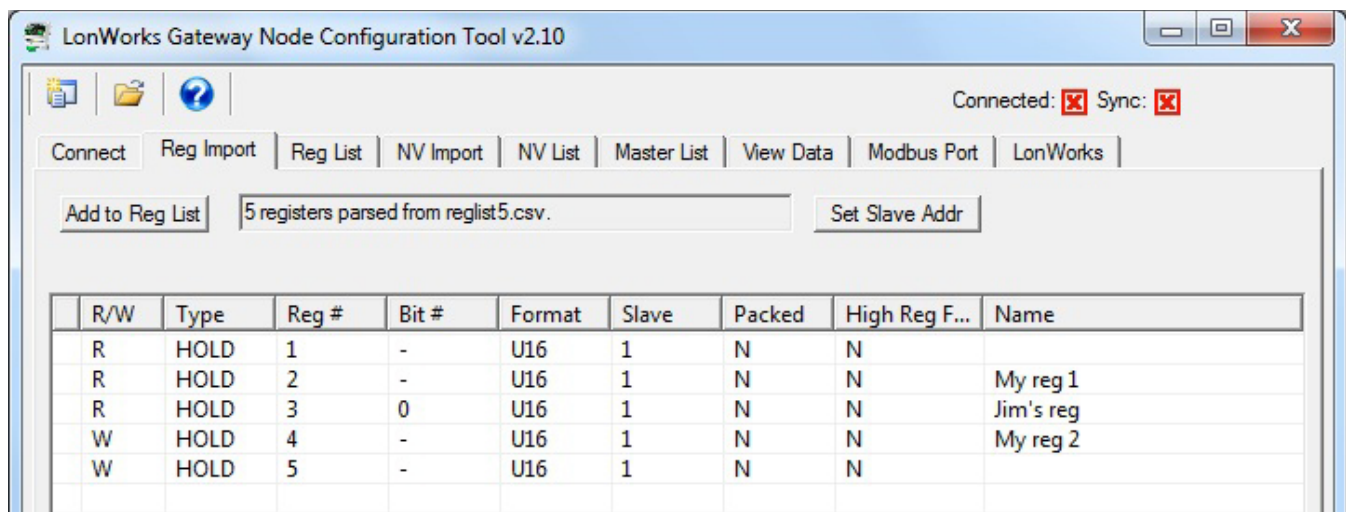
Importing this CSV results in the following screen shot.



```
MODICON,BITNUMBER,RW,NAME
40001,,R
40002,,R,"My reg 1"
40003,0,R,Jim's reg
40004,,W,'My reg 2'
40005,,W
```

An optional register name column is added to the CSV in this example.

Importing this CSV results in the following screen shot.



```

RW,MODICON,FORMAT,SLAVE,BITNUMBER,PACKED,HIGHREGFIRST,NAME
W,40001,U16,1,,F,F,Register 1
W,40002,U16,1,,F,F,Register 2
W,40003,S32,1,,F,F,
W,40005,S16,1,,F,F,
W,40006,U16,1,,F,F,
W,40007,U16,1,,F,F,
W,40008,FP,1,,F,F,
W,40010,FP,1,,F,F,Register 10
R,40012,U16,1,,F,F,
R,40013,S16,1,,F,F,
R,40014,U16,1,,F,F,
R,40015,S16,1,,F,F,
R,40016,U16,1,,F,F,
R,40017,U16,1,,F,F,
R,40018,U16,1,,F,F,

```

This example shows a "full" set of header labels and data columns. If a field is left empty, as denoted by two consecutive commas, the default value will apply.

Importing this CSV results in the following screen shot.

LonWorks Gateway Node Configuration Tool v2.10

Connected: ☒ Sync: ☒

Connect | Reg Import | Reg List | NV Import | NV List | Master List | View Data | Modbus Port | LonWorks

Add to Reg List | 15 registers parsed from regist6.csv. | Set Slave Addr

	R/W	Type	Reg #	Bit #	Format	Slave	Packed	High Reg F...	Name
	W	HOLD	1	-	U16	1	N	N	Register 1
	W	HOLD	2	-	U16	1	N	N	Register 2
	W	HOLD	3	-	S32	1	N	N	
	W	HOLD	5	-	S16	1	N	N	
	W	HOLD	6	-	U16	1	N	N	
	W	HOLD	7	-	U16	1	N	N	
	W	HOLD	8	-	FP	1	N	N	
	W	HOLD	10	-	FP	1	N	N	Register 10
	R	HOLD	12	-	U16	1	N	N	
	R	HOLD	13	-	S16	1	N	N	
	R	HOLD	14	-	U16	1	N	N	
	R	HOLD	15	-	S16	1	N	N	
	R	HOLD	16	-	U16	1	N	N	
	R	HOLD	17	-	U16	1	N	N	
	R	HOLD	18	-	U16	1	N	N	

Appendix E Modbus Slave Register Map

E.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.

Modicon Std	Modicon Extd	Type	Std. Reg. No.	Description
Data object access registers (data registers)				
40001-40400	400001-400400	Holding Register	1-400	Objects 1-400 accessed as signed integer, 16-bit, single Modbus register
41001-41400	401001-401400	Holding Register	1001-1400	Objects 1-400 accessed as unsigned integer, 16-bit, single Modbus register
42001-42800	402001-402800	Holding Register	2001-2800	Objects 1-400 accessed as IEEE754 floating point, 32-bit, double Modbus register
43001-43800	403001-403800	Holding Register	3001-3800	Objects 1-400 accessed as signed integer, 32-bit, double Modbus register
48001-48246	408001-408246	Holding	8001-8246	Modbus error codes for slaves 1-246 (8001 if BB2 is slave) Value will be 0 if no error, exception codes 1..11, or 129=no response, 130=CRC errors.
Data mirror registers, data objects accessed via Modbus function codes other than holding registers				
00001-00400	000001-000400	Coil	1-400	Objects 1-400 accessed as single bit Coil registers
10001-10400	100001-100400	Discrete Input	1-400	Objects 1-400 accessed as single bit Discrete Input registers
30001-30400	300001-300400	Input Register	1-400	Objects 1-400 accessed as unsigned integer, 16-bit, single Modbus register
31001-31400	301001-301400	Input Register	1001-1400	Objects 1-400 accessed as signed integer, 16-bit, single Modbus register
32001-32800	302001-302800	Input Register	2001-2800	Objects 1-400 accessed as IEEE754 floating point, 32-bit, double Modbus register
33001-33800	303001-303800	Input Register	3001-3800	Objects 1-400 accessed as signed integer, 32-bit, double Modbus register

Appendix F LonWorks XIF File Format

F.1 XIF Files

The XIF file may be exported via the configuration tool for those network management tools that explicitly require a file. The content of the XIF file may also be extracted from the device itself by tools such as LonMaker. The only difference between file export and extracting the XIF from the device is that the user given network variable names are not stored in the device. If the XIF is extracted from the device by a network management tool, generic names as originally compiled into code are provided.

The XIF file follows the version 4.402 specification available from LonMark at www.lonmark.org. You will need to obtain a copy of that specification from LonMark if you have a need to study the XIF file in detail.

The XIF file is a text file with a .xif suffix and the file may be opened with any text editor. The beginning of the file will appear as follows, illustrated here in part so that you can identify what an XIF file looks like. **Do not attempt to edit this file!**

```
File: test7.xif generated by BB2-2010 Tool v2.10, XIF Version 4.402
Copyright (c) Control Solutions, Inc. 2014
All Rights Reserved. Run on Thu Mar 20 13:54:20 2014
```

```
90:00:17:47:1E:84:04:01
2 15 1 243 0 8 8 5 5 8 8 11 11 11 11 7 243 16 127 0 1 128 243 0 0 0 0 0 0 0 0 2 15
0 0 0 0 0 2 1 0 0
32 7 19 13 28 0 0 15 5 3 841 4 10000000 1
1 7 1 1 4 4 4 15 200 0
78125 0 0 0 0 0 252 0 0 0 0 0
90 0 240 0 0 0 40 40 0 5 22 9 26 43 44
*
"&3.4@0nodeObject,1[120openLoopSensor,3[120openLoopActuator

VAR nviRequest 0 0 0 0
0 1 63 0 0 1 0 1 0 1 0 0 0
"@0|1
92 * 2
2 0 0 0 0
1 0 0 1 0
VAR nvoStatus 1 0 0 0
0 1 63 1 0 1 0 1 0 1 0 1 0
"@0|2
93 * 26
2 0 0 0 0
3 0 1 0 0
3 1 1 0 0
3 2 1 0 0
3 3 1 0 0
3 4 1 0 0
3 5 1 0 0
3 6 1 0 0
3 7 1 0 0
3 0 1 0 0
3 1 1 0 0
3 2 1 0 0
3 3 1 0 0
```

```
3 4 1 0 0
3 5 1 0 0
3 6 1 0 0
3 7 1 0 0
3 0 1 0 0
3 1 1 0 0
3 2 1 0 0
3 3 1 0 0
3 4 1 0 0
3 5 1 0 0
3 6 1 0 0
3 7 1 0 0
3 0 8 0 0
VAR nvoFileDirectory 2 0 0 0
0 1 63 1 0 1 0 1 0 1 1 0 0
"@0|8
114 * 1
2 0 0 0 0
```

Appendix G Configuration XML File Format

G.1 Configuration Files

The configuration file that is used to save gateway configuration and reload later to reconfigure another gateway the same way is saved in XML format. This makes it easy to read for diagnostic purposes, primarily for Control Solutions' technical support use. However, due to the complexity and interaction between the parts of the file, the **XML FILE IS NOT INTENDED TO BE MANUALLY EDITED.**

There is no reason to manually edit an XML file. If you are looking for a short cut in configuring the gateway, you are looking in the wrong direction. The configuration tool includes the ability to import and export register lists as a CSV file, and import and export XIF files. Manual editing should be limited to creating and modifying the CSV file. Once your files are imported, you can auto-create much of the configuration.

The configuration software includes a number of error checking steps, and these are bypassed in the event you manually edit an XML file. **IF YOU HAVE CREATED PROBLEMS BY MANUALLY EDITING AN XML FILE, CONTROL SOLUTIONS WILL NOT HELP YOU FIX IT.**

Appendix H USB Driver Installation

H.1 Driver Installation

The required USB driver used to be included as a standard part of Windows, and all that needed to be done to “install” the driver was provide a configuration file telling Windows which driver to use. That driver is no longer included in Windows. First Microsoft said we should have no use for serial ports, so those disappeared and we started using USB. Now Microsoft’s official position is that we should no longer have any use for USB to even emulate serial ports. To make things even more frustrating for technical users, Windows 8 has greatly complicated the driver installation process allegedly in the interest of security.

Control Solutions has licensed a USB driver and installer from a software development company that specializes in USB drivers. Drivers are all they do and they do it well, so we feel confident in our choice. Control Solutions paid a significant license fee so that we are able to provide it to our customers at no charge.

The USB driver provided in Control Solutions’ driver package will install in Windows XP, 7, and 8, and both 32-bit and 64-bit versions. It includes the necessary driver signing verified through Verisign (now part of Symantec).

The driver package will show up as a zip file named “csimnUSB.zip”.



Unzip the contents of this file into a directory somewhere on your PC. The contents will look like this:

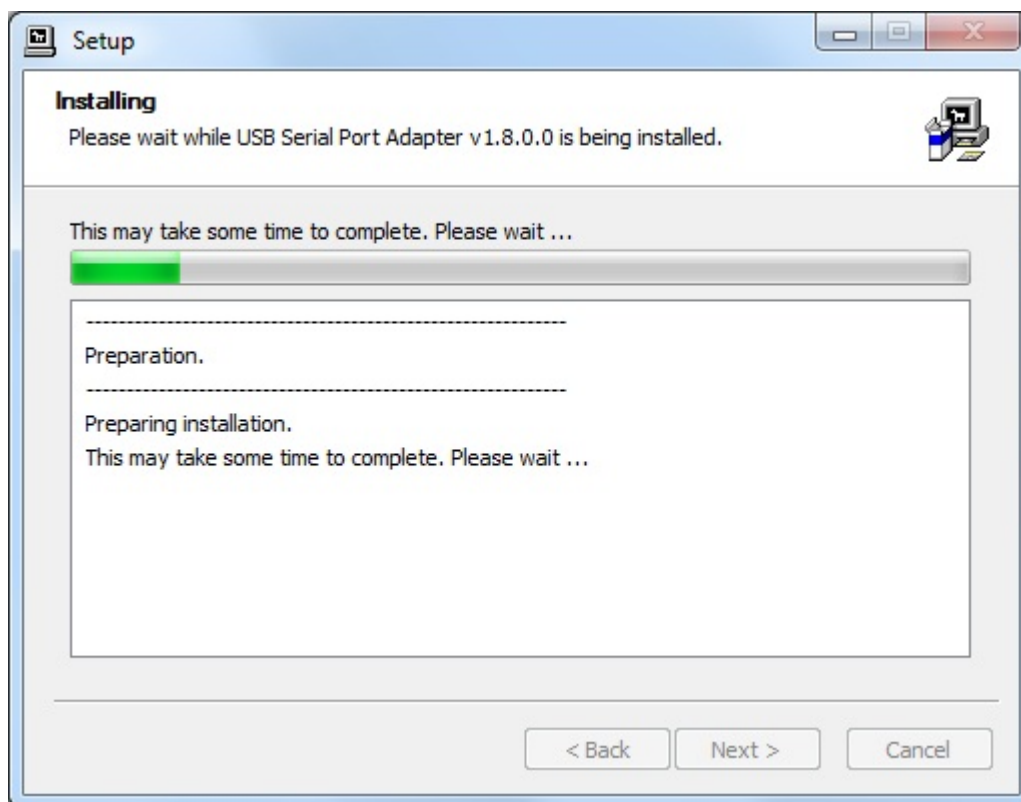
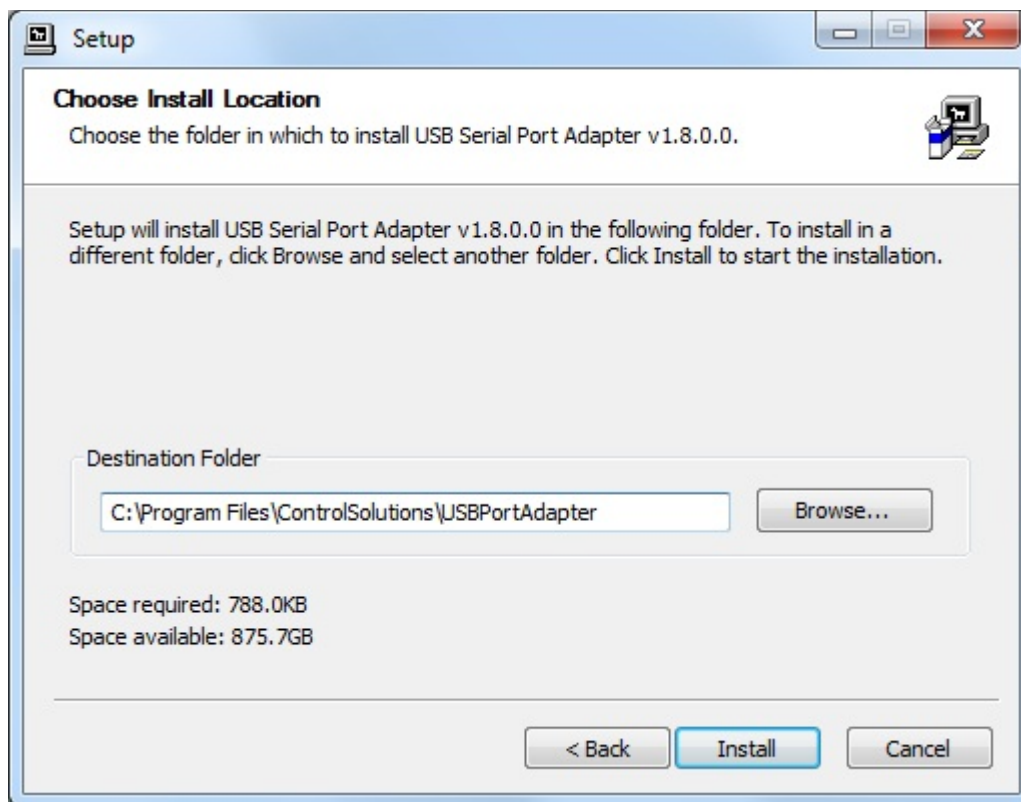


Double click “setup.exe”. Say “yes” to any questions about whether to trust this software. Also, for Windows 8, you should right click on the setup file and “Run as administrator” - you will need to be logged in with administrator privileges.

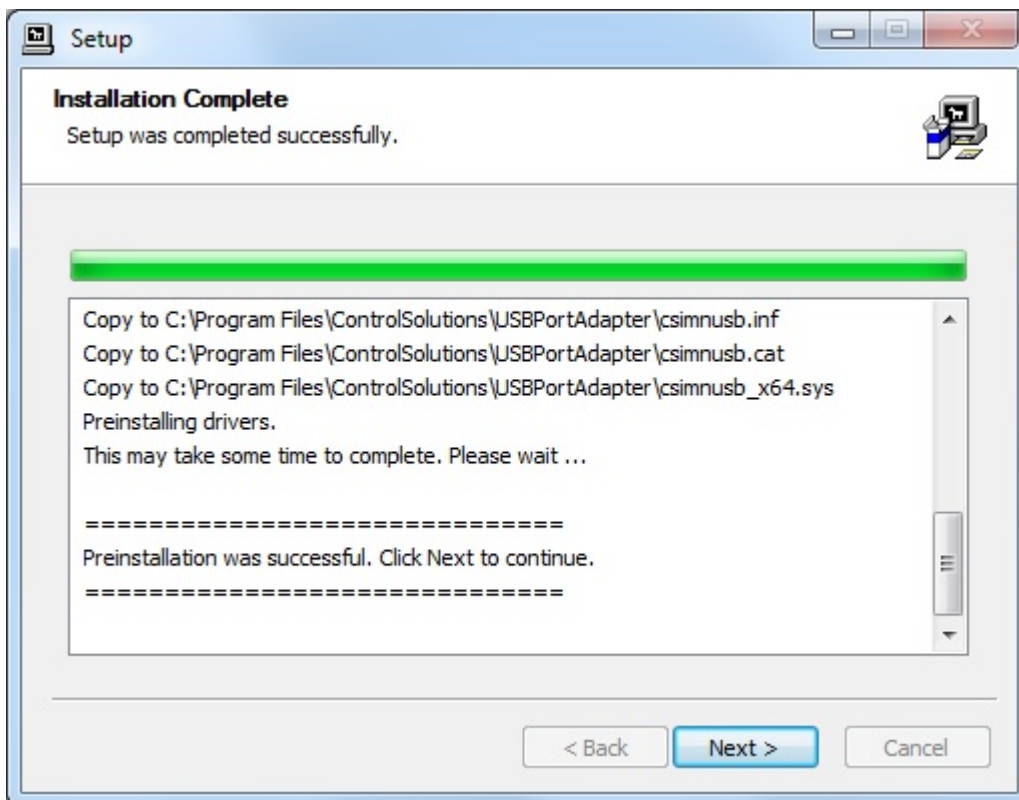
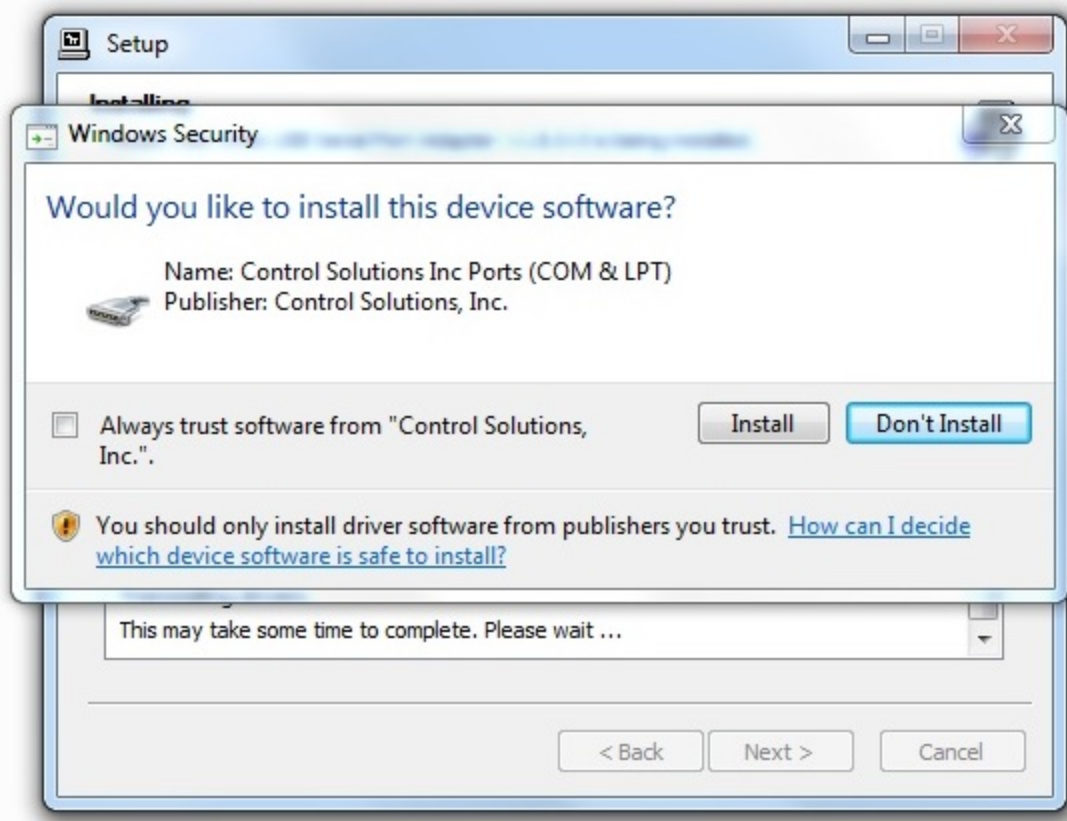
A sample of the series of screens you will see appears below. Basically all you need to do is follow the prompts, and click “yes”, “next”, “continue anyway”, etc, as applicable.

Technically, what you are doing in the process illustrated in the screen shots below is “driver pre-installation”. After initial installation of this package, the device will automatically find the right driver when you plug it in, and driver installation will be finalized. Windows 8 will install the driver quietly and usually say nothing about it. Windows 7 will display a prompt telling you the new device was installed, but will not require responding to any prompts. Windows XP will go through the characteristic “Found new hardware” routine with a series of dialogs and prompts the first time you plug the device in. Tell your PC “no” to searching the Internet, but “yes” to installing automatically, and “continue anyway” when it complains about Windows logo certification. (Windows 7 and 8 will not register any such Windows logo complaint.)



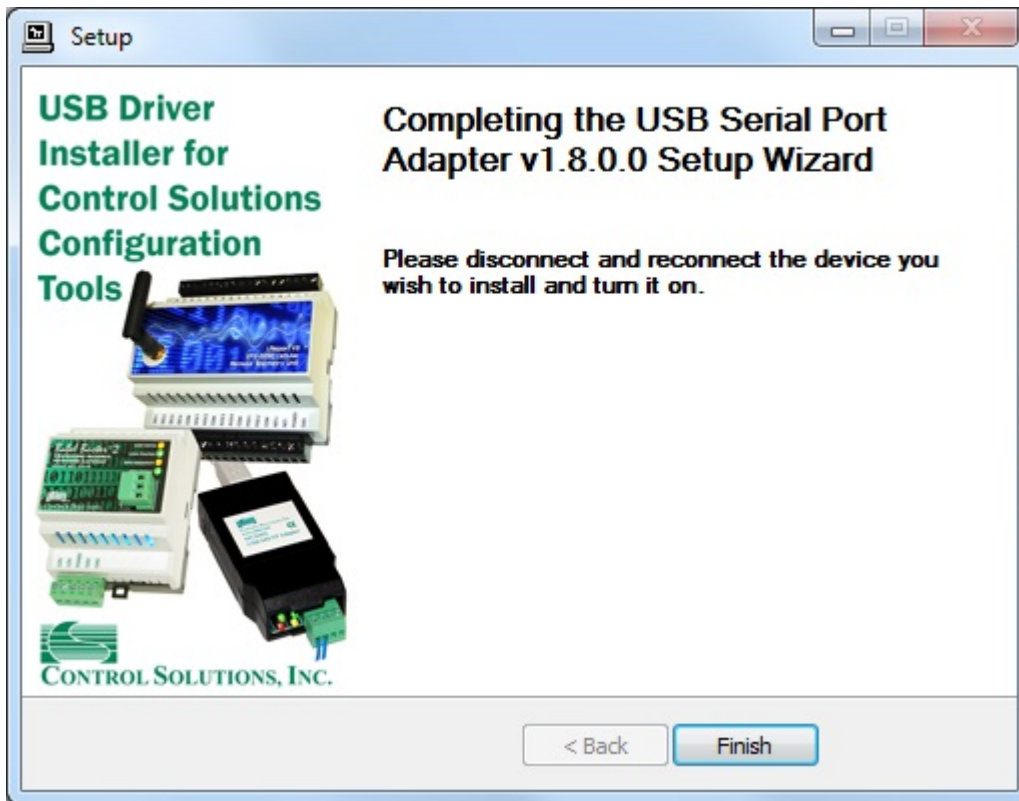


Click on "Install" when you get to this window:



When you get to this screen, you're done. Now plug in your USB device (MTX002, iReport, BB2-LON), allow the PC to finalize installation, and then go to the Device Manager via your PC's control panel to see which port the USB device got assigned to. Select this "COM" port in the Control

Solutions configuration tool's "Connect" page.



Appendix J Hardware Details

J.1 Service Button & USB Connection

Connect USB cable as illustrated. Install the driver package provided by Control Solutions prior to attempting to use the USB connection.

The service button is hidden behind the tab illustrated below. Press lightly on the white tab indicated. You will feel a slight clicking action when the button is pressed. It is not necessary to remove the cover to press the button - it is intended to be actuated by the plastic tab that is part of the cover.



J.2 Front Panel LED Indicators

Power-up LED behavior for Modbus RTU gateways: All LEDs on front panel will turn on yellow or red for half a second, then all will turn on green for half a second. Then they will proceed to indicate as normally defined for the indicators.

Power-up LED behavior for Modbus TCP gateway: Will behave the same as RTU, except the TCP request/reply behavior will be delayed by several seconds after power-up indication on the LonWorks LEDs.



The LonWorks indicator LEDs display LonWorks network activity. Note, however, that activity of these LEDs is not only affected by configuration of the device, but by whether the device is commissioned on the network and whether other devices are also communicating.

Mode	Data LED	Status LED
Wink	Alternates between yellow & green 10 times, then resume normal mode.	Alternates between red & green 10 times, then resume normal mode.
Normal	<p>Yellow flash indicates NV update was sent by gateway.</p> <p>Green flash indicates NV update was received by gateway.</p>	<p>Red indicates Neuron chip is not running.</p> <p>Green indicates gateway's host processor is communicating with LonWorks Neuron chip.</p>

The "wink" behavior is invoked by sending a wink command to the BB2 gateway via the LonWorks network. This is generally just a diagnostic to see if you are successfully communicating with the device via LonWorks. Other than the few seconds it takes to execute the wink, the device will always be in "normal" mode as far as LED indications are concerned in the table above.

The LED indicators on the front of the gateway also indicate Modbus errors. These are global indicators that do not tell you which device or which register is having trouble, but these indicators are a very quick way to observe whether there are problems, and also whether there is any activity at all.



The request and reply LEDs will indicate Modbus traffic as indicated in the table below.

The Ethernet traffic LED will indicate any traffic on the Ethernet network, and does not necessarily indicate Modbus TCP traffic. The traffic LED will typically be off more than on, flashing on each time traffic is indicated. If the traffic LED is on completely solid, the server is not running (normal for a half minute or so during startup).

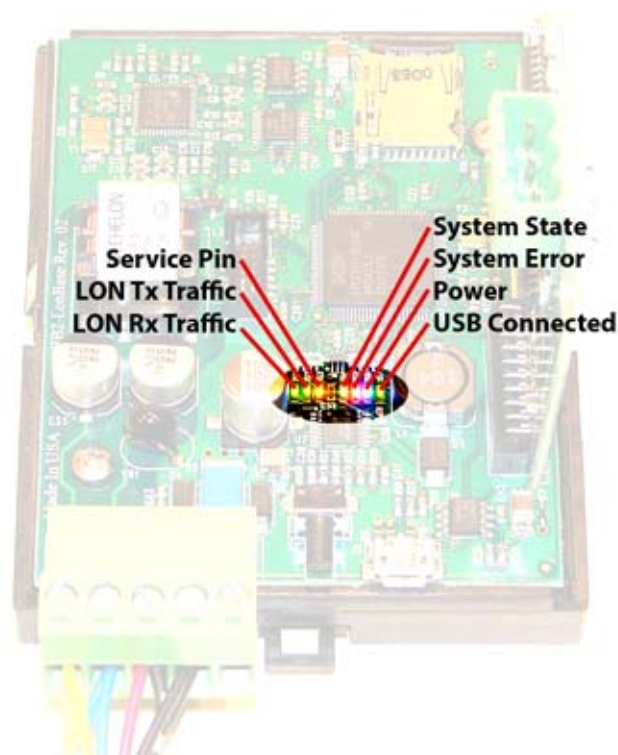
The Ethernet link LED will be on any time there is a connection to the network. If the Ethernet cable is unplugged, this light will go out. If Modbus TCP is failing and this light is out, check Ethernet cables.

Mode	Request LED	Reply LED
Gateway is Master	Flash yellow each time master (gateway) sends a request to a remote slave.	Flash green when master receives a good response. Flash red when master receives exception message from slave, or if timed out with no response from slave.
Gateway is Slave	Flash yellow each time slave (gateway) receives a request from external master.	Flash green when slave recognizes request as good/valid and sends a good reply. Flash red when slave receives a request that results in replying with an exception, or there was a CRC error (RTU only) in the request.
Gateway is Master	TCP only: If TCP is unable to make a connection with the IP address given for the TCP slave, the request LED will not flash yellow (because no request was sent yet), but the reply LED will flash red each time the connection attempt times out or fails.	

J.3 Internal Diagnostic LED Indicators

The internal diagnostic LEDs may be observed through the vent slots in the case. You normally have

minimal need to observe these, but if you are having trouble, you may want to check these.



Service Pin	Flashes yellow any time service pin message is sent by LonWorks Neuron.
LON Tx Traffic	Flashes green when message is transmitted by LonWorks Neuron.
LON Rx Traffic	Flashes green when message is received by LonWorks Neuron.
System State	Flashes green codes indicating system state, normally on with brief flash off once every 2-3 seconds as heartbeat indicator.
System Error	Flashes red error codes if a hardware fault has been detected. Normally off.
Power	Blue, should always be on, indicates power is present.
USB Connected	Turns on green when USB connection is made with PC.

During power-up, the blue LED should turn on immediately, and most other LEDs will flash briefly. The system state and system error LEDs will deliberately flash once, and the system state LED will typically flash a short flash more than once before resuming normal heartbeat indication. If a hardware fault has been detected, the red system error LED will continue to flash a code. During a firmware update, the indicators take on a different set of meanings, and these will be provided along with update instructions as applicable. If any abnormal indications are observed, contact technical support (www.csimn.com/ticket) for additional advice.

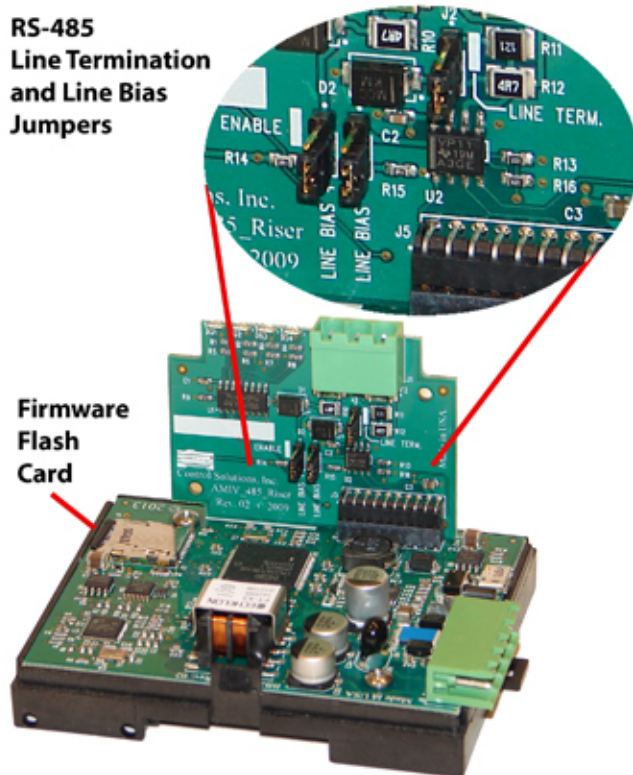
J.4 RS-485 Line Termination & Bias

Enable line termination only when this device is placed at the end of the network. Termination should only be enabled at two points on the network, and these two points must be specifically the end points.

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

start bit and cause loss of communication.

The line conditioning options are enabled when the respective shunt is moved to the position indicated by the white block next to the 3-pin header. Putting the shunt on the opposite 2 pins disables the option, and is simply a place to store the shunt.



J.5 Server Module Init Jumper

The "Init" jumper on the server module should only be used when advised by tech support. Installing this jumper prior to power-up causes the server to go into firmware update mode.

