The following is a printed copy of the help found in the configuration tool software and online.

BB2-2010-NB
BB2-2011-NB
BB2-6020-NB
LonWorks Modbus
Non-Bound Gateway
Rev. 1.0 – August 2015

User Guide

Babel Buster 2

User Guide Contents

User Guide

Babel Buster 2

BB2-2010-NB
BB2-2011-NB
BB2-6020-NB
LonWorks Modbus
Non-Bound Gateway
Rev. 1.0 – August 2015

1 Introduction

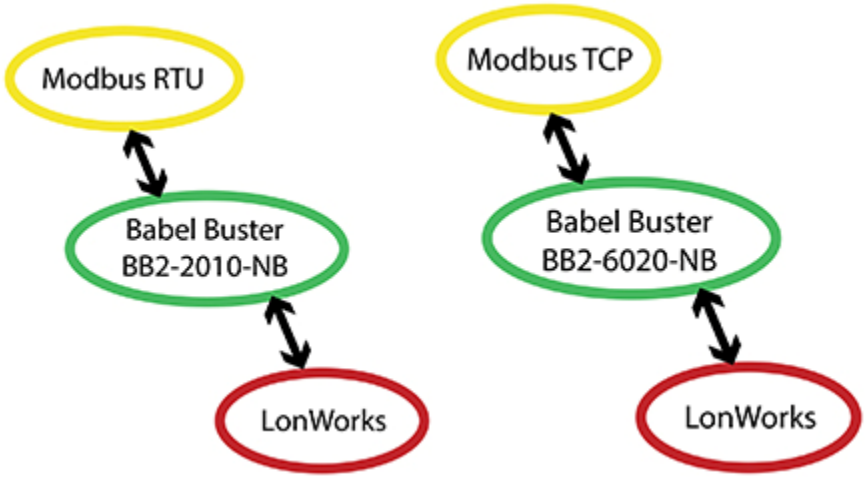
1.1 How to Use This Guide

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.

Note: While this user guide makes frequent reference to BB2-2010-NB, everything said about BB2-2010-NB also applies to BB2-2011-NB. The only difference between these models is whether the RTU line driver is RS485 or RS232.

1.2 Overview of Non-Bound Gateway Devices



What is different about BB2-2010-NB versus BB2-2010 (or BB2-6020-NB versus BB2-6020)? The LonMark certified BB2-2010 and BB2-6020 are designed for accessing Modbus devices from a LonWorks network. The "-NB" version, where NB stands for Non-Bound, is designed for accessing LonWorks devices from a Modbus network. Whereas the LonMark certified gateways require the use of a tool such as LonMaker to bind the gateways to the rest of the network, no such binding tool is required for the NB. The NB uses non-bound polling to access network variables in LonWorks devices and makes that data accessible as Modbus registers on the Modbus network. Device management in the NB is provided by the gateway along with the support of the configuration tool software provided at no charge by Control Solutions.

Babel Buster model BB2-2010-NB is a LonWorks to Modbus RTU gateway. It 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-NB 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-NB 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-NB, BB2-2011-NB, or BB2-6020-NB will have it functioning as a Modbus slave. 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.

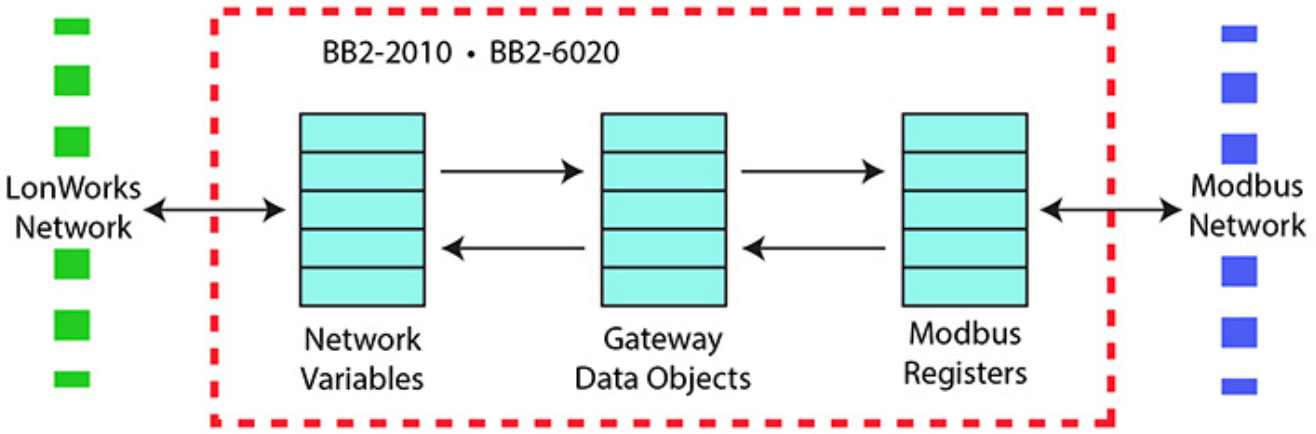
The BB2-2010-NB (or BB2-2011-NB) can be configured as a Modbus RTU master or slave (client or server), but can never be both at the same time (as specified by Modbus protocol). The BB2-6020-NB can be master and slave (client and server) at the same time since Modbus TCP supports this possibility.

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 three 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.

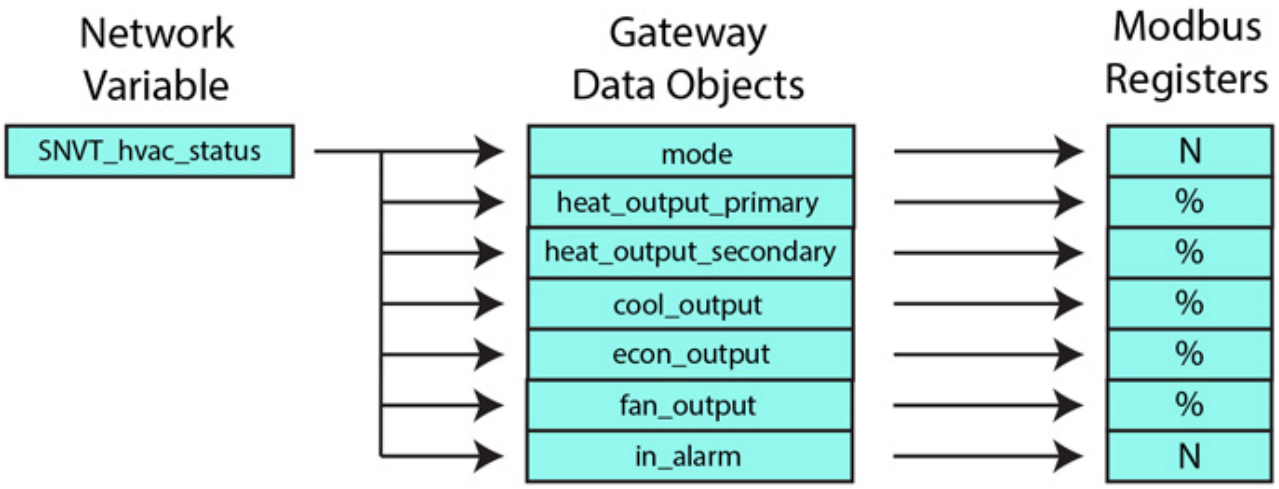
The Network Variable in the diagram below is LonMark type SNVT_temp. The LonMark specification for SNVTs (Standard Network Variable Types) provides the scaling for temperature transmitted over the LonWorks network as SNVT_temp. The raw data is a 16-bit binary number, but scaling provides 0.01 degree 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 Non-Bound Polling of Network Variables

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 NB version of the Babel Buster 2 LonWorks Gateway has only a Node Object to allow it to exist peacefully on a LonWorks network. It does not contain any bindable network variables (other than Node Object variables) because it only does non-bound polling of network variables in other devices.

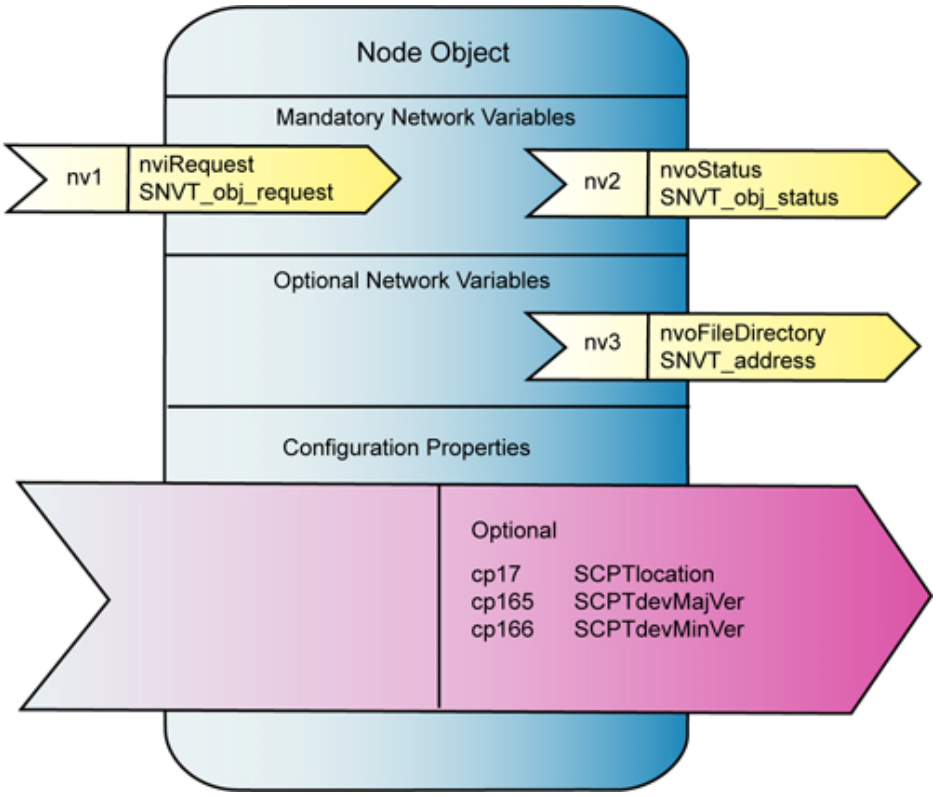
The polling of network variables in other devices is established by mappings in the gateway. Each of the gateway's internal data objects has a map telling it which LonWorks device to query, and which network variable in that device to access. Data is read from other devices using the network management "NV Fetch" request/response. Data is written to other devices using the network management "NV Update" request. Before the NB gateway can communicate with the other LonWorks device, it must establish the other device's network address, and this is done using the network management "Set Domain" request. There is more discussion related to device addressing in Section 12 and Appendix B.

The BB2-2010-NB, BB2-2011-NB, or BB2-6020-NB can poll up to 300 network variables from up to 50 other LonWorks devices.

Object Type	LON Object #	NV index
Node	0	0 .. 2

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 LonWorks network.


Node Object
Mandatory Network Variables
nv1
nviRequest
SNVT_obj_request
nv2
nvoStatus
SNVT_obj_status
Optional Network Variables
nv3
nvoFileDirectory
SNVT_address
Configuration Properties
Optional
cp17 SCPTlocation
cp165 SCPTdevMajVer
cp166 SCPTdevMinVer

3 How to Configure the Gateway

3.1 The Basics

The goal is to make one or more LonWorks devices accessible on a Modbus network, with the content of the LonWorks network variables showing up as Modbus registers. The configuration of the gateway's Modbus port is done on the Modbus Port page. But most of the work is involved in mapping LonWorks Network Variables to Modbus registers.

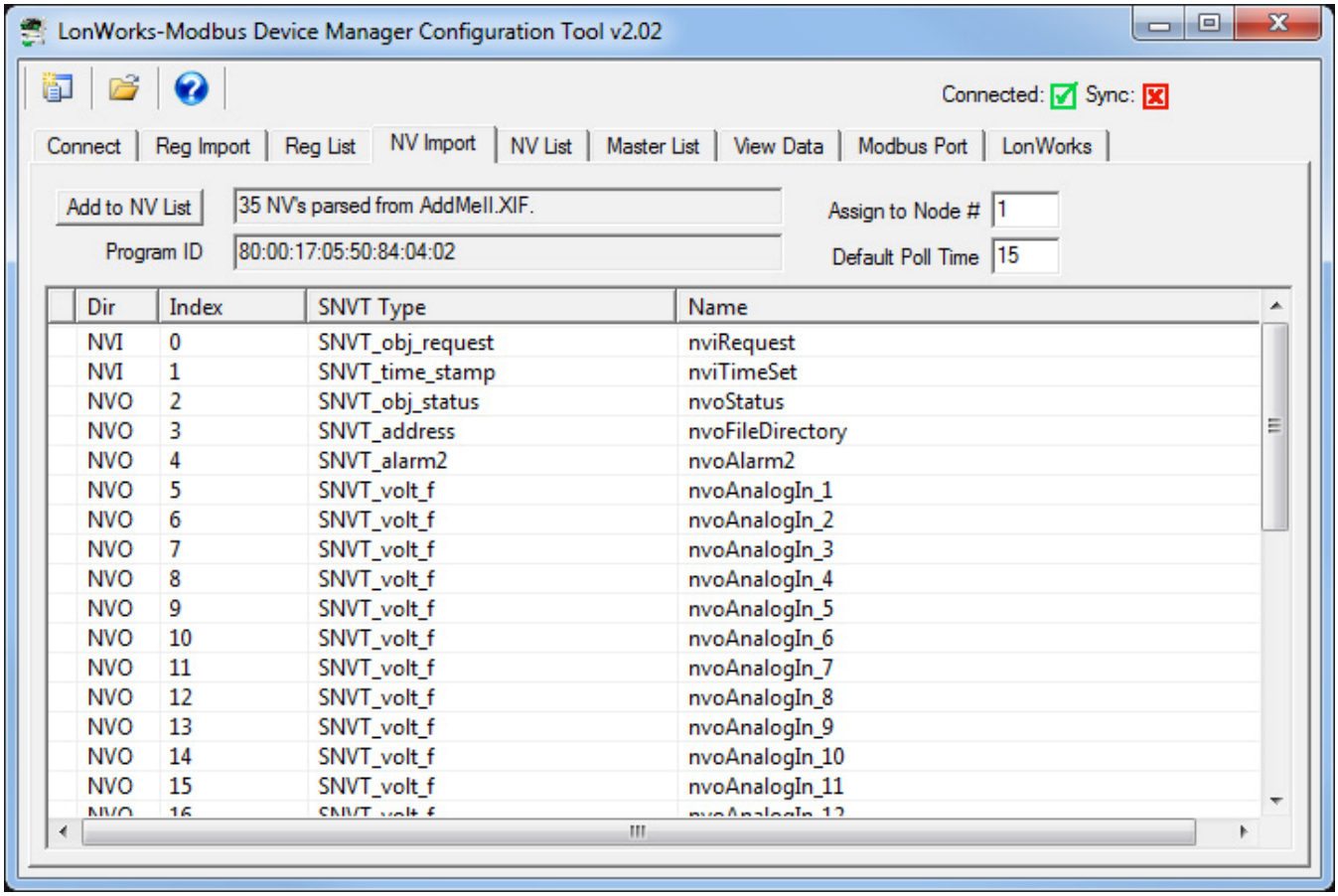
The Babel Buster gateway can be active or passive on the Modbus side, functioning as a master or slave. As a Modbus slave, the LonWorks data shows up in Modbus registers which some other Modbus device can then retrieve by reading holding registers. Your master can also access the LonWorks data as input registers, or coils and discrete inputs if data simply reflects on/off.

You have the option of making the gateway be a Modbus master to actively write data (received from LonWorks) to other Modbus slaves, or read data from other Modbus slaves and send it to the LonWorks device(s). If you want the gateway to do the polling, then you will need to configure the Modbus register mappings to tell the gateway which Modbus devices to talk to, and which Modbus registers in those devices the gateway should interact with. If the Babel Buster gateway will be a Modbus slave, then you can skip all of the discussion pertaining to Modbus register mappings on the Reg List page.

The topic you will spend most of your effort on is identifying the LonWorks network variables that you want to access from Modbus. Since some LonWorks network variables are structures with multiple data items per variable, you will sometimes need to map multiple data objects or registers (within the gateway) to a single LonWorks network variable. There are multiple possible starting points for configuring the gateway, and these are described in the following sections.

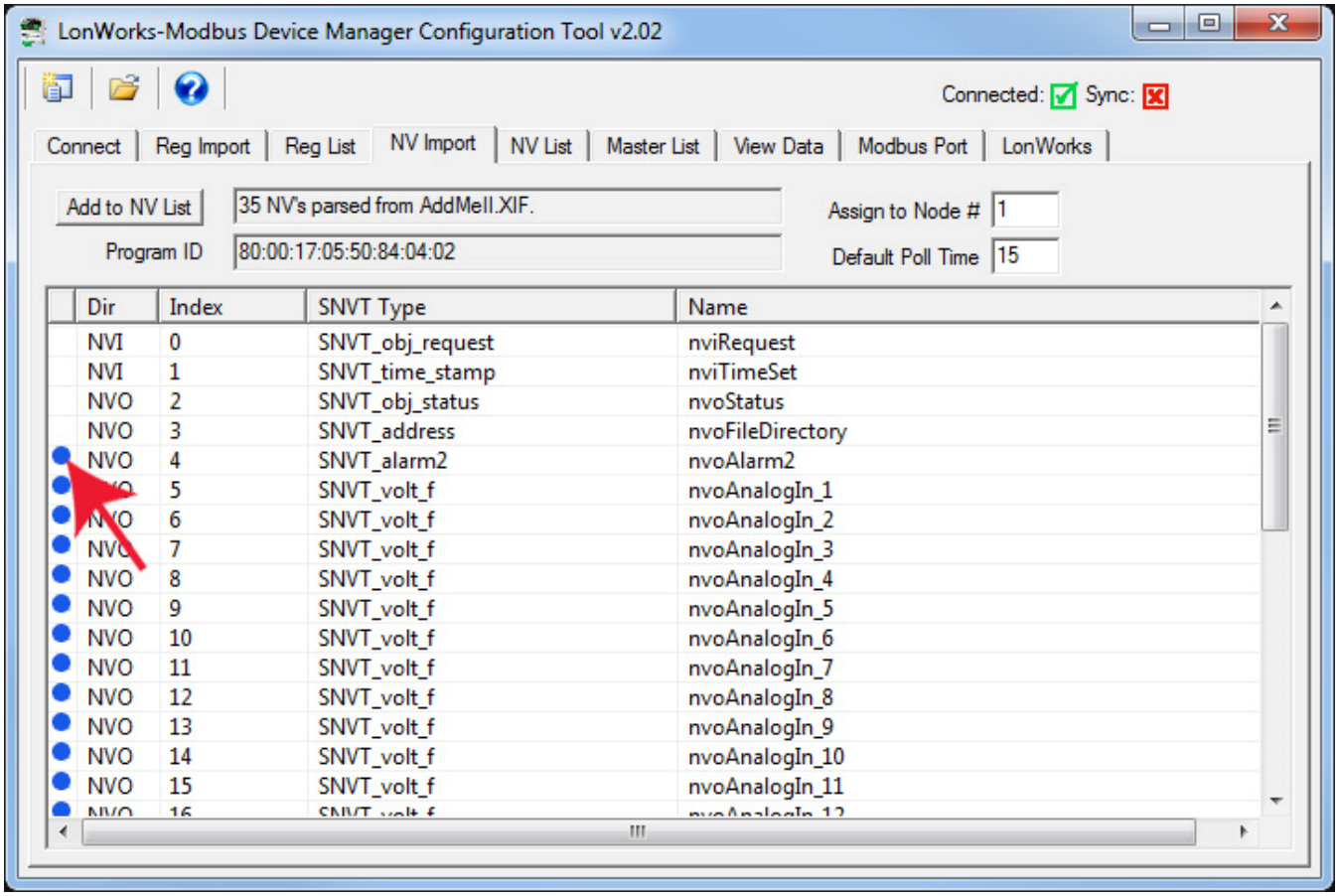
3.2 Build Configuration from XIF File for LonWorks Device

To begin building your configuration from an XIF file (obtained from the manufacturer of the device), 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.

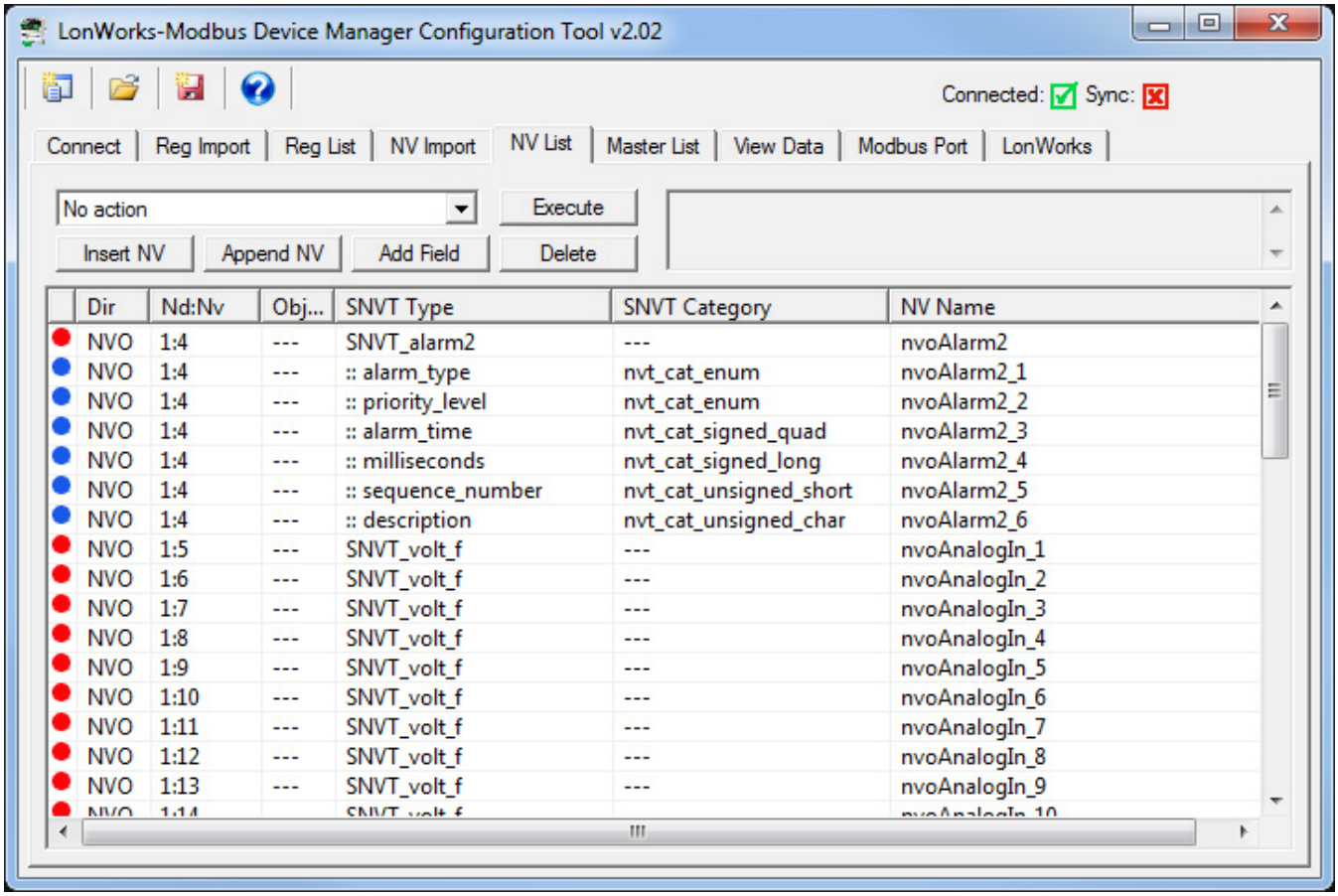


The NV Import page is essentially a scratch pad where you import the content of XIF files, then select which of the available variables you wish to include in your gateway configuration. Click on the icon column header to select all items, or click on the icon column for individual lines to select only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

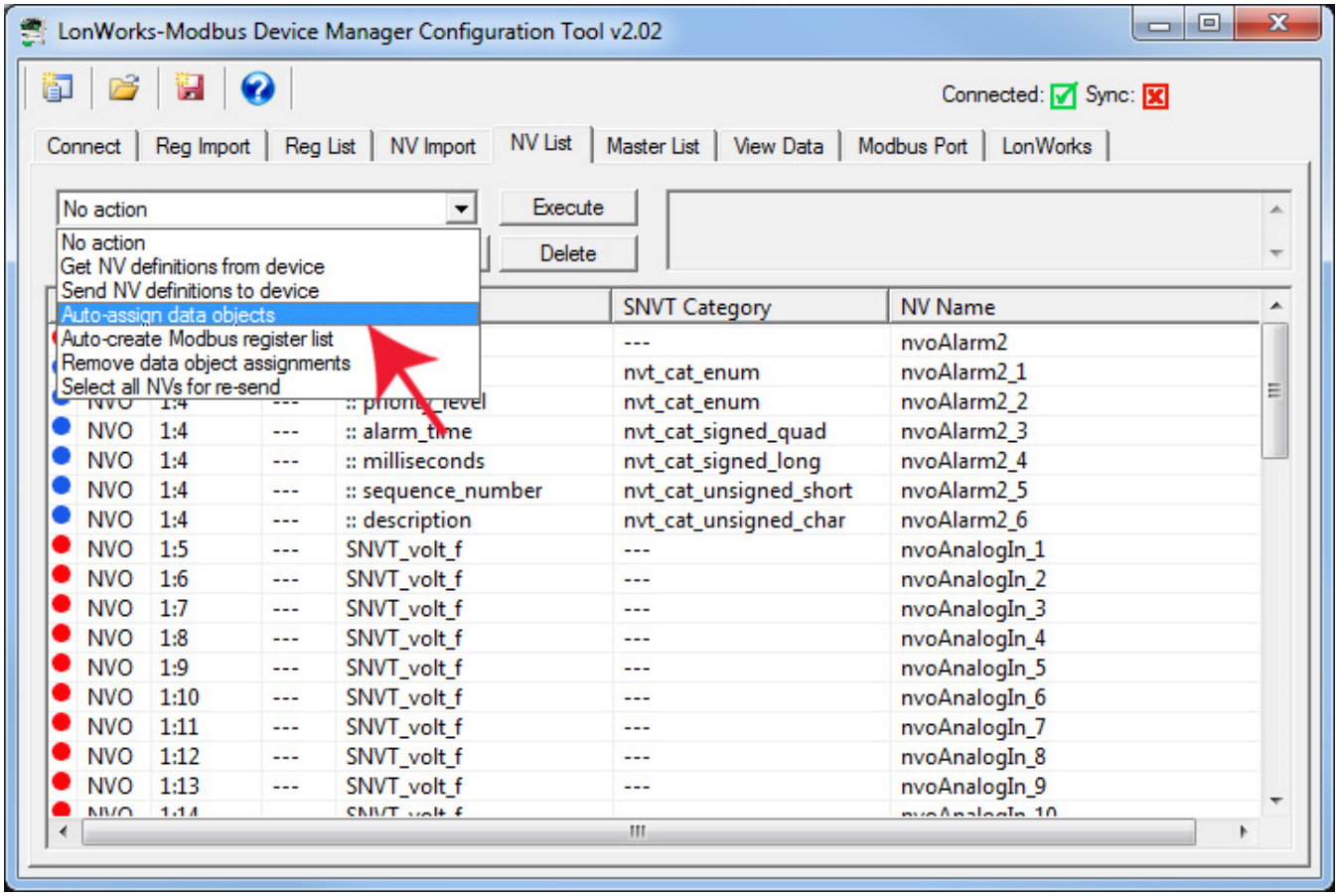
When you have made your selections, click Add to NV List. The selected variables are now copied to the NV List.



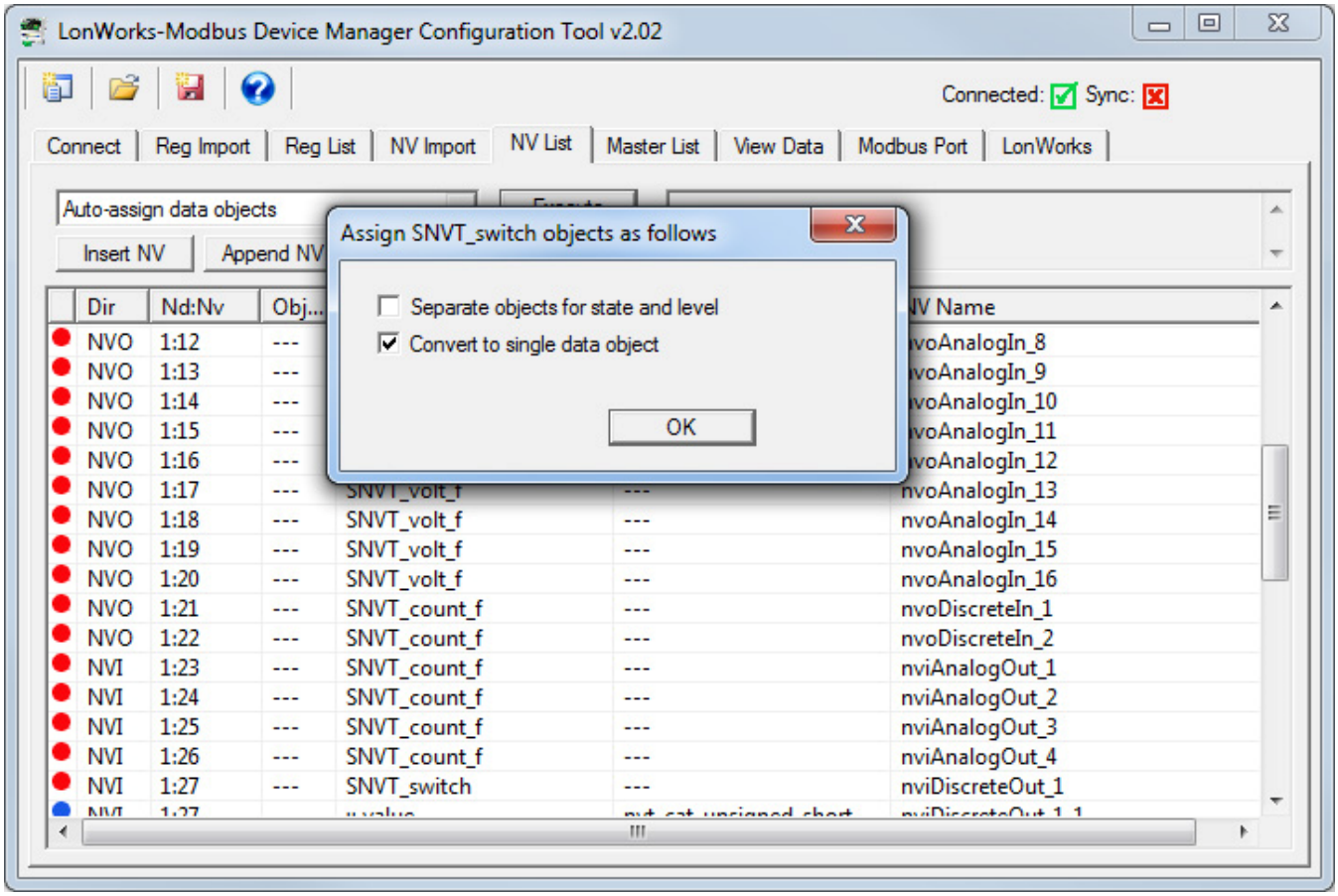
The NV List is the definition of the list of network variables found in other LonWorks devices that will be mapped to objects in your gateway. The icon in the first column will be red if this NV definition has not yet been written to the gateway, and green if it has been written to the gateway. Blue icons indicate fields of structured network variables. All lines with a blue icon are part of the network variable immediately preceding the set of blue icons. There will be only one network variable, but multiple data objects (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).



At this point, you have the option of editing the NV list and also the option of manually assigning variables to data objects. But the easiest way to quickly configure the gateway is to simply select "Auto-assign data objects" from the list and click Execute.



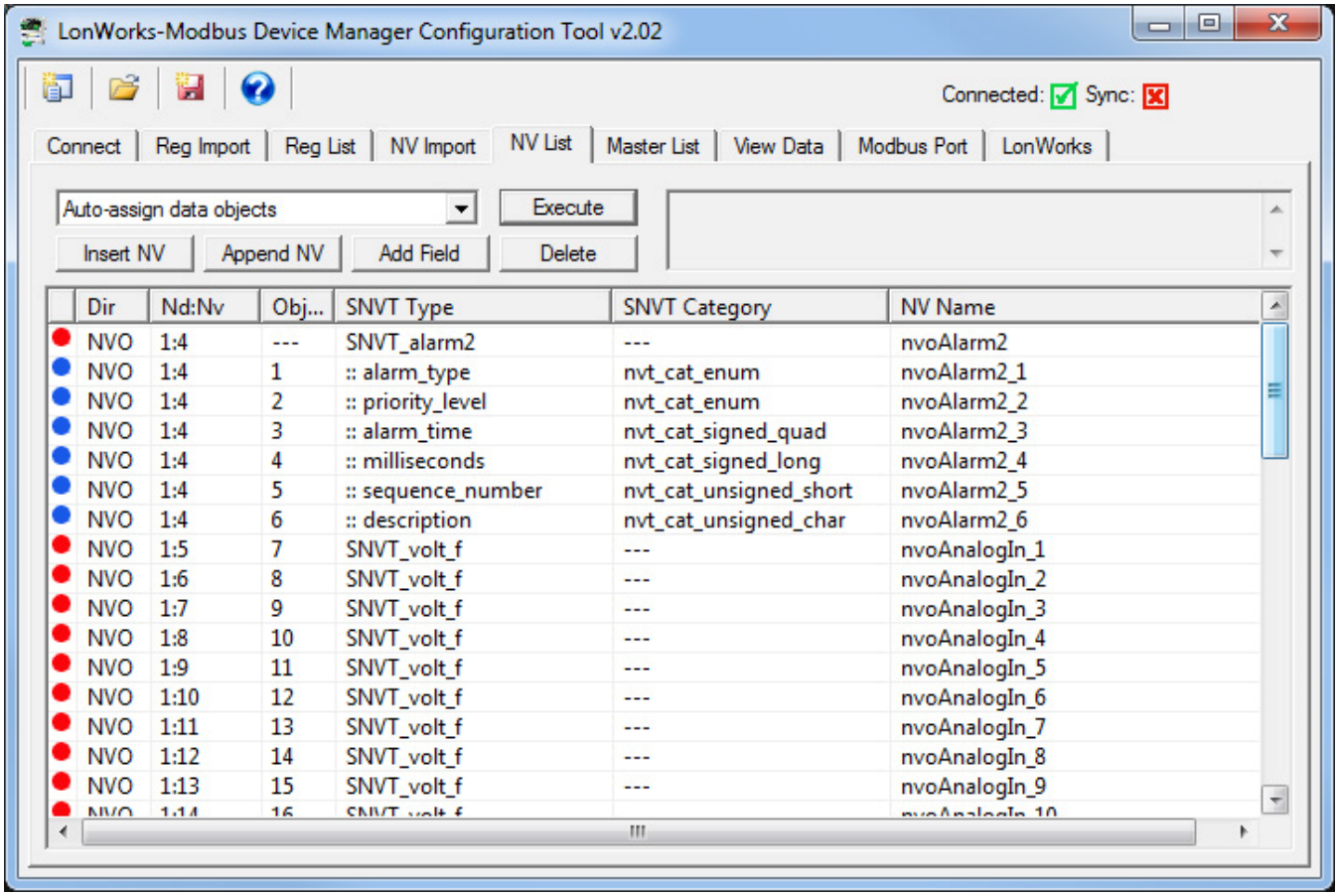
When auto-assigning objects, a dialog will pop up asking about your preference for treatment of any SNVT_switch variables that may be in the list. LonWorks treatment of switches assumes a dimmer type switch, and the SNVT_switch variable has both a state and a level, both of which must be provided on the LonWorks side. You have 2 options for how you will treat this on the Modbus side: (a) You can assign two objects (Modbus registers), one each for state and level; (b) You can assign a single data object which is then anticipating a value between 0 and 100 (percent implied) if accessed as a Modbus holding register, or simply on/off if accessed as a Modbus coil.



After object assignment, the Object column will be populated. The example below shows several network variables that will be polled in node #1 in the node table on the LonWorks page of the tool. The Nd:Nv column shows node number in that table, and NV index that will be queried in that node. The NV index value is found in the XIF file. If creating the NV list by other means, you will need to find out from the device manufacturer's documentation what the NV index is for the variables of interest. This is effectively the "address" of the variable in the LonWorks device.

The "Dir" column shows NVO for Network Variable Output, meaning the LonWorks device will transmit data to the LonWorks network, or NVI for Network Variable Input, meaning the LonWorks device expects to receive data from the LonWorks network.

Looking at nvoAnalogIn_1 as an example, the LonWorks device at node #1 will be providing a floating point data value representing voltage. The Babel Buster gateway will be polling this variable to read its data from the LonWorks device. Your Modbus master can then read the data in local object 1 and see that voltage data. Gateway data object 1 is accessed as Modbus integer holding register 1, floating point register pair at 2001, etc - see Appendix E.

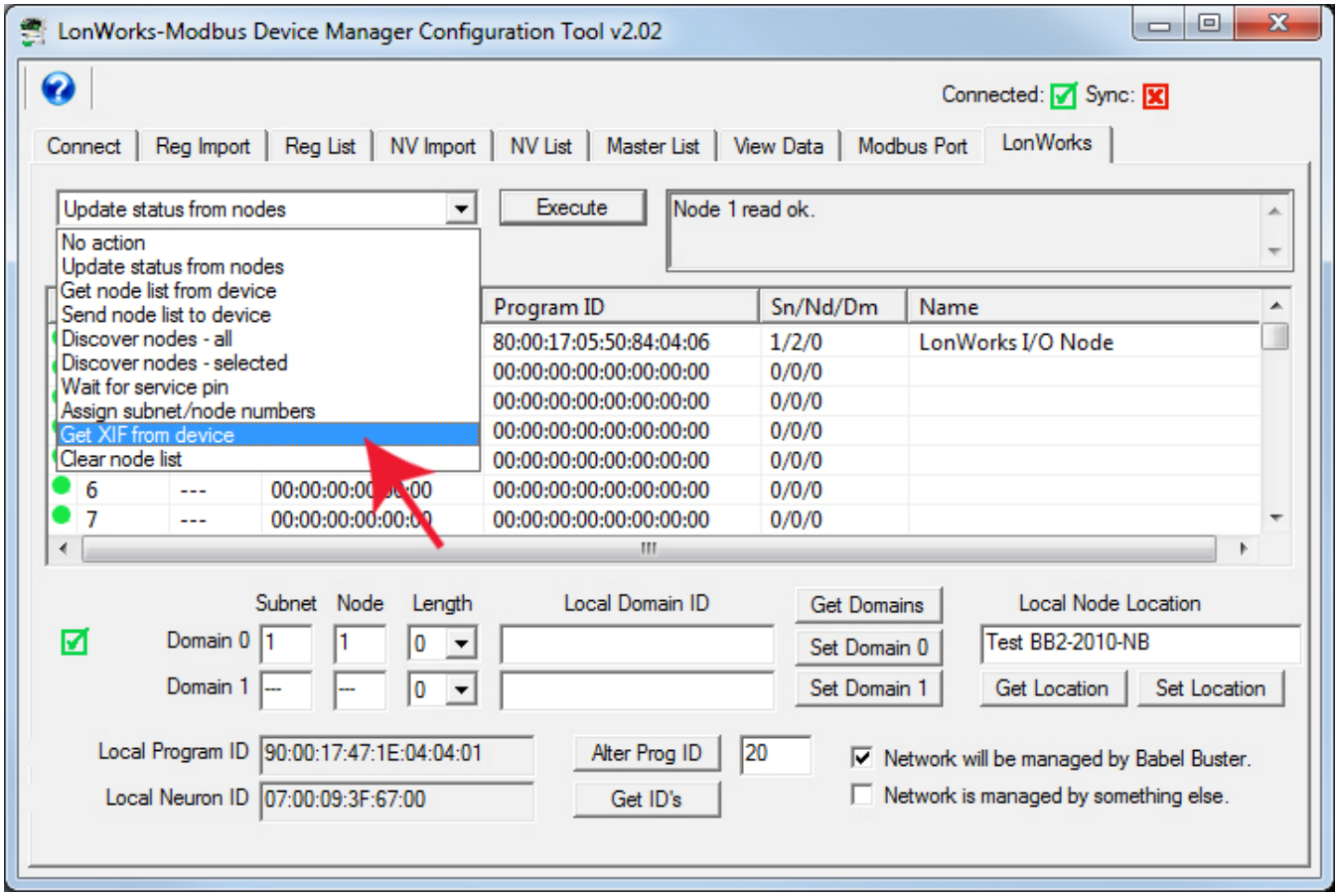


3.3 Build Configuration by Importing XIF File from LonWorks Device

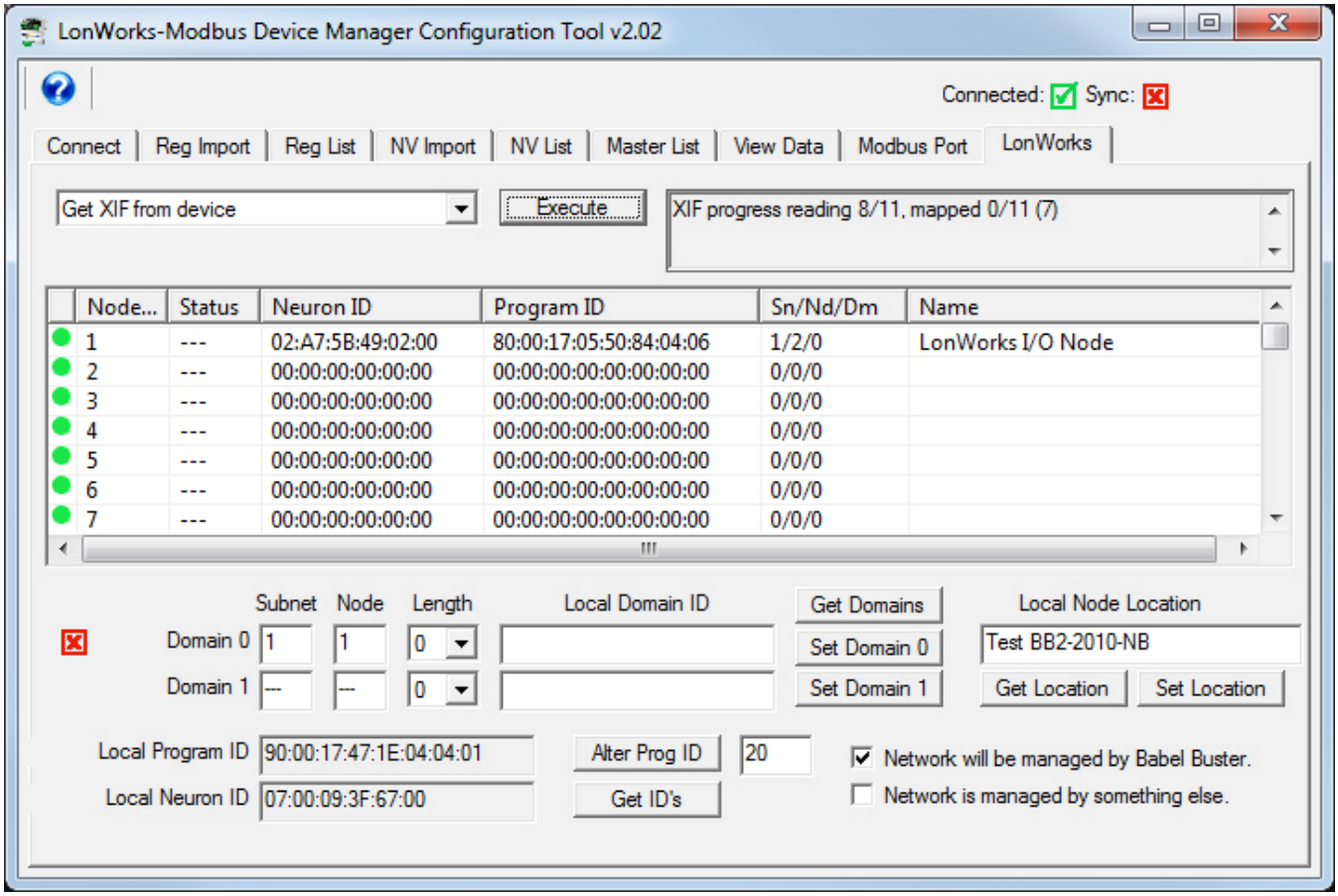
You can usually import the contents of an XIF file from the device itself in the event you do not have ready access to the XIF file. If the device contains more than 300 network variables, then importing from the device will exceed the capacity of the gateway. Also, it is known that some devices block import of the XIF file from the device itself. Therefore, while it is worth a try and will succeed in the majority of cases, there is no guarantee.

You will also need to first connect with the node using the procedures outlined for the LonWorks page. If you cannot communicate with the node, you cannot import its XIF information. Therefore, you must establish communication first (whereas if you have the XIF file, you do not need to be connected to the device at all to initially create a gateway configuration).

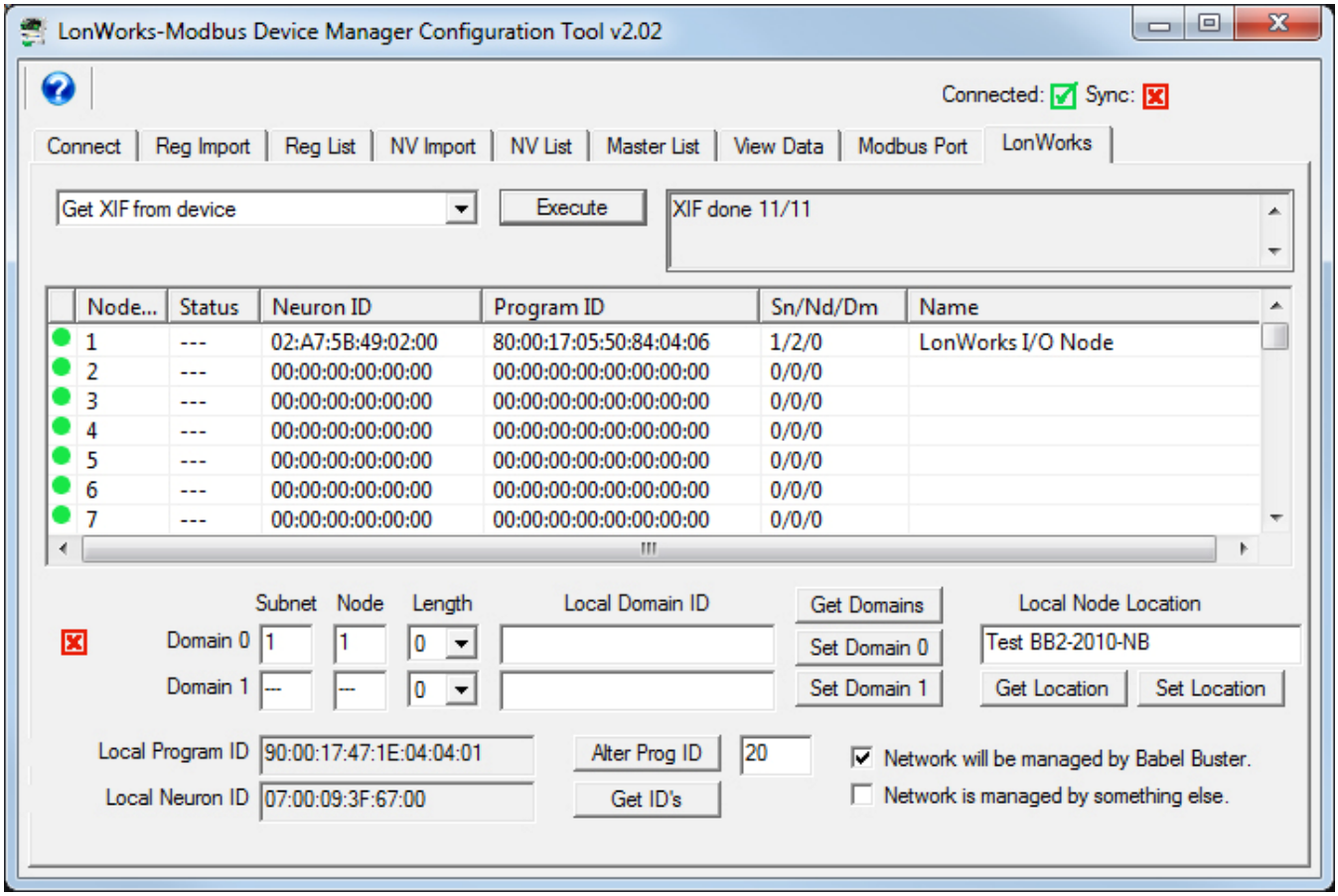
Assuming you have already connected with the node to be imported, and its status is 'Ready', select 'Get XIF from device' from the list on the LonWorks page and click Execute.



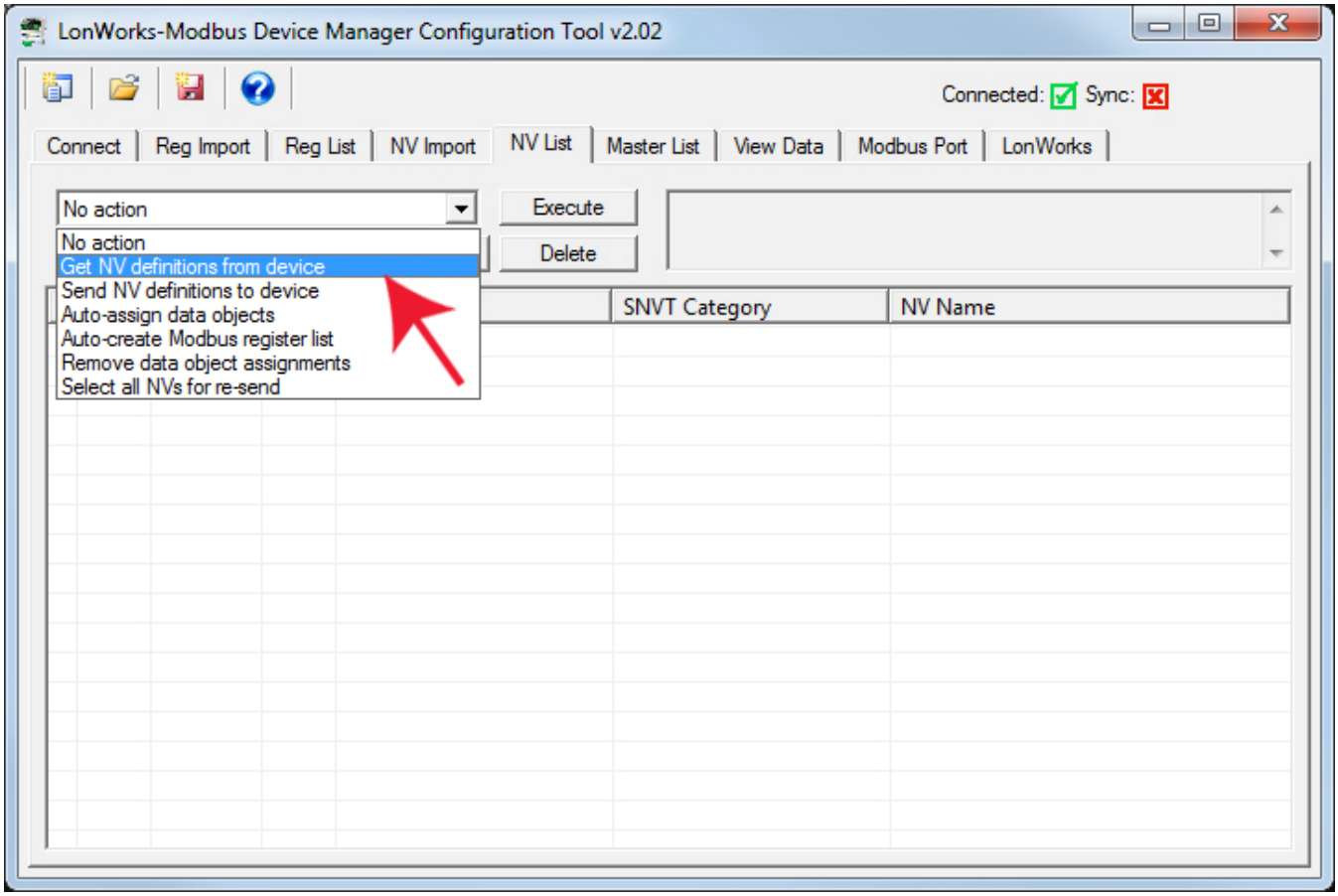
During the XIF import process, progress will be updated in the tool status window from time to time. The process usually involves multiple passes, so do not be alarmed if progress counts revert to zero and begin counting up again. In most cases, three passes are required.



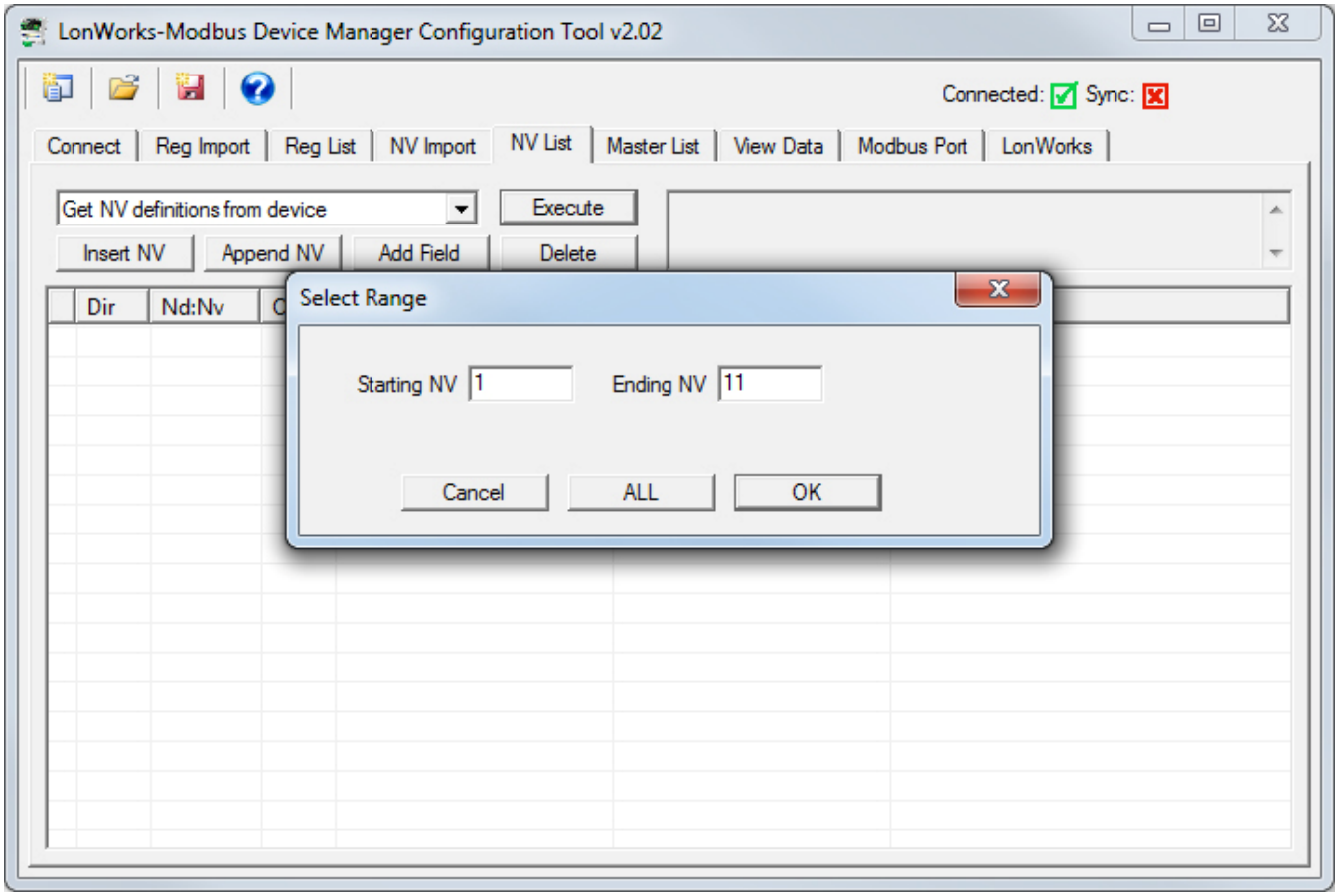
Once complete, the message "XIF done" will appear, along with an indication of how many Network Variables were imported.



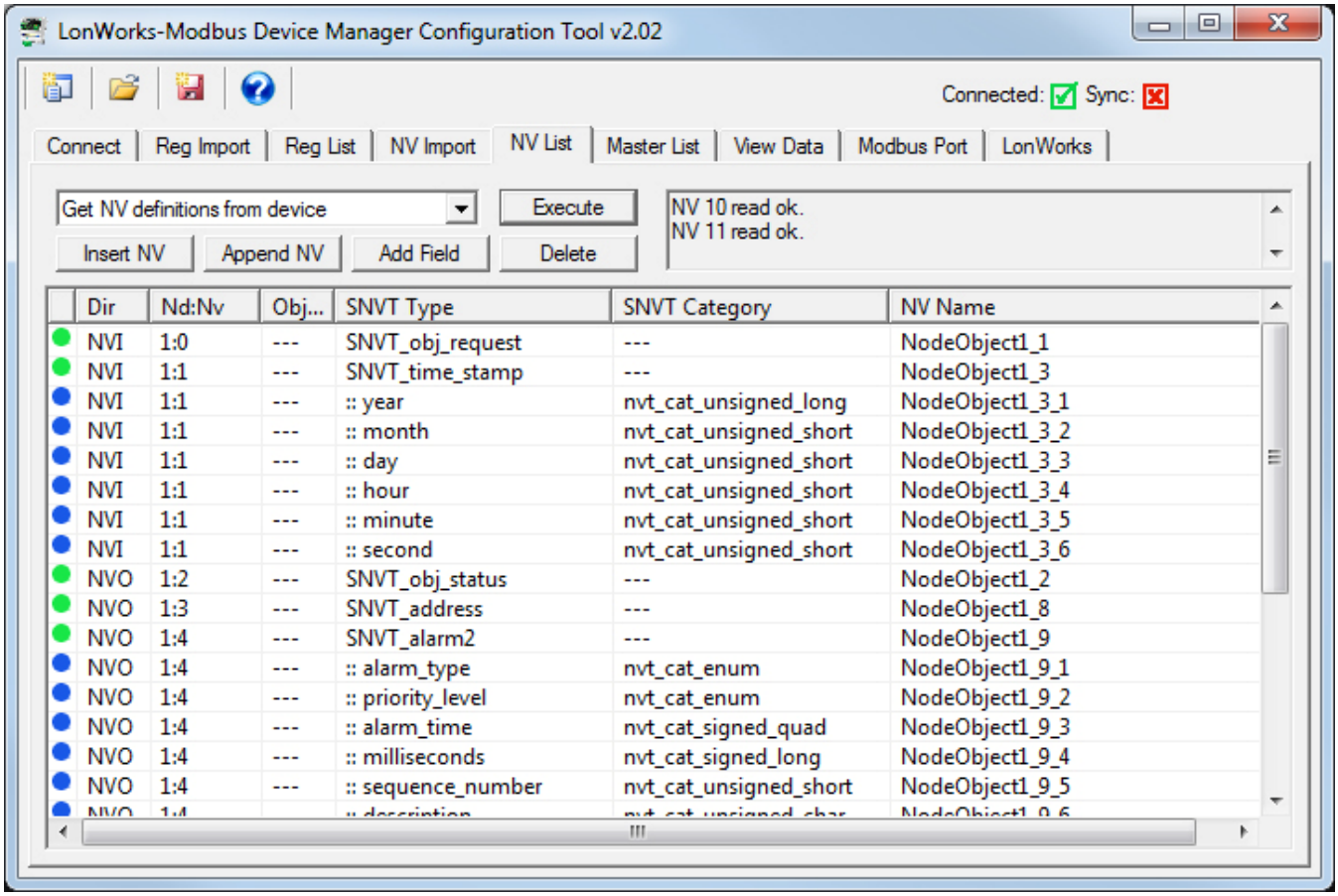
The XIF import process filled up the NV table in the gateway device. To work with this information, you now need to retrieve the NV List from the gateway device. Select 'Get NV definitions from device' and click Execute.



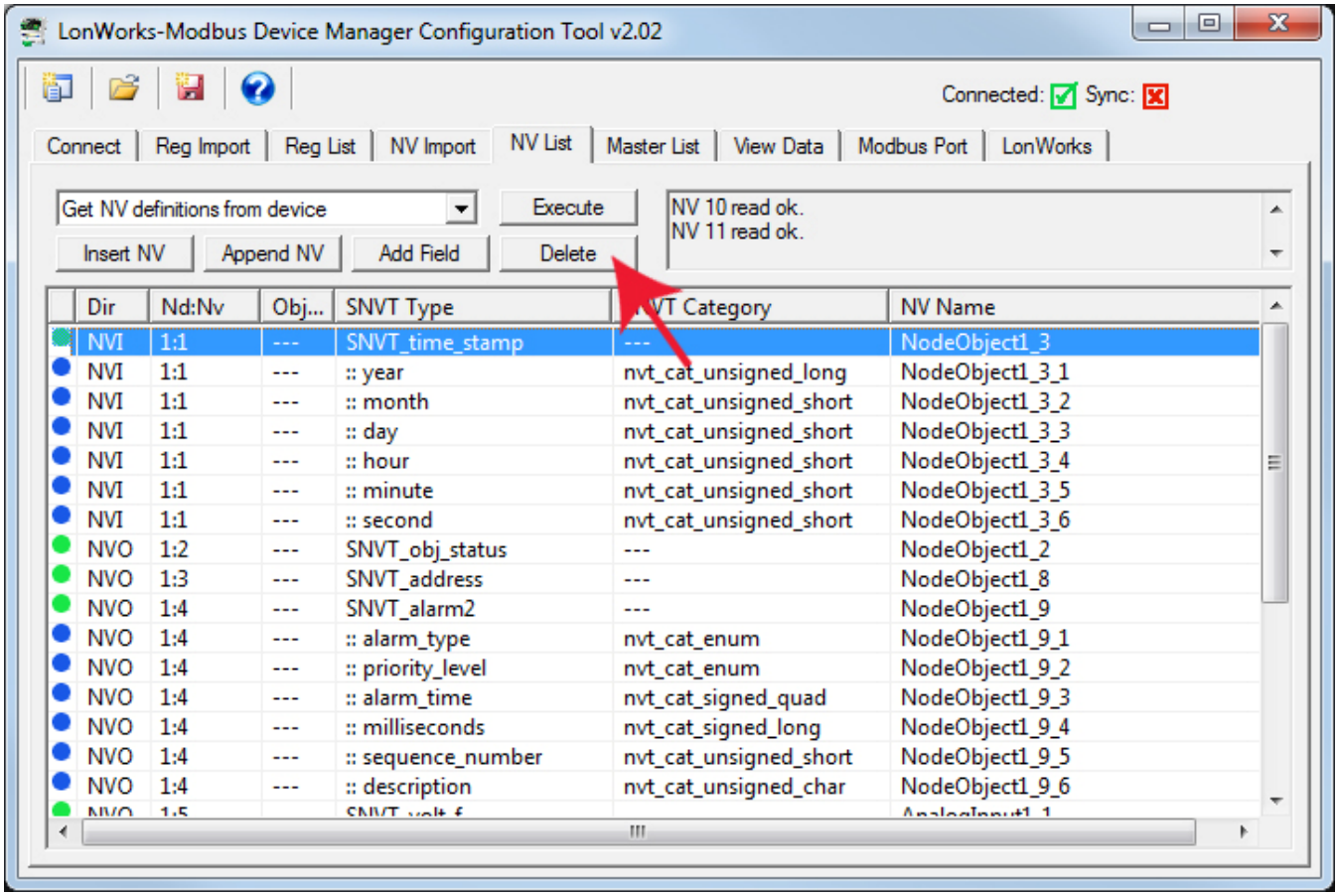
Upon exeuting 'Get NV definitions from device', a dialog will pop up asking for the range of network variables to retrieve. It will default to all of those found in the device. Usually you can just click OK here. If you have imported more than one device, and only want to retrieve information from the most recent import, then assuming you kept track of previous counts, you can enter a range less than all available.



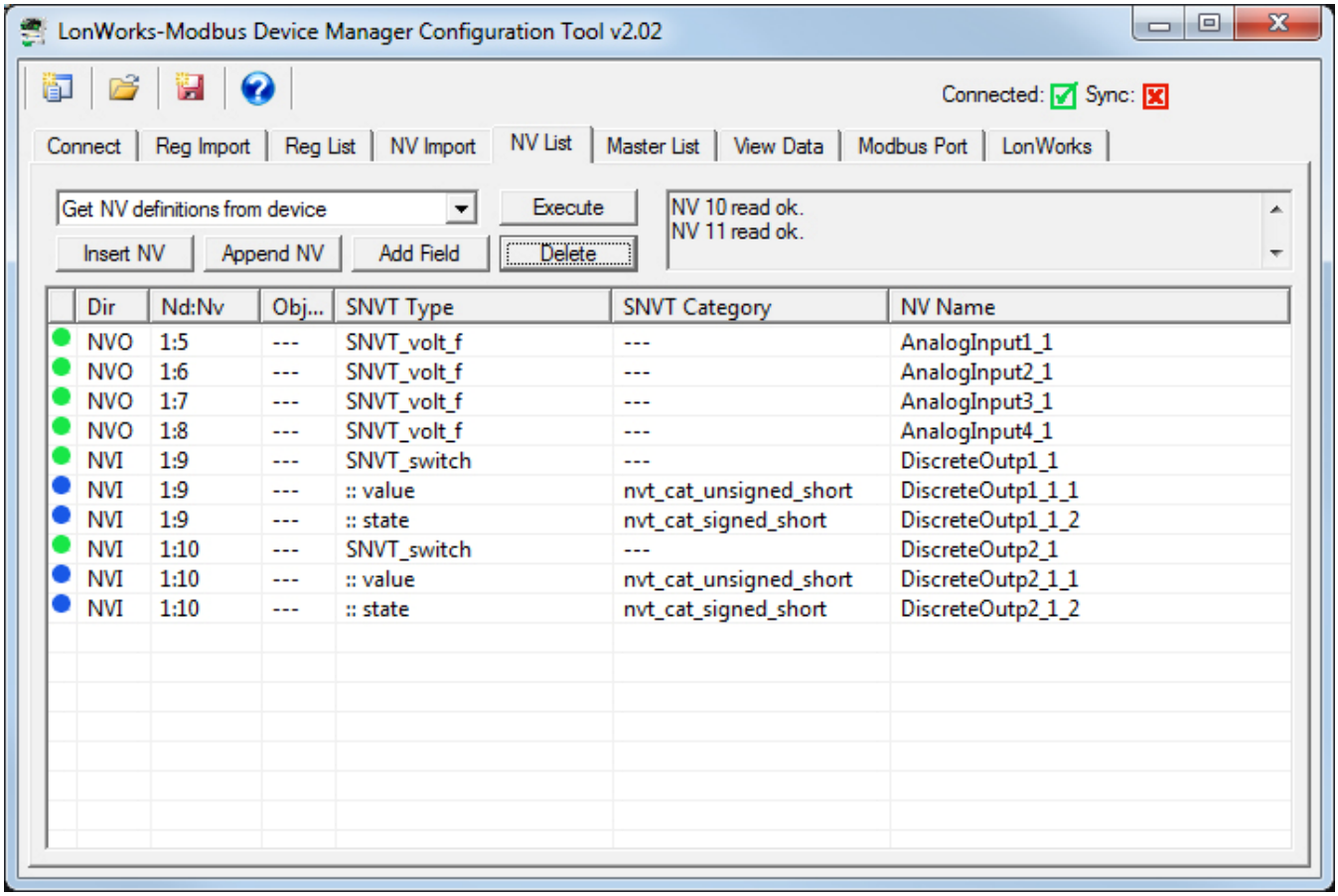
Once the process of retrieving the imported NV List is complete, the process is largely the same as if you had read the XIF from a file.



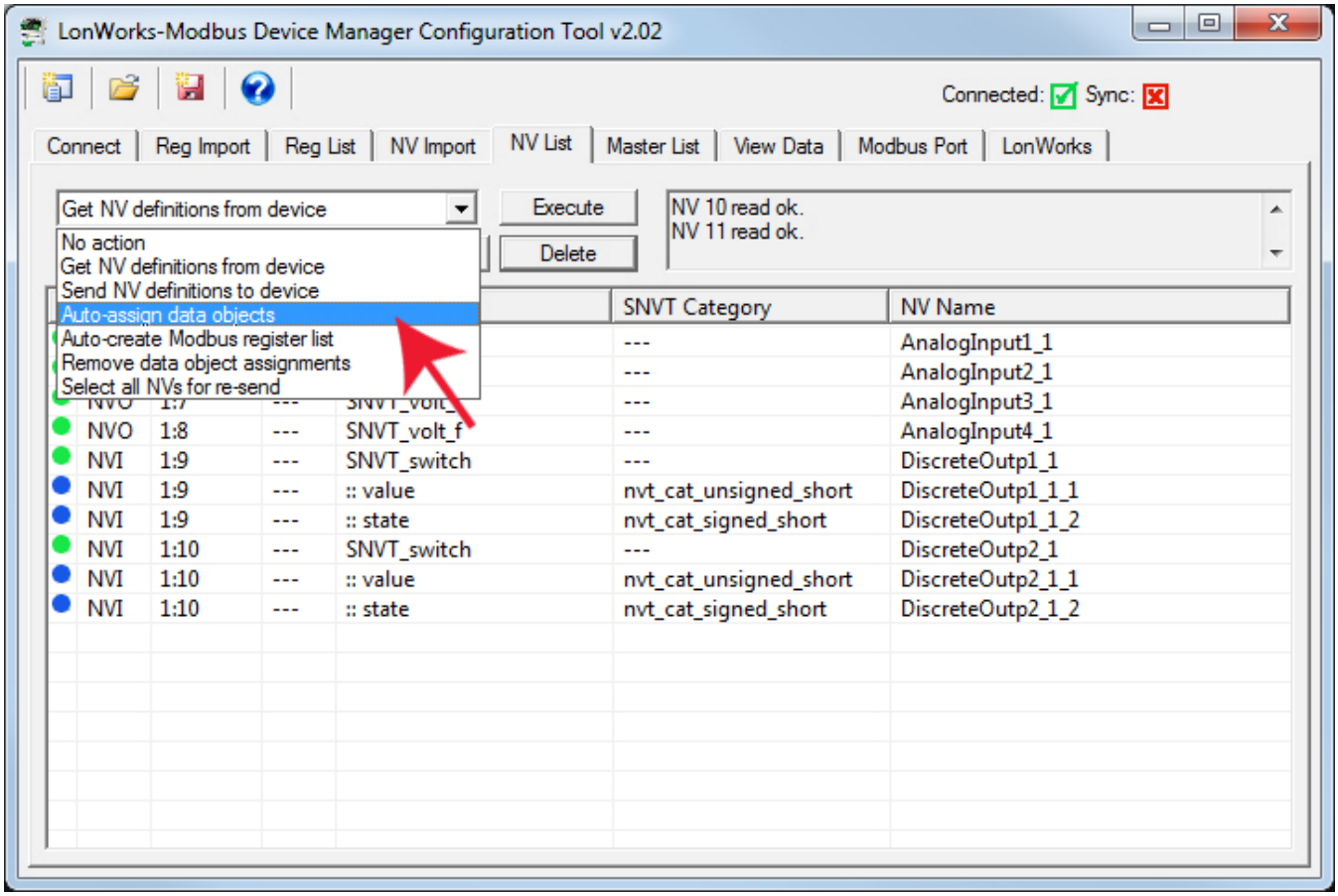
The device may contain more Network Variables than you care to process as Modbus registers. You can select those variables and click Delete to remove them from your configuration.



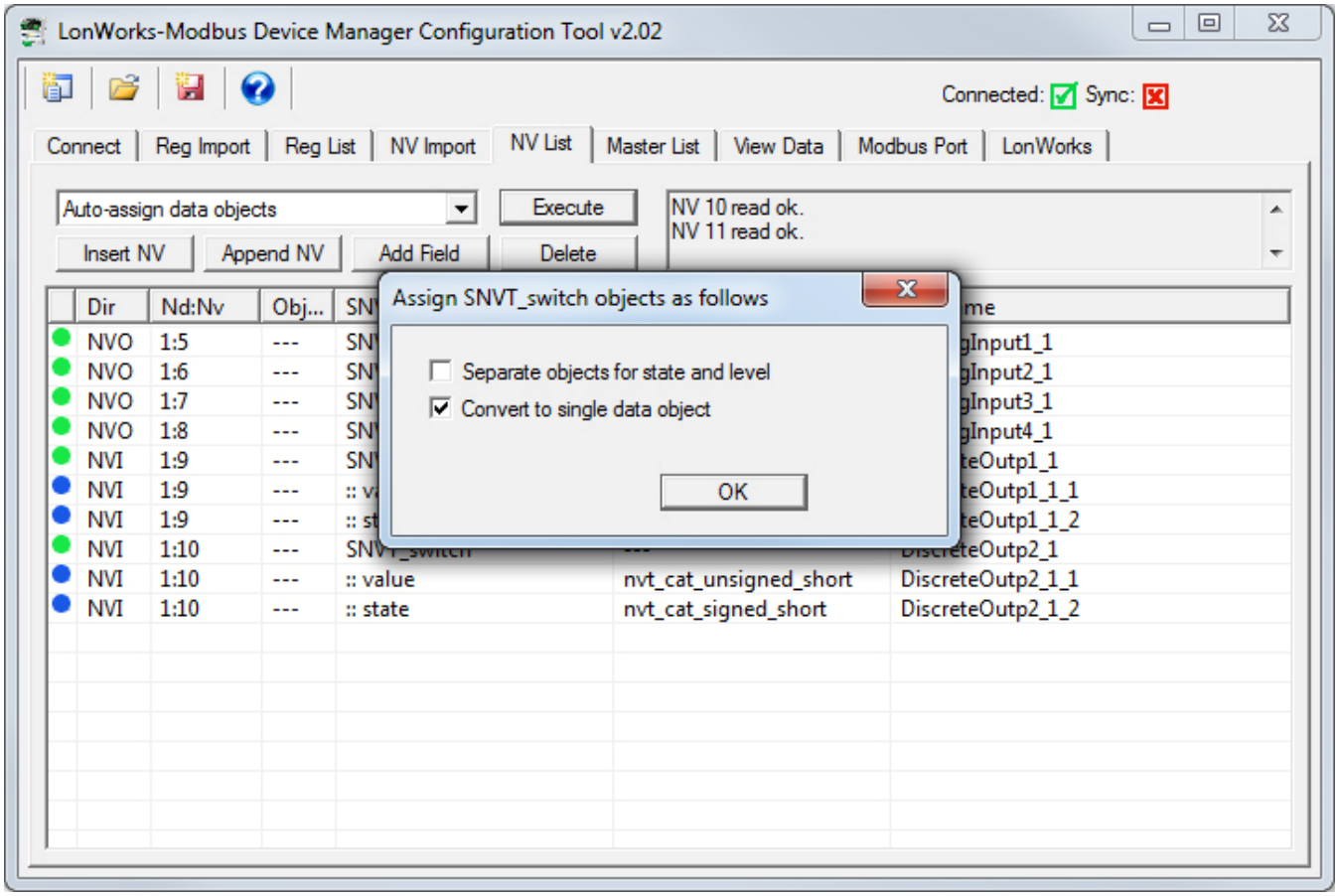
In this example, we have eliminated variables we don't care about, such as node object status (SNVT_obj_status). The list that remains is the actual I/O point list in this particular device.



At this point, you have the option of manually assigning variables to data objects. But the easiest way to quickly configure the gateway is to simply select "Auto-assign data objects" from the list and click Execute.



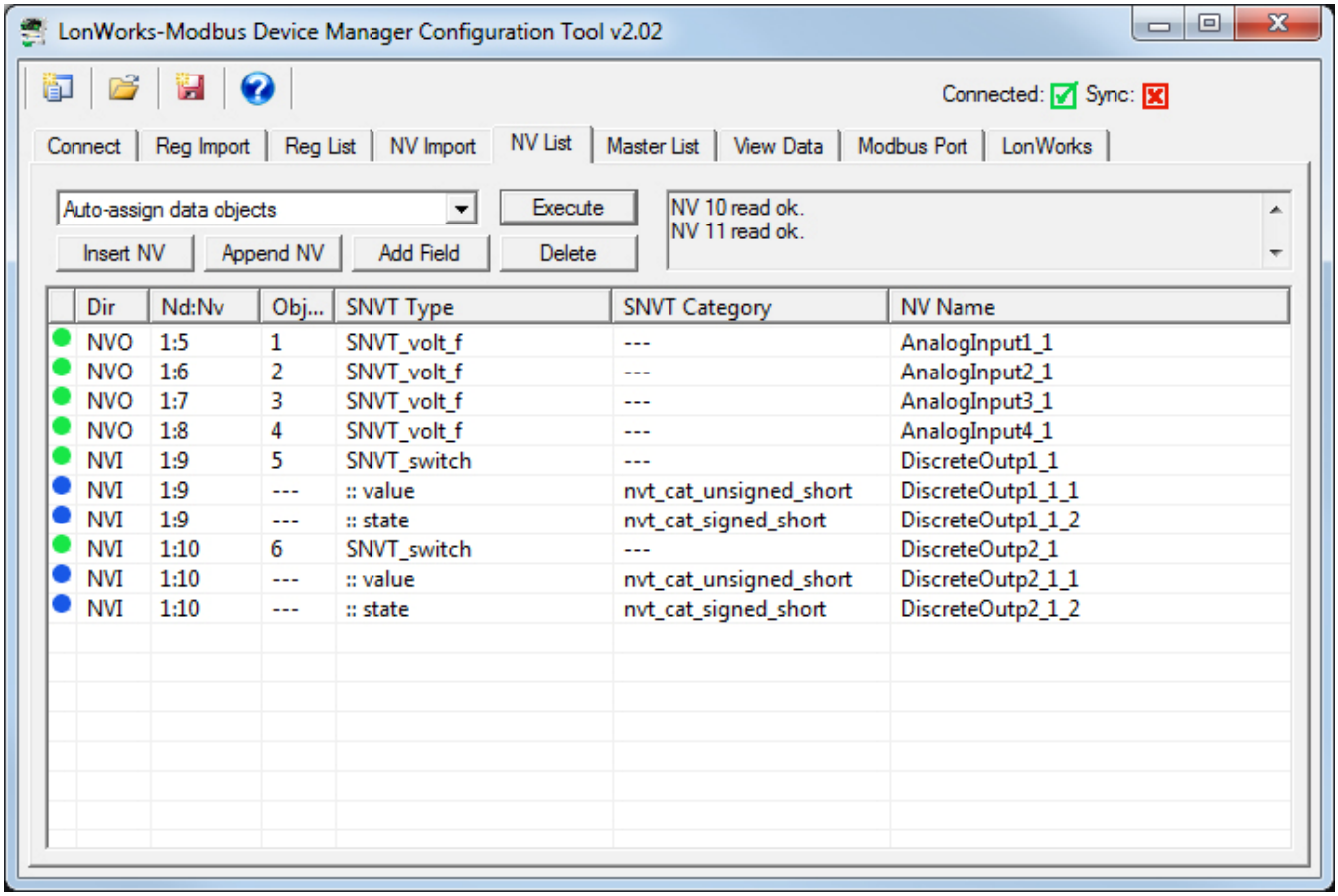
When auto-assigning objects, a dialog will pop up asking about your preference for treatment of any SNVT_switch variables that may be in the list. LonWorks treatment of switches assumes a dimmer type switch, and the SNVT_switch variable has both a state and a level, both of which must be provided on the LonWorks side. You have 2 options for how you will treat this on the Modbus side: (a) You can assign two objects (Modbus registers), one each for state and level; (b) You can assign a single data object which is then anticipating a value between 0 and 100 (percent implied) if accessed as a Modbus holding register, or simply on/off if accessed as a Modbus coil.



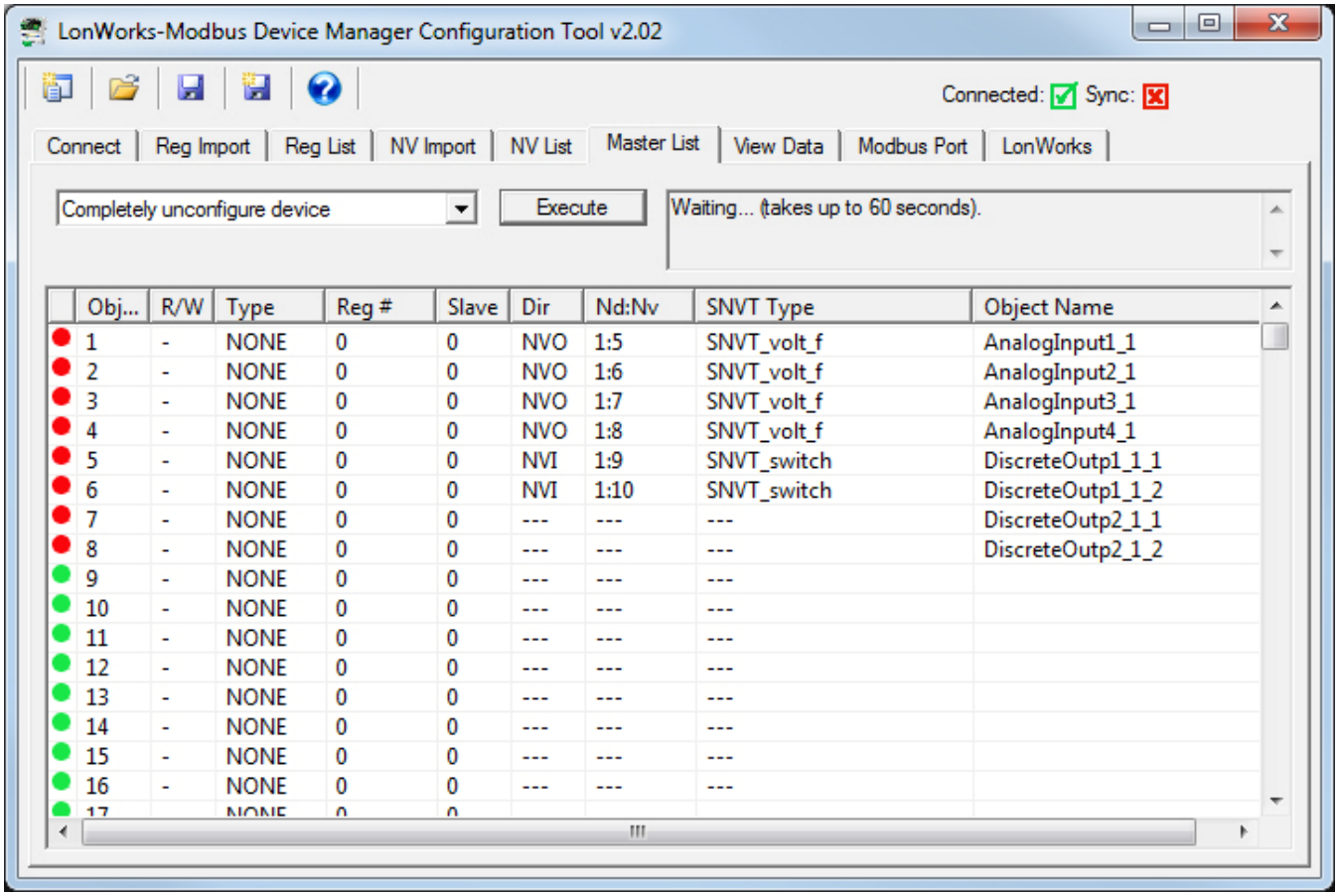
After object assignment, the Object column will be populated. The example below shows several network variables that will be polled in node #1 in the node table on the LonWorks page of the tool. The Nd:Nv column shows node number in that table, and NV index that will be queried in that node. The NV index value is found in the XIF file. If creating the NV list by other means, you will need to find out from the device manufacturer's documentation what the NV index is for the variables of interest. This is effectively the "address" of the variable in the LonWorks device.

The "Dir" column shows NVO for Network Variable Output, meaning the LonWorks device will transmit data to the LonWorks network, or NVI for Network Variable Input, meaning the LonWorks device expects to receive data from the LonWorks network.

Looking at nvoAnalogIn_1 as an example, the LonWorks device at node #1 will be providing a floating point data value representing voltage. The Babel Buster gateway will be polling this variable to read its data from the LonWorks device. Your Modbus master can then read the data in local object 1 and see that voltage data. Gateway data object 1 is accessed as Modbus integer holding register 1, floating point register pair at 2001, etc - see Appendix E.

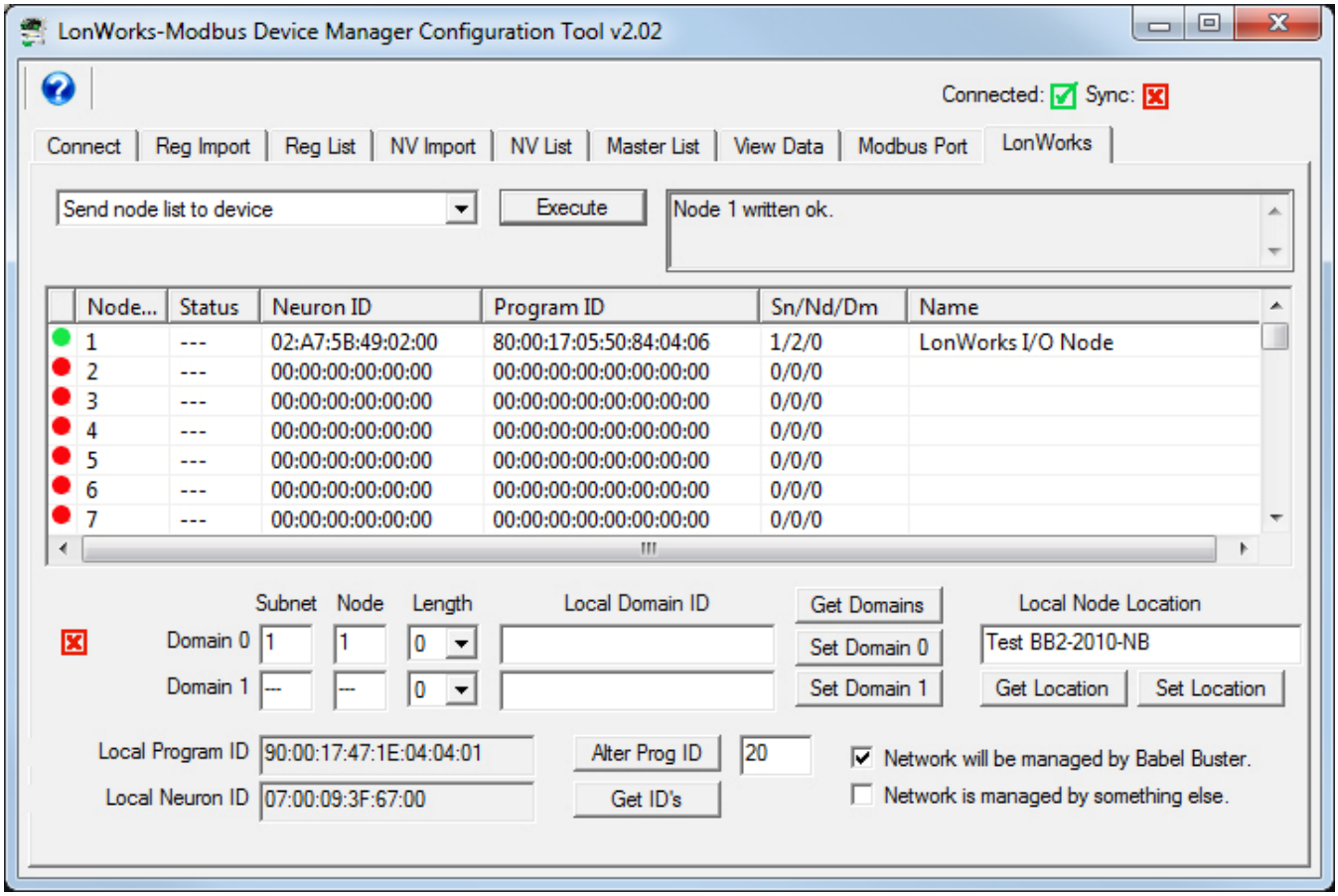


There is one IMPORTANT bit of housekeeping to do here before sending your revised NV List to the gateway device. The gateway itself still contains the originally imported list of variables. You don't want those. Therefore, select 'Completely unconfigure device' from the list and click Execute. This is the quicker way to clear the list. The other option is to select ALL when sending the NV List to the gateway device. Sending all 300 NV entries will take some time, but will make sure that the NV List in the device matches what you see in the configuration tool.

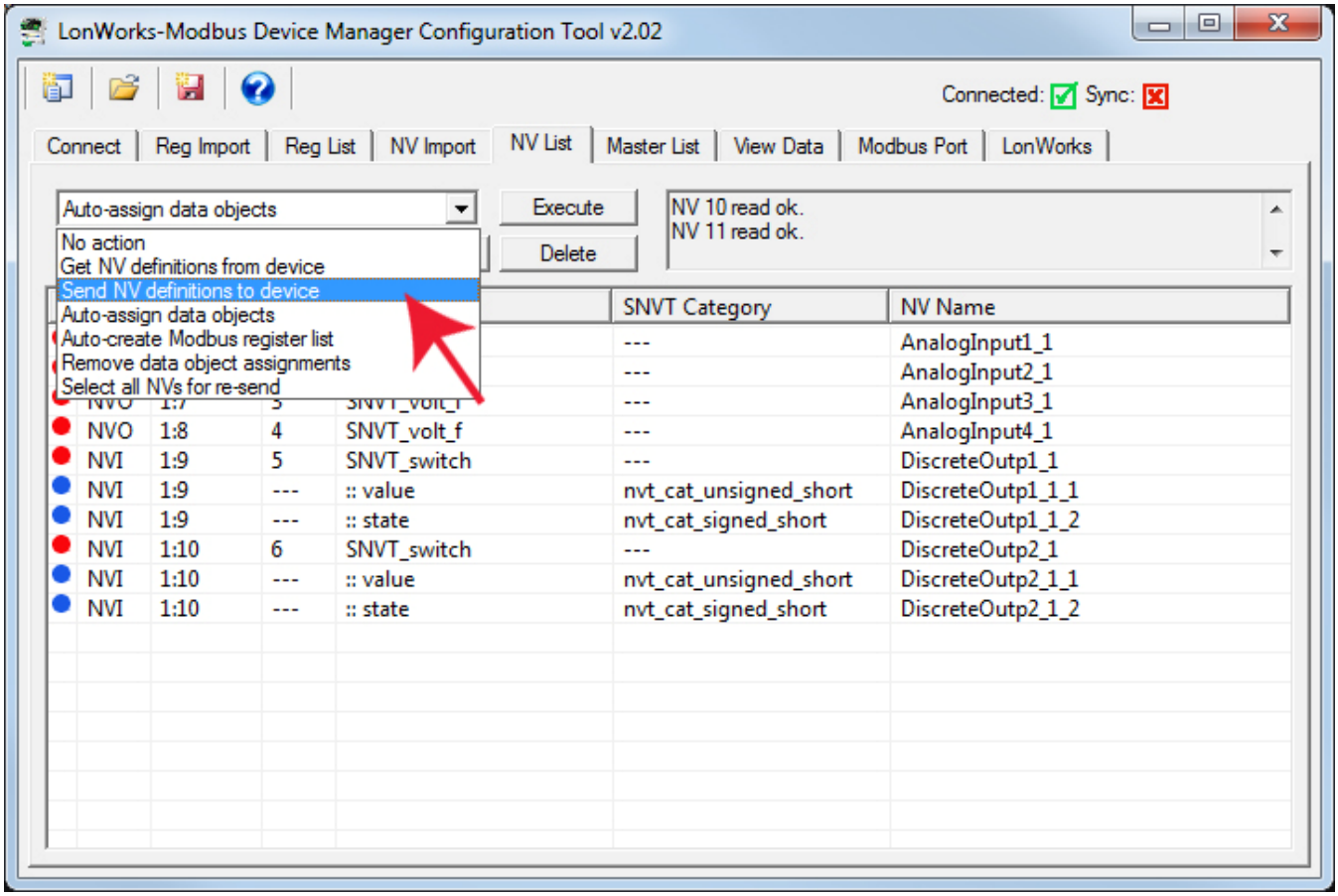


	Obj...	R/W	Type	Reg #	Slave	Dir	Nd:Nv	SNVT Type	Object Name
	1	-	NONE	0	0	NVO	1:5	SNVT_volt_f	AnalogInput1_1
	2	-	NONE	0	0	NVO	1:6	SNVT_volt_f	AnalogInput2_1
	3	-	NONE	0	0	NVO	1:7	SNVT_volt_f	AnalogInput3_1
	4	-	NONE	0	0	NVO	1:8	SNVT_volt_f	AnalogInput4_1
	5	-	NONE	0	0	NVI	1:9	SNVT_switch	DiscreteOutp1_1_1
	6	-	NONE	0	0	NVI	1:10	SNVT_switch	DiscreteOutp1_1_2
	7	-	NONE	0	0	---	---	---	DiscreteOutp2_1_1
	8	-	NONE	0	0	---	---	---	DiscreteOutp2_1_2
	9	-	NONE	0	0	---	---	---	
	10	-	NONE	0	0	---	---	---	
	11	-	NONE	0	0	---	---	---	
	12	-	NONE	0	0	---	---	---	
	13	-	NONE	0	0	---	---	---	
	14	-	NONE	0	0	---	---	---	
	15	-	NONE	0	0	---	---	---	
	16	-	NONE	0	0	---	---	---	

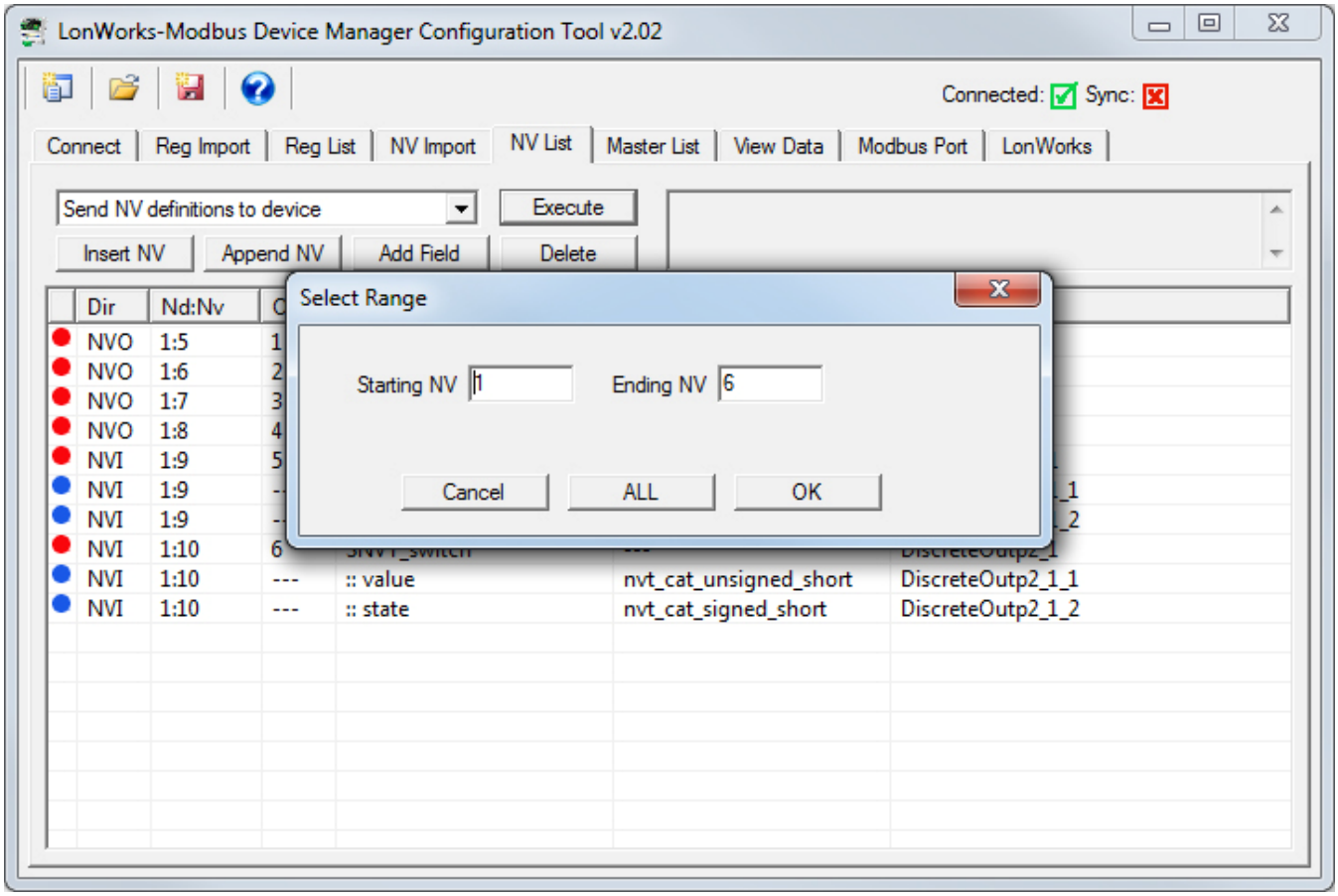
Assuming you did the 'Completely unconfigure device', you now need to restore the node table. You need to either keep the tool open and resend the list retained by the tool, or reload the configuration file you previously saved on the Master List page. Resend the node list by selecting 'Send node list to device' and click Execute.



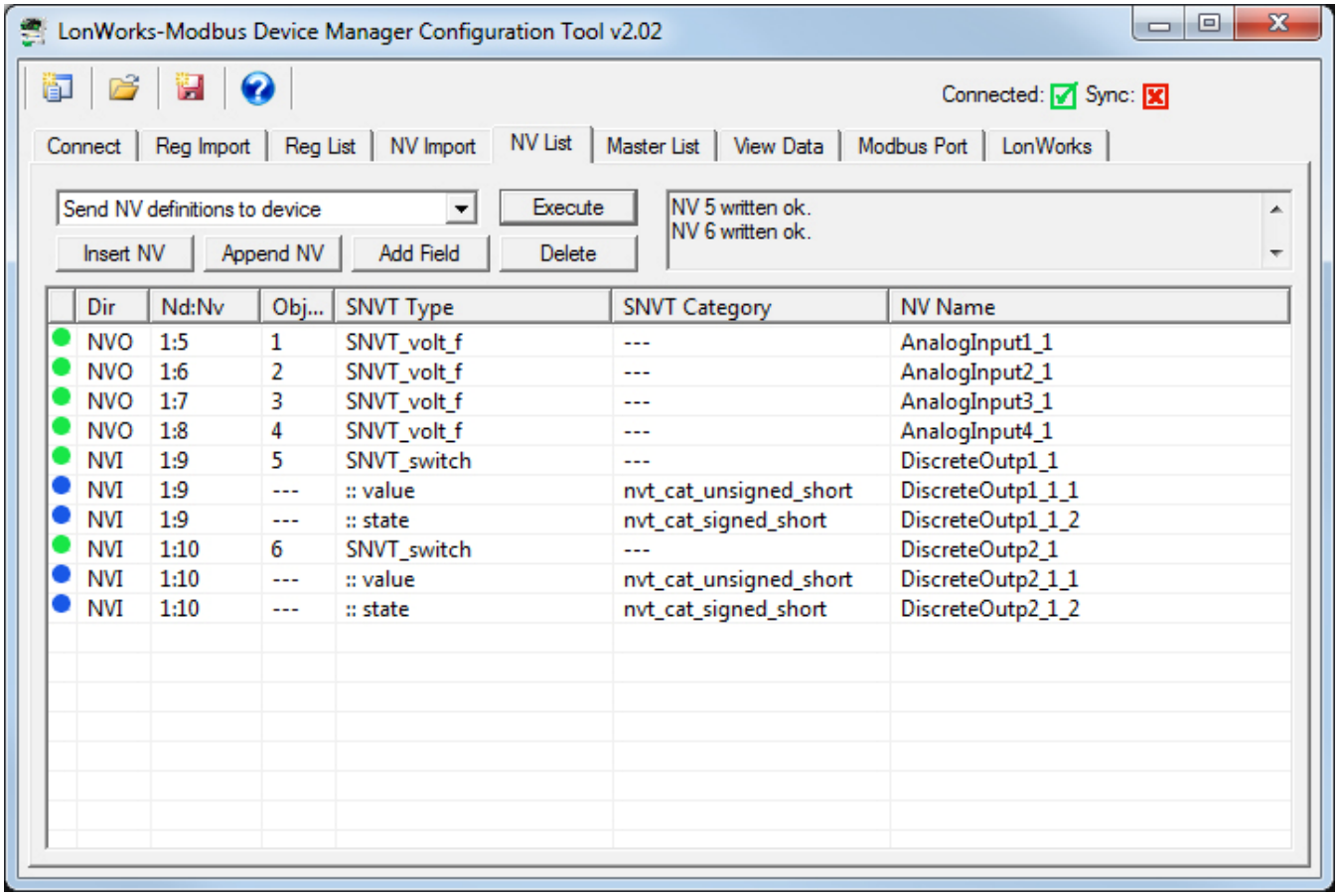
Assuming your node table has been sent to the gateway from the LonWorks page, you can proceed to send the updated NV List to the gateway device. Select 'Send NV definitions to device' from the list on the NV List page, and click Execute.



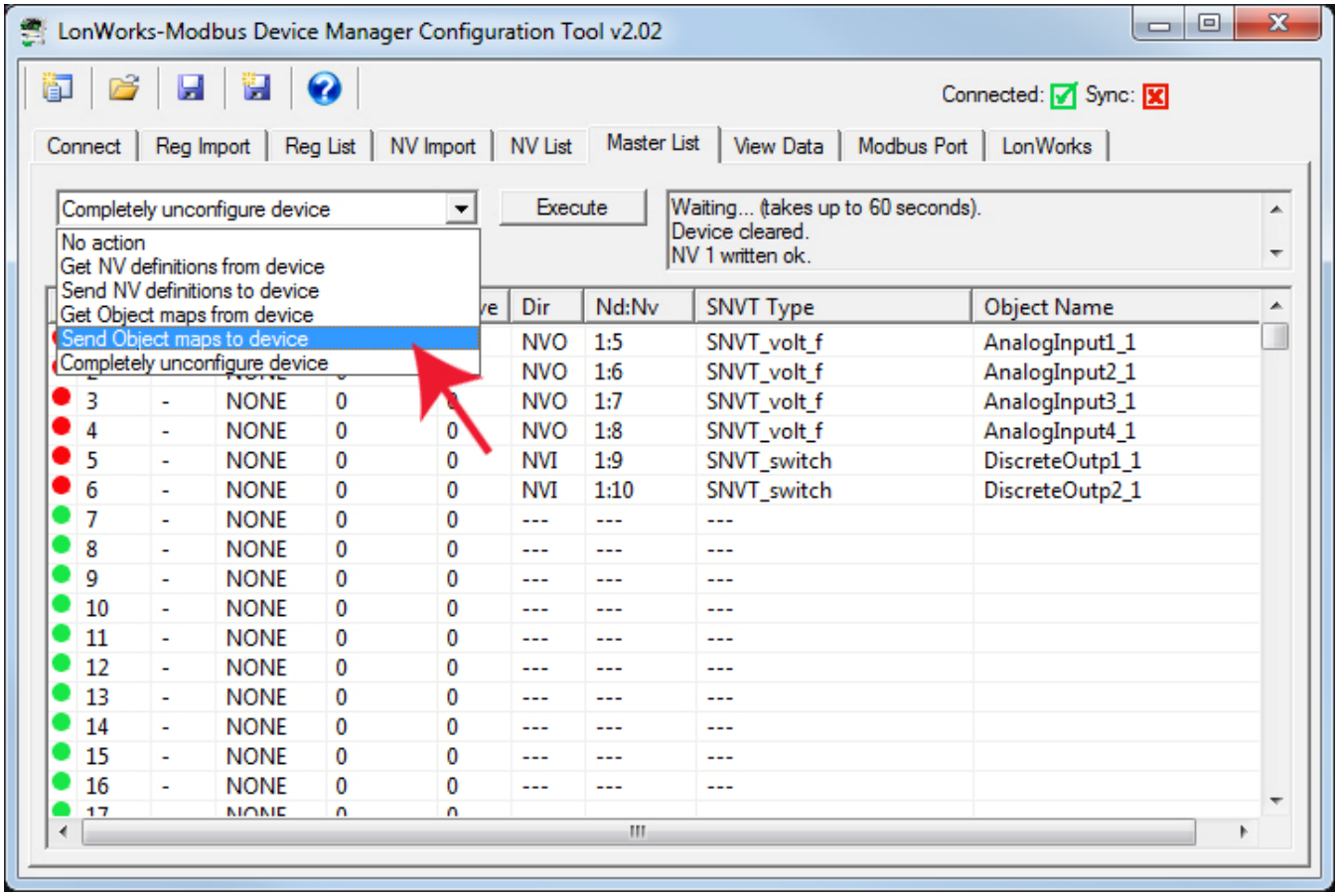
Upon executing 'Send NV definitions to device' you will be given the option of selecting a range. It will default to sending all unconfigured Network Variables, and you can generally just click OK here. If you have previously sent some variables but made changes such that only some remain unconfigured in the device, this dialog will find the range that needs to be sent. If all have been previouly sent and you are simply resending, it will default to all Network Variables found in the NV List.



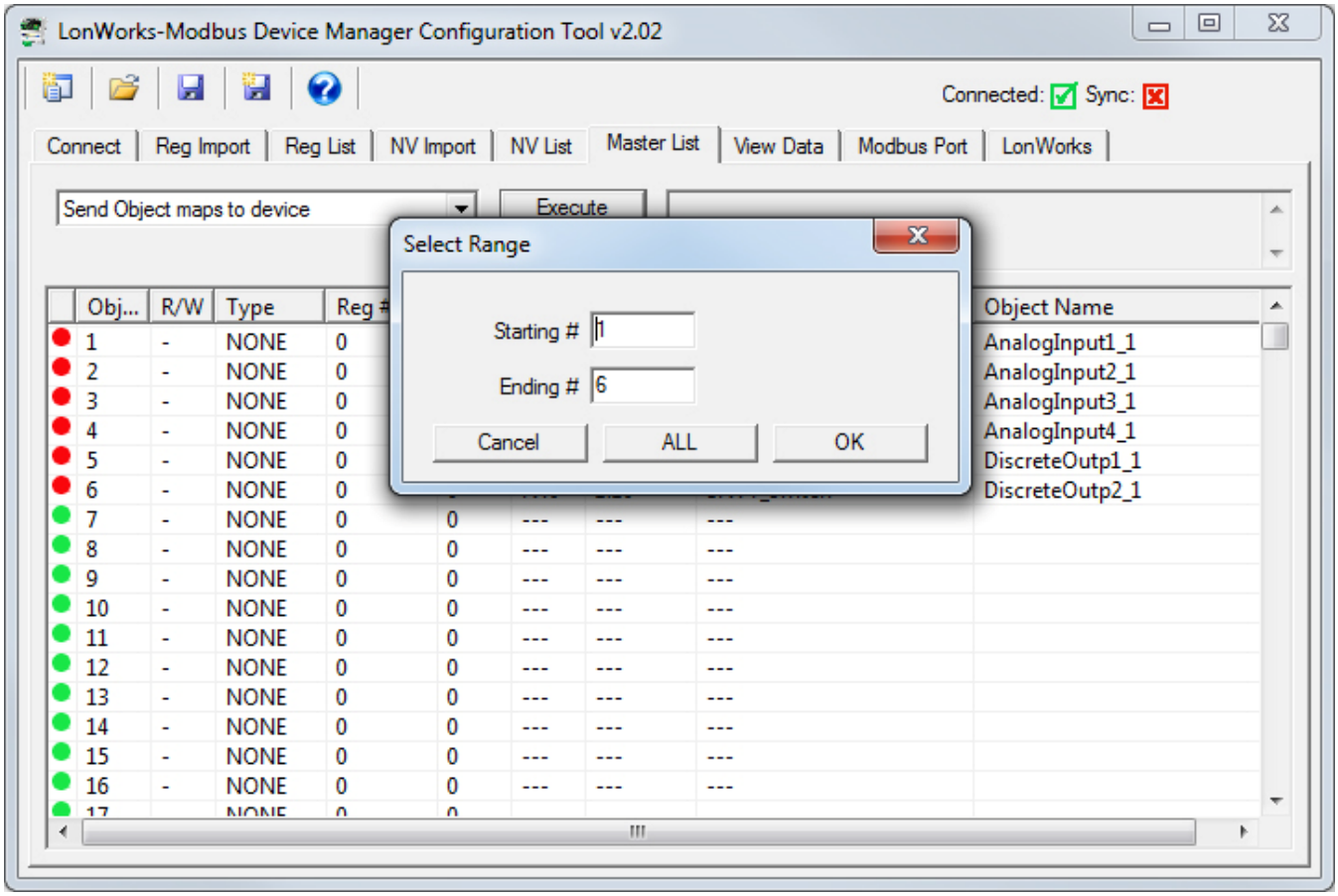
Progress will be indicated as the NV List is being sent.



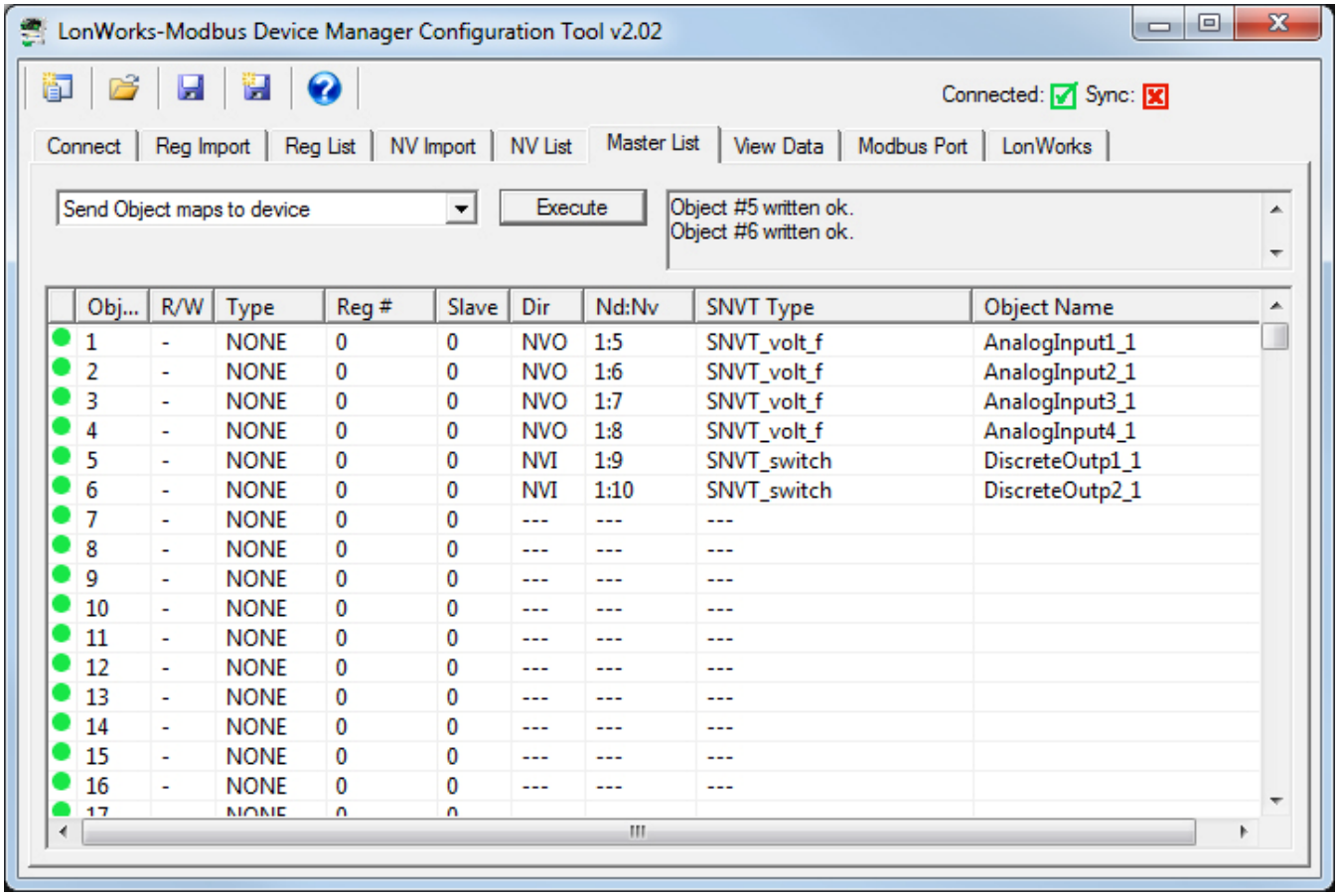
Next, you should proceed to the Master List page and send the object maps. The NV List only defines the list of network variables that will be polled by the gateway. The object maps make the connection between those Network Variables and the Modbus data objects you have assigned.



Upon executing 'Send Object maps to device', you will see a dialog pop up with a range of objects to send. The range will default to all objects remaining unconfigured, or all objects if resending a list that the tool believes is already completely configured in the device.

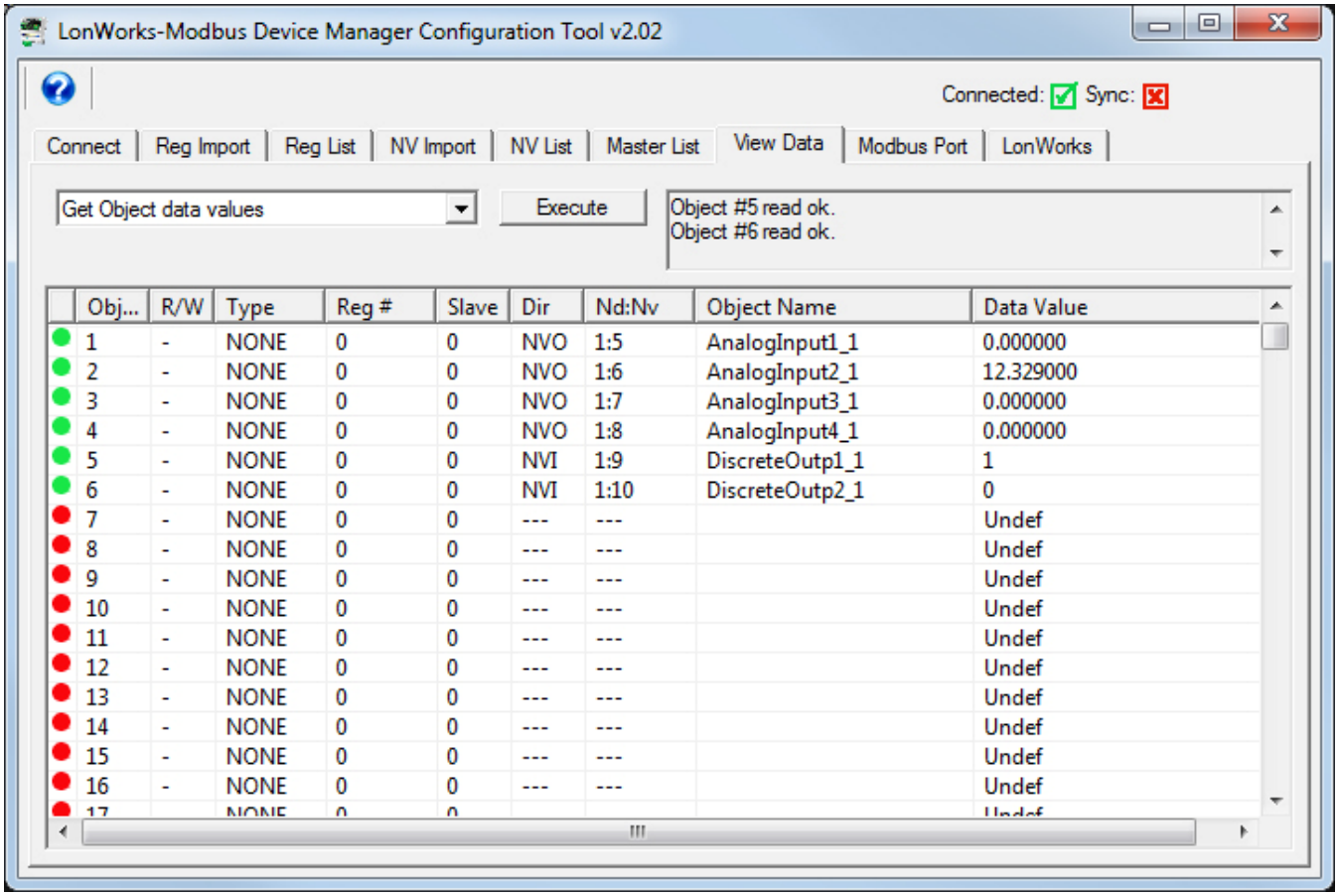


Progress will be indicated as objects are sent.



	Obj...	R/W	Type	Reg #	Slave	Dir	Nd:Nv	SNVT Type	Object Name
	1	-	NONE	0	0	NVO	1:5	SNVT_volt_f	AnalogInput1_1
	2	-	NONE	0	0	NVO	1:6	SNVT_volt_f	AnalogInput2_1
	3	-	NONE	0	0	NVO	1:7	SNVT_volt_f	AnalogInput3_1
	4	-	NONE	0	0	NVO	1:8	SNVT_volt_f	AnalogInput4_1
	5	-	NONE	0	0	NVI	1:9	SNVT_switch	DiscreteOutp1_1
	6	-	NONE	0	0	NVI	1:10	SNVT_switch	DiscreteOutp2_1
	7	-	NONE	0	0	---	---	---	
	8	-	NONE	0	0	---	---	---	
	9	-	NONE	0	0	---	---	---	
	10	-	NONE	0	0	---	---	---	
	11	-	NONE	0	0	---	---	---	
	12	-	NONE	0	0	---	---	---	
	13	-	NONE	0	0	---	---	---	
	14	-	NONE	0	0	---	---	---	
	15	-	NONE	0	0	---	---	---	
	16	-	NONE	0	0	---	---	---	

Assuming the NV List has been sent, the object maps have been sent from the Master List, and the node is connected from the LonWorks page, you can now view data being obtained from (or sent to) the LonWorks device. Go to the View Data page, select 'Get Object data values' and click Execute. You can also change data written to the LonWorks device from this page (refer to section on View Data page).



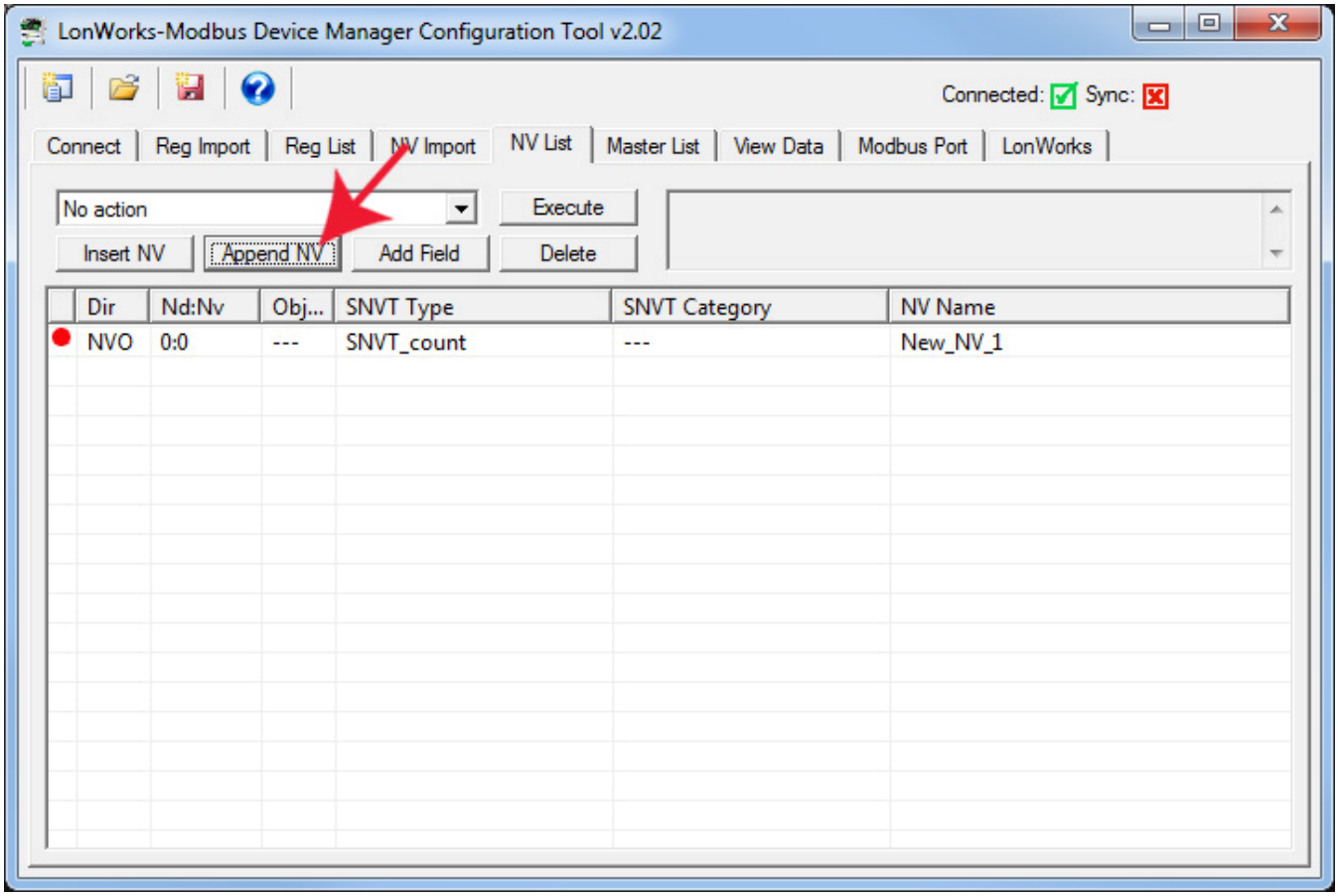
Obj...	R/W	Type	Reg #	Slave	Dir	Nd:Nv	Object Name	Data Value
1	-	NONE	0	0	NVO	1:5	AnalogInput1_1	0.000000
2	-	NONE	0	0	NVO	1:6	AnalogInput2_1	12.329000
3	-	NONE	0	0	NVO	1:7	AnalogInput3_1	0.000000
4	-	NONE	0	0	NVO	1:8	AnalogInput4_1	0.000000
5	-	NONE	0	0	NVI	1:9	DiscreteOutp1_1	1
6	-	NONE	0	0	NVI	1:10	DiscreteOutp2_1	0
7	-	NONE	0	0	---	---		Undef
8	-	NONE	0	0	---	---		Undef
9	-	NONE	0	0	---	---		Undef
10	-	NONE	0	0	---	---		Undef
11	-	NONE	0	0	---	---		Undef
12	-	NONE	0	0	---	---		Undef
13	-	NONE	0	0	---	---		Undef
14	-	NONE	0	0	---	---		Undef
15	-	NONE	0	0	---	---		Undef
16	-	NONE	0	0	---	---		Undef

3.4 Build Configuration from Scratch

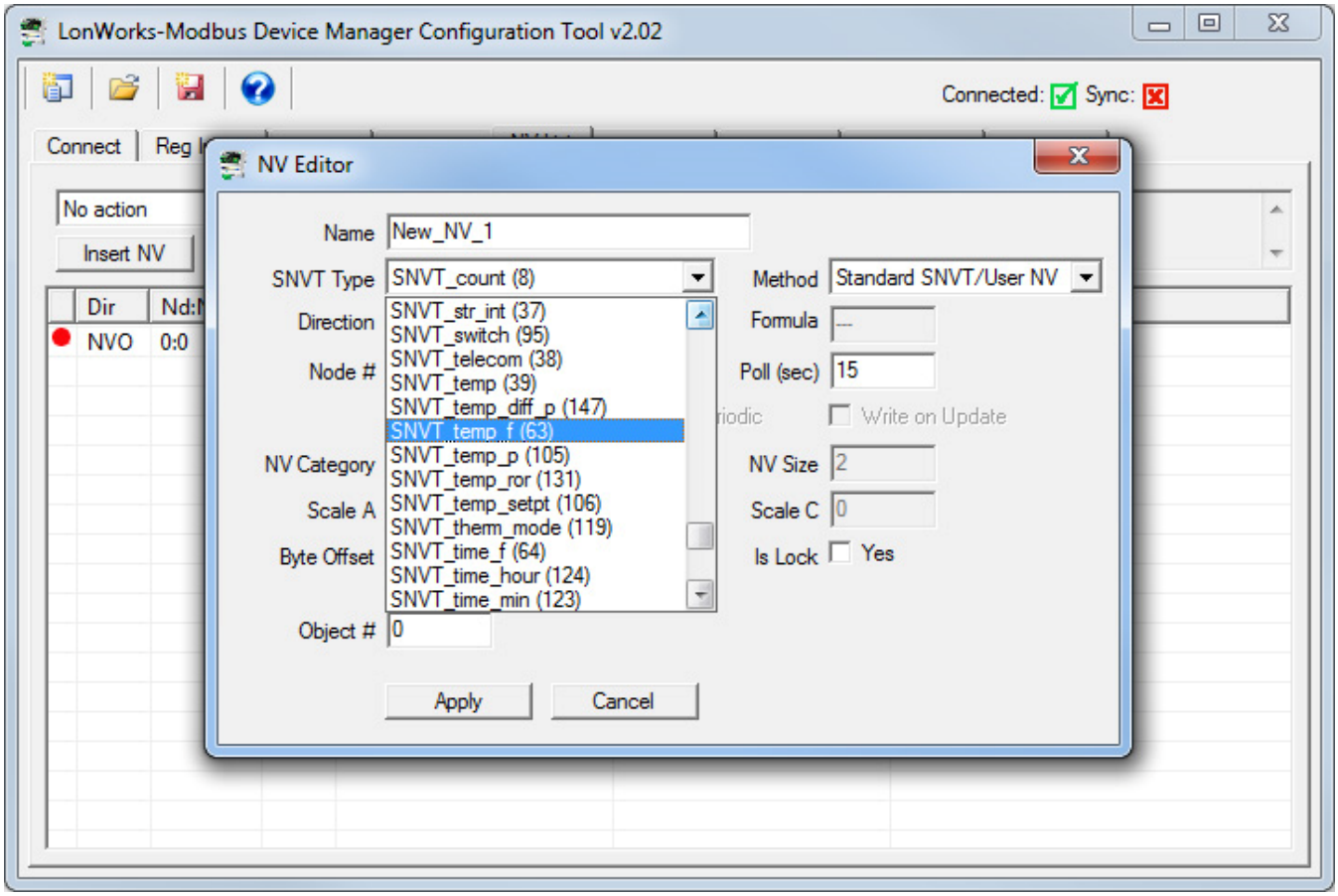
It is not required that you start with either a CSV file or an XIF file. You can use the configuration tool to start with a blank slate and build your configuration from scratch.

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 or Append NV.



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.

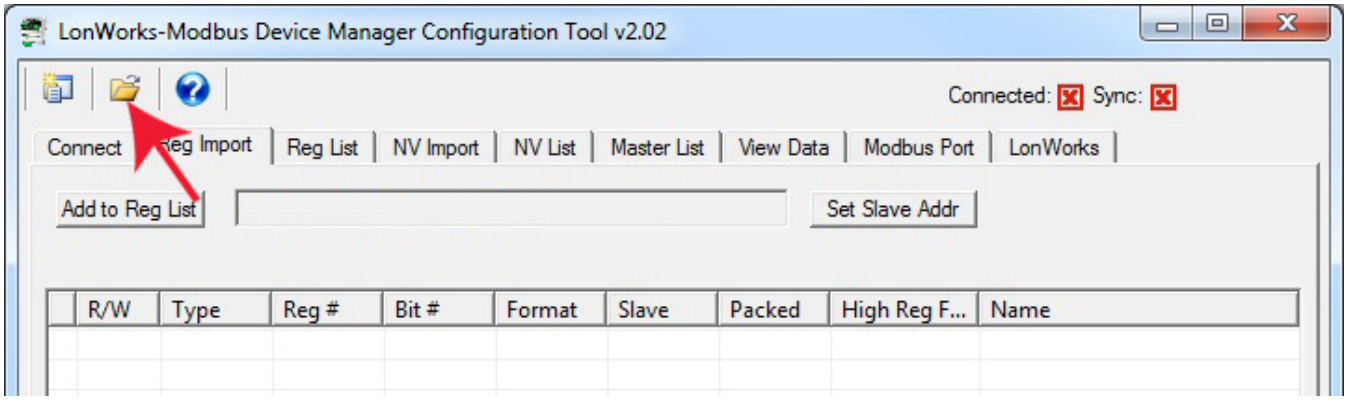


Once you have your list of variables created, you need to assign data objects in the same manner as discussed for building a configuration starting from an XIF file. What you have done by manually entering network variables is replace the XIF import process with a manually imported list. Additional details regarding creation of user defined network variables and structures can be found in the section dedicated to the NV List page.

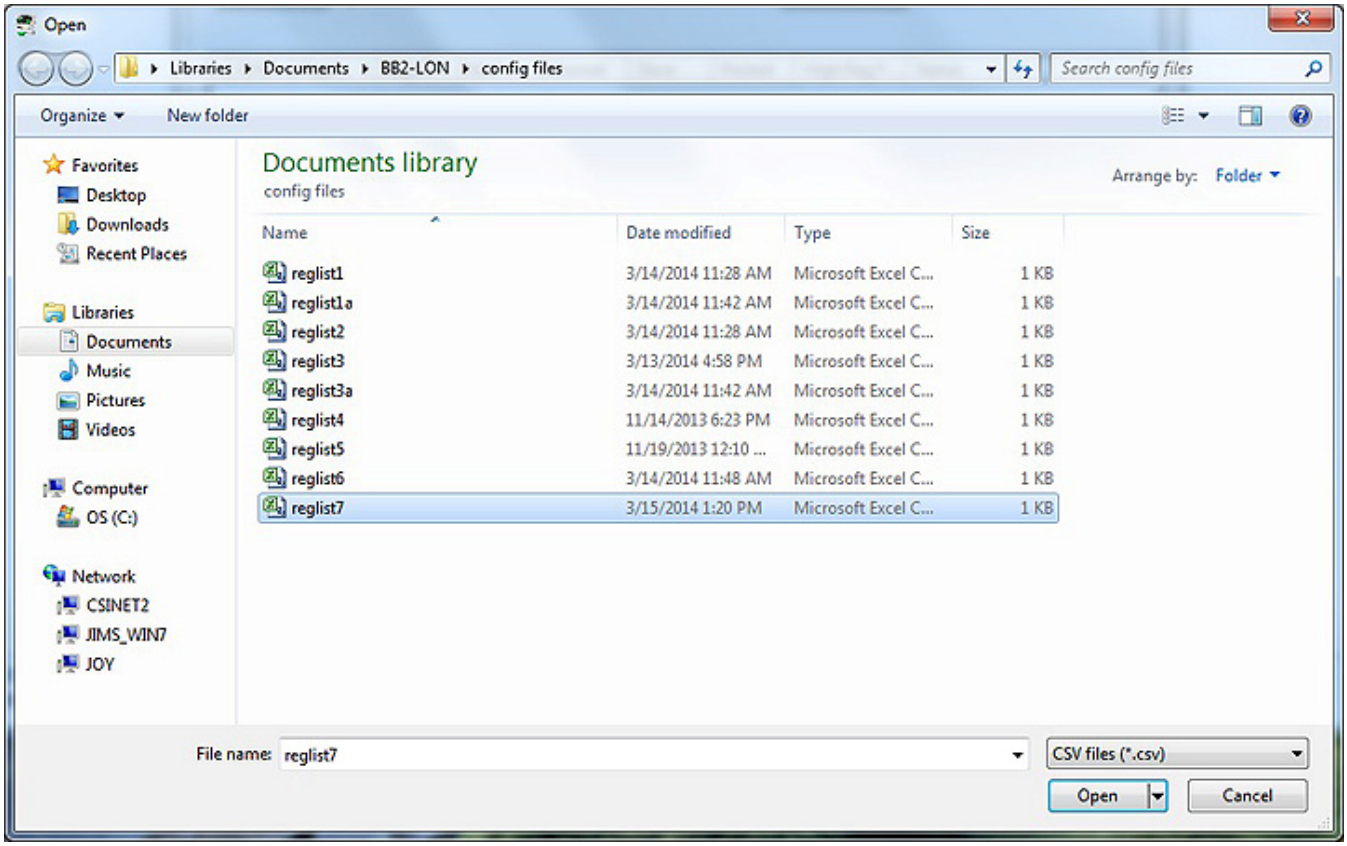
3.5 Build Configuration from CSV List of Modbus Registers

Starting with a Modbus register list is only useful if the Babel Buster gateway will be functioning as a Modbus master. If the Babel Buster will only be a slave on the Modbus network, then the CSV object import is of little use to you.

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.

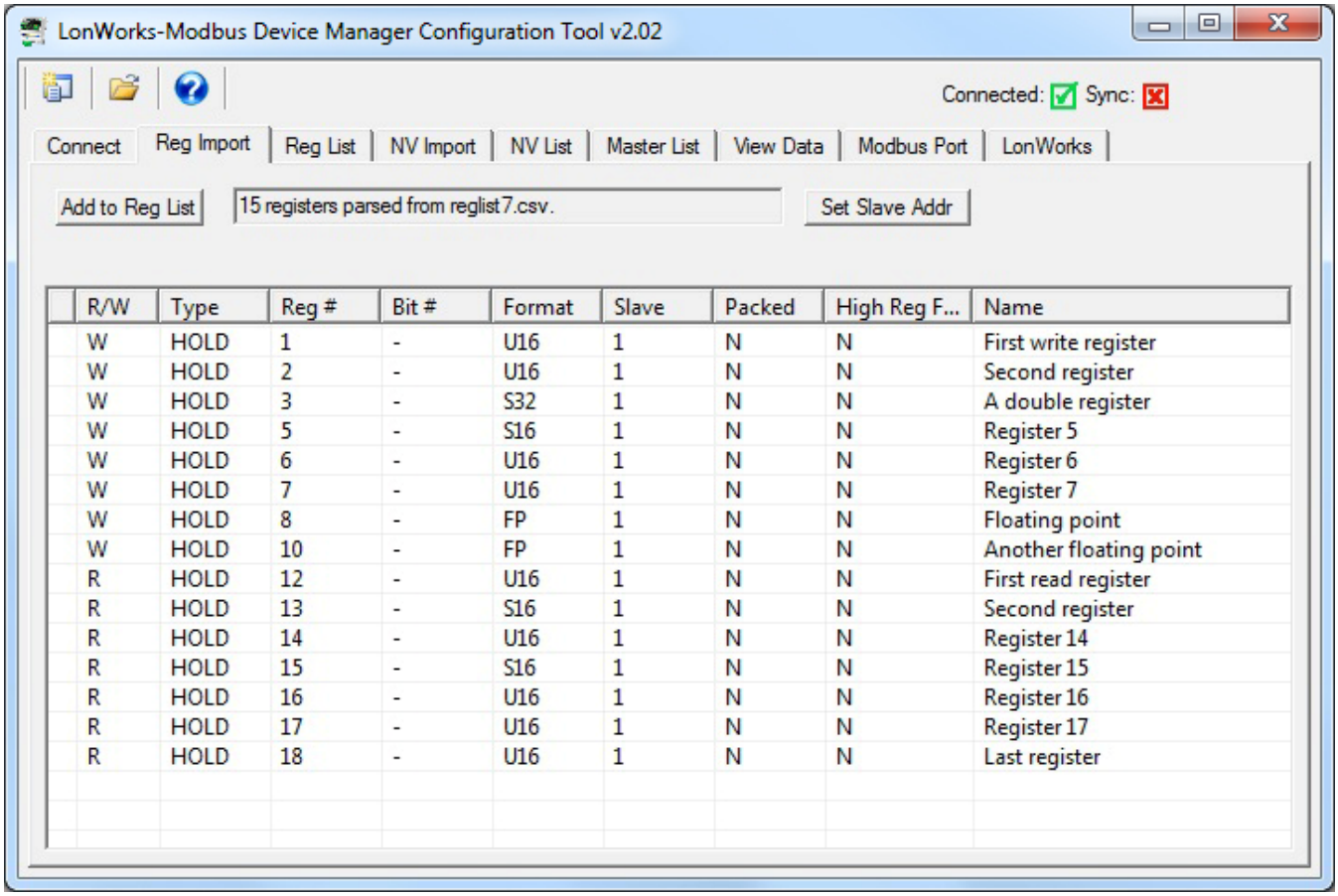


You will see the familiar file open dialog. Select your CSV file.



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
```

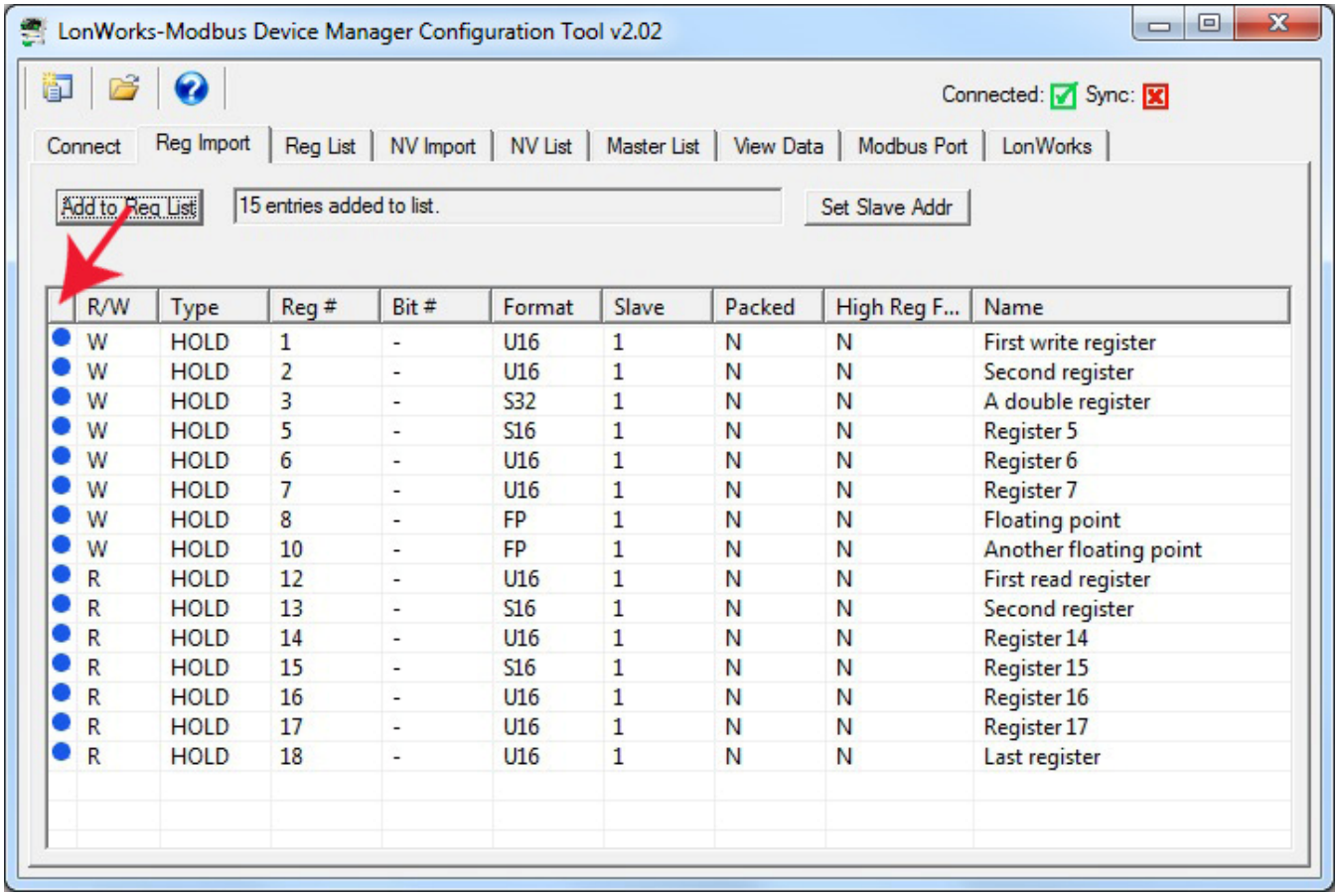


R/W	Type	Reg #	Bit #	Format	Slave	Packed	High Reg F...	Name
W	HOLD	1	-	U16	1	N	N	First write register
W	HOLD	2	-	U16	1	N	N	Second register
W	HOLD	3	-	S32	1	N	N	A double register
W	HOLD	5	-	S16	1	N	N	Register 5
W	HOLD	6	-	U16	1	N	N	Register 6
W	HOLD	7	-	U16	1	N	N	Register 7
W	HOLD	8	-	FP	1	N	N	Floating point
W	HOLD	10	-	FP	1	N	N	Another floating point
R	HOLD	12	-	U16	1	N	N	First read register
R	HOLD	13	-	S16	1	N	N	Second register
R	HOLD	14	-	U16	1	N	N	Register 14
R	HOLD	15	-	S16	1	N	N	Register 15
R	HOLD	16	-	U16	1	N	N	Register 16
R	HOLD	17	-	U16	1	N	N	Register 17
R	HOLD	18	-	U16	1	N	N	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.

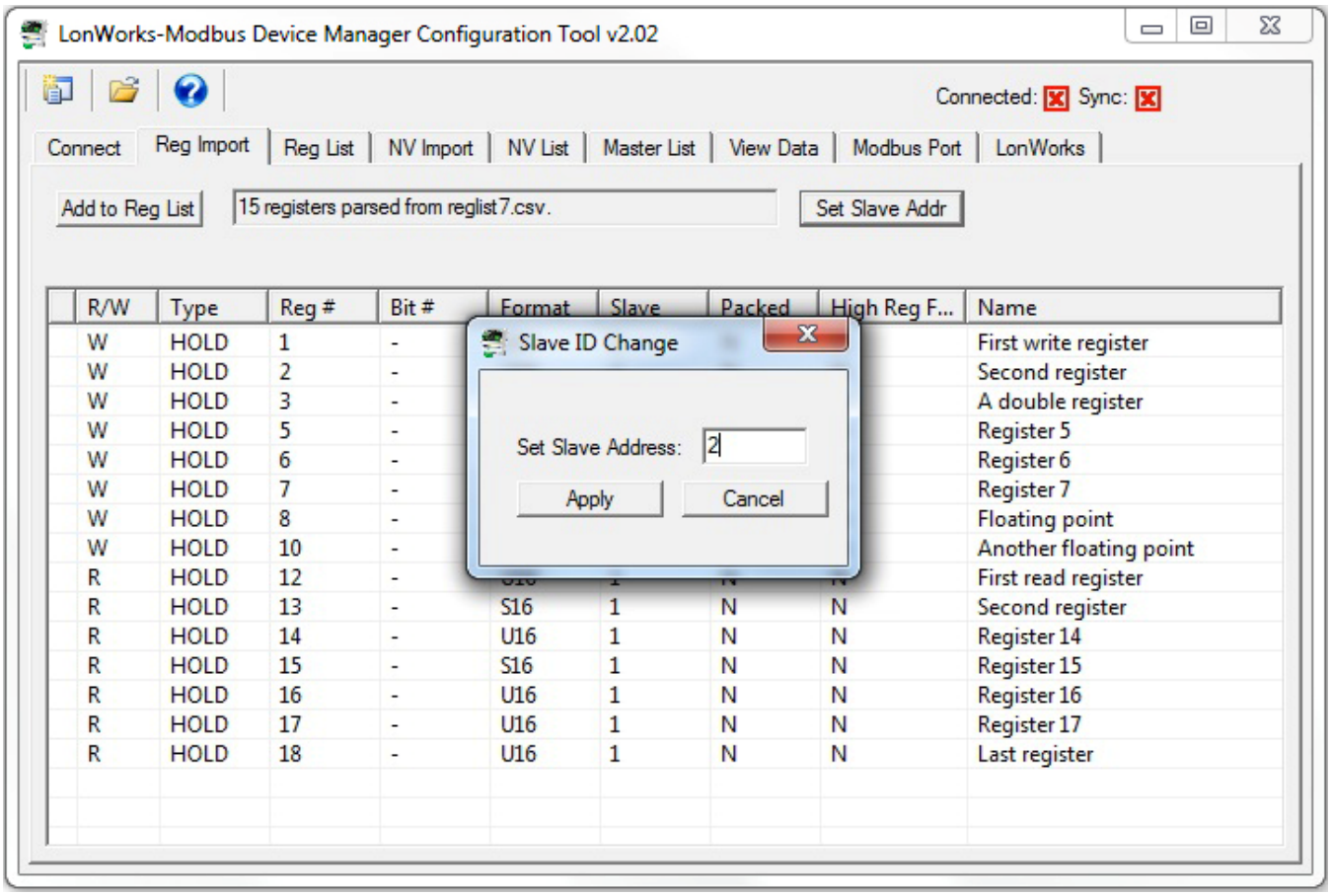
To select all registers, click the icon column header as illustrated by the red arrow below. This will cause the blue dot to appear on each line, which indicates that this register has been selected to be included in the configuration. To select only certain registers, click the icon area of only those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



	R/W	Type	Reg #	Bit #	Format	Slave	Packed	High Reg F...	Name
●	W	HOLD	1	-	U16	1	N	N	First write register
●	W	HOLD	2	-	U16	1	N	N	Second register
●	W	HOLD	3	-	S32	1	N	N	A double register
●	W	HOLD	5	-	S16	1	N	N	Register 5
●	W	HOLD	6	-	U16	1	N	N	Register 6
●	W	HOLD	7	-	U16	1	N	N	Register 7
●	W	HOLD	8	-	FP	1	N	N	Floating point
●	W	HOLD	10	-	FP	1	N	N	Another floating point
●	R	HOLD	12	-	U16	1	N	N	First read register
●	R	HOLD	13	-	S16	1	N	N	Second register
●	R	HOLD	14	-	U16	1	N	N	Register 14
●	R	HOLD	15	-	S16	1	N	N	Register 15
●	R	HOLD	16	-	U16	1	N	N	Register 16
●	R	HOLD	17	-	U16	1	N	N	Register 17
●	R	HOLD	18	-	U16	1	N	N	Last register

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.

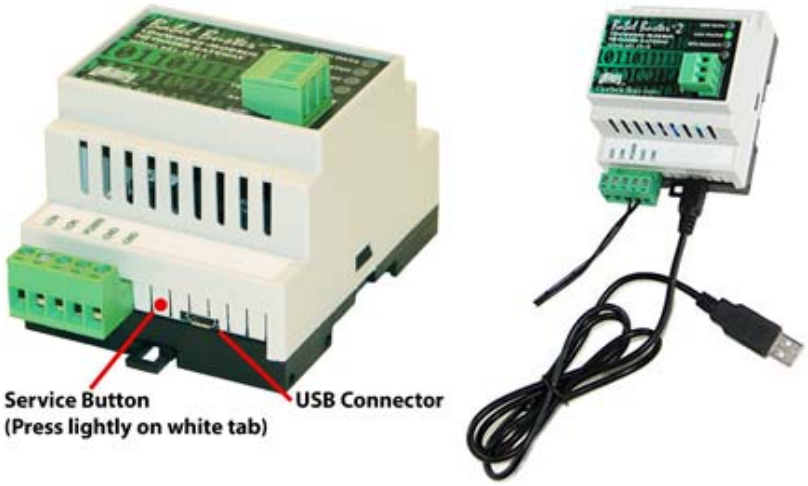


Once you have imported your register list(s), go to the Reg List page. Here is where you begin the auto-build of the rest of your configuration. Refer to the Reg List page section of this user guide to proceed from here.

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-NB, BB2-2011-NB, and BB2-6020-NB. The only significant difference is that TCP/IP settings will show up on the Modbus Port page when BB2-6020-NB is selected as Device Model.

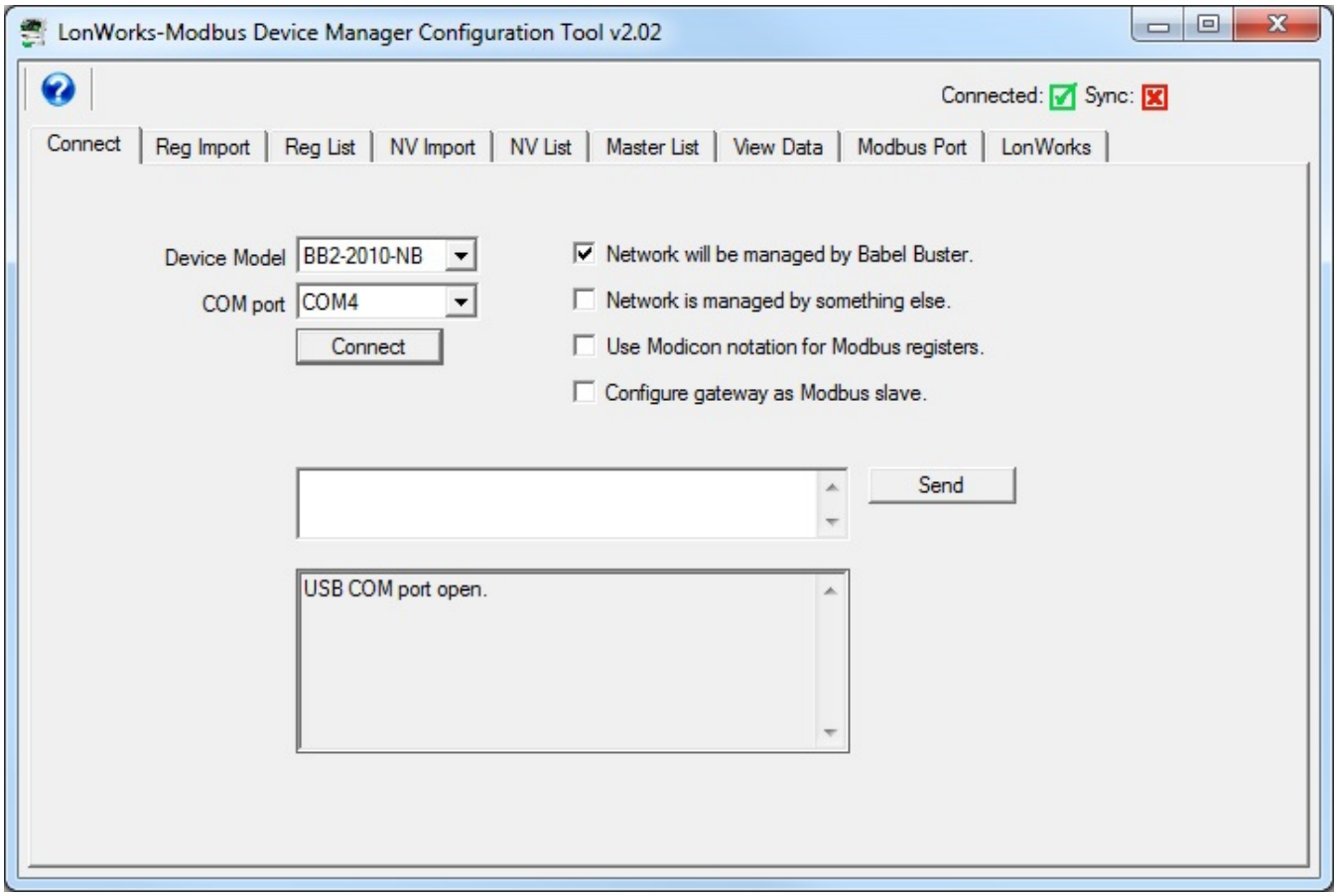
If the BB2-2010-NB or BB2-6020-NB will be managing the devices on the LonWorks side, then leave the default selection of "Network will be managed by Babel Buster". If the LonWorks devices are on a managed network that has been commissioned and bound by a tool such as Echelon's LonMaker, then select "Network is managed by something else".

IMPORTANT: If the LonWorks network has been commissioned and bound by a network management tool (e.g. LonMaker) and the LonWorks network is expected to remain functional, then you MUST select "Network is managed by something else". If you allow Babel Buster to attempt to manage a network that is already managed by something else, YOU WILL BREAK THE NETWORK!

The configuration will default to Modbus master. Check the 'Configure gateway as Modbus 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 object related configuration will change based on whether the gateway will be master or slave.

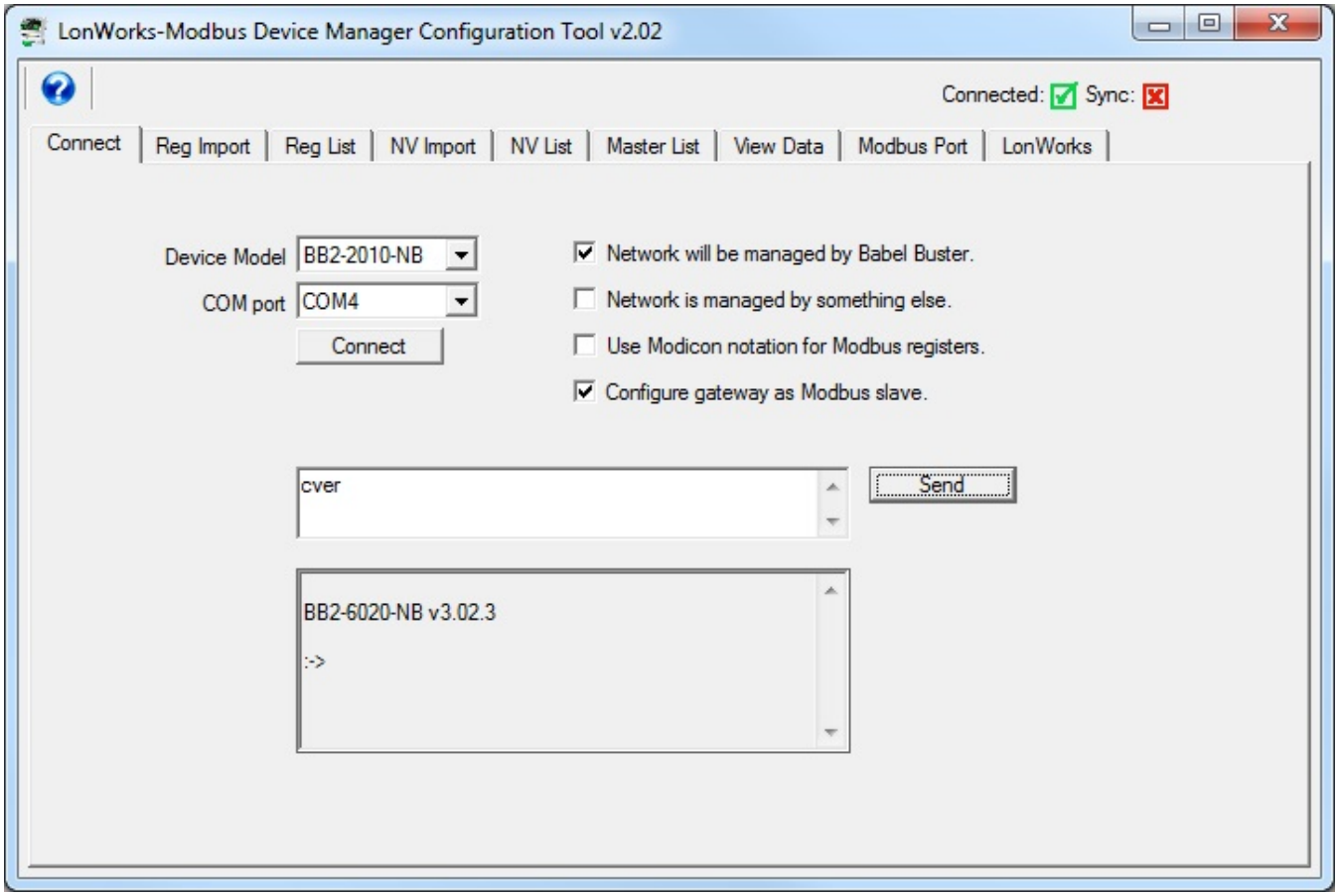
IMPORTANT: If you have connected USB to the gateway, and then power cycle the gateway, the USB connection will be lost. You will need to unplug the USB cable from the gateway and reconnect, then reconnect the configuration tool. It may also sometimes be necessary to close and restart the configuration 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.

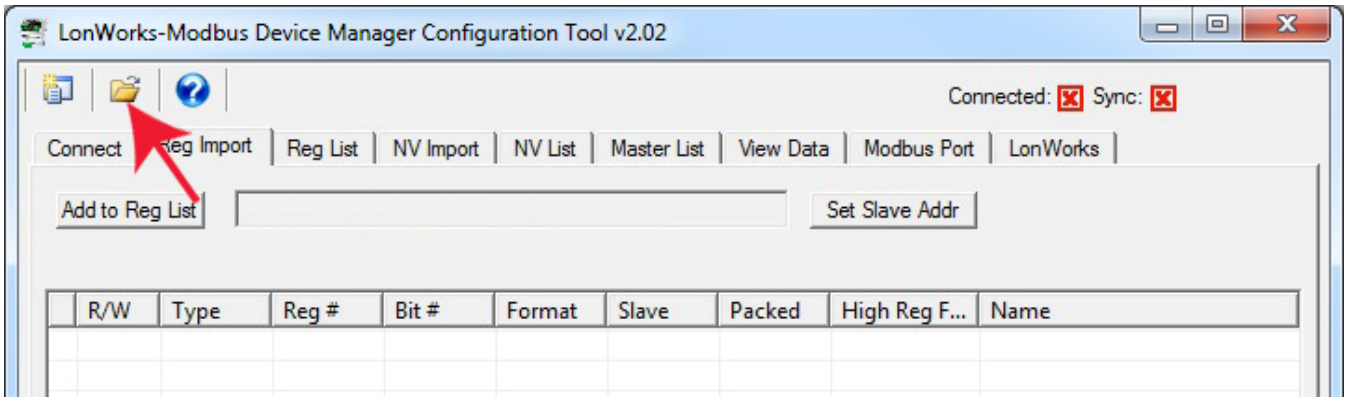

LonWorks-Modbus Device Manager Configuration Tool v2.02
Connected:
Sync:
Connect
Reg Import
Reg List
NV Import
NV List
Master List
View Data
Modbus Port
LonWorks
Device Model
BB2-2010-NB
COM port
COM4
Connect
Network will be managed by Babel Buster.
Network is managed by something else.
Use Modicon notation for Modbus registers.
Configure gateway as Modbus slave.
cver
Send
BB2-6020-NB v3.02.3
:->

5 Tool 'Reg Import' Page

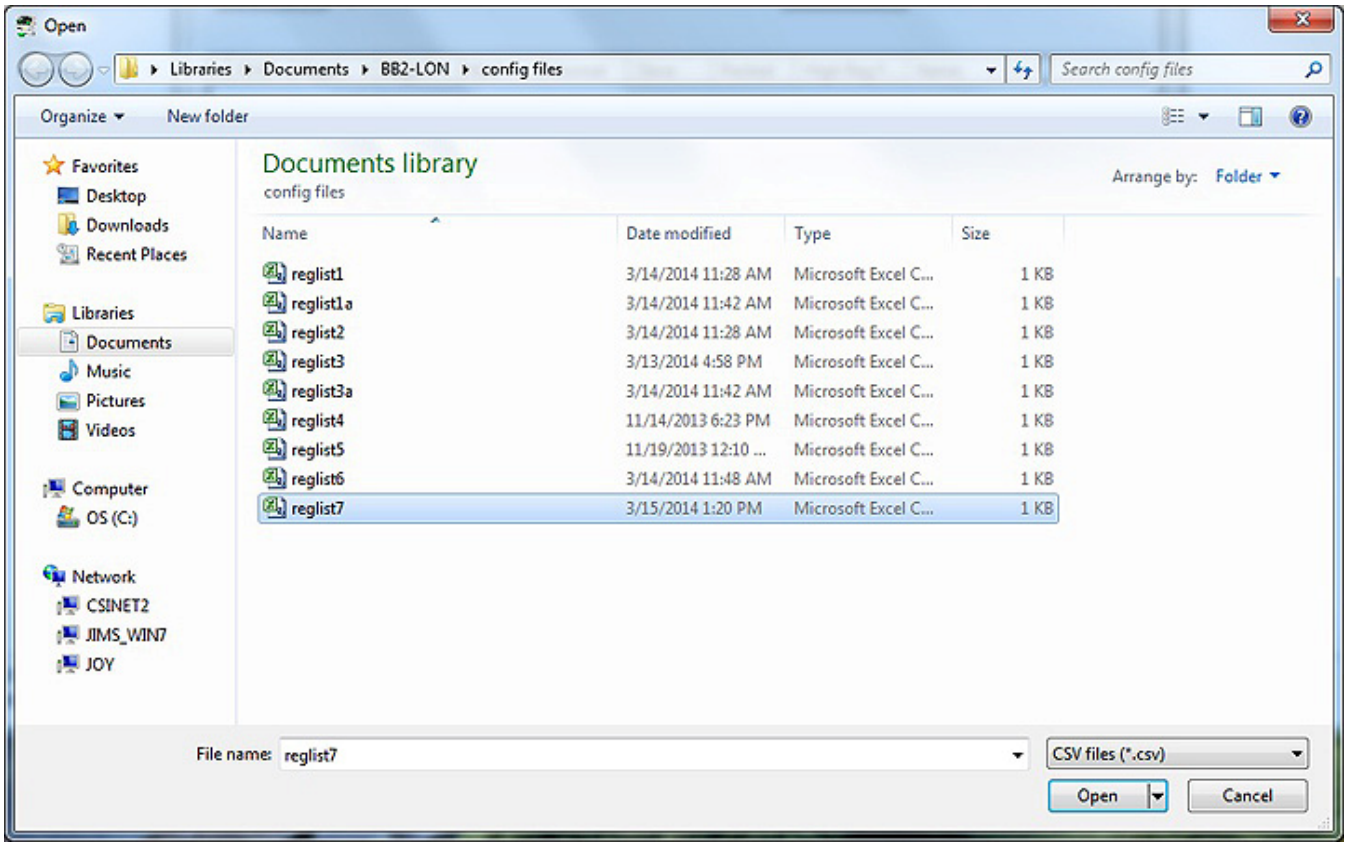
5.1 Importing a CSV Register List

The purpose of the object import capability is to allow you to use a standard spreadsheet program to create object mappings, and probably more importantly allow you to edit those mappings easily. The object mappings are used when the gateway will function as a Modbus master. If your gateway will function as a Modbus slave only, then you do not have any need to import registers here. The registers imported here are a list of registers in other Modbus devices that the Babel Buster gateway, function as Modbus master, will poll, meaning read or write according to the rules you define. Once the register mappings are imported, you will still need to assign local registers on the "Reg List" page. Refer to Appendix D for a reference guide on the format of this CSV file.

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.

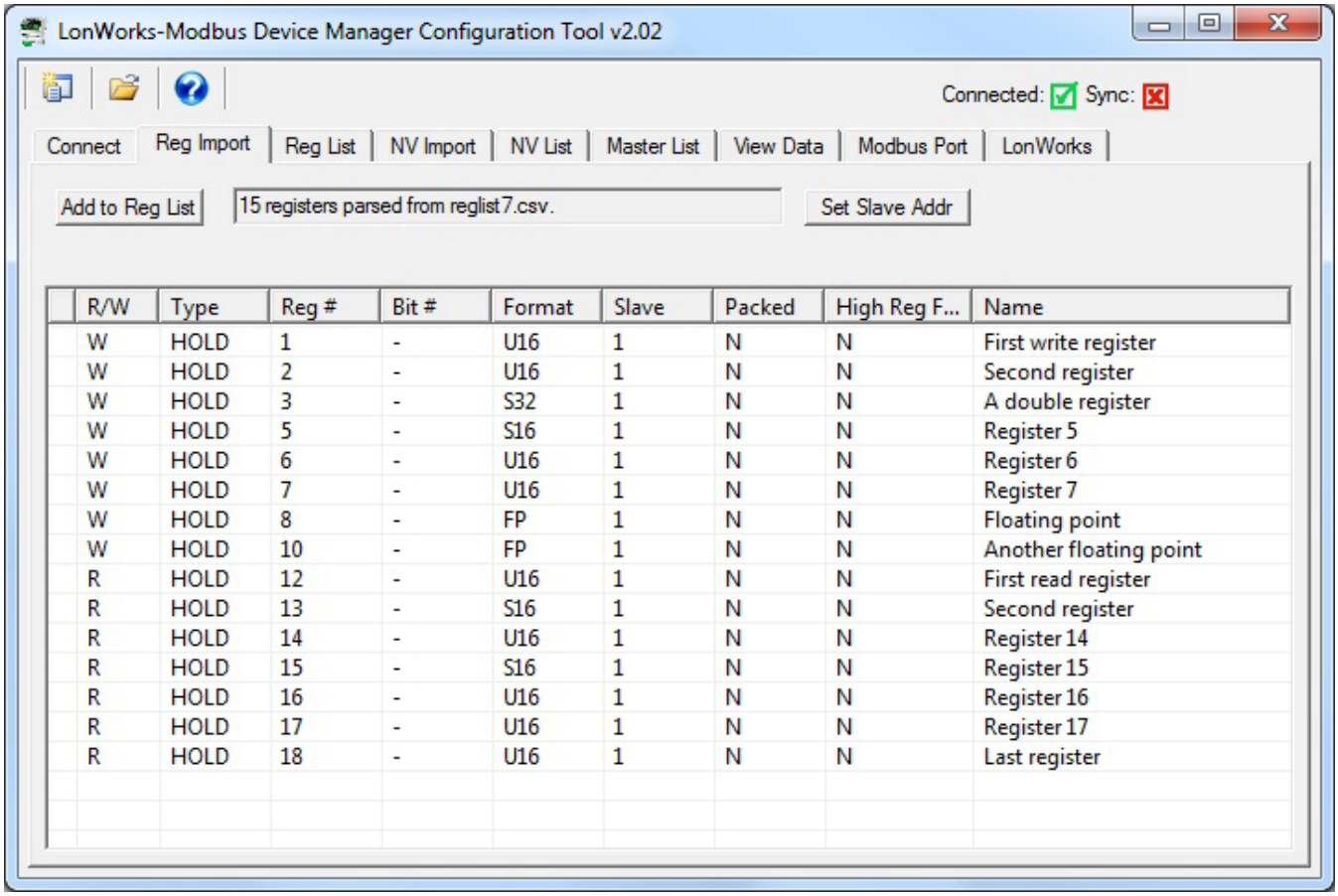


You will see the familiar file open dialog. Select your CSV file.



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
```

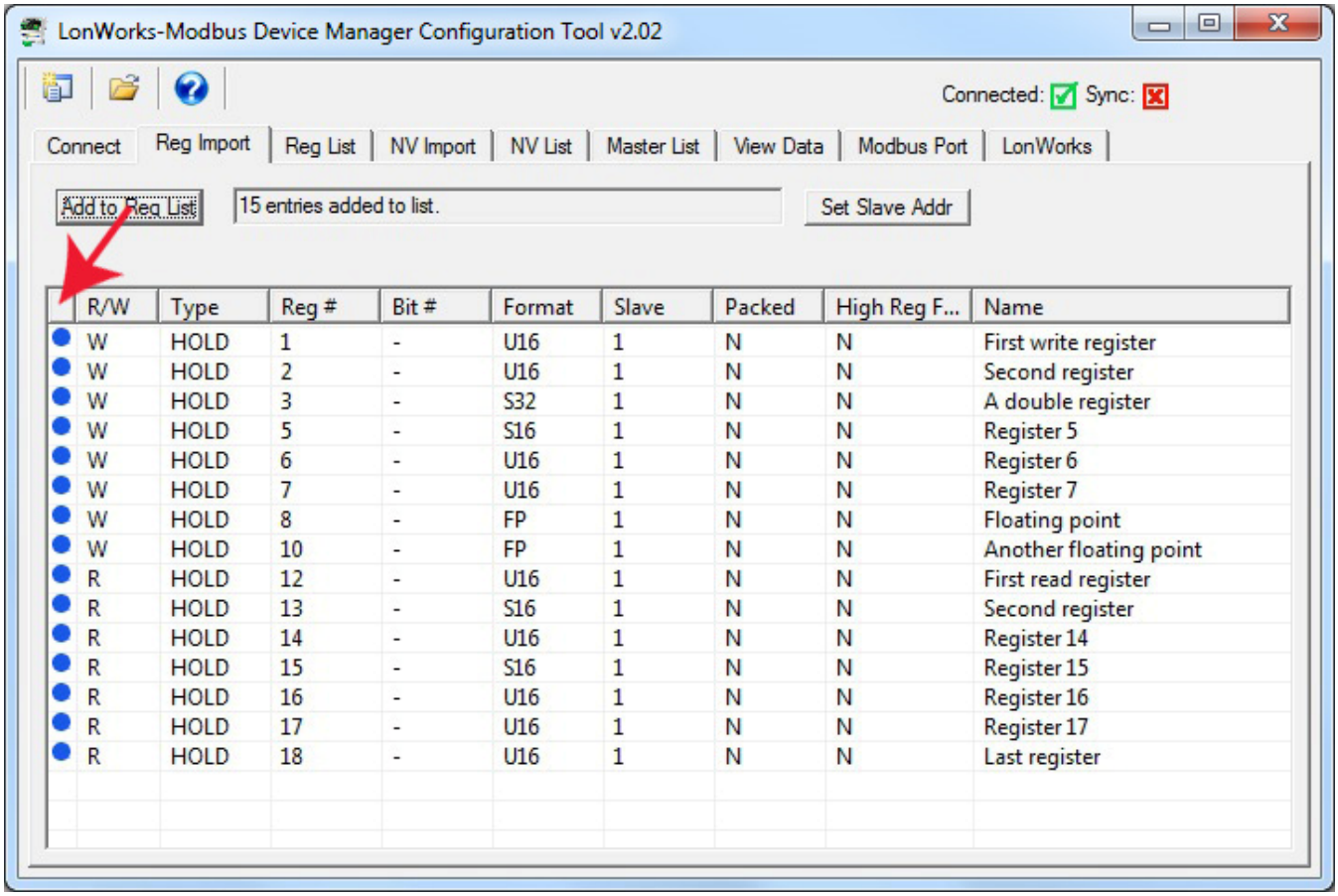


R/W	Type	Reg #	Bit #	Format	Slave	Packed	High Reg F...	Name
W	HOLD	1	-	U16	1	N	N	First write register
W	HOLD	2	-	U16	1	N	N	Second register
W	HOLD	3	-	S32	1	N	N	A double register
W	HOLD	5	-	S16	1	N	N	Register 5
W	HOLD	6	-	U16	1	N	N	Register 6
W	HOLD	7	-	U16	1	N	N	Register 7
W	HOLD	8	-	FP	1	N	N	Floating point
W	HOLD	10	-	FP	1	N	N	Another floating point
R	HOLD	12	-	U16	1	N	N	First read register
R	HOLD	13	-	S16	1	N	N	Second register
R	HOLD	14	-	U16	1	N	N	Register 14
R	HOLD	15	-	S16	1	N	N	Register 15
R	HOLD	16	-	U16	1	N	N	Register 16
R	HOLD	17	-	U16	1	N	N	Register 17
R	HOLD	18	-	U16	1	N	N	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.

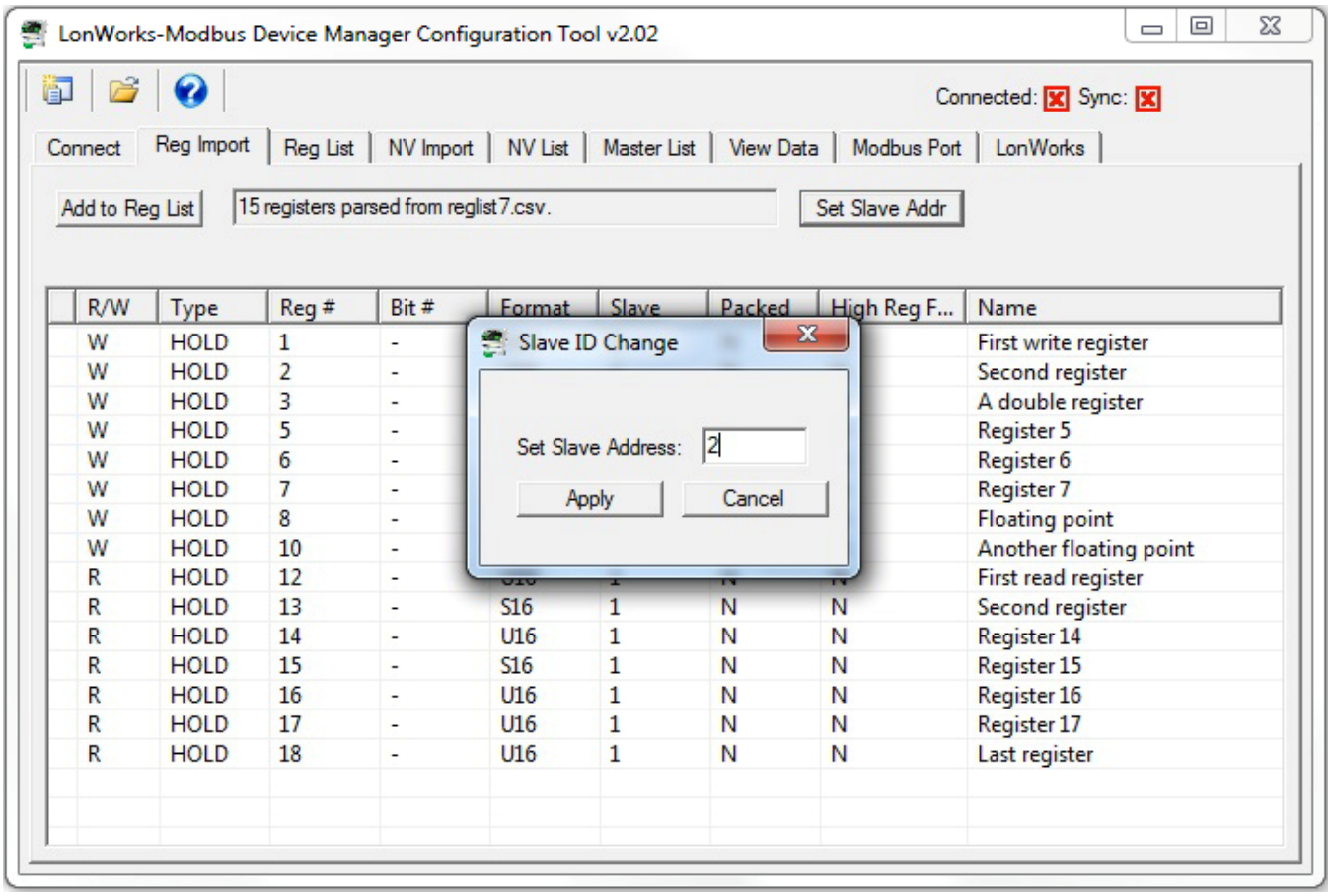
To select all registers, click the icon column header as illustrated by the red arrow below. This will cause the blue dot to appear on each line, which indicates that this register has been selected to be included in the configuration. To select only certain registers, click the icon area of only those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



	R/W	Type	Reg #	Bit #	Format	Slave	Packed	High Reg F...	Name
●	W	HOLD	1	-	U16	1	N	N	First write register
●	W	HOLD	2	-	U16	1	N	N	Second register
●	W	HOLD	3	-	S32	1	N	N	A double register
●	W	HOLD	5	-	S16	1	N	N	Register 5
●	W	HOLD	6	-	U16	1	N	N	Register 6
●	W	HOLD	7	-	U16	1	N	N	Register 7
●	W	HOLD	8	-	FP	1	N	N	Floating point
●	W	HOLD	10	-	FP	1	N	N	Another floating point
●	R	HOLD	12	-	U16	1	N	N	First read register
●	R	HOLD	13	-	S16	1	N	N	Second register
●	R	HOLD	14	-	U16	1	N	N	Register 14
●	R	HOLD	15	-	S16	1	N	N	Register 15
●	R	HOLD	16	-	U16	1	N	N	Register 16
●	R	HOLD	17	-	U16	1	N	N	Register 17
●	R	HOLD	18	-	U16	1	N	N	Last register

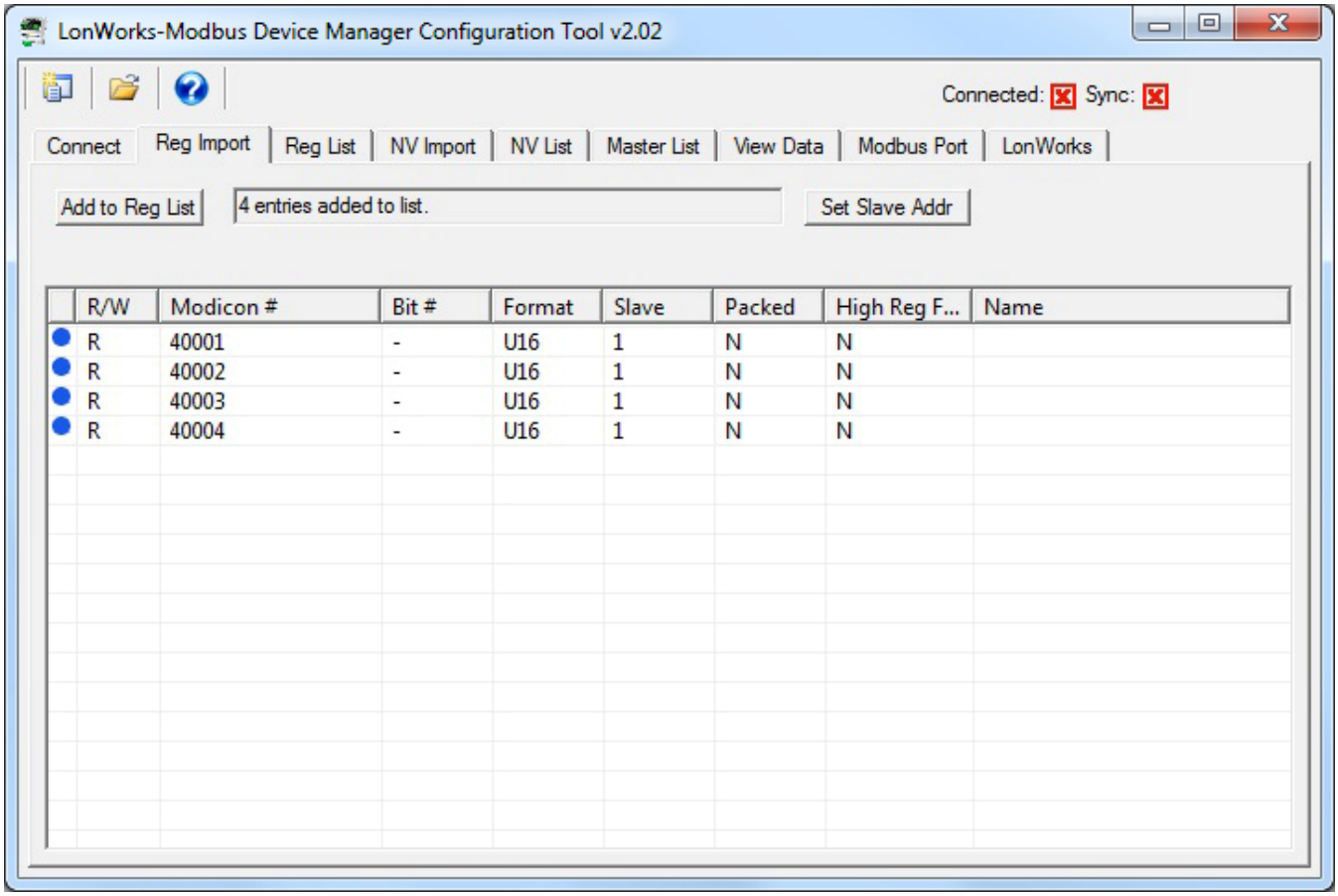
If you will be connecting two or more of the same type of Modbus device, each having identical register sets, click "Add to Reg List". Then use "Set Slave Addr" to select the next device's slave address. Now click "Add to Reg List" again. This adds the same set of registers a second time, but with a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



R/W	Type	Reg #	Bit #	Format	Slave	Packed	High Reg F...	Name
W	HOLD	1	-					First write register
W	HOLD	2	-					Second register
W	HOLD	3	-					A double register
W	HOLD	5	-					Register 5
W	HOLD	6	-					Register 6
W	HOLD	7	-					Register 7
W	HOLD	8	-					Floating point
W	HOLD	10	-					Another floating point
R	HOLD	12	-					First read register
R	HOLD	13	-	S16	1	N	N	Second register
R	HOLD	14	-	U16	1	N	N	Register 14
R	HOLD	15	-	S16	1	N	N	Register 15
R	HOLD	16	-	U16	1	N	N	Register 16
R	HOLD	17	-	U16	1	N	N	Register 17
R	HOLD	18	-	U16	1	N	N	Last register

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.

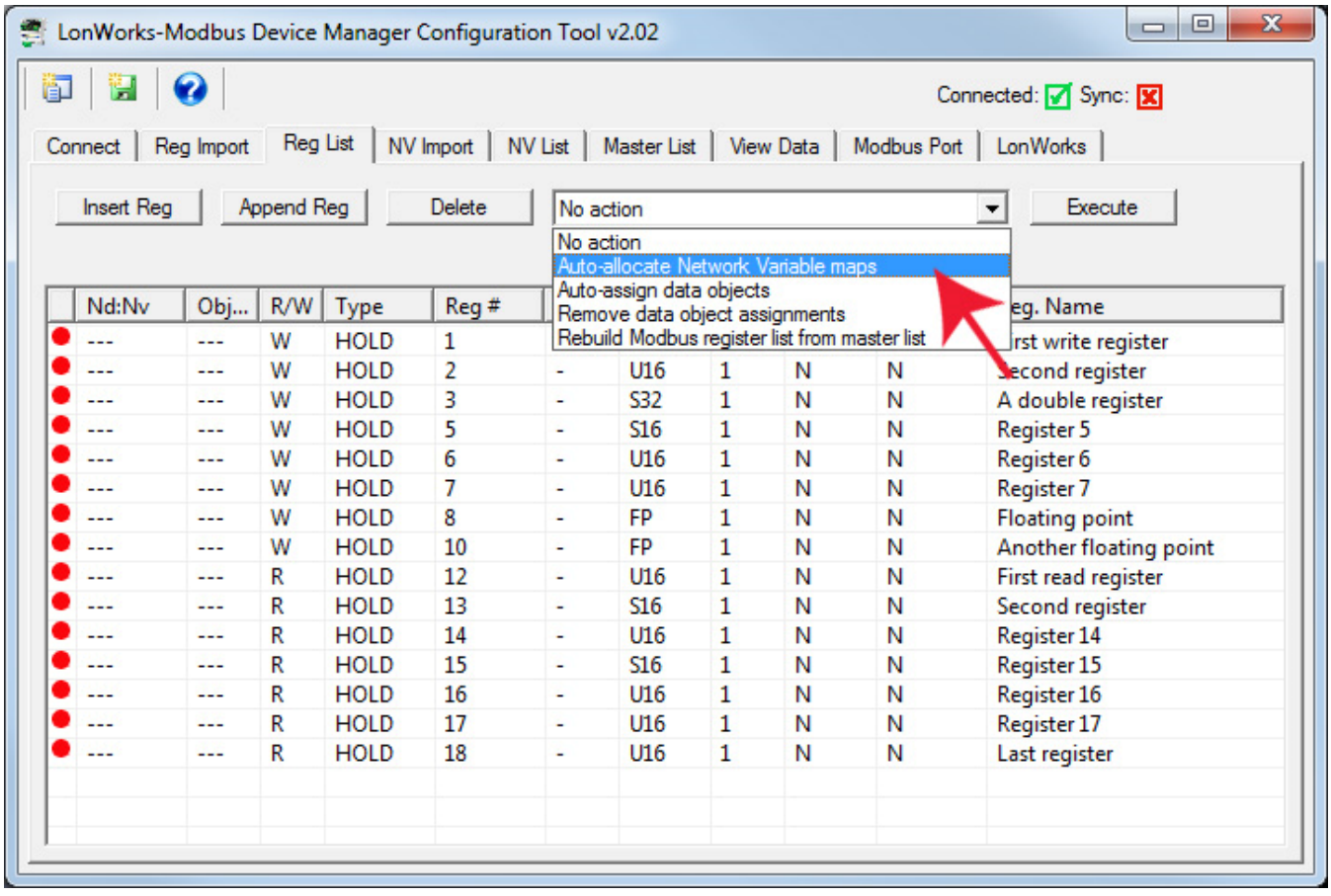

LonWorks-Modbus Device Manager Configuration Tool v2.02
Connected:
Sync:
Connect
Reg Import
Reg List
NV Import
NV List
Master List
View Data
Modbus Port
LonWorks
Add to Reg List
4 entries added to list.
Set Slave Addr
R/W
Modicon #
Bit #
Format
Slave
Packed
High Reg F...
Name
R 40001 - U16 1 N N
R 40002 - U16 1 N N
R 40003 - U16 1 N N
R 40004 - U16 1 N N

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-allocate Network Variable maps". Then click the Execute button.

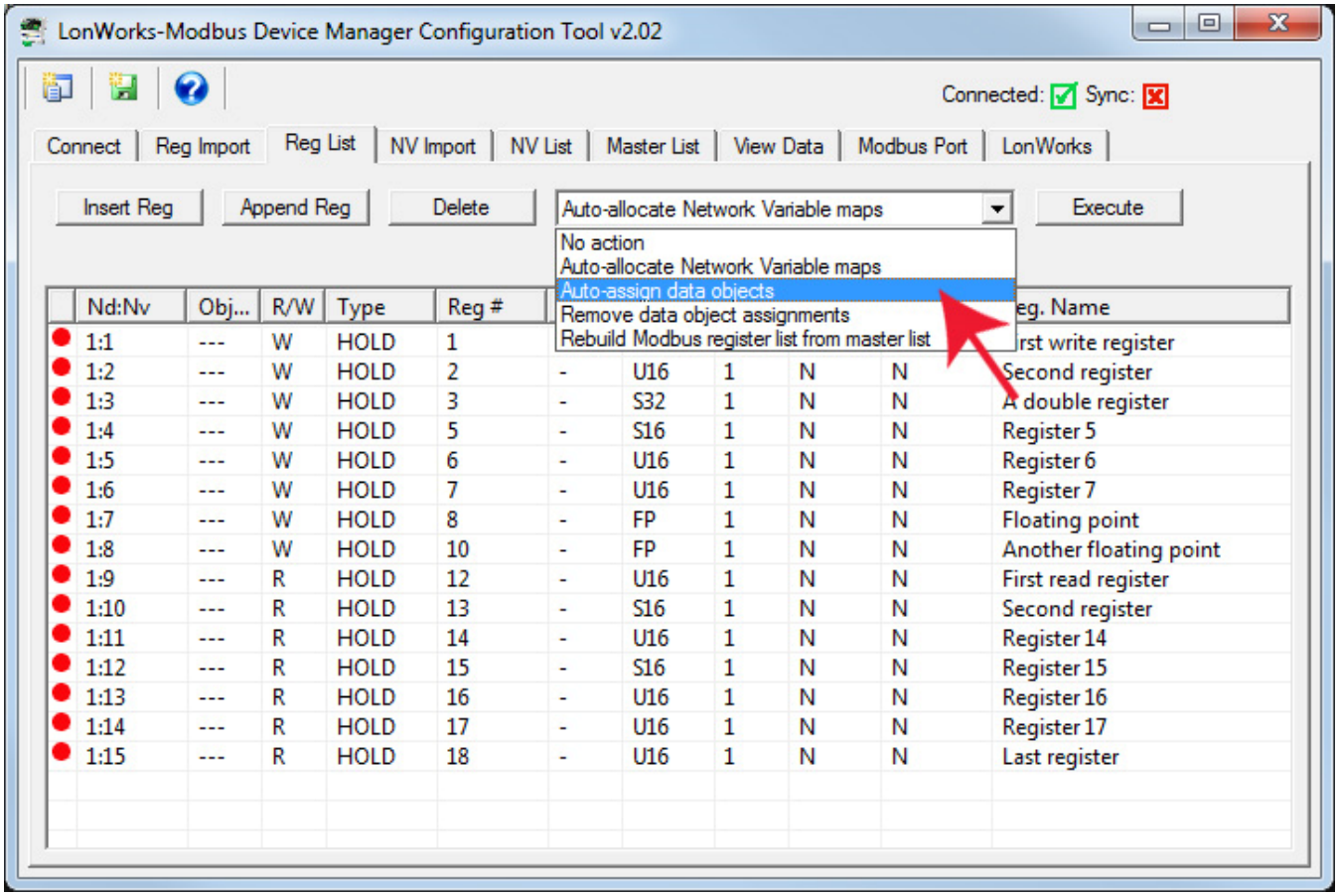
Note: If you are starting your configuration from a Modbus register list, the network variables automatically allocated might not match the LonWorks device you will be communicating with. You will need to manually edit the network variables as applicable. The more common and possibly more useful approach is to start on the NV Import page. The Modbus register list import approach would be used if you have a specific list of Modbus registers that you require matching, and is only useful if the Babel Buster gateway will be operating as a Modbus master. If the gateway will be a Modbus slave, then matching network variables is more important and you should start on the NV Import page instead.



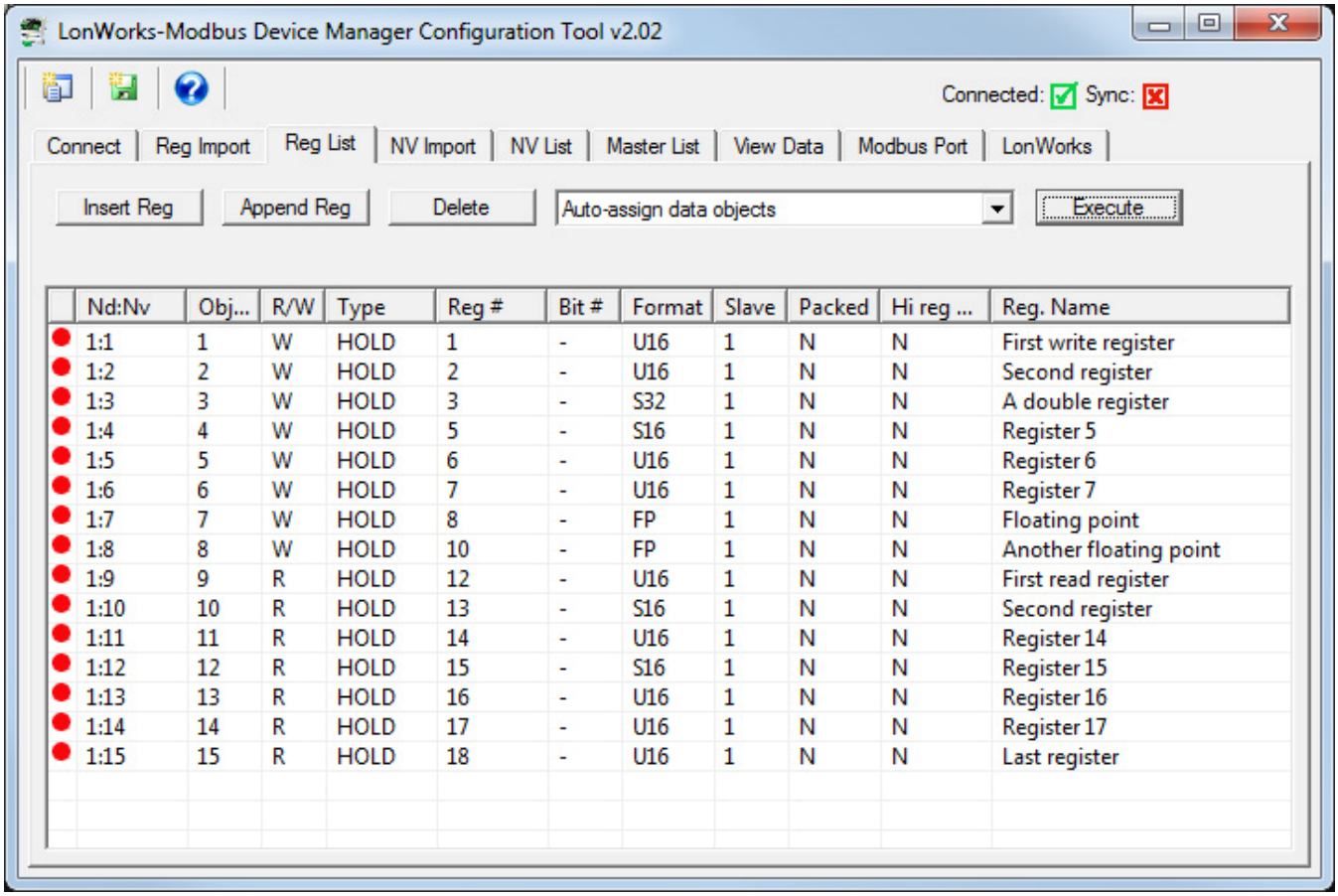
Upon executing the Auto-allocate Network Variable maps, the Nd:Nv (Node:Network Variable) column will be populated. Modbus registers designated as "W" for write, meaning the gateway will write to these registers in the Modbus device (assuming gateway is master), will be assigned to a Network Variable map having a Network Variable Output (NVO). The assumption here is that you want to read data from LonWorks and write it to Modbus.

Modbus registers designated as "R" for read, meaning the gateway will read these registers from the Modbus device, will be assigned to a Network Variable map having a Network Variable Input (NVI). The assumption here is that you want to read from Modbus and write that data to LonWorks.

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.

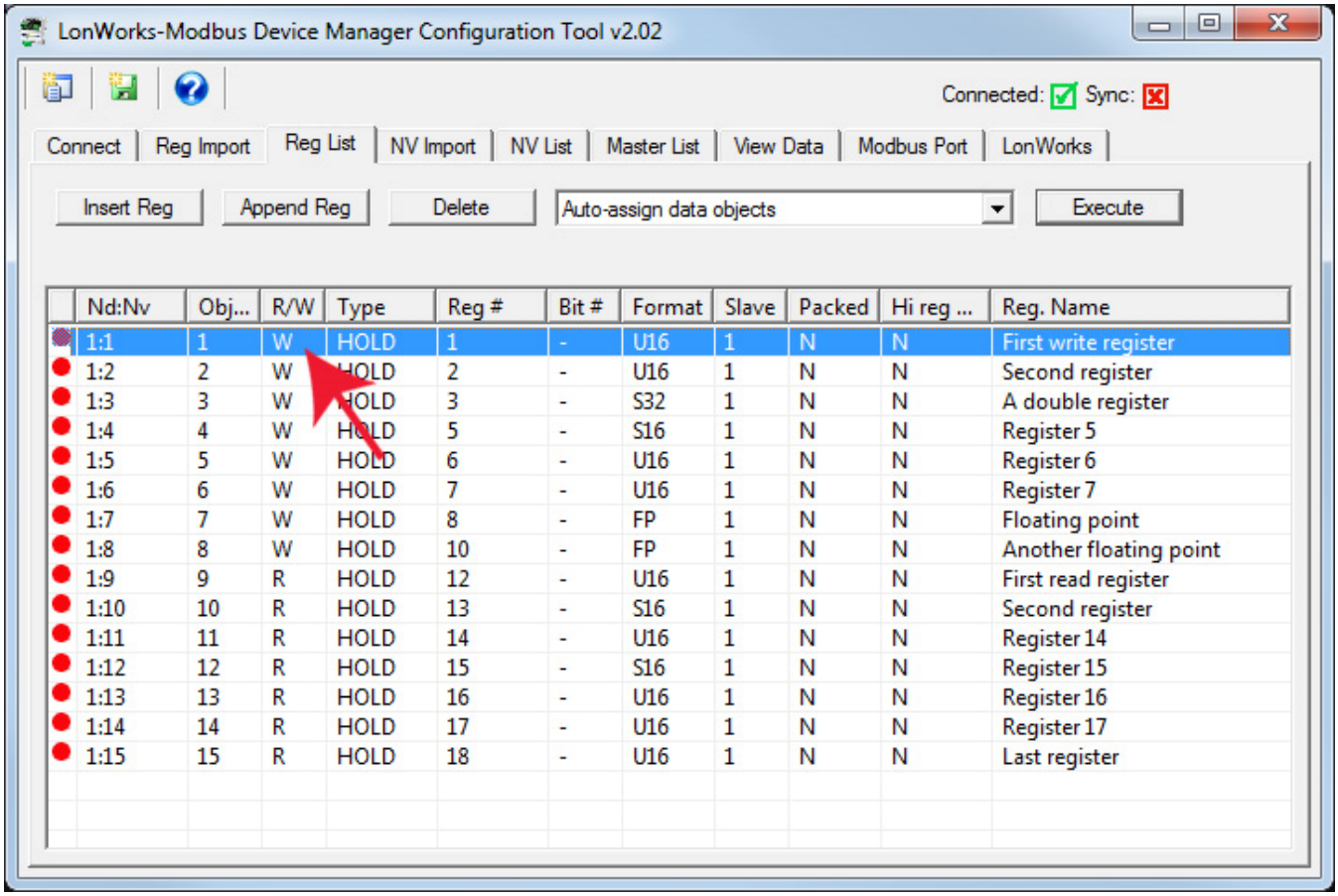


Nd:Nv	Obj...	R/W	Type	Reg #	Bit #	Format	Slave	Packed	Hi reg ...	Reg. Name
1:1	1	W	HOLD	1	-	U16	1	N	N	First write register
1:2	2	W	HOLD	2	-	U16	1	N	N	Second register
1:3	3	W	HOLD	3	-	S32	1	N	N	A double register
1:4	4	W	HOLD	5	-	S16	1	N	N	Register 5
1:5	5	W	HOLD	6	-	U16	1	N	N	Register 6
1:6	6	W	HOLD	7	-	U16	1	N	N	Register 7
1:7	7	W	HOLD	8	-	FP	1	N	N	Floating point
1:8	8	W	HOLD	10	-	FP	1	N	N	Another floating point
1:9	9	R	HOLD	12	-	U16	1	N	N	First read register
1:10	10	R	HOLD	13	-	S16	1	N	N	Second register
1:11	11	R	HOLD	14	-	U16	1	N	N	Register 14
1:12	12	R	HOLD	15	-	S16	1	N	N	Register 15
1:13	13	R	HOLD	16	-	U16	1	N	N	Register 16
1:14	14	R	HOLD	17	-	U16	1	N	N	Register 17
1:15	15	R	HOLD	18	-	U16	1	N	N	Last register

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.

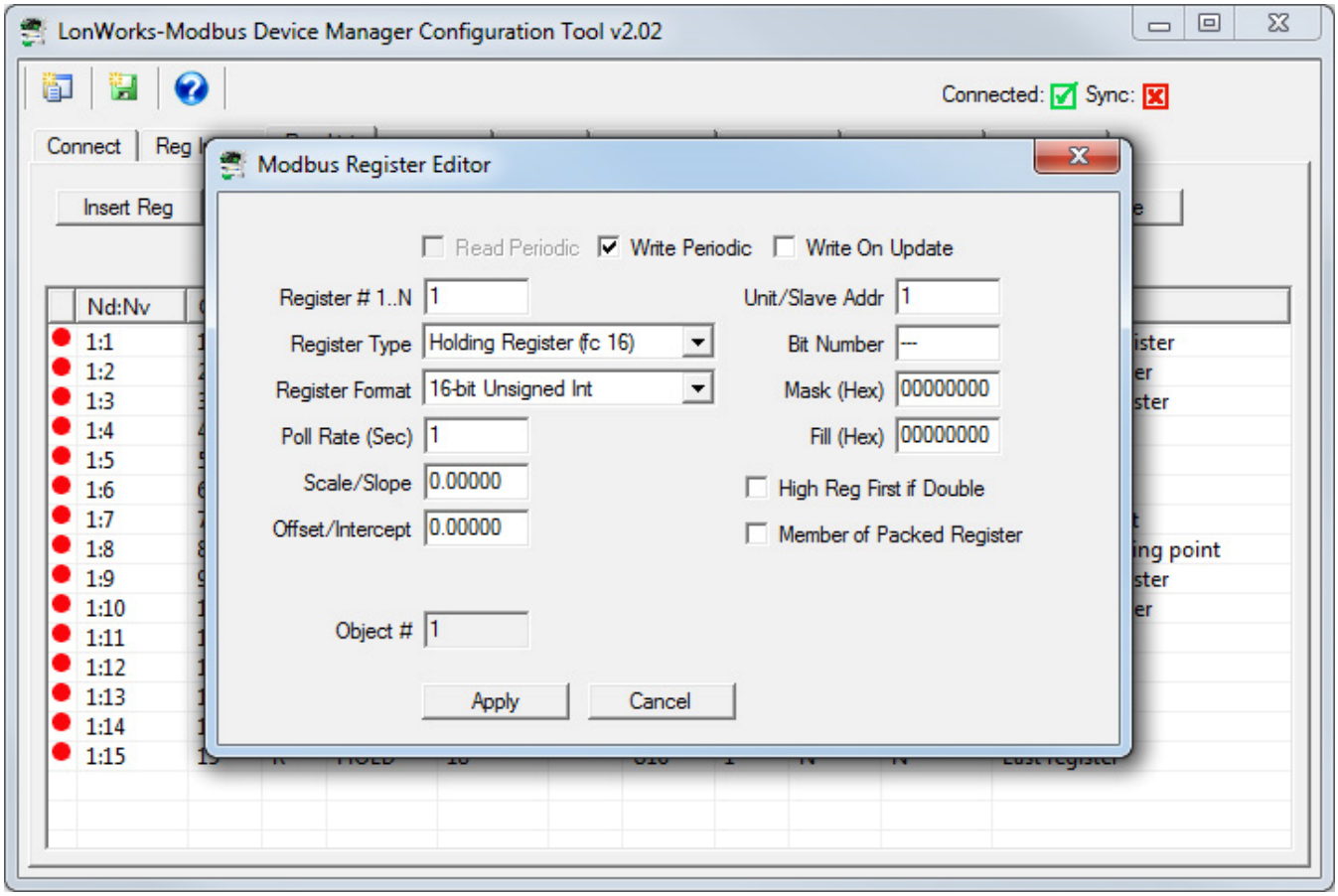
6.2 Editing the Register List

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



Nd:Nv	Obj...	R/W	Type	Reg #	Bit #	Format	Slave	Packed	Hi reg ...	Reg. Name
1:1	1	W	HOLD	1	-	U16	1	N	N	First write register
1:2	2	W	HOLD	2	-	U16	1	N	N	Second register
1:3	3	W	HOLD	3	-	S32	1	N	N	A double register
1:4	4	W	HOLD	5	-	S16	1	N	N	Register 5
1:5	5	W	HOLD	6	-	U16	1	N	N	Register 6
1:6	6	W	HOLD	7	-	U16	1	N	N	Register 7
1:7	7	W	HOLD	8	-	FP	1	N	N	Floating point
1:8	8	W	HOLD	10	-	FP	1	N	N	Another floating point
1:9	9	R	HOLD	12	-	U16	1	N	N	First read register
1:10	10	R	HOLD	13	-	S16	1	N	N	Second register
1:11	11	R	HOLD	14	-	U16	1	N	N	Register 14
1:12	12	R	HOLD	15	-	S16	1	N	N	Register 15
1:13	13	R	HOLD	16	-	U16	1	N	N	Register 16
1:14	14	R	HOLD	17	-	U16	1	N	N	Register 17
1:15	15	R	HOLD	18	-	U16	1	N	N	Last register

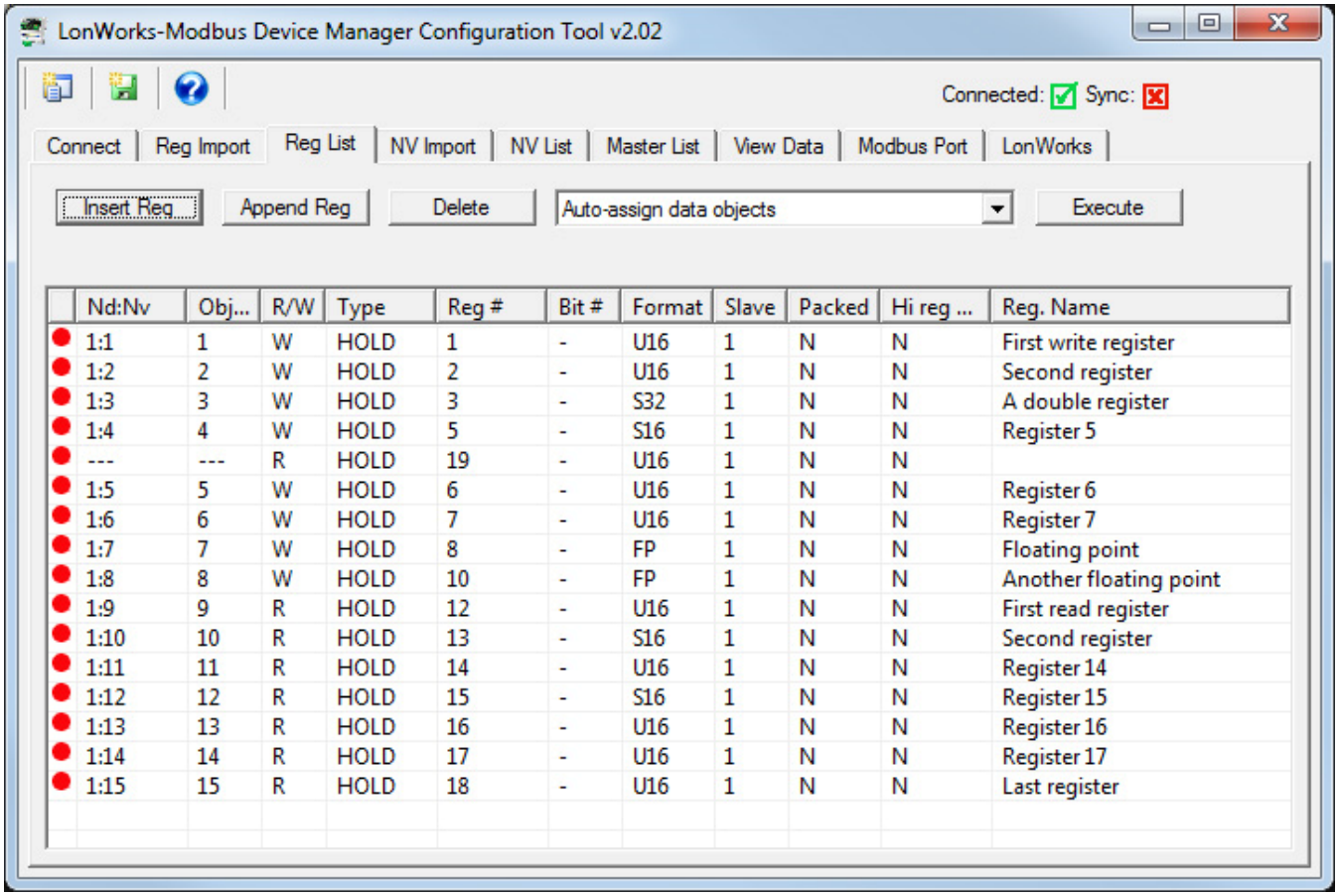
Upon double clicking a line on the Reg List, the dialog shown below will pop up for editing that entry.



After a register list has been imported, you can alter that list here. 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.

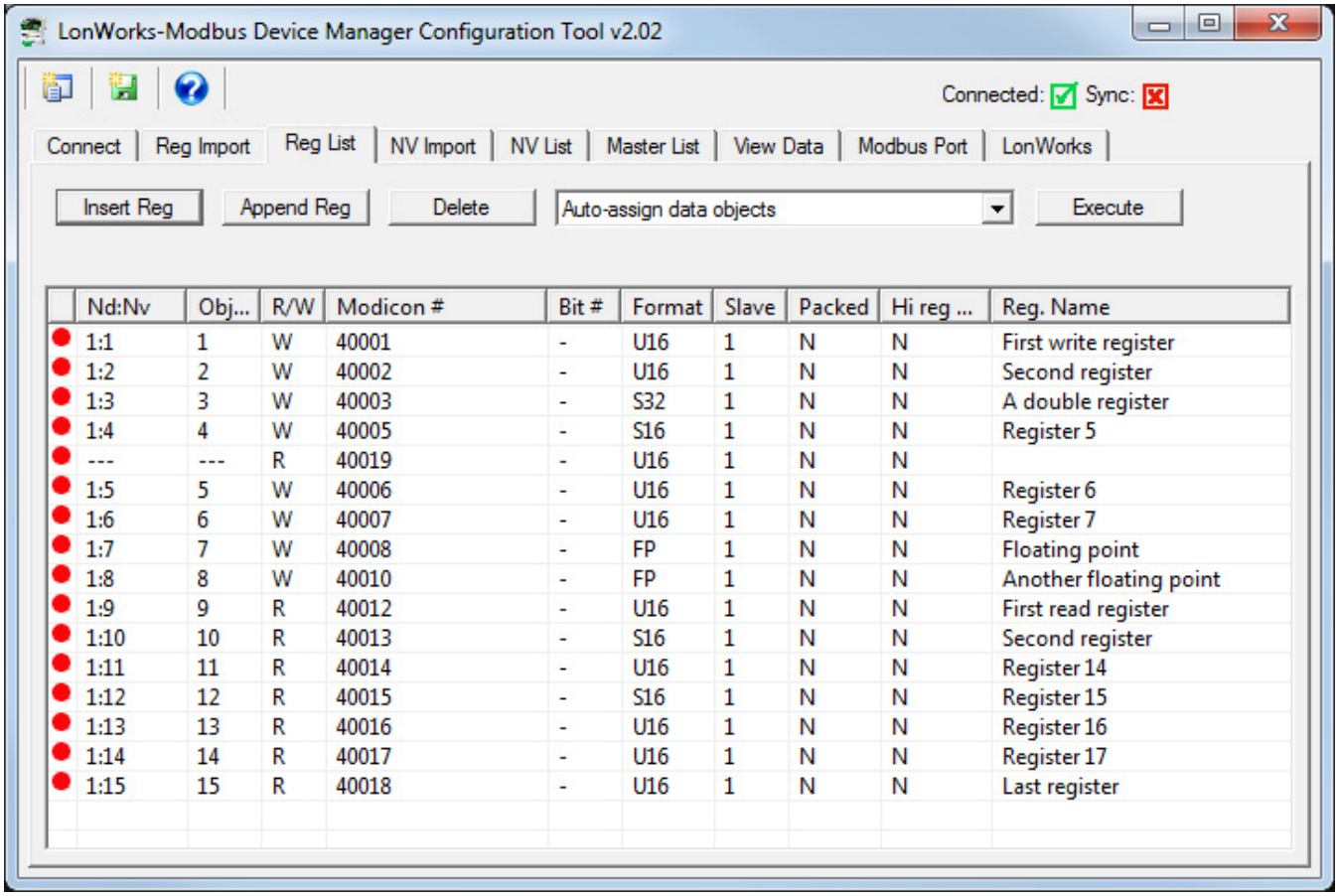
If Network Variable maps have not yet been allocated, you can change the Read/Write direction of the Modbus register map. Once a Network Variable map has been assigned, you cannot change the Modbus register direction; however, you can go to the NV List page, edit the NV there (changing its direction), then come back here and change the corresponding Modbus map direction.

The screen shot below illustrates a new register having been inserted between "Register 5" and "Register 6".



Nd:Nv	Obj...	R/W	Type	Reg #	Bit #	Format	Slave	Packed	Hi reg ...	Reg. Name
1:1	1	W	HOLD	1	-	U16	1	N	N	First write register
1:2	2	W	HOLD	2	-	U16	1	N	N	Second register
1:3	3	W	HOLD	3	-	S32	1	N	N	A double register
1:4	4	W	HOLD	5	-	S16	1	N	N	Register 5
---	---	R	HOLD	19	-	U16	1	N	N	
1:5	5	W	HOLD	6	-	U16	1	N	N	Register 6
1:6	6	W	HOLD	7	-	U16	1	N	N	Register 7
1:7	7	W	HOLD	8	-	FP	1	N	N	Floating point
1:8	8	W	HOLD	10	-	FP	1	N	N	Another floating point
1:9	9	R	HOLD	12	-	U16	1	N	N	First read register
1:10	10	R	HOLD	13	-	S16	1	N	N	Second register
1:11	11	R	HOLD	14	-	U16	1	N	N	Register 14
1:12	12	R	HOLD	15	-	S16	1	N	N	Register 15
1:13	13	R	HOLD	16	-	U16	1	N	N	Register 16
1:14	14	R	HOLD	17	-	U16	1	N	N	Register 17
1:15	15	R	HOLD	18	-	U16	1	N	N	Last register

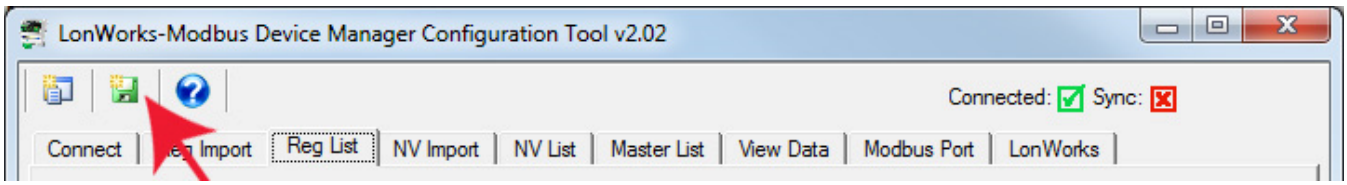
When Modicon display of register numbers has been selected on the Connect page, the Type and Reg # columns are replaced with a single Modicon # column. All other aspects of the Reg List page remain the same regardless of how register numbers are displayed.



Nd:Nv	Obj...	R/W	Modicon #	Bit #	Format	Slave	Packed	Hi reg ...	Reg. Name
1:1	1	W	40001	-	U16	1	N	N	First write register
1:2	2	W	40002	-	U16	1	N	N	Second register
1:3	3	W	40003	-	S32	1	N	N	A double register
1:4	4	W	40005	-	S16	1	N	N	Register 5
---	---	R	40019	-	U16	1	N	N	
1:5	5	W	40006	-	U16	1	N	N	Register 6
1:6	6	W	40007	-	U16	1	N	N	Register 7
1:7	7	W	40008	-	FP	1	N	N	Floating point
1:8	8	W	40010	-	FP	1	N	N	Another floating point
1:9	9	R	40012	-	U16	1	N	N	First read register
1:10	10	R	40013	-	S16	1	N	N	Second register
1:11	11	R	40014	-	U16	1	N	N	Register 14
1:12	12	R	40015	-	S16	1	N	N	Register 15
1:13	13	R	40016	-	U16	1	N	N	Register 16
1:14	14	R	40017	-	U16	1	N	N	Register 17
1:15	15	R	40018	-	U16	1	N	N	Last register

6.3 Register List Export

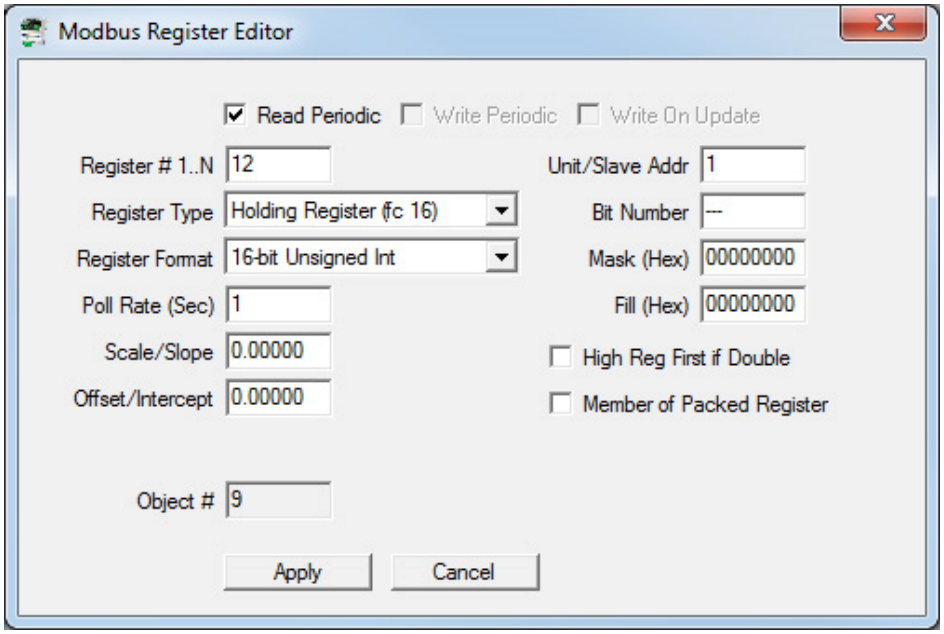
Once you have created a Modbus register list, you can export your register list.

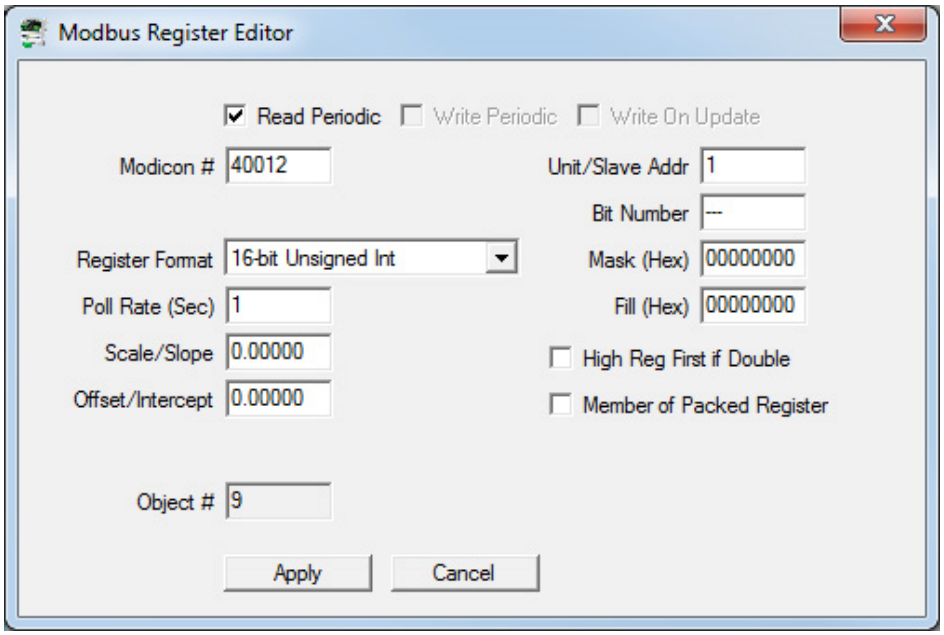


To export, click the new file icon on the toolbar. The CSV file that is exported will be in the file format described in Appendix D of this user guide. This means you can import the same file back into the configuration tool (on Reg Import page) at a later time, or open the file using a spreadsheet program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

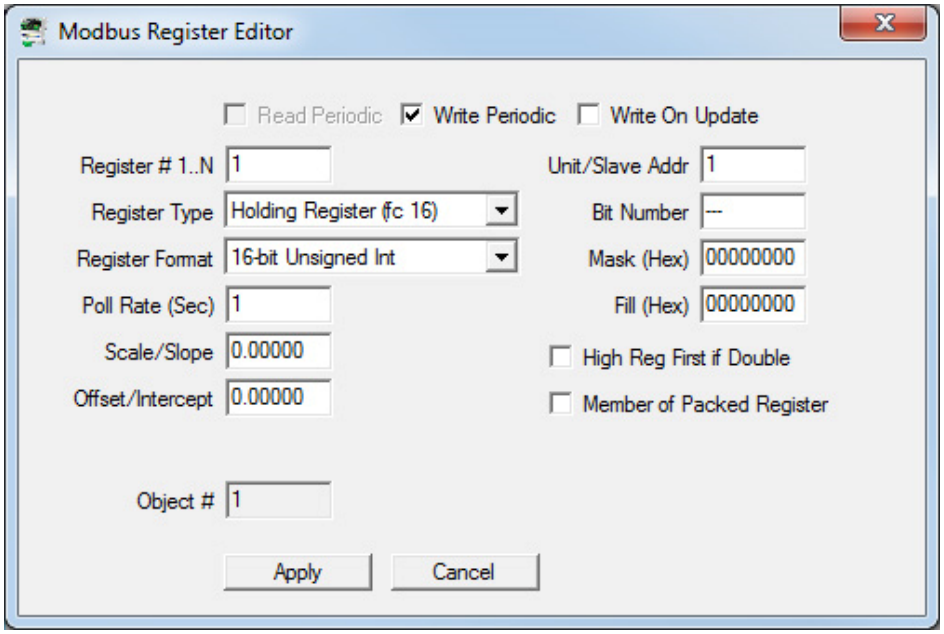
6.4 Definition of Modbus Register Configuration Parameters

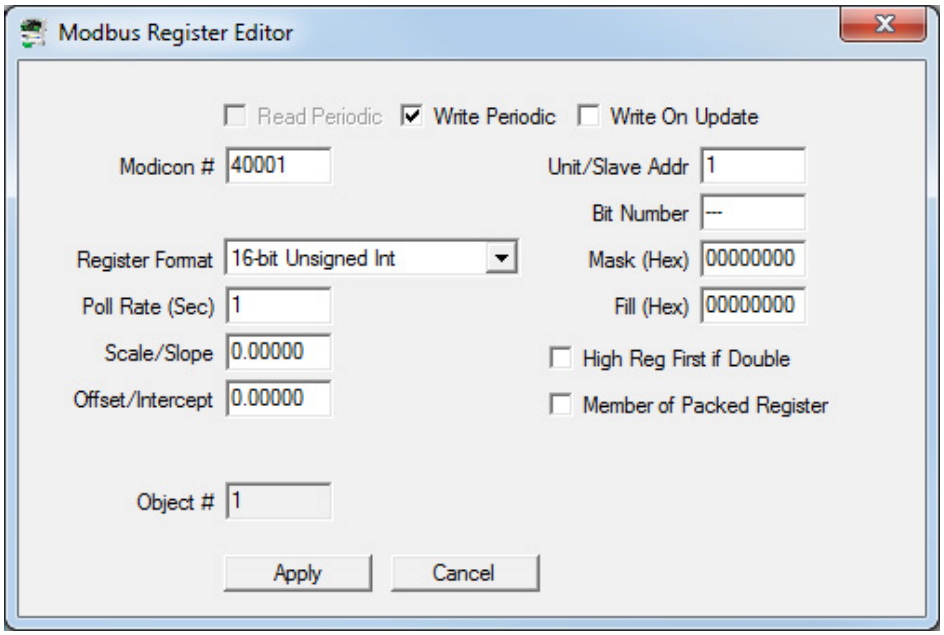
The Modbus Register Editor dialog is accessed by clicking on a line on the Reg List page, or by clicking the Modbus portion of a line on the Master List page. The dialog will vary slightly depending on whether the register is associated with an NVI or NVO, and whether or not you are using Modicon display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





A Modbus register associated with a Network Variable Input (NVI) will read from a Modbus slave when the gateway is functioning as Modbus master. Data will be read from Modbus and written to LonWorks. An NVI at the remote LonWorks device means it is a network input to that device and therefore will accept data from the gateway.





A Modbus register associated with a Network Variable Output (NVO) will write to a Modbus slave when the gateway is functioning as Modbus master. Data will be read from the LonWorks and written to Modbus. An NVO at the remote LonWorks device means it is a network output from that device and therefore provides data to other devices on the network.

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.

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.
Register Type	Valid register types are Coil, Discrete Input, Input Register, and Holding Register. Coil and holding register have two options, each indicated by a 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	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. 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. 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.
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	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 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. 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.
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	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. If you are getting seemingly random results when reading double registers, try swapping them to see if the order is reversed.
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.
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:

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 referred to as Modicon.

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.

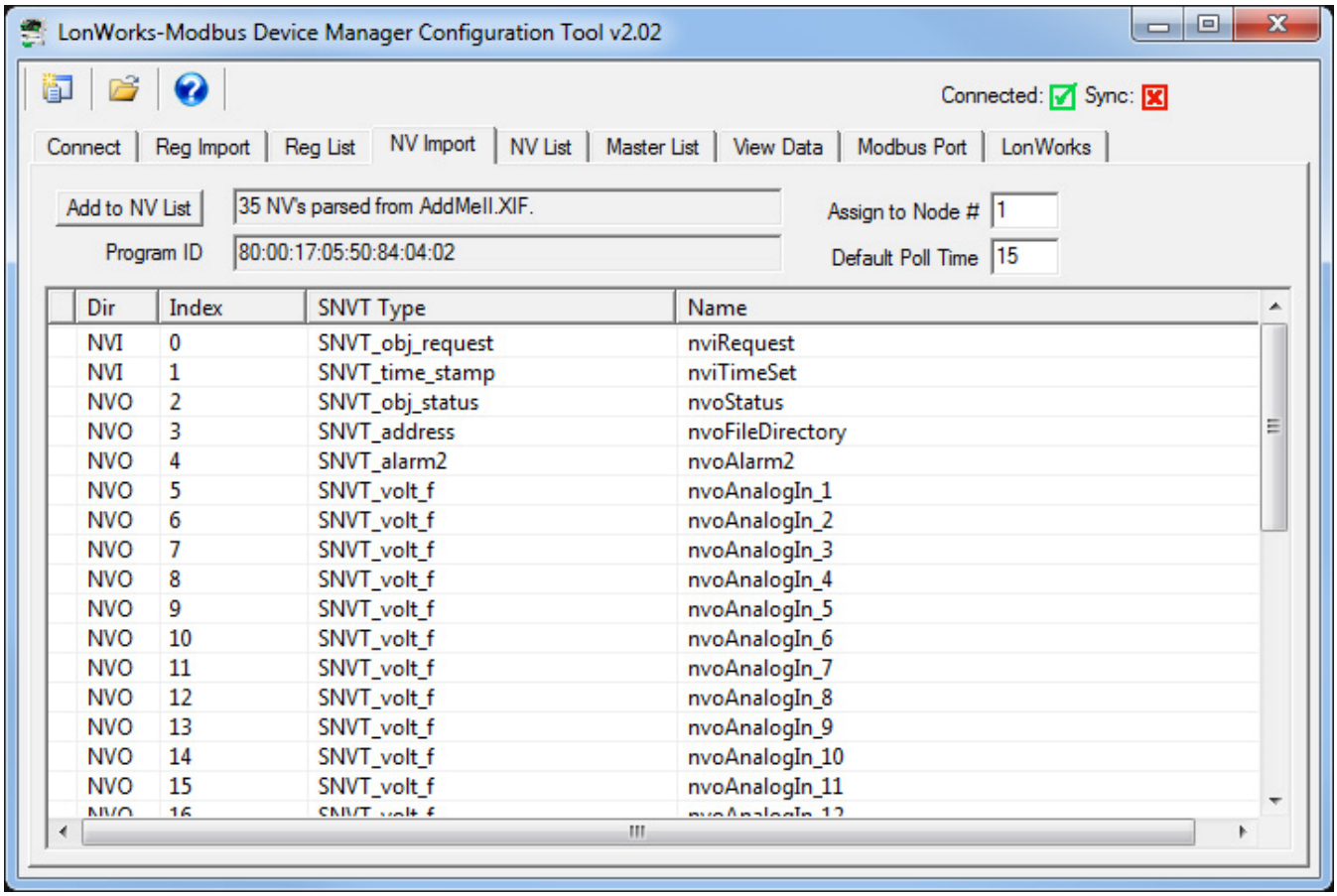
7 Tool 'NV Import' Page

7.1 Importing an XIF File

Starting from an XIF file for a specific LonWorks device, and auto-building the data object list, is the approach you would take if you want to make a LonWorks device accessible as a Modbus device on a Modbus network. This is the most common and most useful approach for the -NB versions of the LonWorks-Modbus gateways.

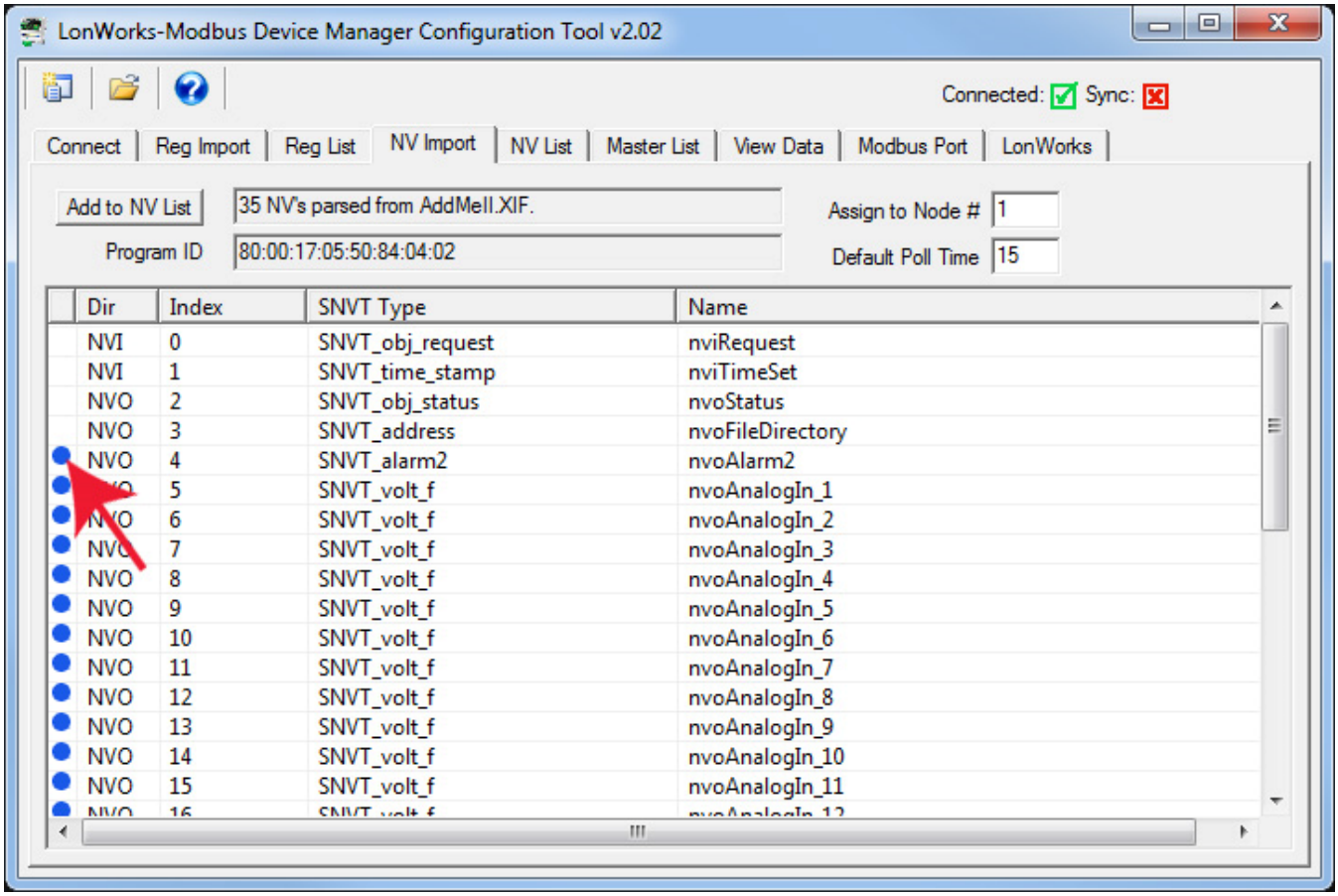
To begin building your configuration from an XIF file, go to the NV Import page. Click on the file icon to open an XIF file. Once the XIF is imported, the list of network variables will be displayed on the NV Import page.

Note: The XIF file is something you would obtain from the manufacturer of the LonWorks device. You can also use Echelon's NodeUtil tool to retrieve the XIF file from the device itself if you have a LonWorks network interface available on your PC. The third option is to use the Babel Buster gateway in conjunction with the configuration tool to retrieve the XIF information from the device. Although slightly more cumbersome, using the gateway to discover network variables in the device is an option and you would start this process on the LonWorks page of the configuration tool.



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.

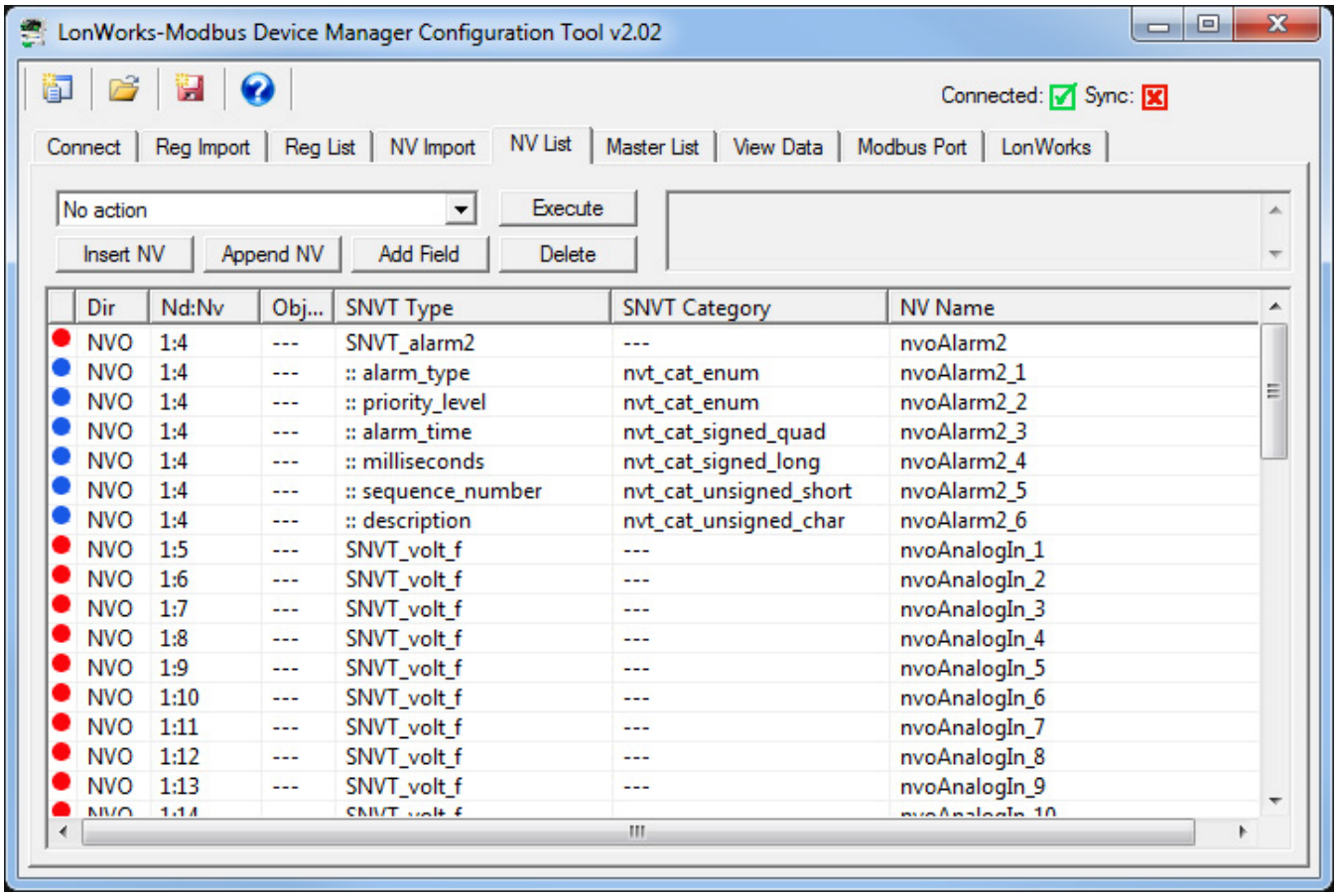
Although the maximum list size is set at 1000, the NV List is will never actually contain 1000 network variables. The BB2-2010-NB or BB2-6020-NB has a network variable mapping capacity of 300 network variables. If structured, the NV will map to multiple data objects. There is a pool of 400 data objects (local Modbus registers) to work with.

8 Tool 'NV List' Page

8.1 Configuration from XIF File

Regardless of whether you imported an XIF file starting on the NV Import page, or imported the XIF information from the device starting on the LonWorks page, you will end up with a list of network variables on the NV List page, and need to continue the configuration process here.

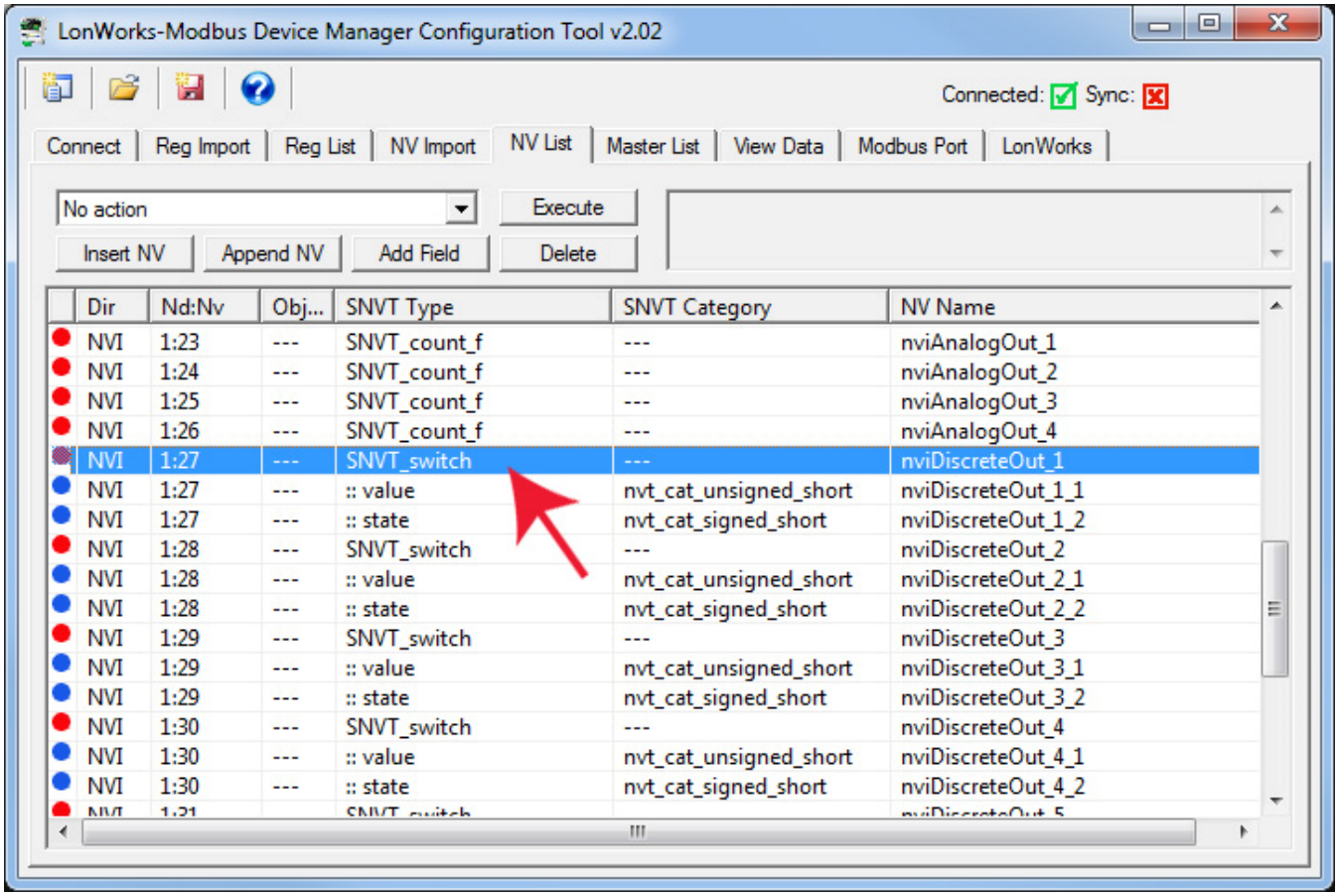
The NV List is the definition of the list of network variables in other LonWorks devices that will be polled by the gateway. The icon in the first column will be red if this NV definition has not yet been written to the gateway, and green if it has been written to the gateway. Blue icons indicate fields of structured network variables. All lines with a blue icon are part of the network variable immediately preceding the set of blue icons. There will be only one network variable, but multiple Modbus registers, for a structured network variable.



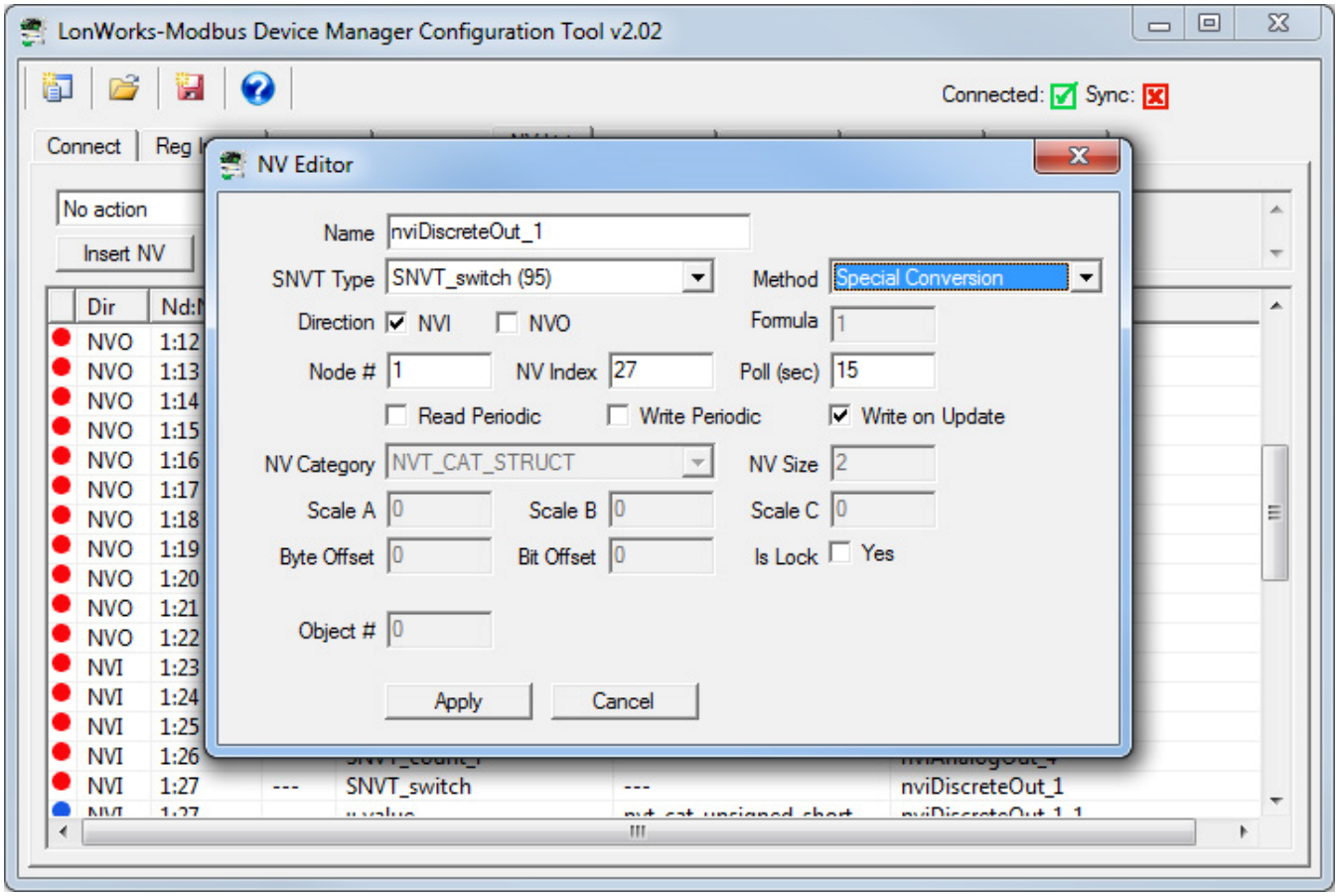
If you wish to make changes to the NV List, this is the point where you should do so. Do not proceed to assign data objects until the NV List is finalized.

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.

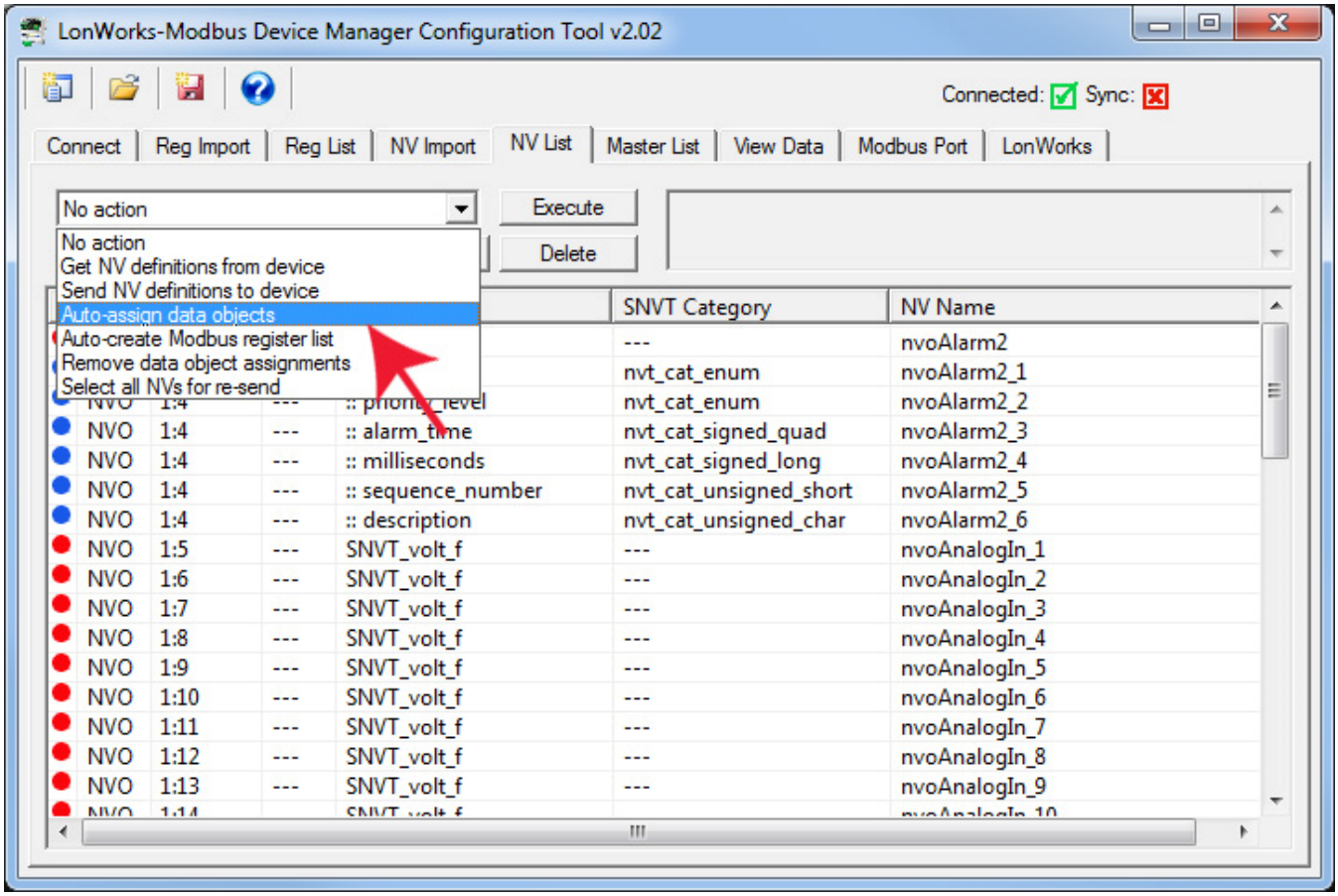
Some network variable types provide for special conversions. SNVT_switch is one such NV. To modify the NV definition, double click on the respective line in the NV List.



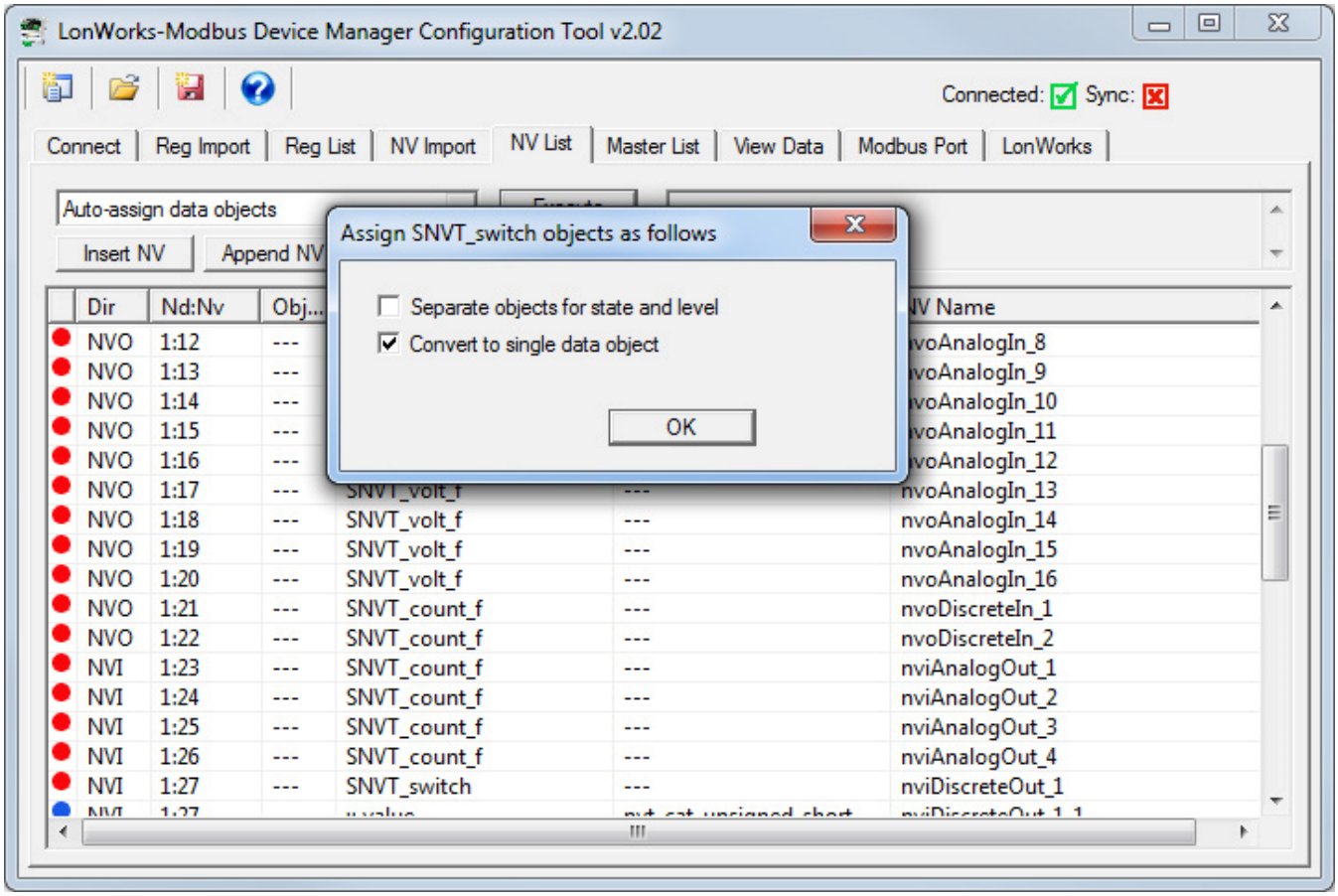
Upon double clicking a network variable in the NV List, the NV Editor dialog will appear. The SNVT_switch example is illustrated here. Your selection of whether to use the special conversion will decide whether SNVT_switch maps to one or two data objects, and therefore whether SNVT_switch maps to one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



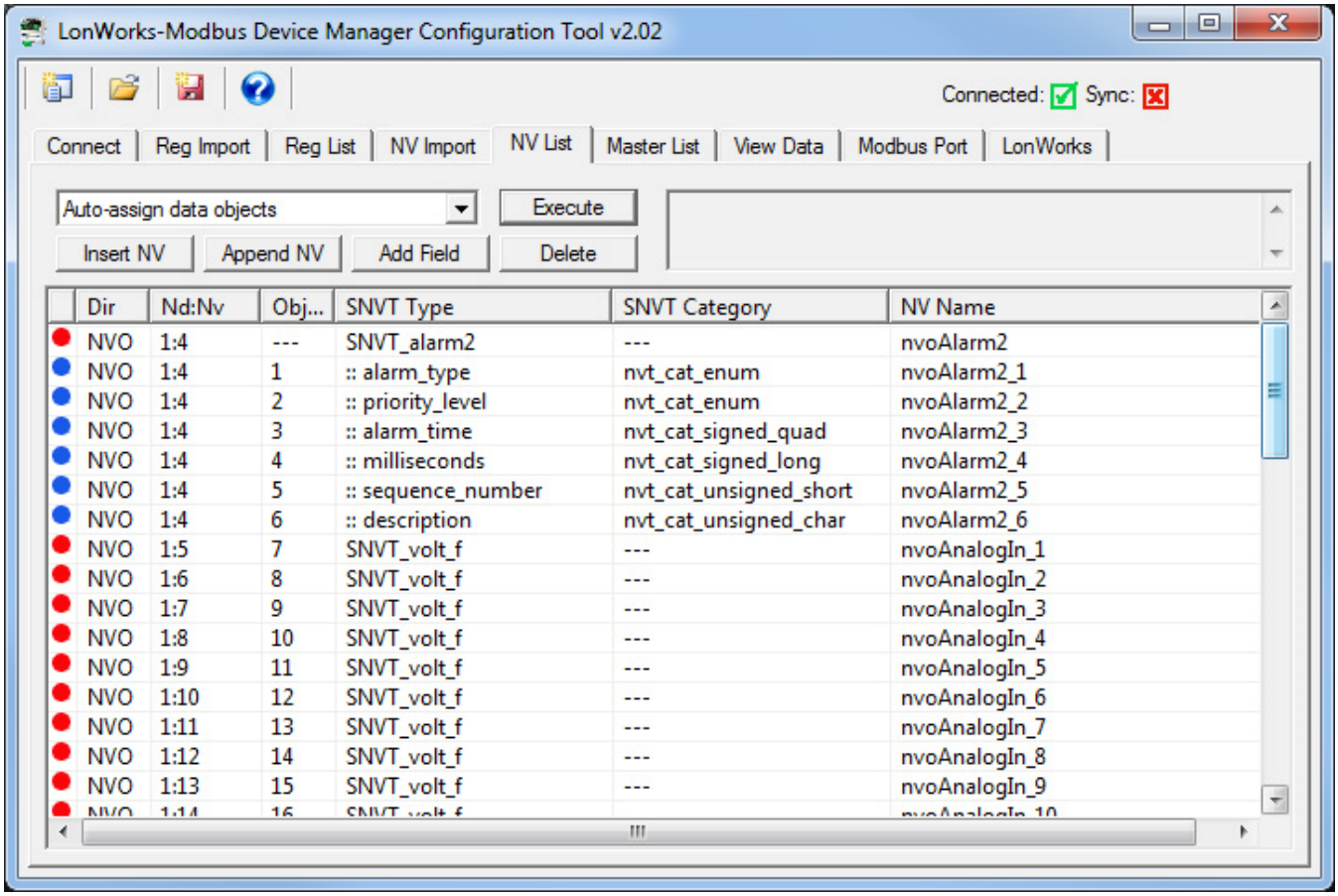
Once the list of network variables is acceptable to you, select and execute "Auto-assign data objects".



When auto-assigning objects, a dialog will pop up asking about your preference for treatment of any SNVT_switch variables that may be in the list. LonWorks treatment of switches assumes a dimmer type switch, and the SNVT_switch variable has both a state and a level, both of which must be provided on the LonWorks side. You have 2 options for how you will treat this on the Modbus side: (a) You can assign two objects (Modbus registers), one each for state and level; (b) You can assign a single data object which is then anticipating a value between 0 and 100 (percent implied) if accessed as a Modbus holding register, or simply on/off if accessed as a Modbus coil.

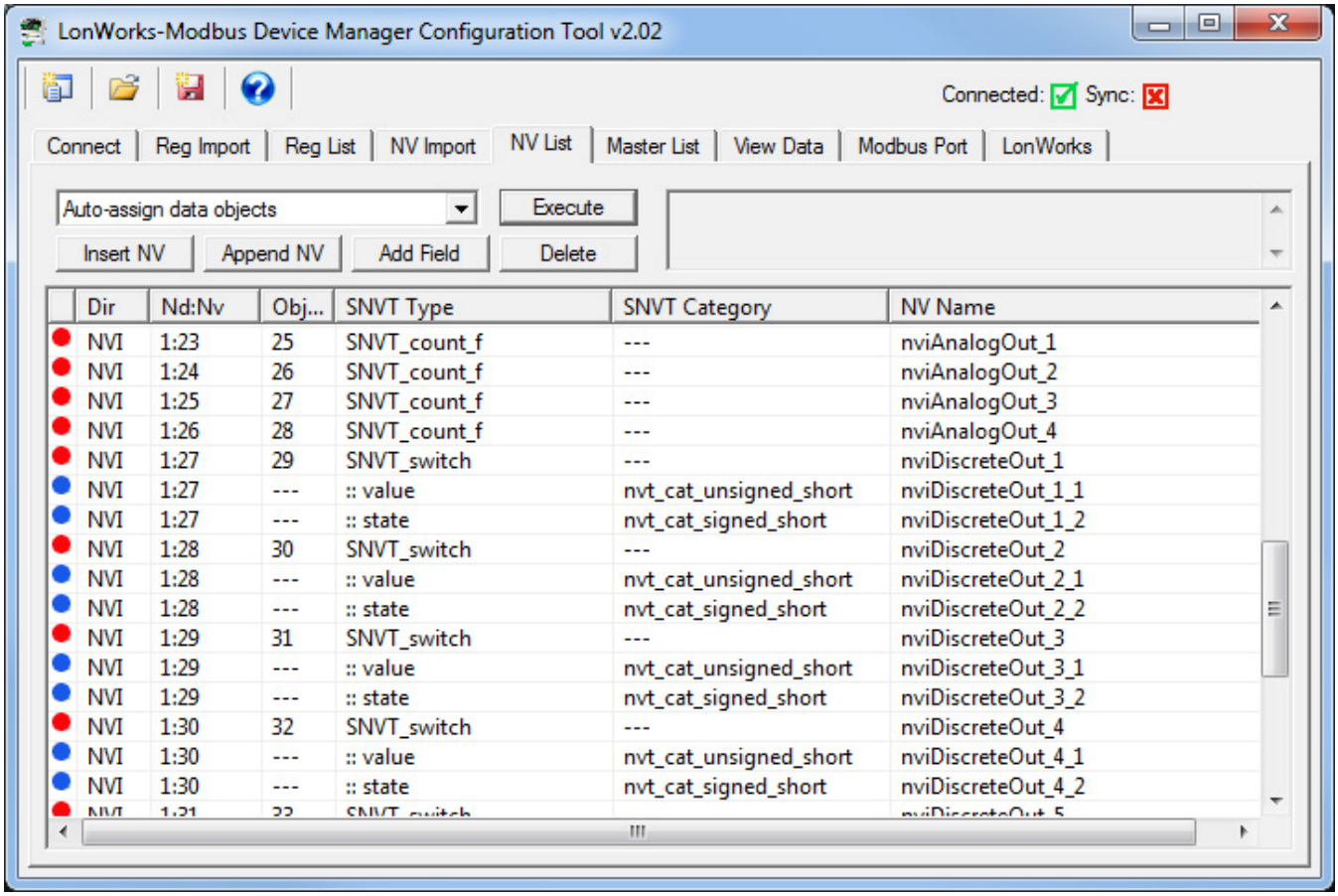


After data object assignment, the Obj... (Object) column will be populated. Note that if the NV is structured, the fields of the structures will be assigned to data objects, not the parent NV entry.



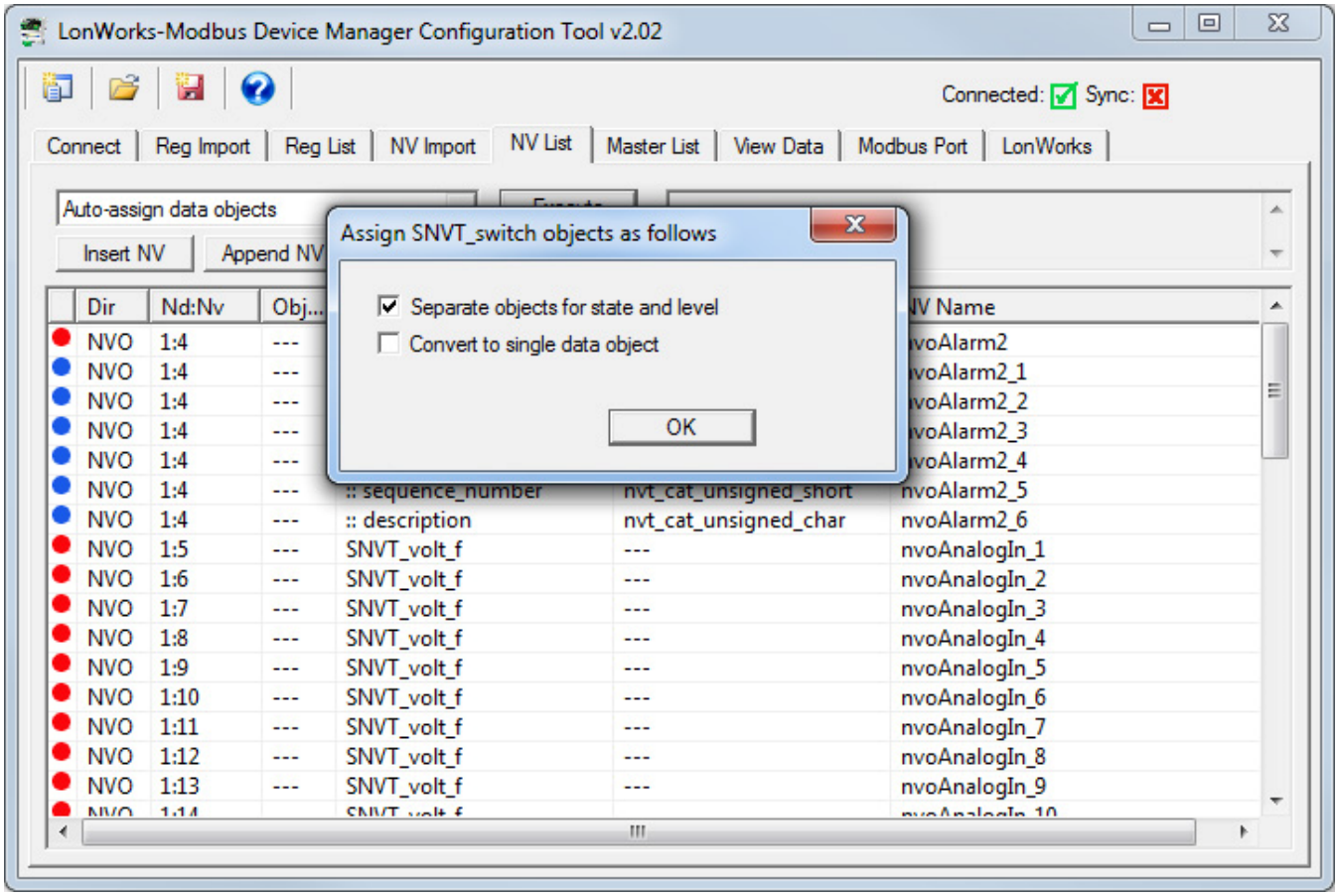
Dir	Nd:Nv	Obj...	SNVT Type	SNVT Category	NV Name
NVO	1:4	---	SNVT_alarm2	---	nvoAlarm2
NVO	1:4	1	:: alarm_type	nvt_cat_enum	nvoAlarm2_1
NVO	1:4	2	:: priority_level	nvt_cat_enum	nvoAlarm2_2
NVO	1:4	3	:: alarm_time	nvt_cat_signed_quad	nvoAlarm2_3
NVO	1:4	4	:: milliseconds	nvt_cat_signed_long	nvoAlarm2_4
NVO	1:4	5	:: sequence_number	nvt_cat_unsigned_short	nvoAlarm2_5
NVO	1:4	6	:: description	nvt_cat_unsigned_char	nvoAlarm2_6
NVO	1:5	7	SNVT_volt_f	---	nvoAnalogIn_1
NVO	1:6	8	SNVT_volt_f	---	nvoAnalogIn_2
NVO	1:7	9	SNVT_volt_f	---	nvoAnalogIn_3
NVO	1:8	10	SNVT_volt_f	---	nvoAnalogIn_4
NVO	1:9	11	SNVT_volt_f	---	nvoAnalogIn_5
NVO	1:10	12	SNVT_volt_f	---	nvoAnalogIn_6
NVO	1:11	13	SNVT_volt_f	---	nvoAnalogIn_7
NVO	1:12	14	SNVT_volt_f	---	nvoAnalogIn_8
NVO	1:13	15	SNVT_volt_f	---	nvoAnalogIn_9

The example configuration below shows that SNVT_switch variables have been converted to a single register.

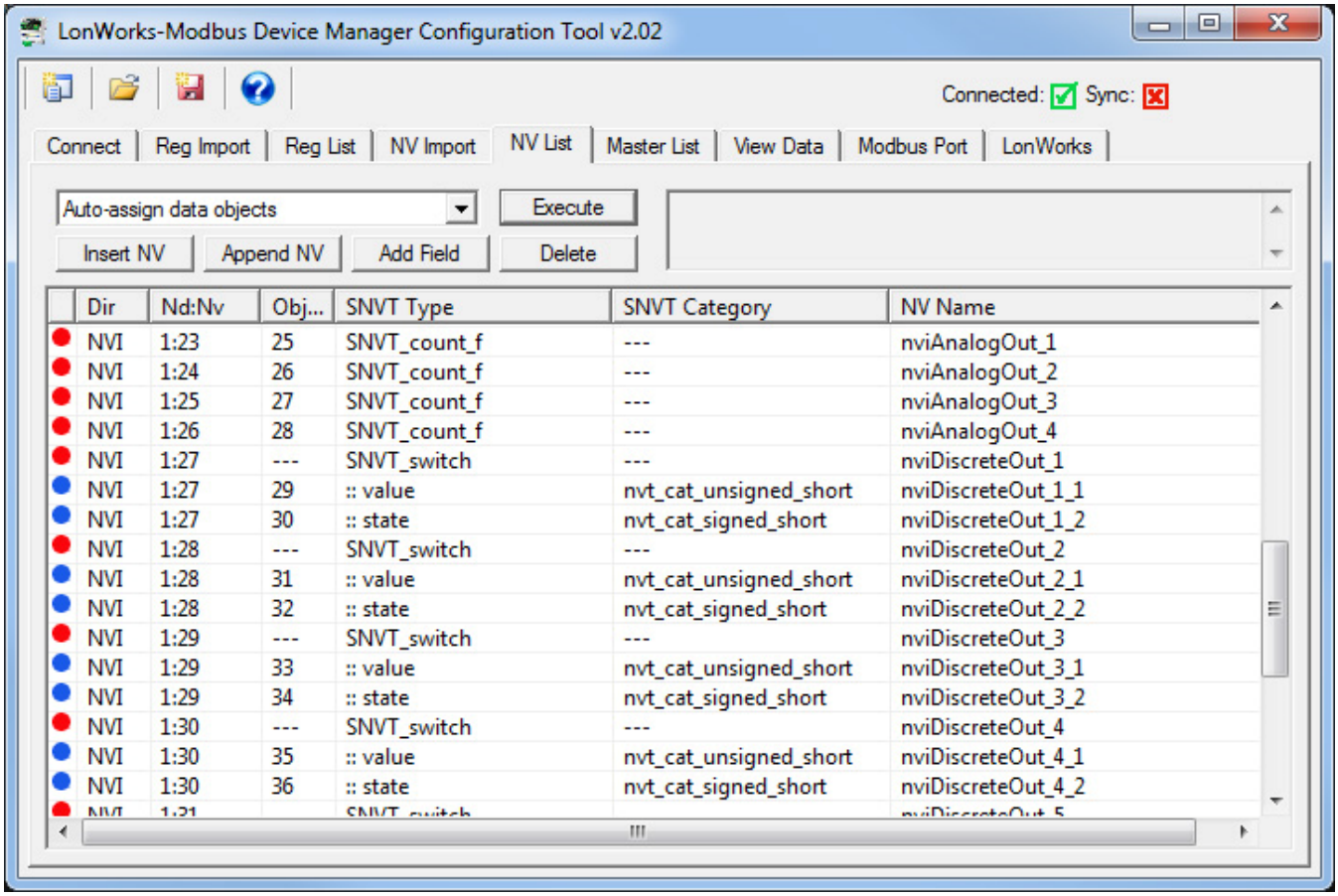


Dir	Nd:Nv	Obj...	SNVT Type	SNVT Category	NV Name
NVI	1:23	25	SNVT_count_f	---	nviAnalogOut_1
NVI	1:24	26	SNVT_count_f	---	nviAnalogOut_2
NVI	1:25	27	SNVT_count_f	---	nviAnalogOut_3
NVI	1:26	28	SNVT_count_f	---	nviAnalogOut_4
NVI	1:27	29	SNVT_switch	---	nviDiscreteOut_1
NVI	1:27	---	:: value	nvt_cat_unsigned_short	nviDiscreteOut_1_1
NVI	1:27	---	:: state	nvt_cat_signed_short	nviDiscreteOut_1_2
NVI	1:28	30	SNVT_switch	---	nviDiscreteOut_2
NVI	1:28	---	:: value	nvt_cat_unsigned_short	nviDiscreteOut_2_1
NVI	1:28	---	:: state	nvt_cat_signed_short	nviDiscreteOut_2_2
NVI	1:29	31	SNVT_switch	---	nviDiscreteOut_3
NVI	1:29	---	:: value	nvt_cat_unsigned_short	nviDiscreteOut_3_1
NVI	1:29	---	:: state	nvt_cat_signed_short	nviDiscreteOut_3_2
NVI	1:30	32	SNVT_switch	---	nviDiscreteOut_4
NVI	1:30	---	:: value	nvt_cat_unsigned_short	nviDiscreteOut_4_1
NVI	1:30	---	:: state	nvt_cat_signed_short	nviDiscreteOut_4_2

You might also select 'Separate objects for state and level'.



The resulting automatic assignment of objects will then appear as follows.

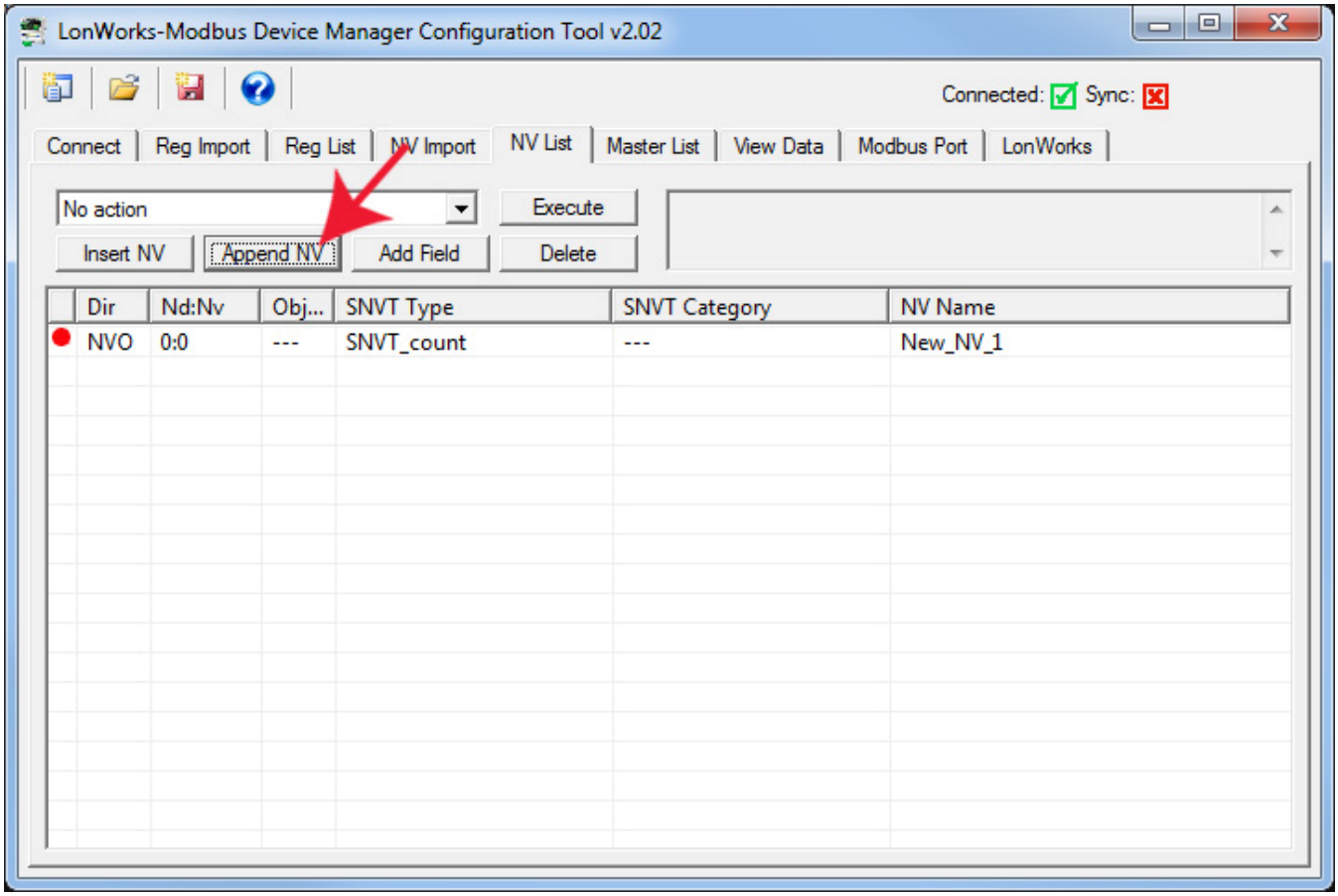


8.2 Building Configuration Manually

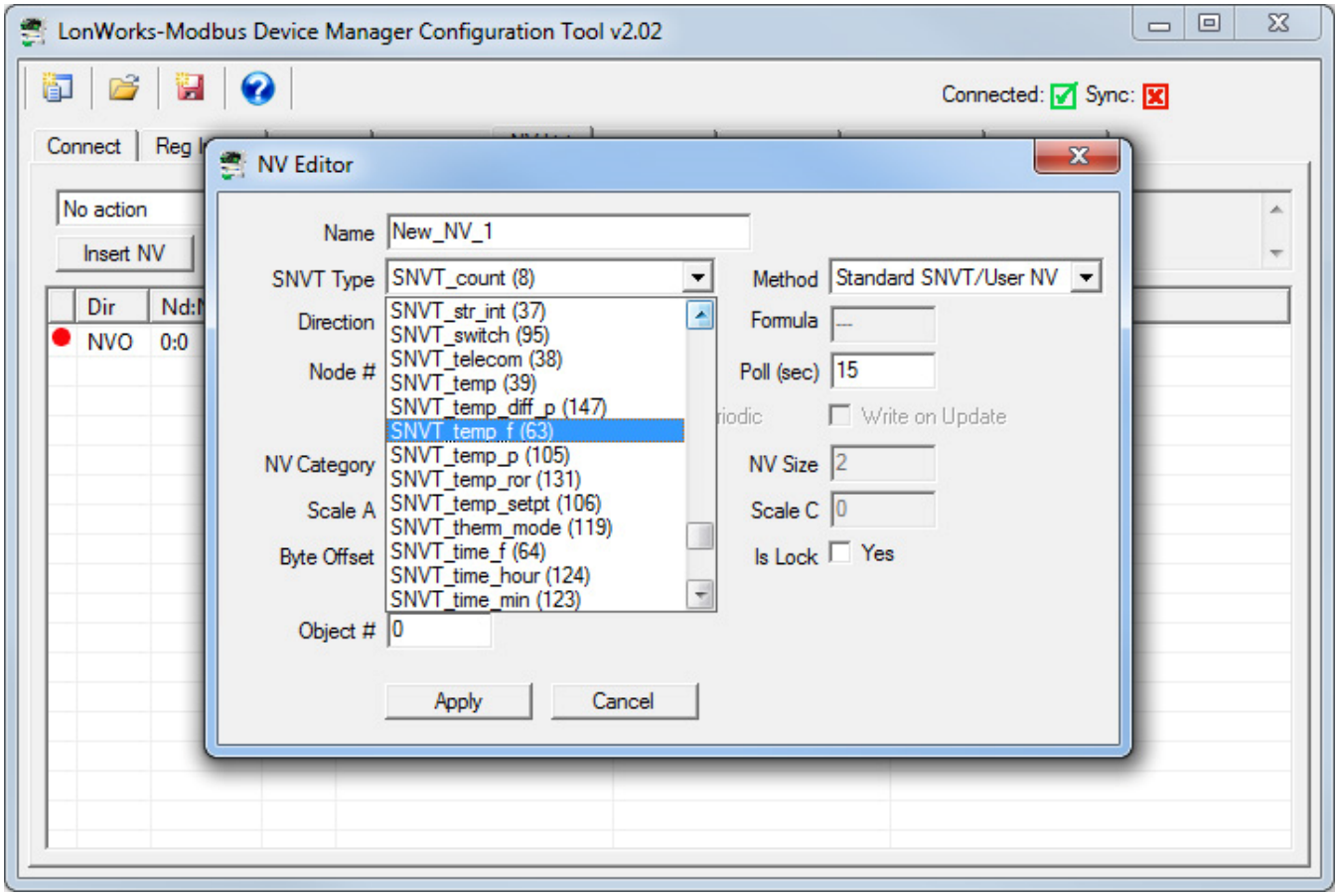
It is not required that you start with either a CSV file or an XIF file. You can use the configuration tool to start with a blank slate and build your configuration from scratch.

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.

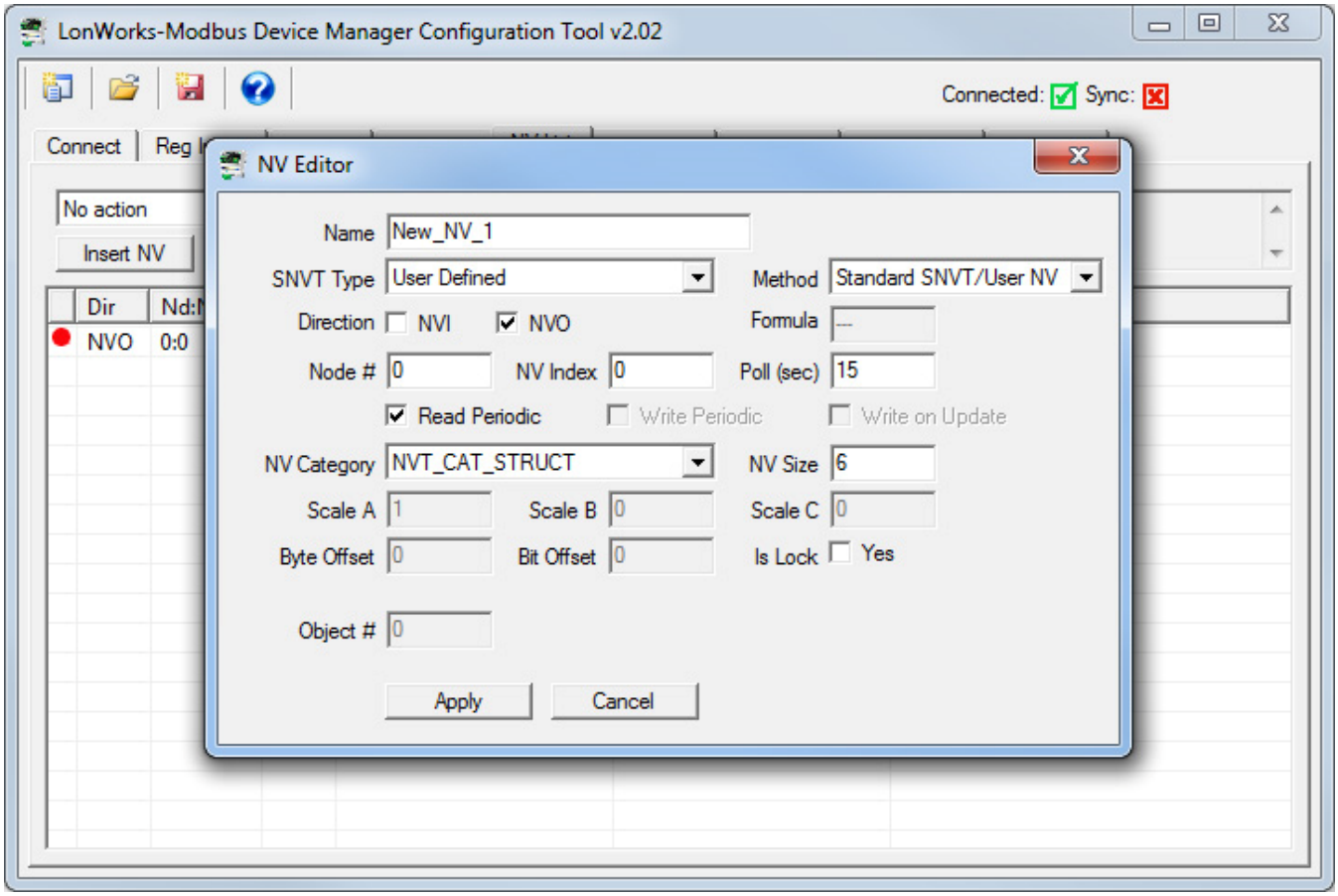
For each network variable you wish to add, click Append NV to add one at the end of the list, or Insert NV to add one immediately before the selected line on the list (if any).



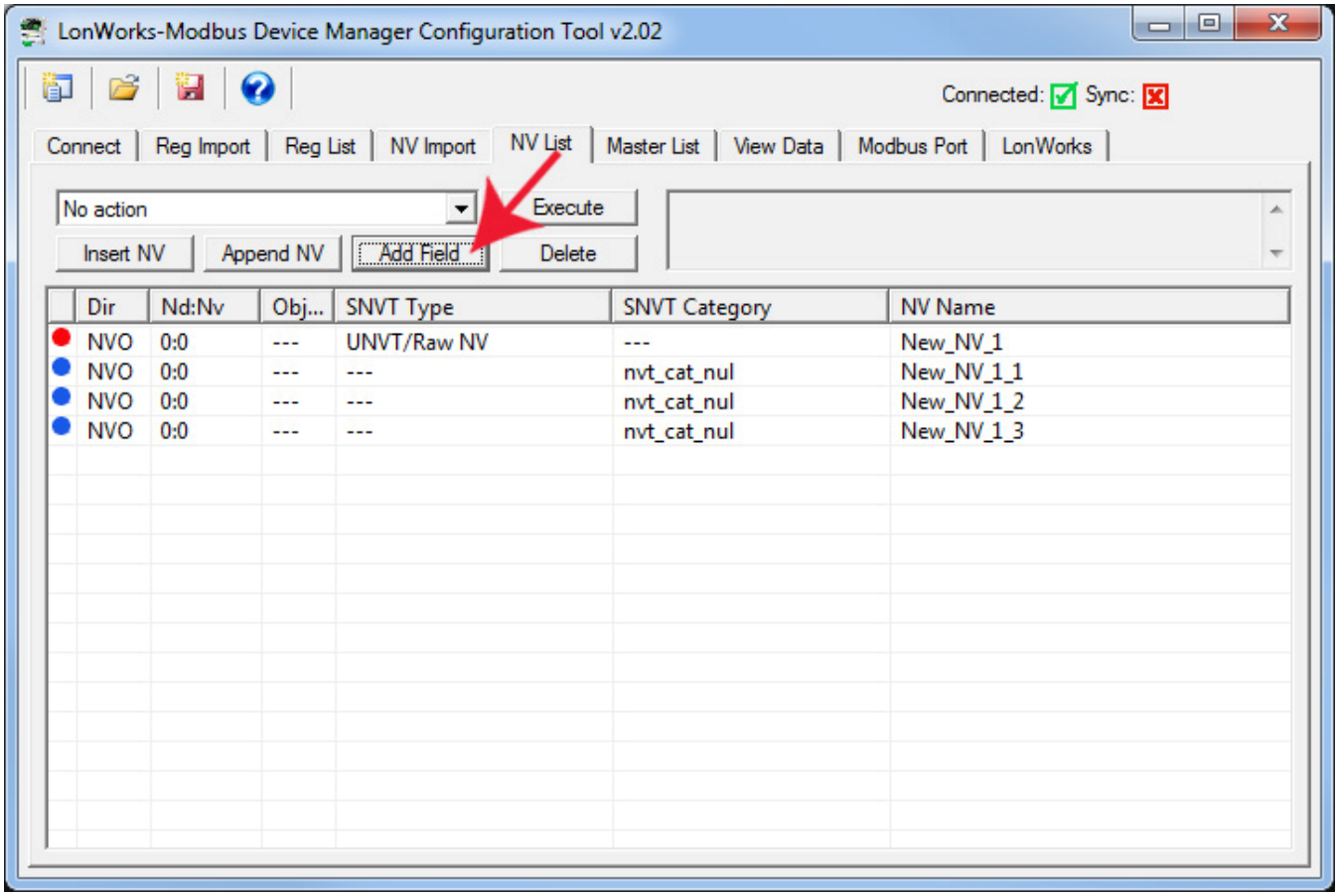
The newly inserted NV will default to SNVT_count. If you wish to change it, double click on the line to be changed, and the NV Editor dialog will appear. In the example that follows, we will complicate things to the maximum by creating a user defined structured network variable.



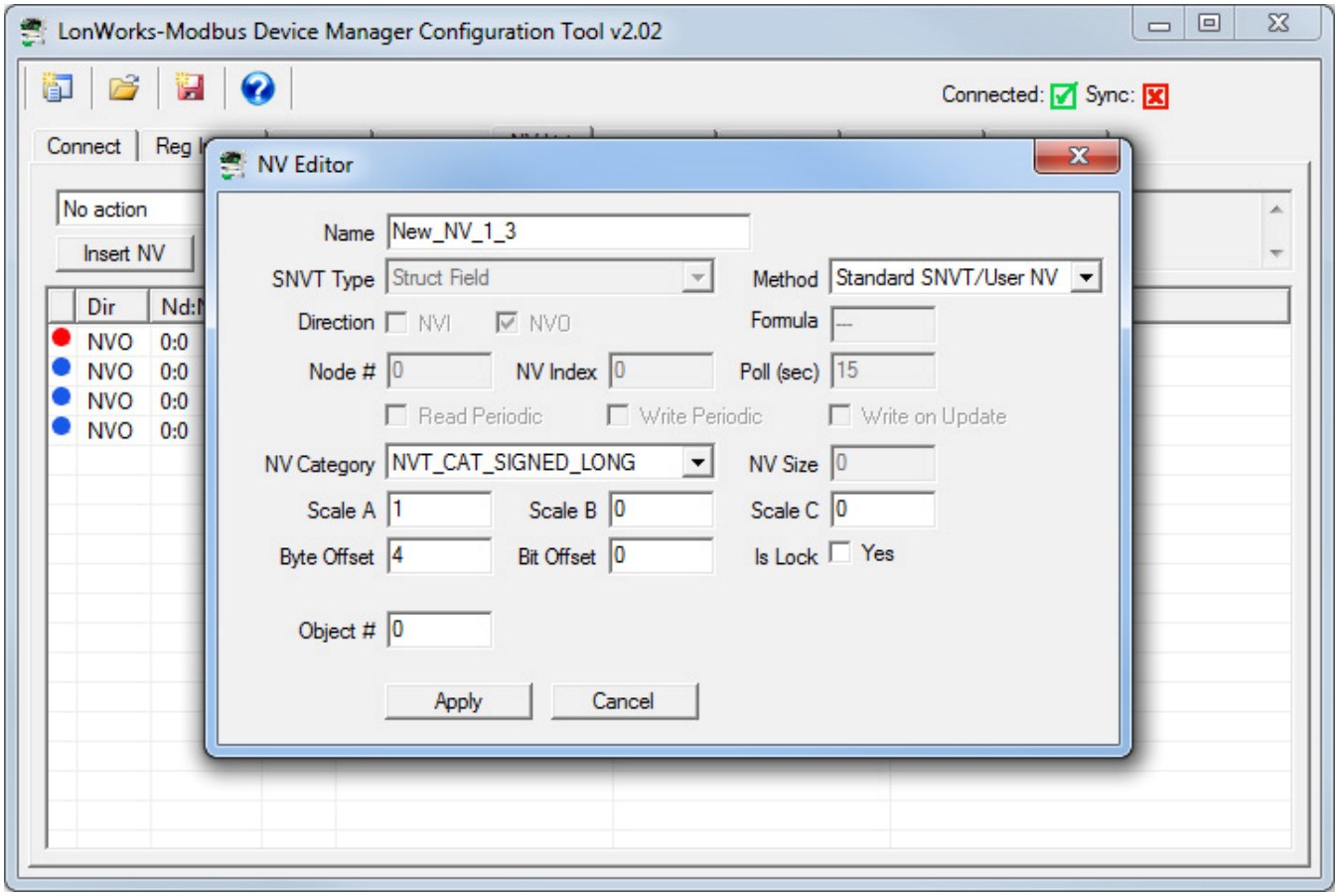
When creating a structure, the NV needs to be set to NV Category NVT_CAT_STRUCT and the NV Size needs to be set to the number of bytes that make up this variable in the LonWorks device.



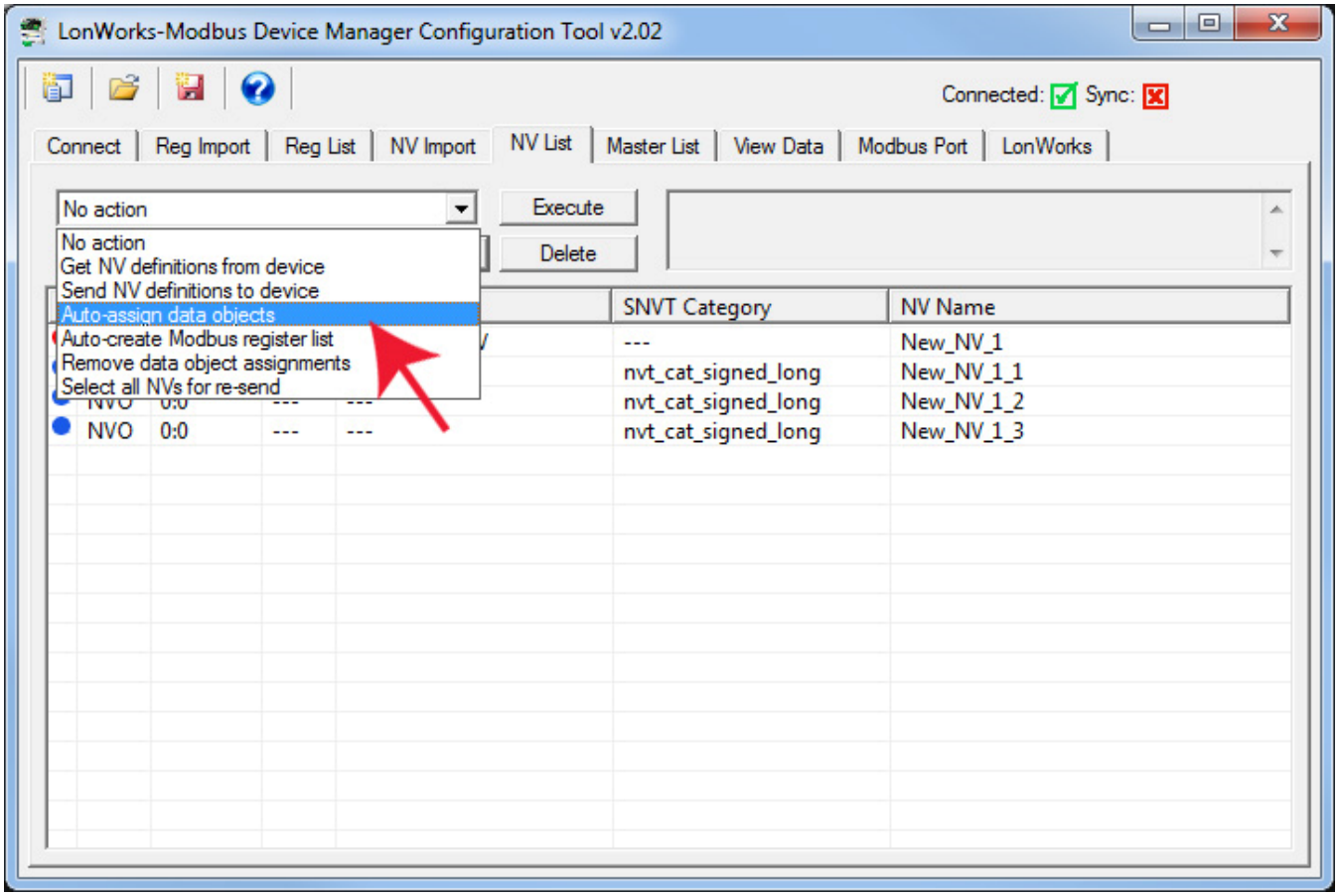
After defining the NV as a structure, you next need to add fields to the structure. Do this by clicking Add Field.



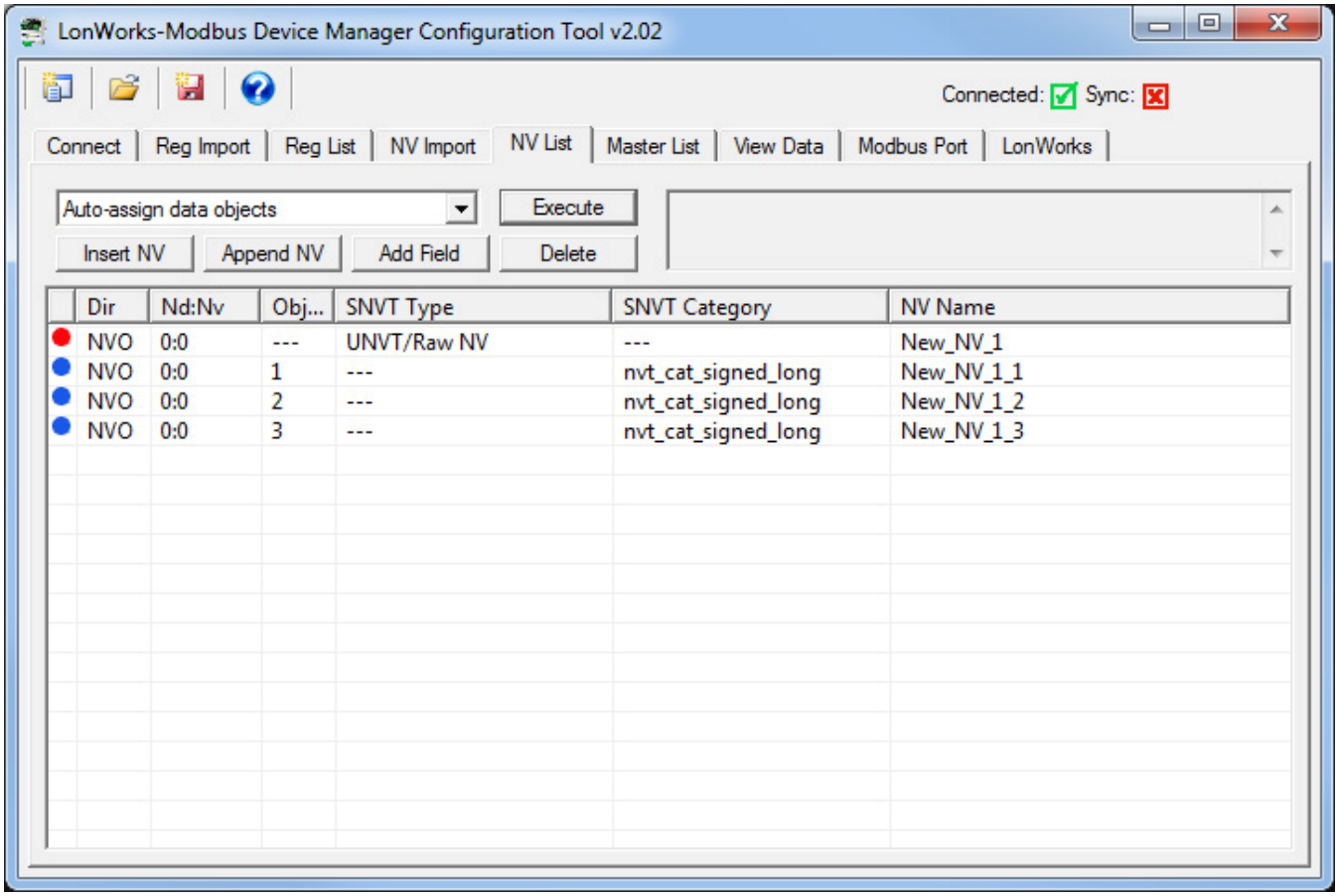
Next, edit each field by double clicking on that line. Specify the data type for the field. It is also important to specify the Byte Offset in the structure. An offset of zero means this field occupies the first byte(s) of the structure. You also need to enter the scale values. These follow the LonMark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



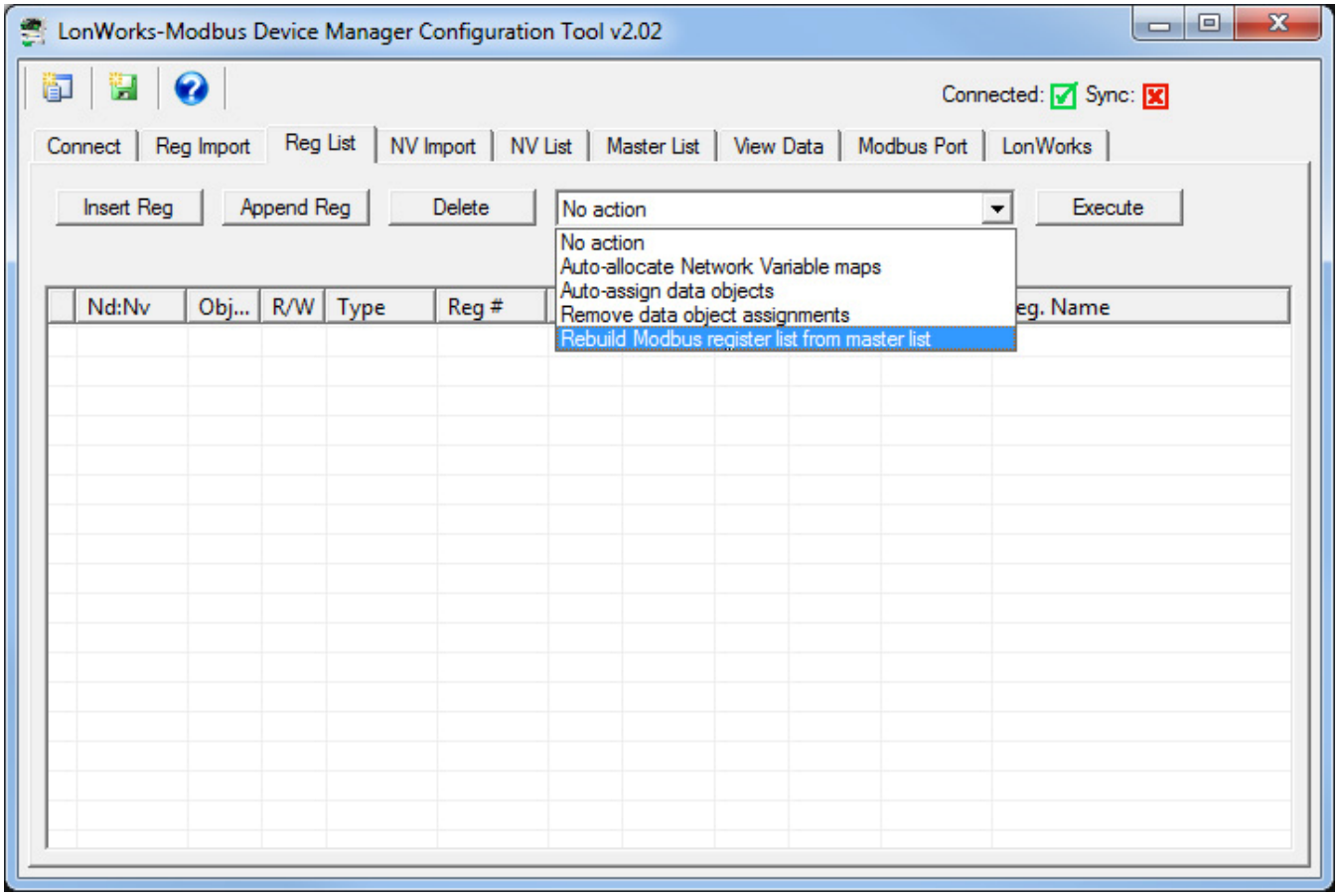
In addition to setting up the Network Variable itself, you need to assign data objects. These can be manually entered, or automatically assigned by the configuration tool. Once assigned, you need to go to the Master List page, click on the object portion of the Master List entry, and change type or instance there.



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



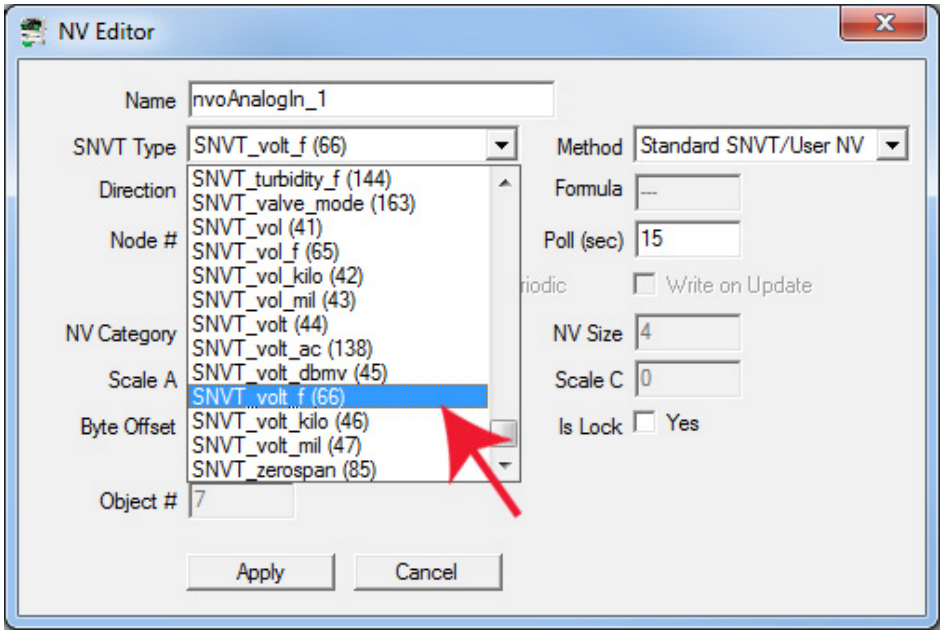
IMPORTANT: If you have manually assigned data objects using the NV Editor dialog, you must select 'Rebuild Modbus register list from master list' from the action list on the NV List page and click Execute. Until you do this, registers will not appear in the register list on the Reg List page.



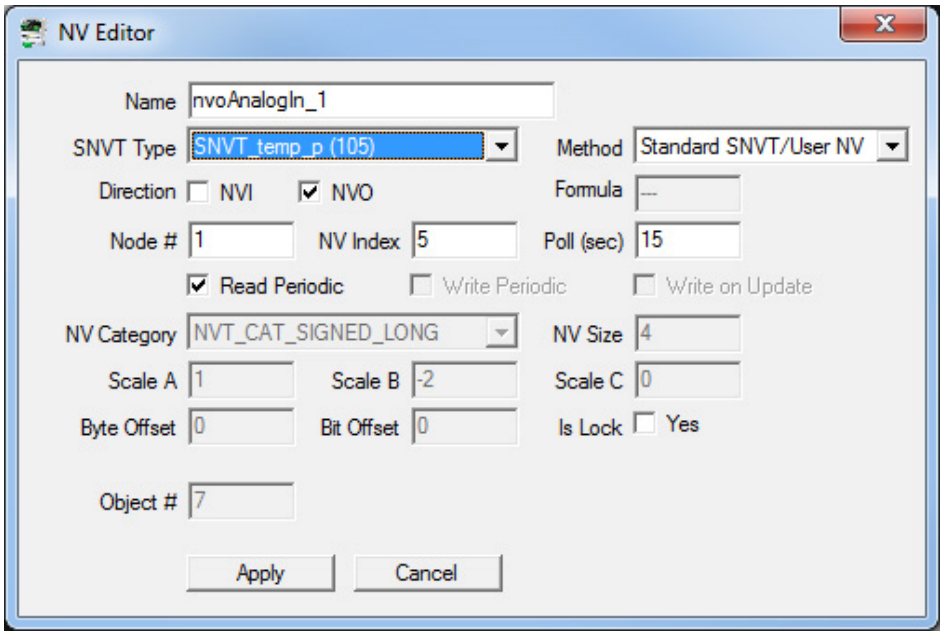
8.3 Using the NV Editor

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.

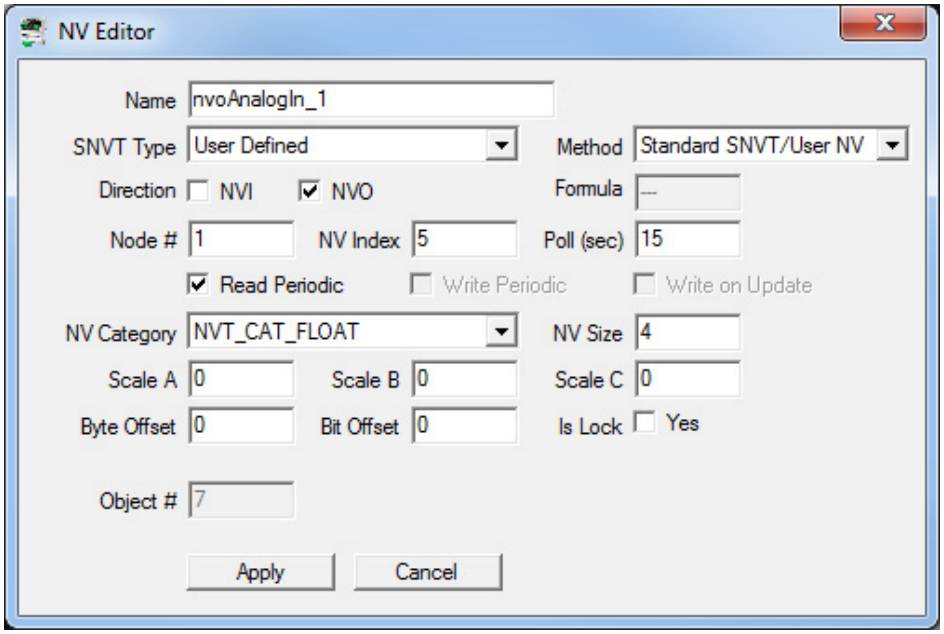
The following example illustrates the most simple of the NV Editor operations, namely just selecting a different standard SNVT type from the list of SNVT Types. When a standard LonMark SNVT type is selected, most of the configuration parameters will be fixed by LonMark and set to read-only in the editor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



The following is an example of a standard SNVT selection:

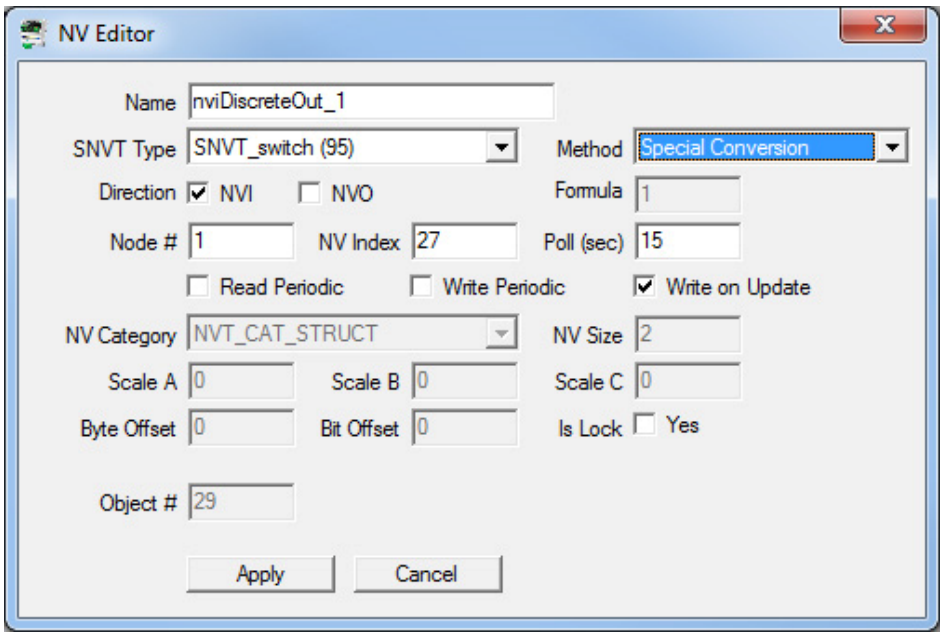


If you were to create a 'user defined' NV that provided the exact same results as SNVT_temp, the NV Editor dialog would appear as follows:



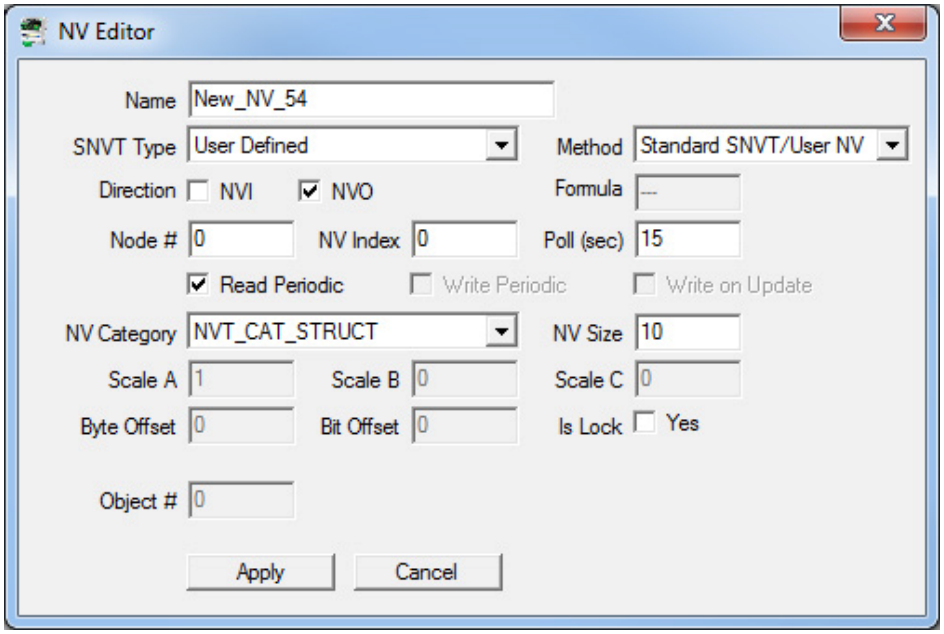
The SNVT_switch example is illustrated below. The LonMark definition of a "switch" actually contains two elements of data, a state and a level (as would be used for a dimmer switch). You cannot correctly control anything via a SNVT_switch without properly dealing with both elements of data. In Modbus 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. The special conversion is also extended to check specifically for Modbus coils, and convert off and on states to 0% and 100% levels for SNVT_switch respectively.

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 data objects.

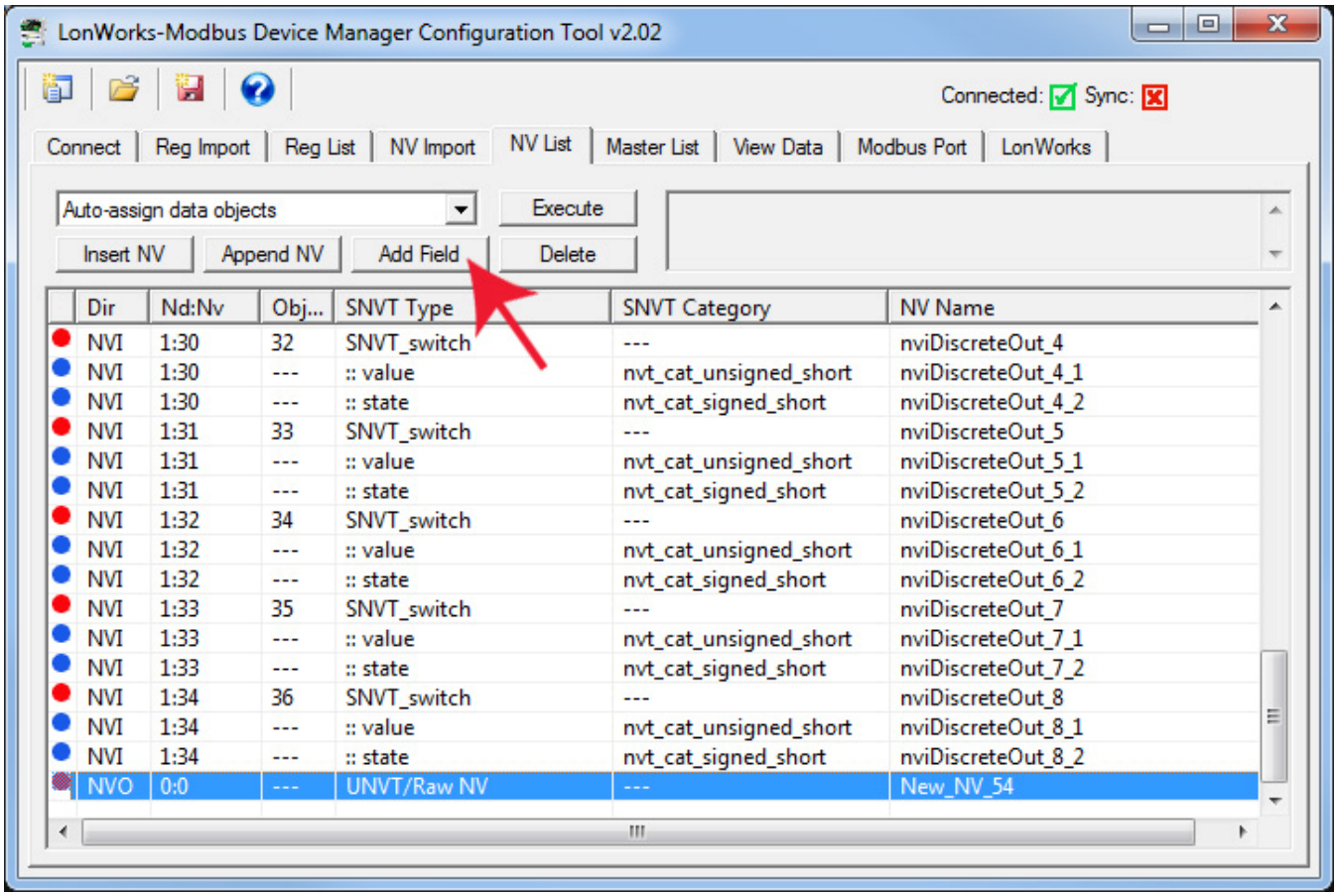


Creating a user defined structured network variable is illustrated below. The parent NV must have

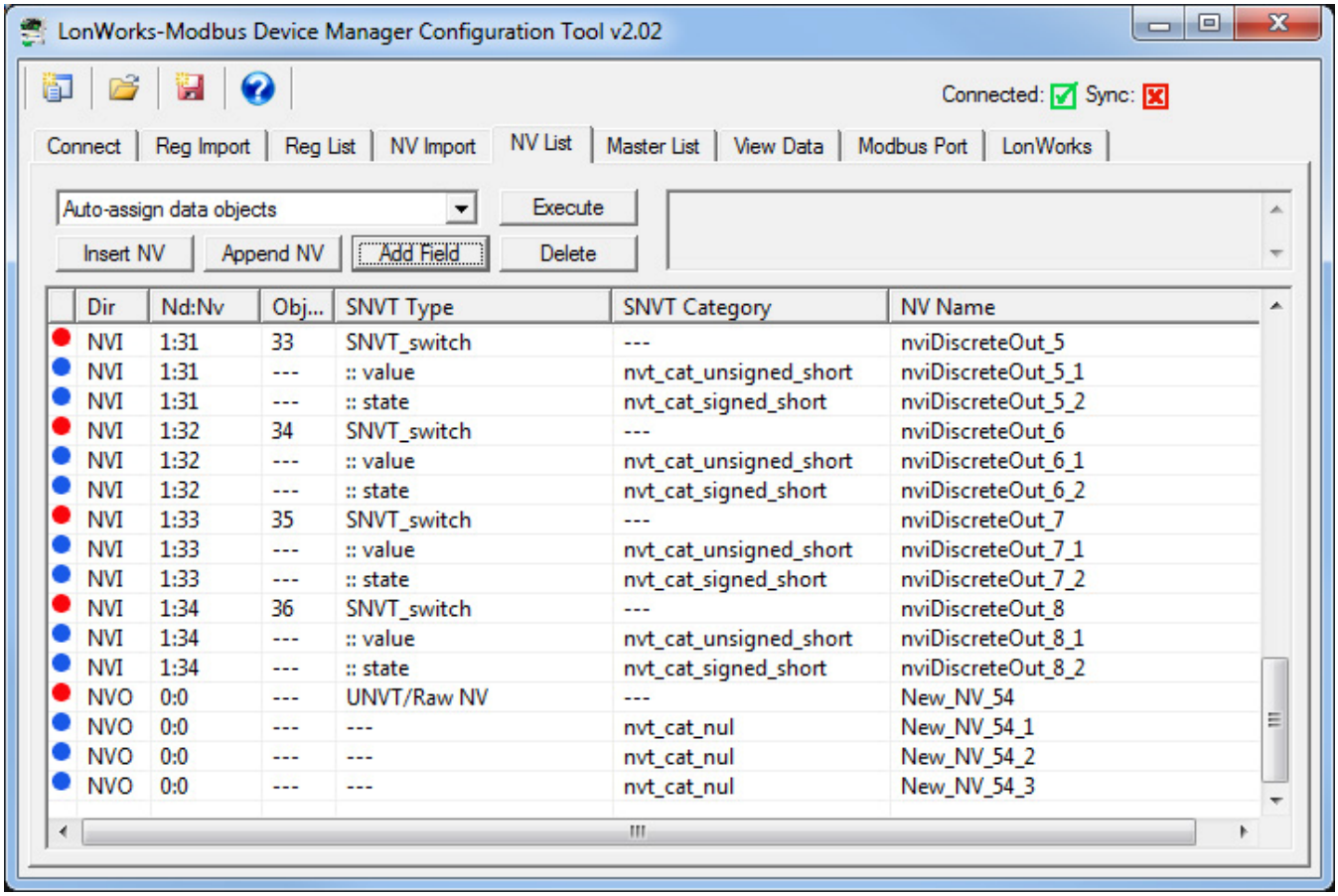
SNVT type set to User Defined. The NV Category must be NVT_CAT_STRUCT and the NV Size must be set to the total number of bytes of data that this NV will occupy, including all fields of the structure.



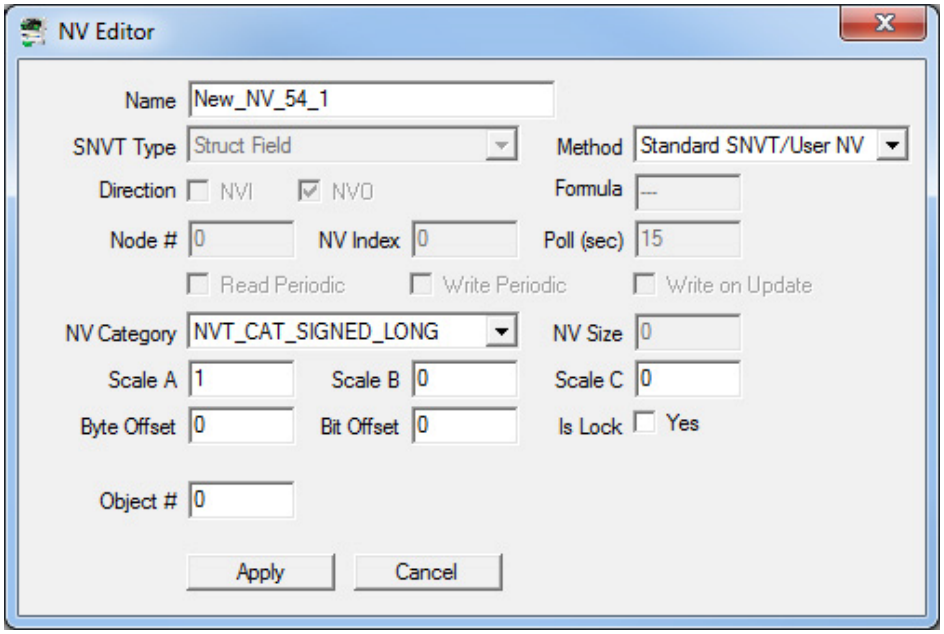
Once a user defined structured Network Variable is created, you need to add data fields to the structure by clicking Add Field (after selecting the UNVT by single clicking on it in the list).



The unconfigured, empty fields of the structure will look something like the following example.

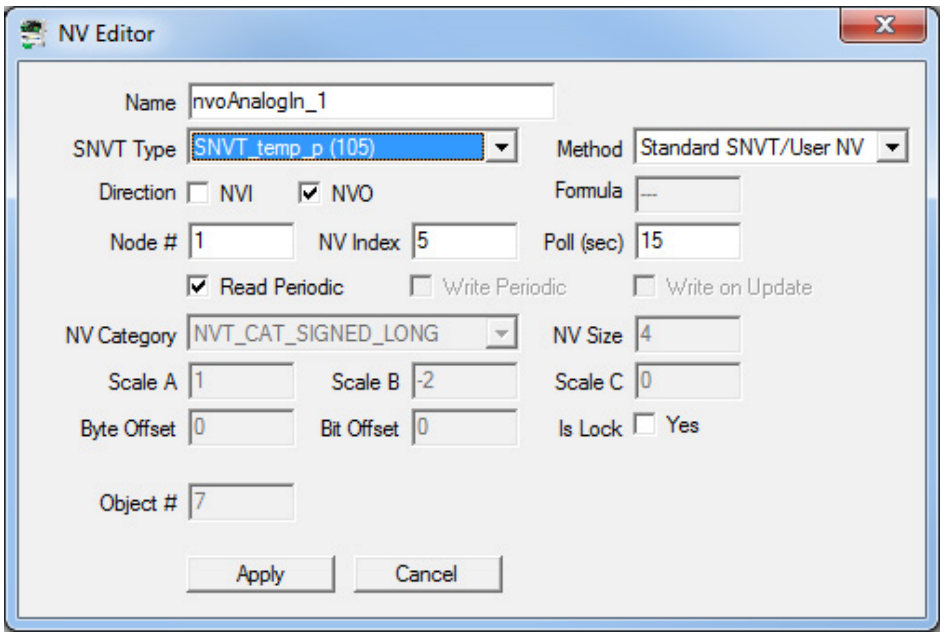


For each field of the structured NV, select a standard data type for NV Category. The first field will have a byte offset of zero. But all subsequent fields will need to have a non-zero offset so that the data is taken from the right place in the structure. When the network variable (NV) is read from 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 initally 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 millisconds is converted to a single floating point value of seconds.



NV Editor fields are used as follows:

Name	NV name that will be saved in the gateway device just for reference. The name is limited to 20 characters.
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.
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.
Node #	Indicates which node entry in the node table will be polled for this NV mapping.
NV Index	Specifies the NV index in the remote LonWorks device that will be polled. This is usually taken from the XIF file, otherwise must be obtained from the device manufacturer's documentation. This is effectively the 'address' of the variable within the device, and cannot be arbitrary.
Poll (sec)	Polling period in seconds indicates how frequently the gateway will read or periodically write a Network Variable in the remote LonWorks device.
Read/Write	The choice for read/write must match up with the direction. You can read an NVI or NVO. However, you can only write to an NVI in the remote device. Writing can be periodic at the poll rate, or 'on update' meaning it will only be written to when the Modbus register mapped to this NV is written to.
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	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. The scaling formula is S = a*10^b*(R+c) where R is raw data, and S is scaled data.
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.
	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

Is Lock	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.
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.

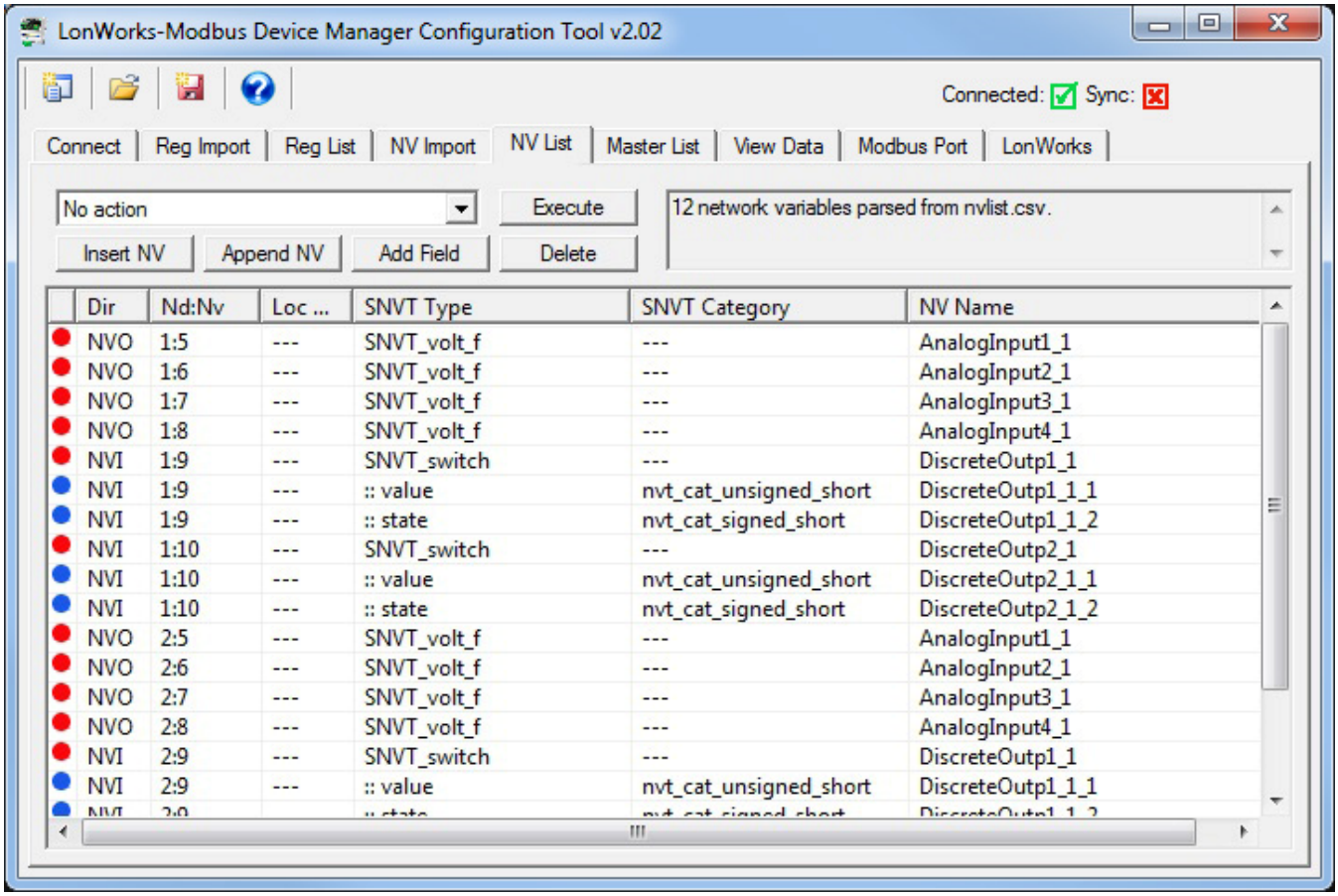
8.4 Using NV CSV Files to Build Multi-Device Configuration

The ability to save and re-load NV lists from a CSV file has been added to this configuration tool. If you have used the XIF import from device, it becomes more manageable to import the NV list, clear the gateway, then import more devices. Save each individual device's NV list to a CSV file. Then use a spread sheet program to combine and edit the lists.

The format expected for the NV CSV file is given in Appendix F.

```
NODE,NVINDEX,DIR,SERVICE,SNVT,UNVTSIZE,NAME
1,5,0,R,66,0,AnalogInput1_1
1,6,0,R,66,0,AnalogInput2_1
1,7,0,R,66,0,AnalogInput3_1
1,8,0,R,66,0,AnalogInput4_1
1,9,1,W+,95,0,DiscreteOutp1_1
1,10,1,W+,95,0,DiscreteOutp2_1
2,5,0,R,66,0,AnalogInput1_1
2,6,0,R,66,0,AnalogInput2_1
2,7,0,R,66,0,AnalogInput3_1
2,8,0,R,66,0,AnalogInput4_1
2,9,1,W+,95,0,DiscreteOutp1_1
2,10,1,W+,95,0,DiscreteOutp2_1
```

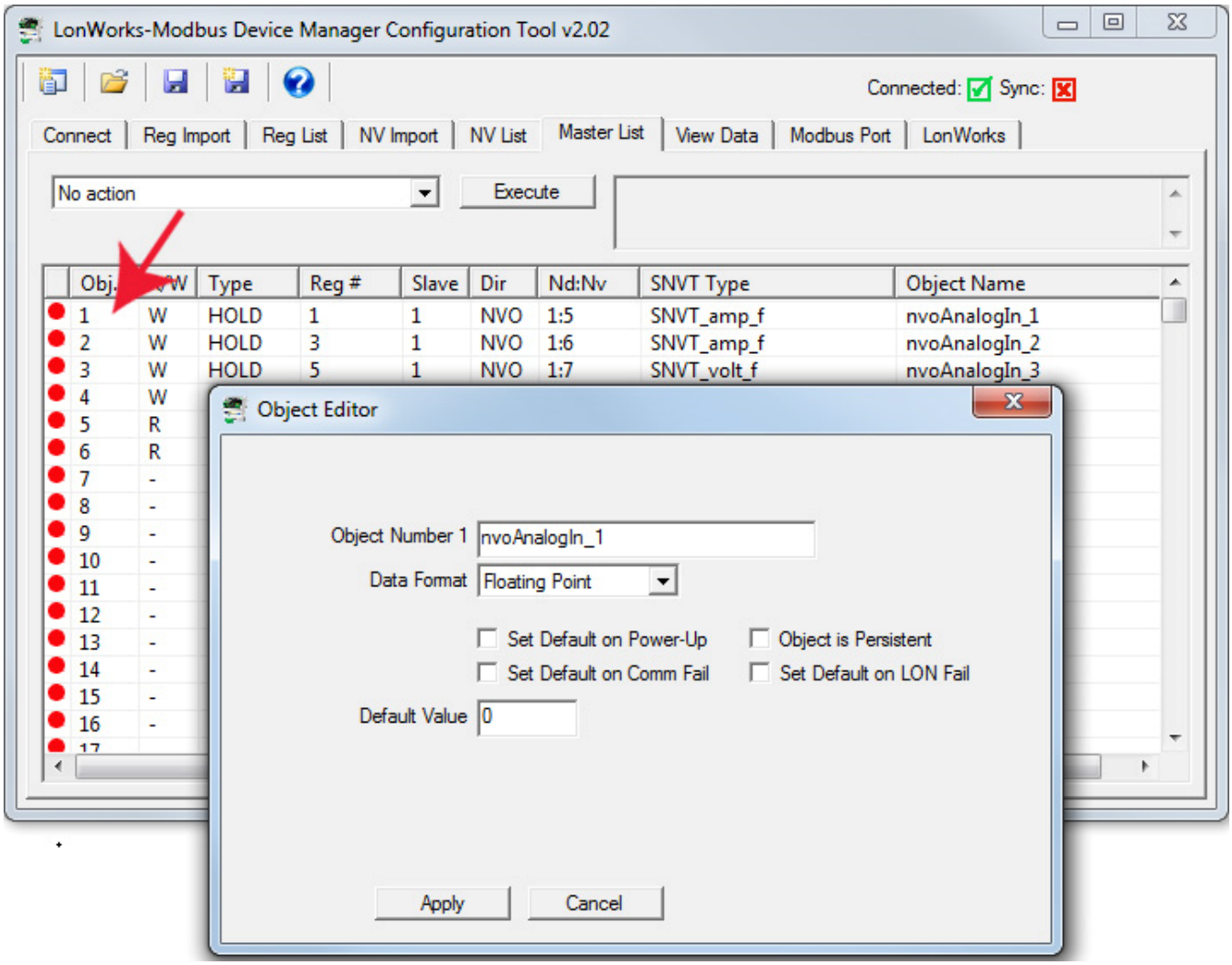
Importing this CSV, reflecting network variables in two different LonWorks nodes, results in the following screen shot.


LonWorks-Modbus Device Manager Configuration Tool v2.02
Connected:
Sync:
Connect
Reg Import
Reg List
NV Import
NV List
Master List
View Data
Modbus Port
LonWorks
No action
Execute
12 network variables parsed from nvlist.csv.
Insert NV
Append NV
Add Field
Delete
Dir Nd:Nv Loc ... SNVT Type SNVT Category NV Name
NVO 1:5 --- SNVT_volt_f --- AnalogInput1_1
NVO 1:6 --- SNVT_volt_f --- AnalogInput2_1
NVO 1:7 --- SNVT_volt_f --- AnalogInput3_1
NVO 1:8 --- SNVT_volt_f --- AnalogInput4_1
NVI 1:9 --- SNVT_switch --- DiscreteOutp1_1
NVI 1:9 --- :: value nvt_cat_unsigned_short DiscreteOutp1_1_1
NVI 1:9 --- :: state nvt_cat_signed_short DiscreteOutp1_1_2
NVI 1:10 --- SNVT_switch --- DiscreteOutp2_1
NVI 1:10 --- :: value nvt_cat_unsigned_short DiscreteOutp2_1_1
NVI 1:10 --- :: state nvt_cat_signed_short DiscreteOutp2_1_2
NVO 2:5 --- SNVT_volt_f --- AnalogInput1_1
NVO 2:6 --- SNVT_volt_f --- AnalogInput2_1
NVO 2:7 --- SNVT_volt_f --- AnalogInput3_1
NVO 2:8 --- SNVT_volt_f --- AnalogInput4_1
NVI 2:9 --- SNVT_switch --- DiscreteOutp1_1
NVI 2:9 --- :: value nvt_cat_unsigned_short DiscreteOutp1_1_1

9 Tool 'Master List' Page

9.1 Editing Configuration from Master List Page

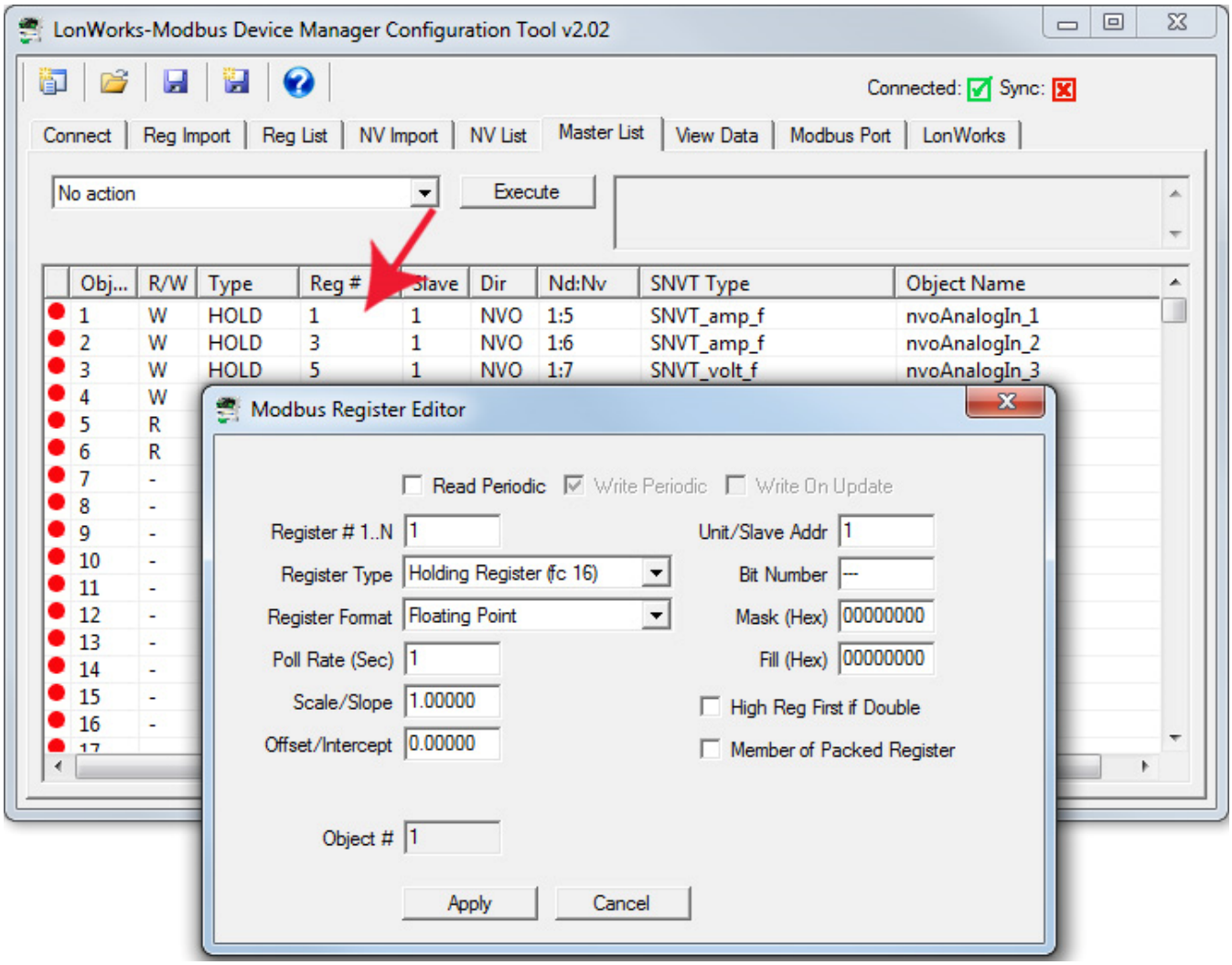
Clicking on the Object number column will open the Object Editor dialog as illustrated below.



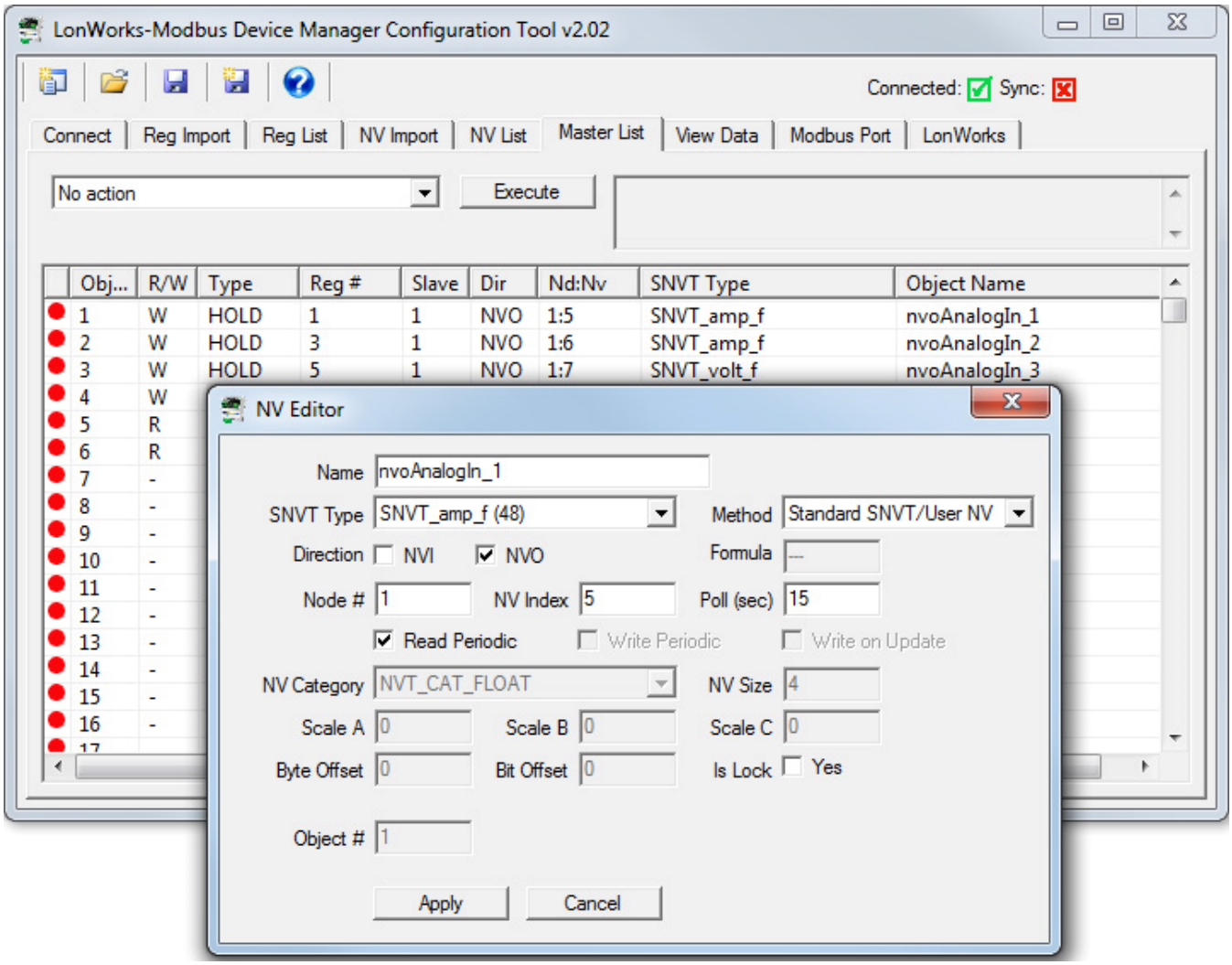
Object Number/Name	The object number you clicked on will be displayed, along with a window where you may enter an object name. This name is primarily for documentation purposes since Modbus cannot read names, and LonWorks can only use the Network Variable name.
Data Format	The internal data format of the object is selected here. If the object as 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.



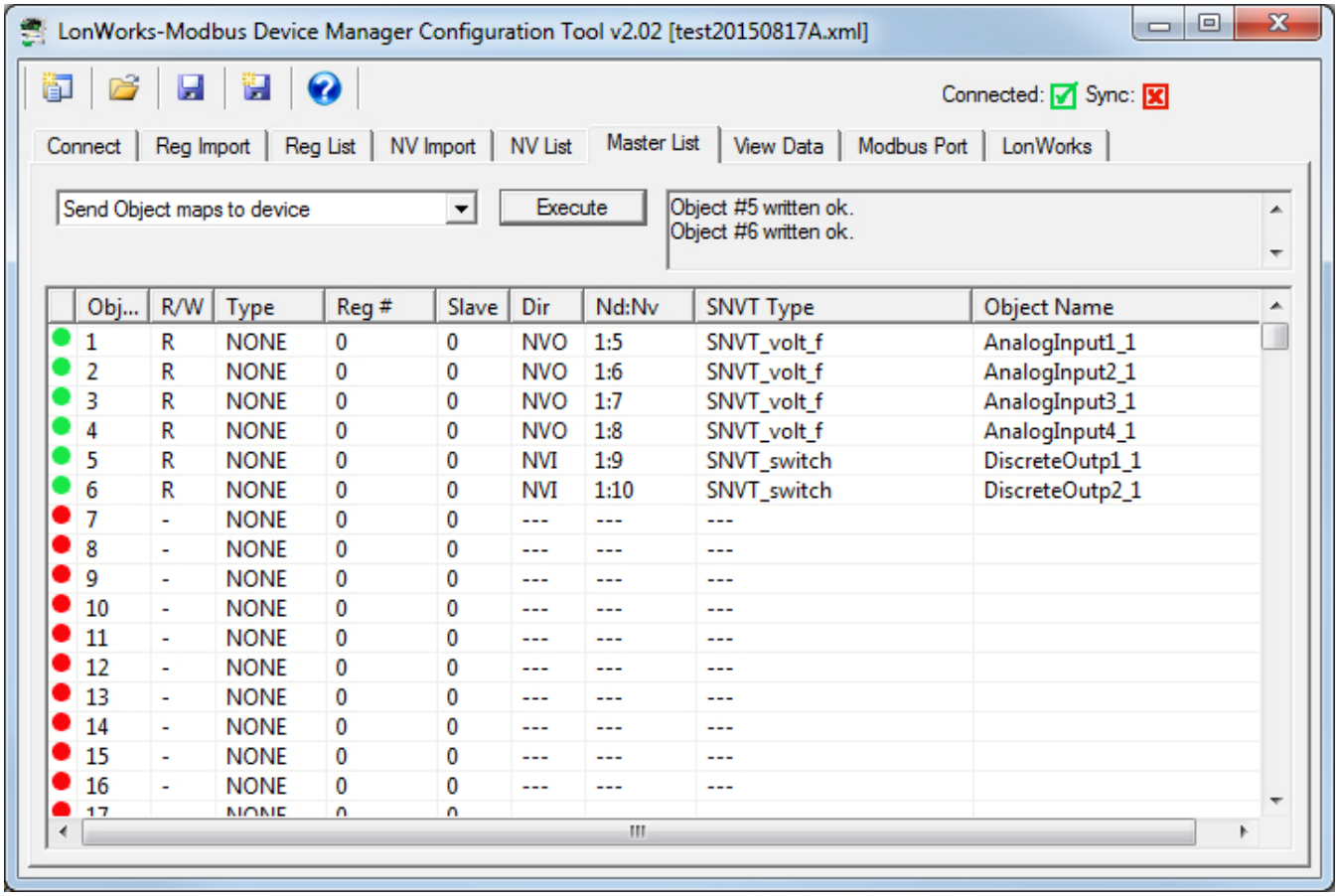
Clicking on the Network Variable area of a line on the Master List will open the NV Editor dialog as illustrated below. This is the same editor described in Section 8 of this user guide. Refer to that section for details about the NV Editor.



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, and generally in this order:

- Modbus Port: Set baud rate or IP address, etc., as applicable
- LonWorks: Set Location, send node list (see LonWork section about populating node table)
- NV List (or Master List): Send NV definitions to device
- Master List: Send Object maps to device (includes register mapping from Reg List)



Obj...	R/W	Type	Reg #	Slave	Dir	Nd:Nv	SNVT Type	Object Name
1	R	NONE	0	0	NVO	1:5	SNVT_volt_f	AnalogInput1_1
2	R	NONE	0	0	NVO	1:6	SNVT_volt_f	AnalogInput2_1
3	R	NONE	0	0	NVO	1:7	SNVT_volt_f	AnalogInput3_1
4	R	NONE	0	0	NVO	1:8	SNVT_volt_f	AnalogInput4_1
5	R	NONE	0	0	NVI	1:9	SNVT_switch	DiscreteOutp1_1
6	R	NONE	0	0	NVI	1:10	SNVT_switch	DiscreteOutp2_1
7	-	NONE	0	0	---	---	---	
8	-	NONE	0	0	---	---	---	
9	-	NONE	0	0	---	---	---	
10	-	NONE	0	0	---	---	---	
11	-	NONE	0	0	---	---	---	
12	-	NONE	0	0	---	---	---	
13	-	NONE	0	0	---	---	---	
14	-	NONE	0	0	---	---	---	
15	-	NONE	0	0	---	---	---	
16	-	NONE	0	0	---	---	---	

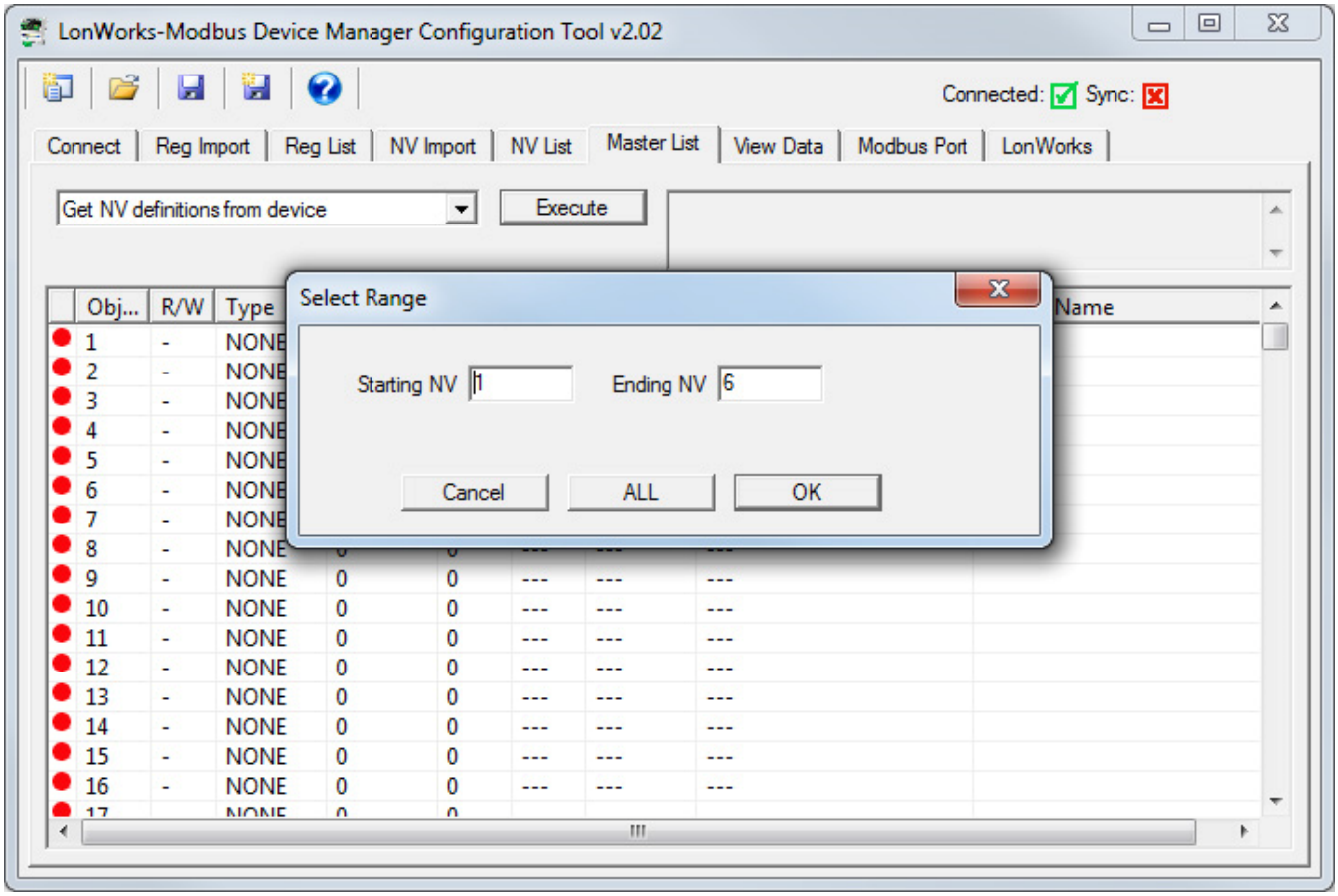
9.3 Getting Configuration From Device

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.

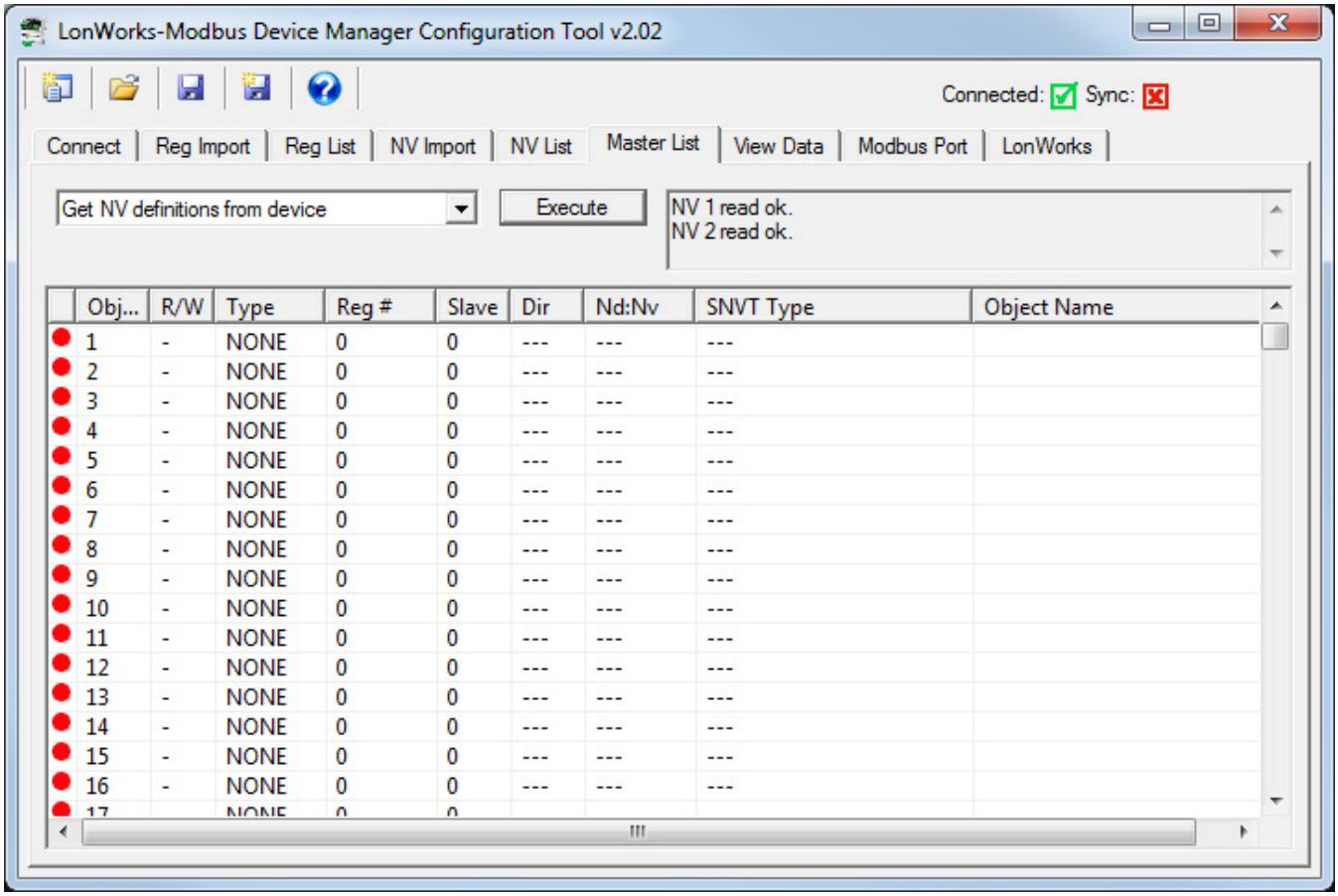
There are a total of four pages containing information that must be read from the gateway to obtain the complete configuration.

- NV List (or Master List): Get NV definitions from device
- Master List: Get Object maps from device (should Get Counts first)
- Modbus Port: Get port info and TCP mappings if applicable
- LonWorks: Get location and node list

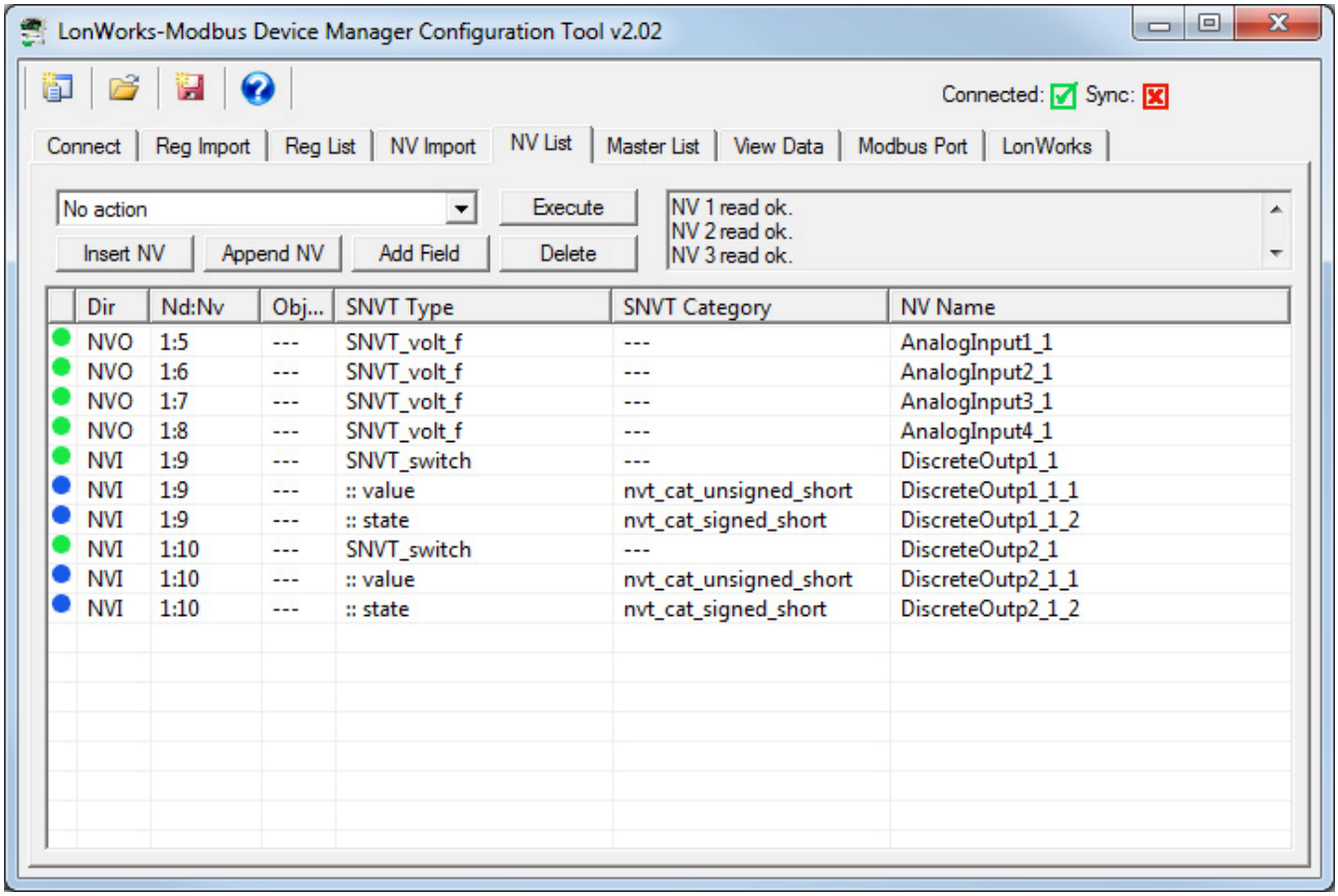
Upon clicking Execute with "Get NV definitions from device" selected, the tool will query the gateway to see how many NVs are actually in use. The range will appear in the dialog, allowing you to get only those NVs that are in use rather than all 300 of them.



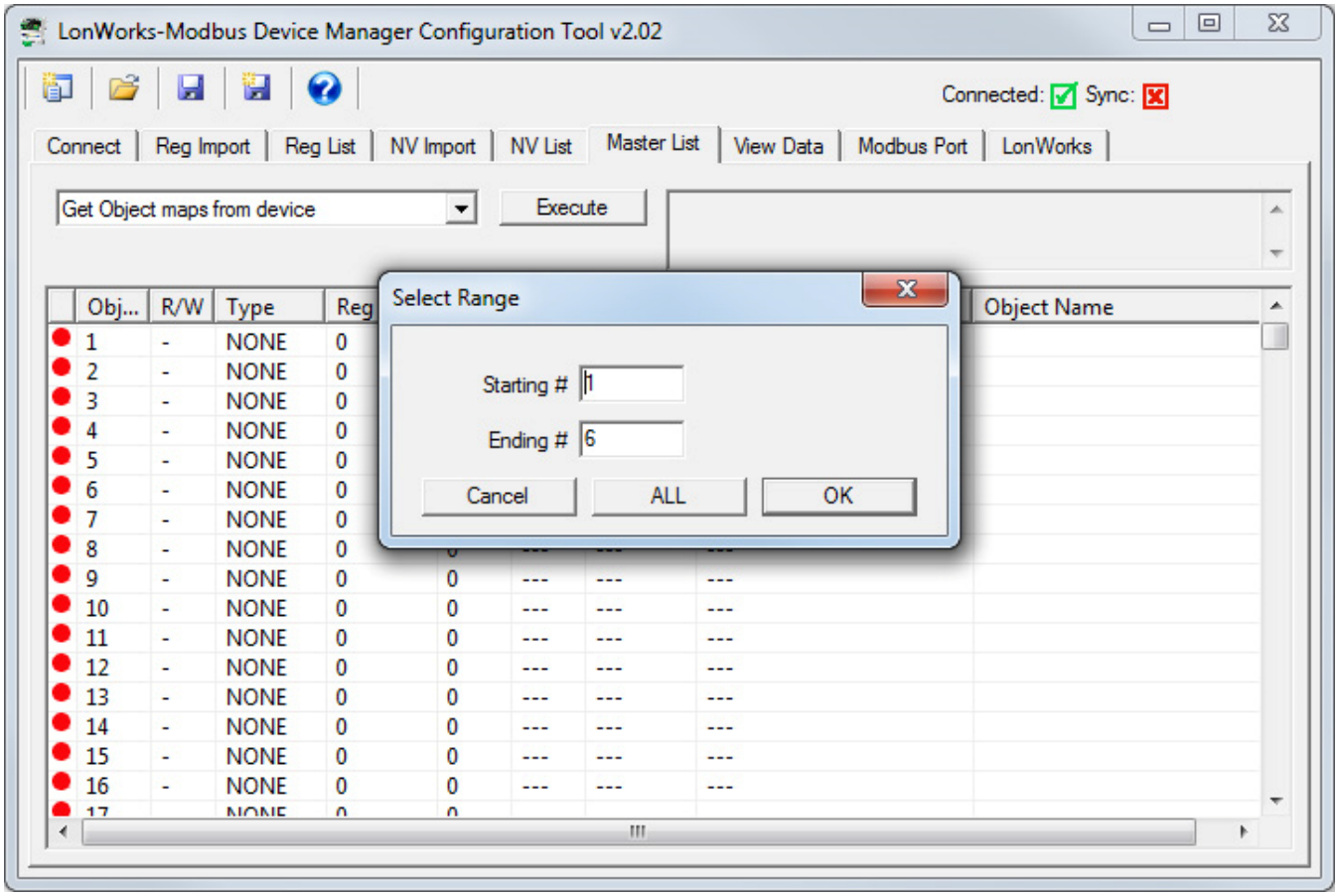
Upon completion of reading the list, the NVs will not show up in the Master List, but will show up in the NV List. You need to also read the objects before they will show up in the Master List.



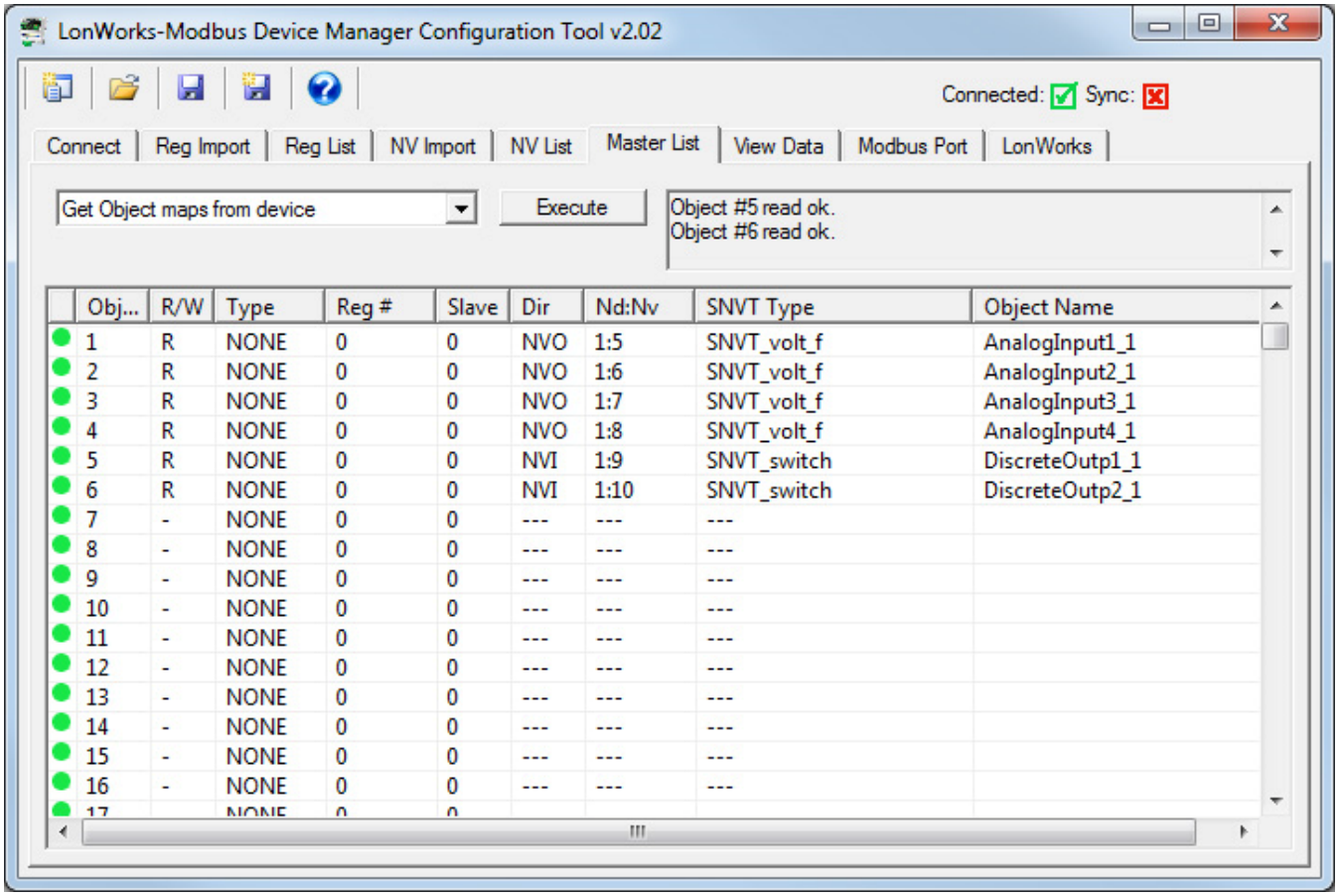
The NV List will appear something like the screen below following the Get NVs. Note that no local objects are yet known.



Upon clicking Execute with "Get Object maps from device" selected, the tool will quiery the gateway to see how many objects of each type are actually in use. The range will appear in the dialog, allowing you to get only those objects that are in use rather than all 400 of them.



Upon completion of reading the list, the Master List will now be fully populated.



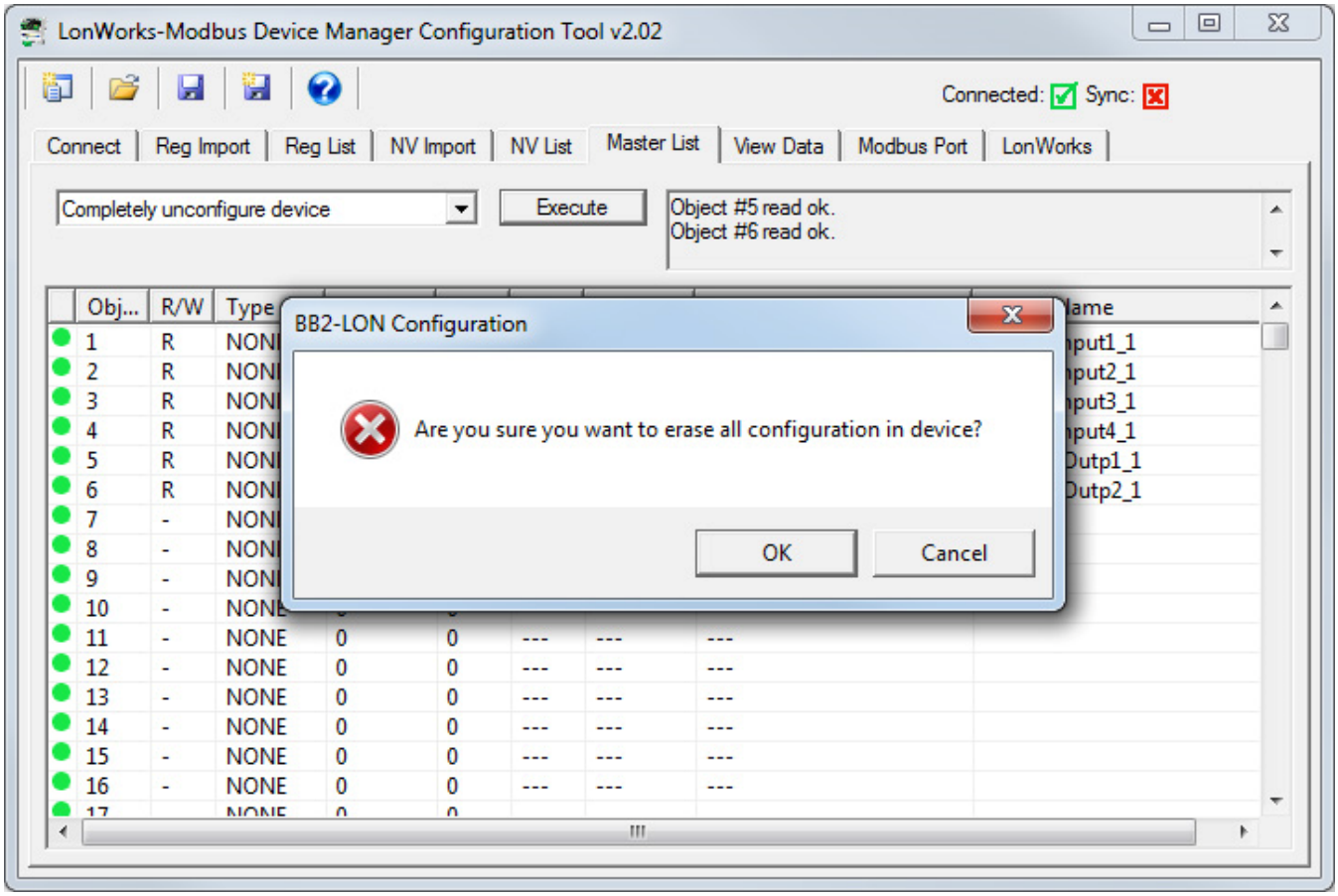
Obj...	R/W	Type	Reg #	Slave	Dir	Nd:Nv	SNVT Type	Object Name
1	R	NONE	0	0	NVO	1:5	SNVT_volt_f	AnalogInput1_1
2	R	NONE	0	0	NVO	1:6	SNVT_volt_f	AnalogInput2_1
3	R	NONE	0	0	NVO	1:7	SNVT_volt_f	AnalogInput3_1
4	R	NONE	0	0	NVO	1:8	SNVT_volt_f	AnalogInput4_1
5	R	NONE	0	0	NVI	1:9	SNVT_switch	DiscreteOutp1_1
6	R	NONE	0	0	NVI	1:10	SNVT_switch	DiscreteOutp2_1
7	-	NONE	0	0	---	---	---	
8	-	NONE	0	0	---	---	---	
9	-	NONE	0	0	---	---	---	
10	-	NONE	0	0	---	---	---	
11	-	NONE	0	0	---	---	---	
12	-	NONE	0	0	---	---	---	
13	-	NONE	0	0	---	---	---	
14	-	NONE	0	0	---	---	---	
15	-	NONE	0	0	---	---	---	
16	-	NONE	0	0	---	---	---	

9.4 Fixing Conflicts

You may run into the error message "Conflicting parameters" 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.

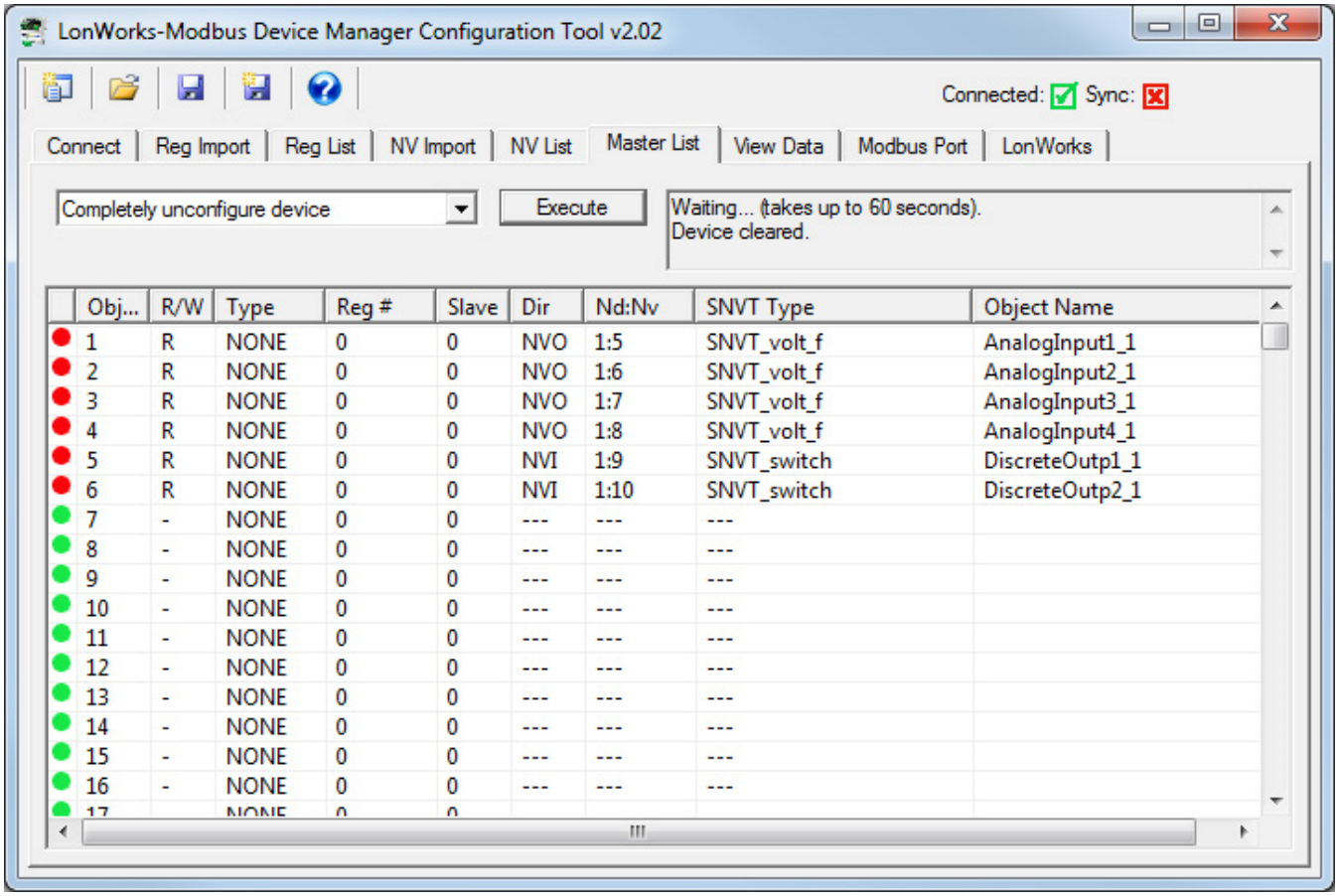
The easiest way to resolve configuration conflicts is to completely clear the device configuration and resend network variables, followed by object maps. When you execute "Completely unconfigure device", there will be a delay of typically 60 seconds while the device reprograms all of its internal non-volatile configuration memory, restoring it to an 'unused' state. Once complete, proceed with sending configuration to the device as noted above.

You will be prompted with a last chance to cancel the erasing of all configuration.



You will see the message "Device cleared" when finished. Note that the configuration held by the tool is not cleared by this operation. You can now re-send the configuration from the tool to the device. The NV List and Object List will be cleared in the device but not in the configuration tool. You will need to resend the NV List, Object List, and Node List (LonWorks page) to become fully operational.

If you want to also clear the tool, click the 'new' icon in the upper left corner of the toolbar.



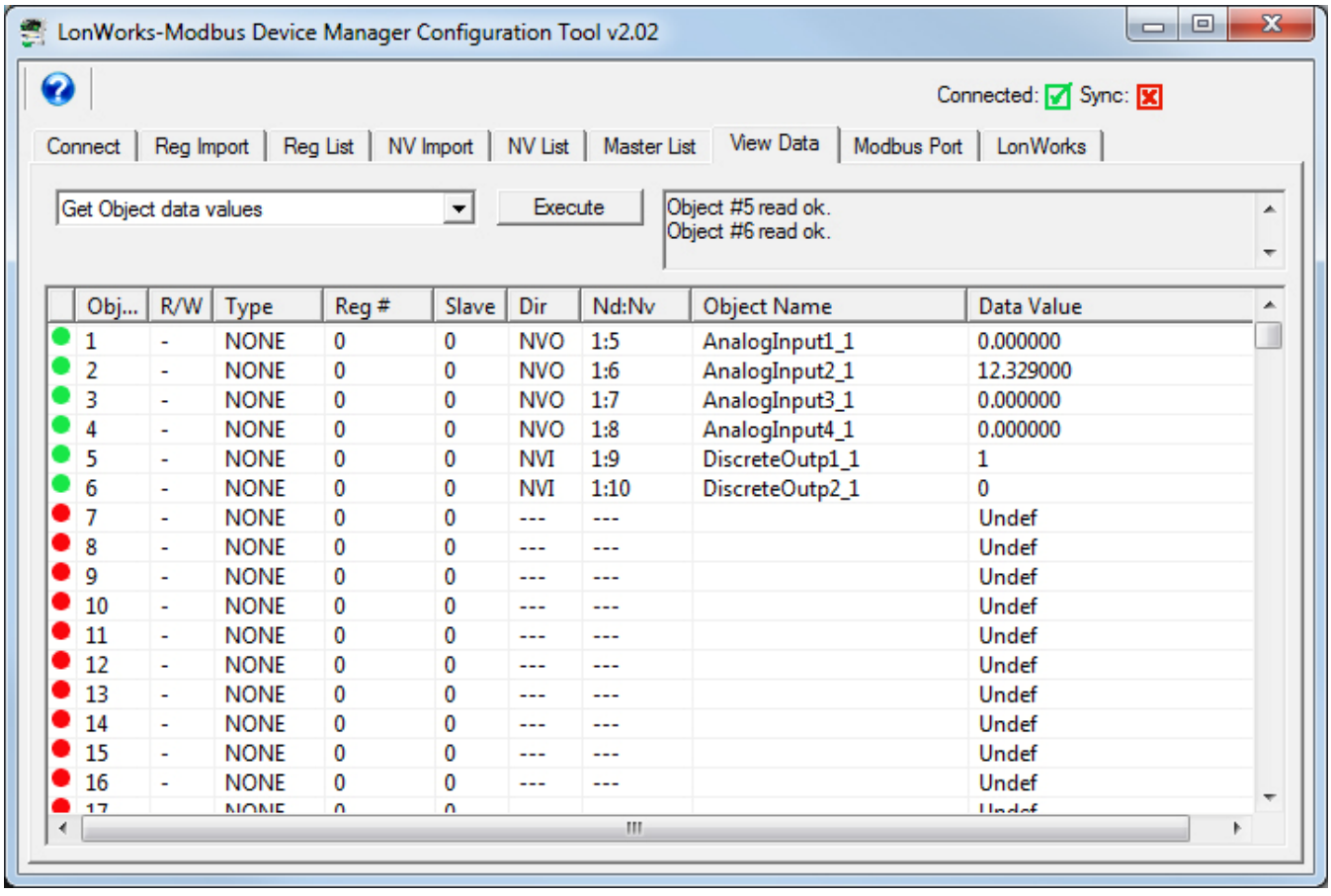
	Obj...	R/W	Type	Reg #	Slave	Dir	Nd:Nv	SNVT Type	Object Name
	1	R	NONE	0	0	NVO	1:5	SNVT_volt_f	AnalogInput1_1
	2	R	NONE	0	0	NVO	1:6	SNVT_volt_f	AnalogInput2_1
	3	R	NONE	0	0	NVO	1:7	SNVT_volt_f	AnalogInput3_1
	4	R	NONE	0	0	NVO	1:8	SNVT_volt_f	AnalogInput4_1
	5	R	NONE	0	0	NVI	1:9	SNVT_switch	DiscreteOutp1_1
	6	R	NONE	0	0	NVI	1:10	SNVT_switch	DiscreteOutp2_1
	7	-	NONE	0	0	---	---	---	
	8	-	NONE	0	0	---	---	---	
	9	-	NONE	0	0	---	---	---	
	10	-	NONE	0	0	---	---	---	
	11	-	NONE	0	0	---	---	---	
	12	-	NONE	0	0	---	---	---	
	13	-	NONE	0	0	---	---	---	
	14	-	NONE	0	0	---	---	---	
	15	-	NONE	0	0	---	---	---	
	16	-	NONE	0	0	---	---	---	
	17		NONE	0	0				

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.

To fully view data on the LonWorks network, you need a LonWorks network interface (such as Echelon's U10) to connect to the LonWorks network. You can then use Nodeutil to view NV data in raw form via the LonWorks network. The other option is that you can view NV data in raw form via the USB diagnostic console (Appendix A).

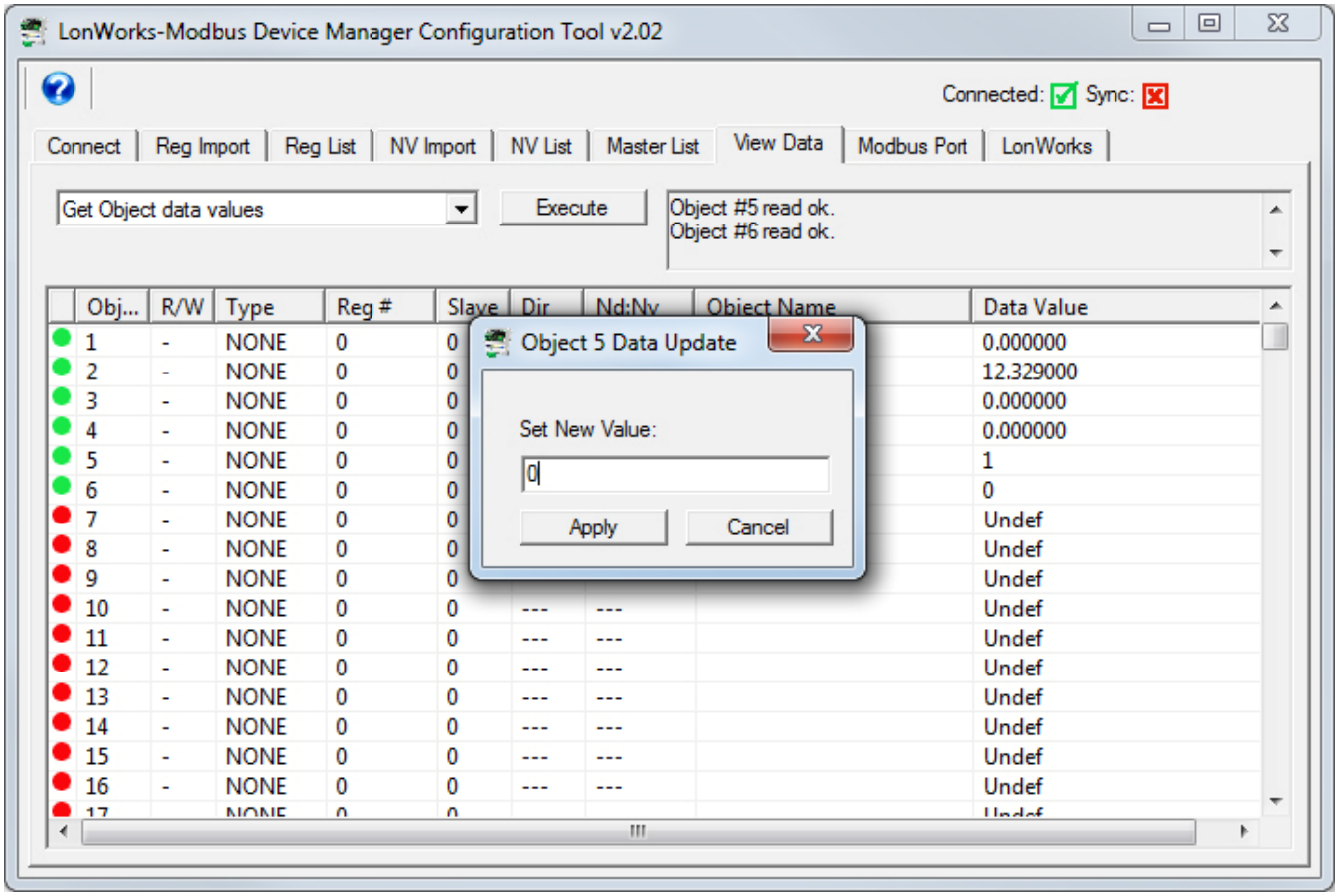


	Obj...	R/W	Type	Reg #	Slave	Dir	Nd:Nv	Object Name	Data Value
	1	-	NONE	0	0	NVO	1:5	AnalogInput1_1	0.000000
	2	-	NONE	0	0	NVO	1:6	AnalogInput2_1	12.329000
	3	-	NONE	0	0	NVO	1:7	AnalogInput3_1	0.000000
	4	-	NONE	0	0	NVO	1:8	AnalogInput4_1	0.000000
	5	-	NONE	0	0	NVI	1:9	DiscreteOutp1_1	1
	6	-	NONE	0	0	NVI	1:10	DiscreteOutp2_1	0
	7	-	NONE	0	0	---	---		Undef
	8	-	NONE	0	0	---	---		Undef
	9	-	NONE	0	0	---	---		Undef
	10	-	NONE	0	0	---	---		Undef
	11	-	NONE	0	0	---	---		Undef
	12	-	NONE	0	0	---	---		Undef
	13	-	NONE	0	0	---	---		Undef
	14	-	NONE	0	0	---	---		Undef
	15	-	NONE	0	0	---	---		Undef
	16	-	NONE	0	0	---	---		Undef

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.

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 register is read and placed into this object.

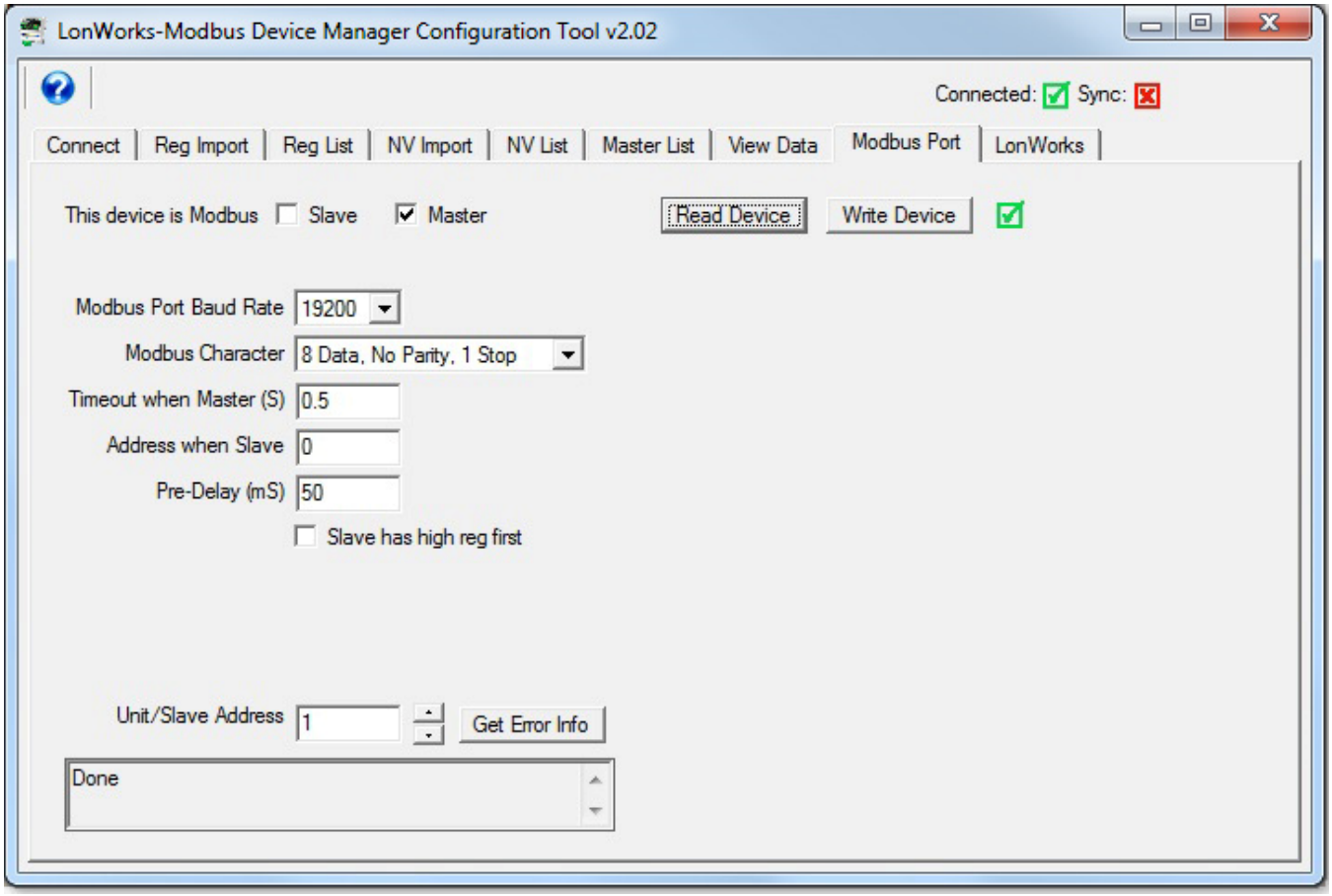

LonWorks-Modbus Device Manager Configuration Tool v2.02
Connected: Sync:
Connect
Reg Import
Reg List
NV Import
NV List
Master List
View Data
Modbus Port
LonWorks
Get Object data values
Execute
Object #5 read ok.
Object #6 read ok.
Obj...
R/W
Type
Reg #
Slave
Dir
Nd:Nv
Object Name
Data Value
Object 5 Data Update
Set New Value:
0
Apply
Cancel

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 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

Timeout	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	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. 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.

11.2 Modbus TCP Port Settings

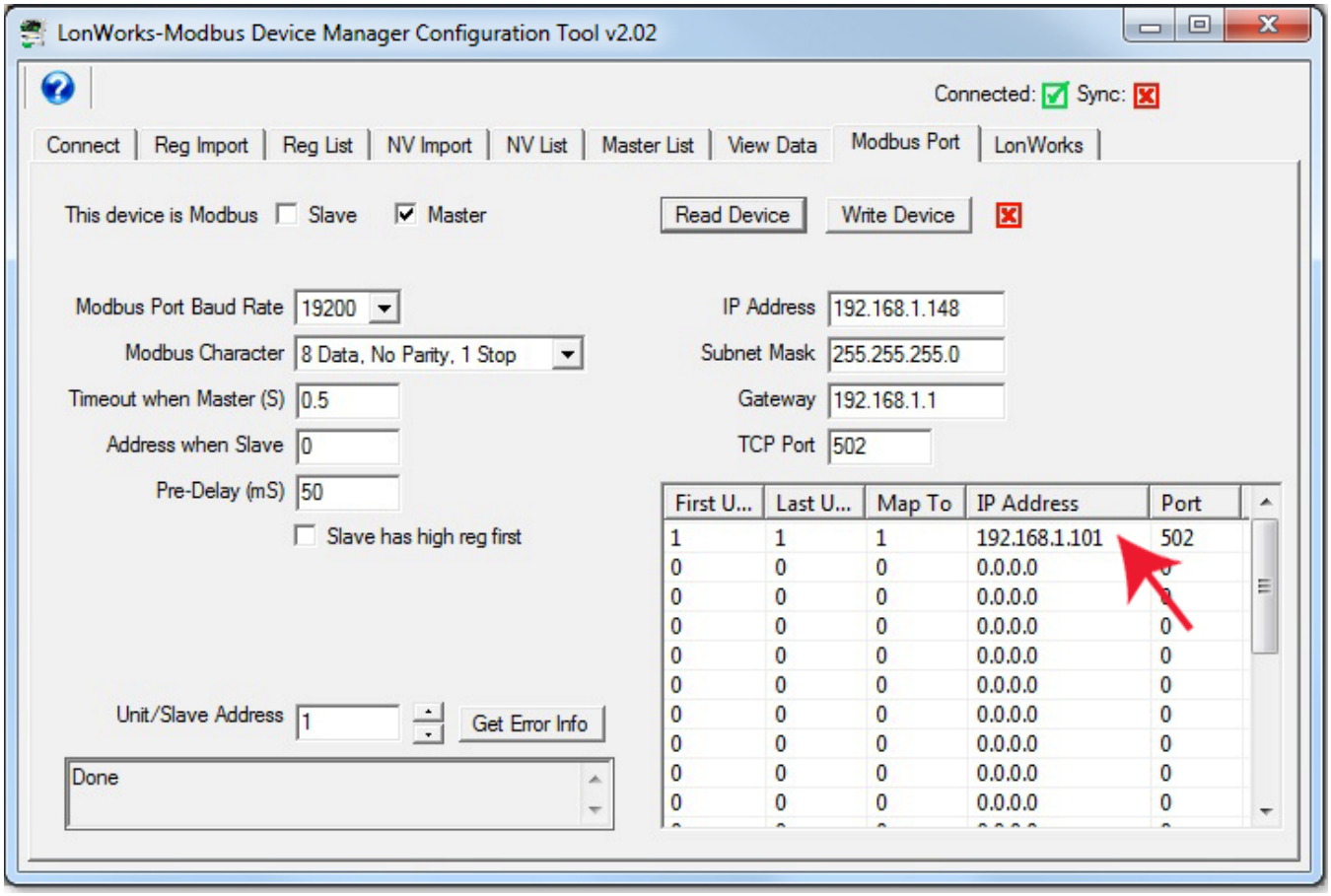
There only RTU settings that also apply to TCP are the Timeout when Master, and "Slave has high reg first" selection. Modbus TCP is both master and slave at the same time, so it is not necessary to select one or the other. Modbus TCP in the BB2-6020-NB gateway will respond to any and all unit numbers,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

Note that as a slave, the BB2-6020-NB 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.

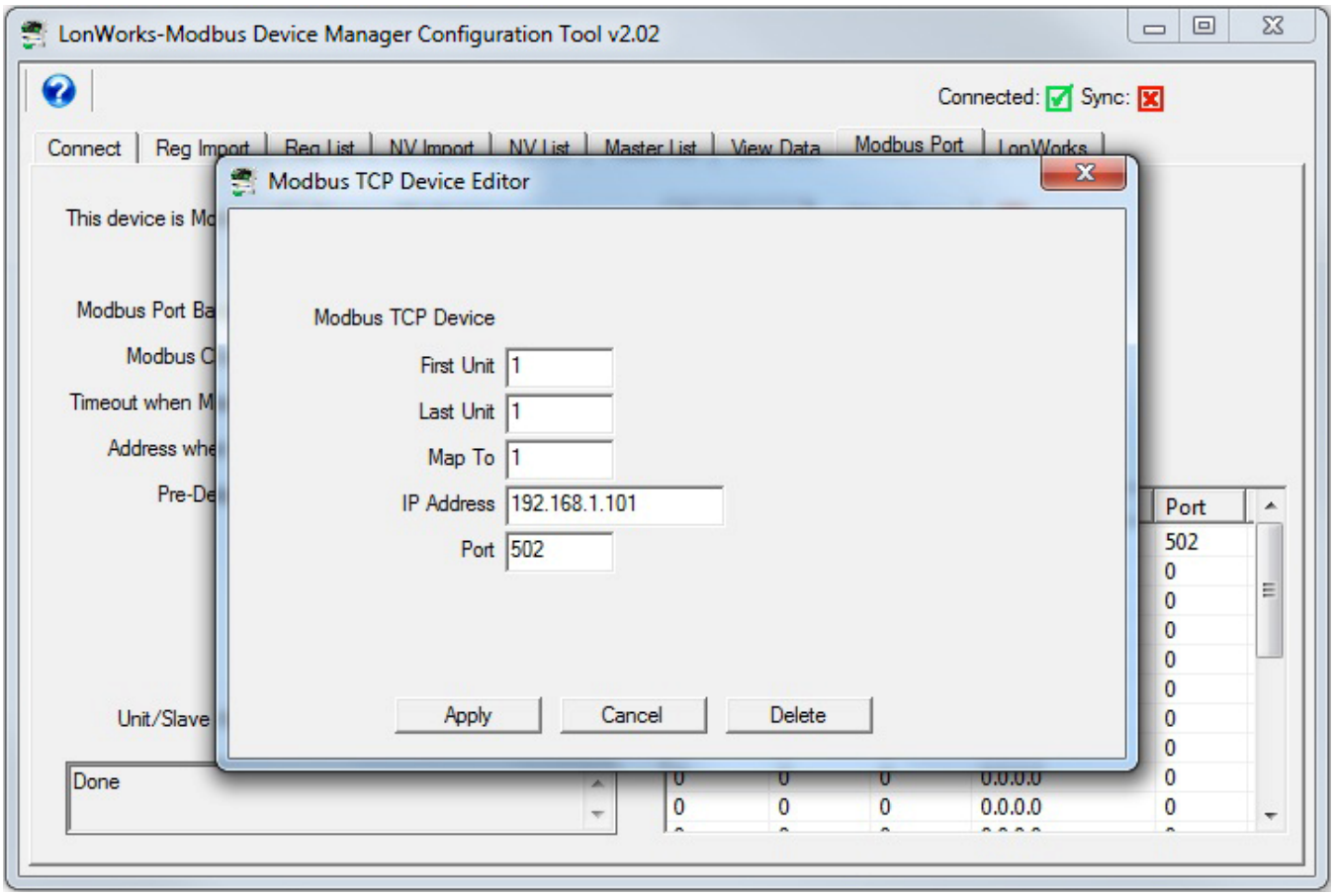
Configuring the BB2-6020-NB for Modbus TCP requires that you provide an IP address for the gateway itself, along with the applicable subnet mask as illustrated below. The default TCP port for Modbus TCP is 502.

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.



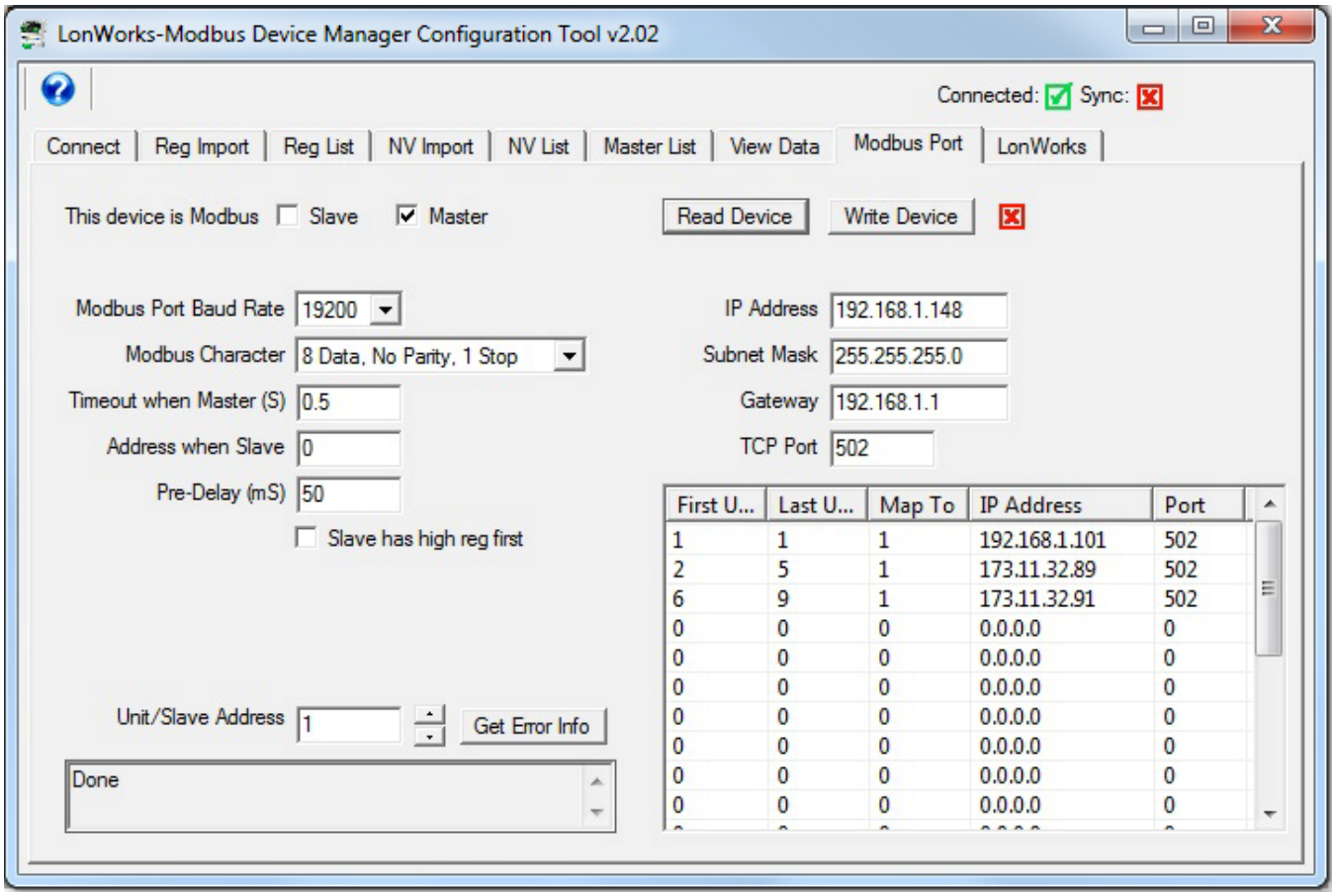
In addition to setting the IP address of the gateway itself, you need to provide an IP address of at least one Modbus TCP slave if the BB2-6020-NB will be functioning as a master. Double click on a line in the IP address list, and enter unit numbers and IP address as illustrated below. If you have only 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.



11.4 Get Error Info

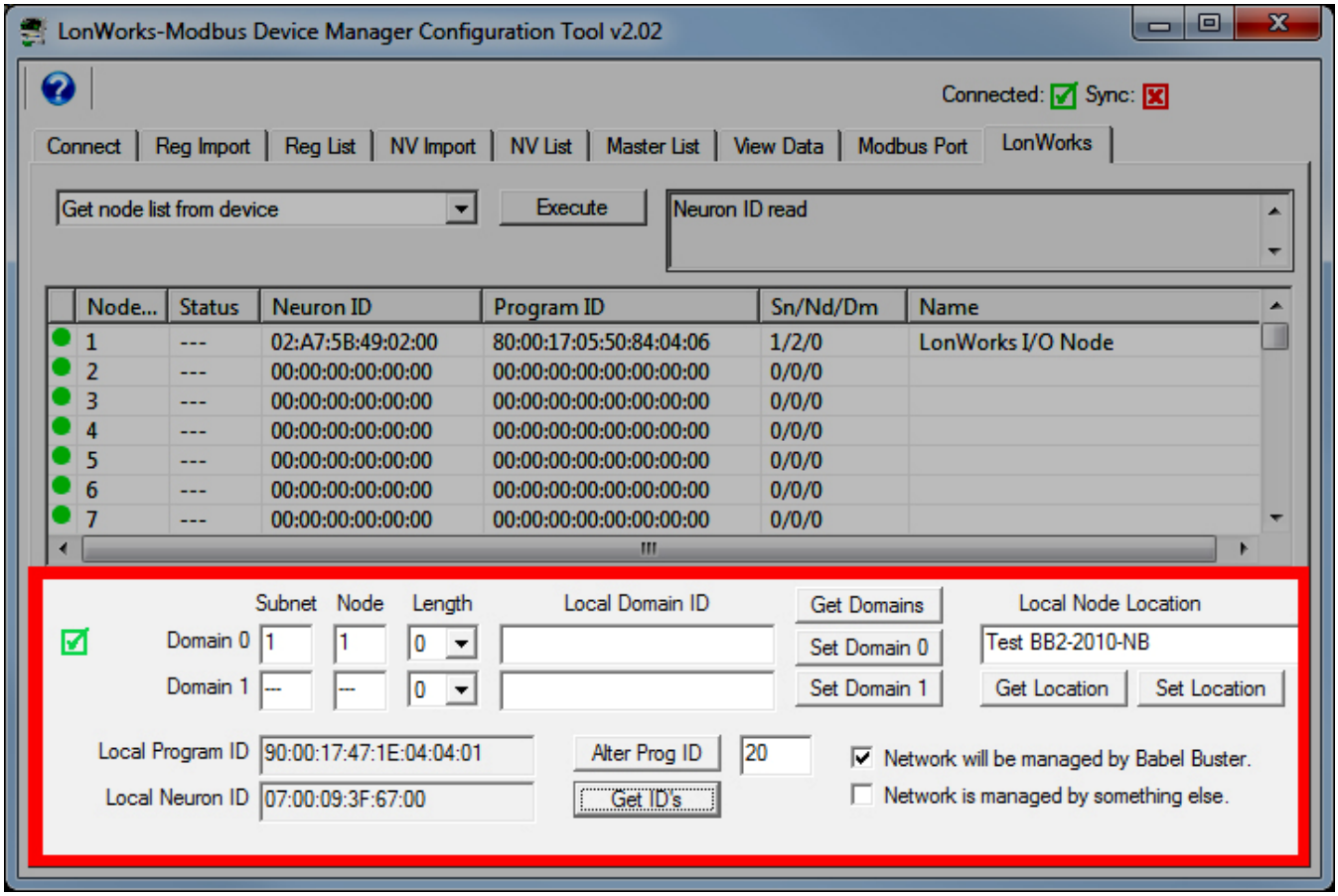
The Babel Buster gateway keeps error counts for each slave device. The counts increment for CRC and no-response (timeout) errors. The most recent exception code is retained for exceptions. Click Get Error Info to retrieve error information, which will be displayed in the log window just below the button. 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 LonWorks Identity of the Gateway

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:04:04:01 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-NB, BB2-2011-NB, or BB2-6020-NB.

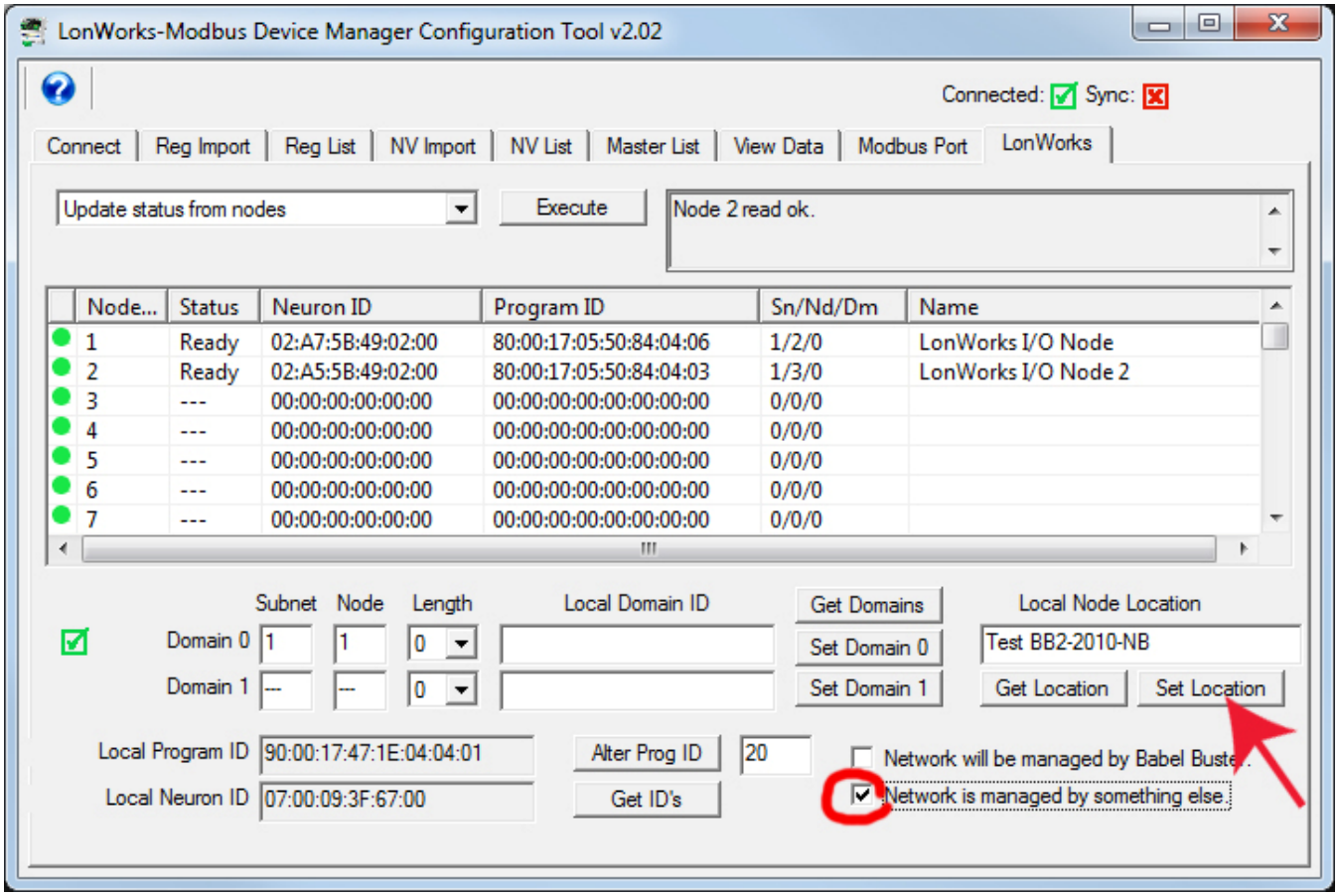
There is no restriction on what the Neuron ID might be. It is displayed here for reference just in case you are trying to correlate traffic in the LonScanner protocol analyzer.



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, if the gateway is being installed on a network where 'Network is managed by something else'.

You have the option of viewing current domain settings using the Get Domains button, or setting the domains with the respective Set Domain buttons. If the gateway is being used on a network where 'Network is managed by something else', you will need to find out via the network management tool involved what the domain should be, and set it accordingly. If 'Network will be managed by Babel Buster', then the default should work, unless you run into trouble as noted in Appendix B.3 or B.4.

IMPORTANT: If you change the state of "Network is/will be managed..." as illustrated below, you MUST click 'Set Location' to send this change of state to the Babel Buster gateway device. While you may not care about the location string, the state of network management is included in the packet of information sent with location.



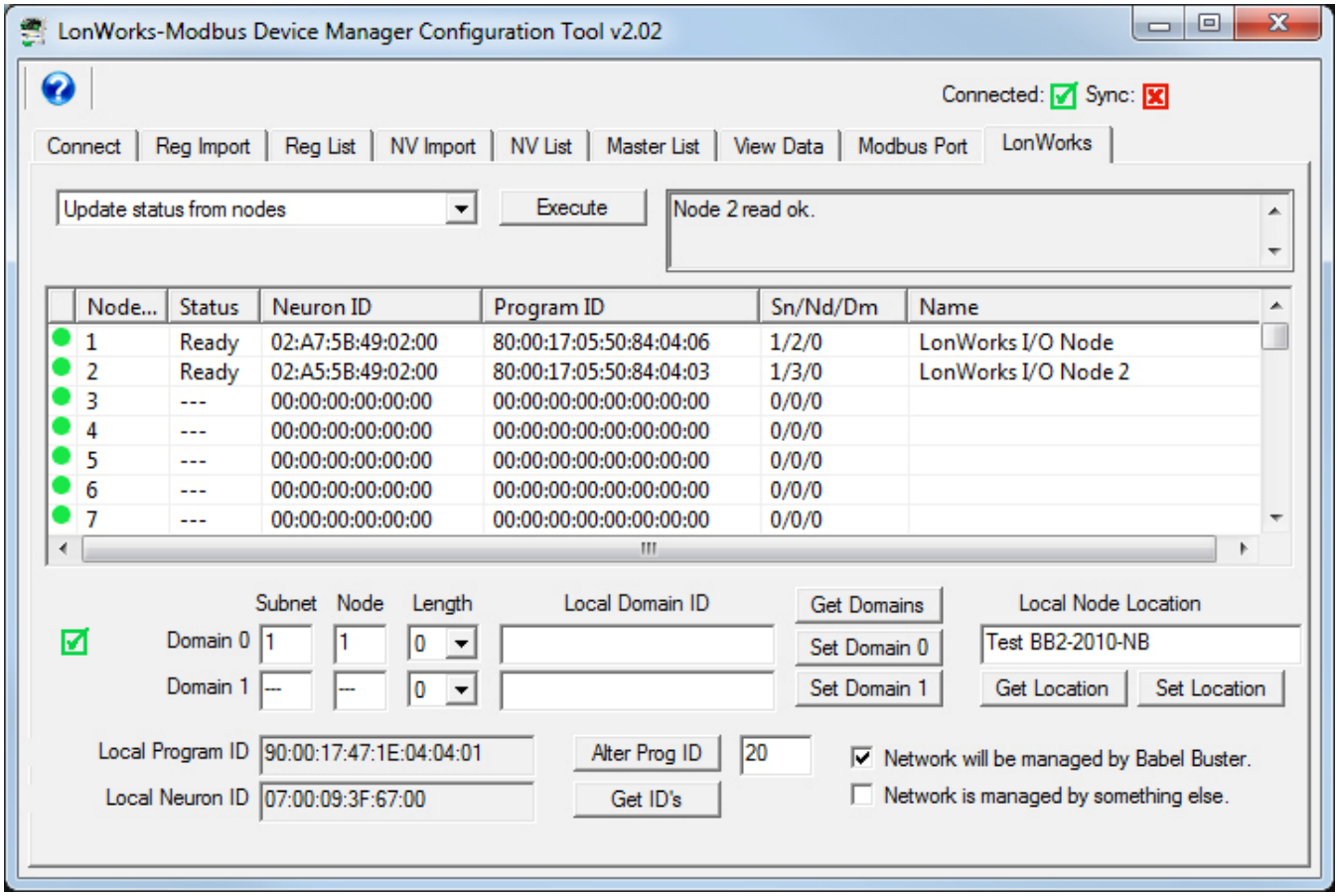
12.2 Changing Program ID of the Gateway

Changing program ID is only applicable if 'Network will be managed by something else' meaning Babel Buster is not in control of the network. You should normally have no reason to change the program ID. However, if you are pre-configuring multiple Babel Buster gateways and require them to have different program IDs, then you may use this option.

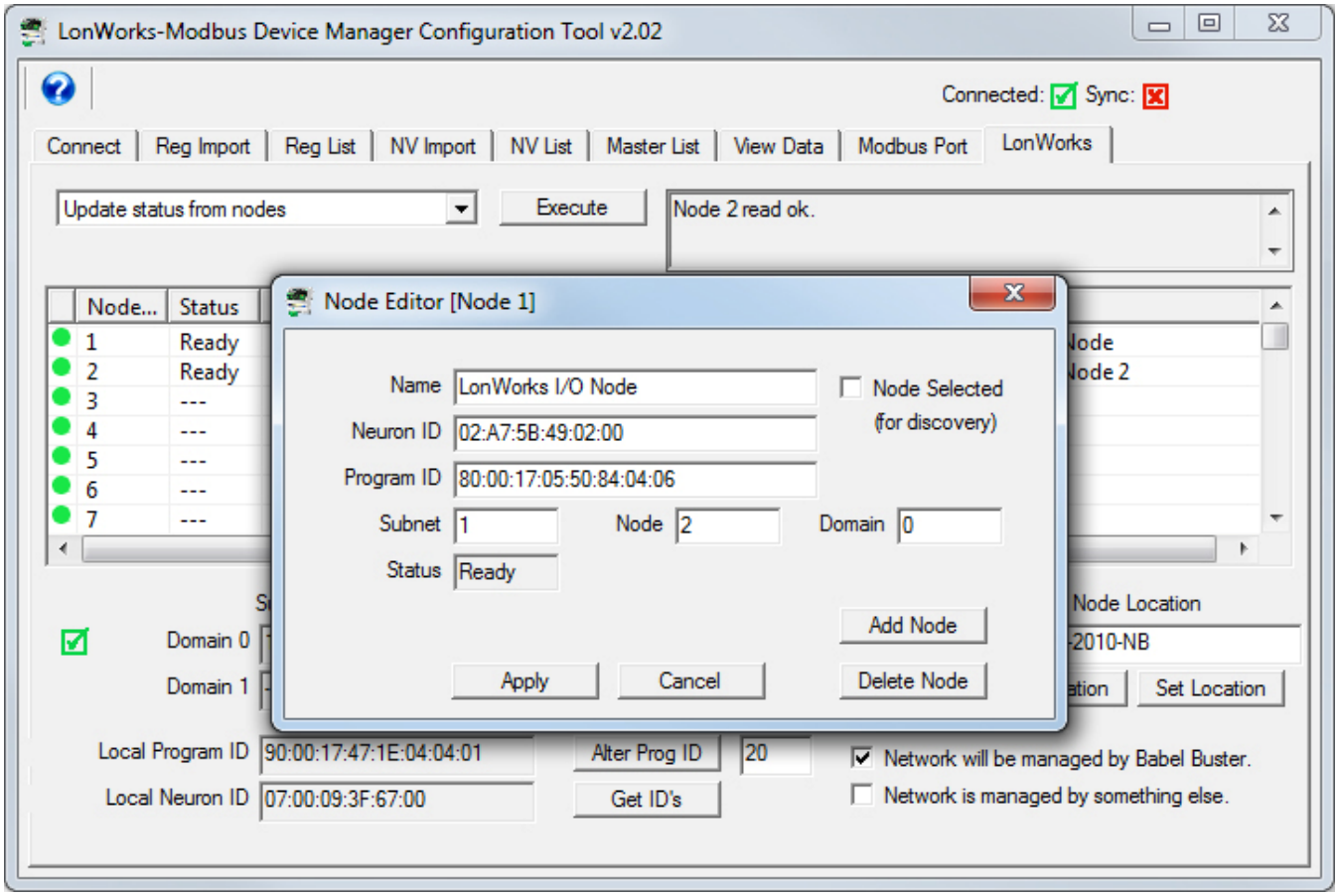
Note that ID numbers 1 through 20 are reserved for Control Solutions standard gateway ID numbers. User defined program ID numbers may be 21 through 255. Note that the input window accepts a decimal number even though the complete program ID is displayed as hexadecimal.

12.3 Viewing Identity and Status of Other LonWorks Devices

To view the identity and status of whatever LonWorks devices are known to the Babel Buster gateway, select 'Get node list from device', and click Execute. When it completes, then select 'Update status from nodes' and click Execute again. You will now see the list of known LonWorks devices and a status indicating whether or not they are communicating. If they are communicating normally, the status will be listed as "Ready". If not, then a code indicating where the gateway is at in its attempt will be displayed instead.



There is more than one way to intially populate the node list. One way is to manually enter a Neuron ID and let the gateway try to talk to it. There are a couple of more automates ways of letting the gateway just to find out who's out there, as discussed in following sections. In any event, you can edit the information for a given node by double-clicking that node in the list to get the Node Editor dialog to pop up. If you make changes, enter a new Neuron ID, or discover new nodes as discussed further on, you will need to send that node configuration to the gateway before the gateway will begin attempting to communicate with it.



Node status will hopefully indicate "Ready". But it may indicate other codes showing the gateway's progress in attempting to communicate. The most common "non-ready" indication will be "SFN". This means it is completely unable to reach that particular node.

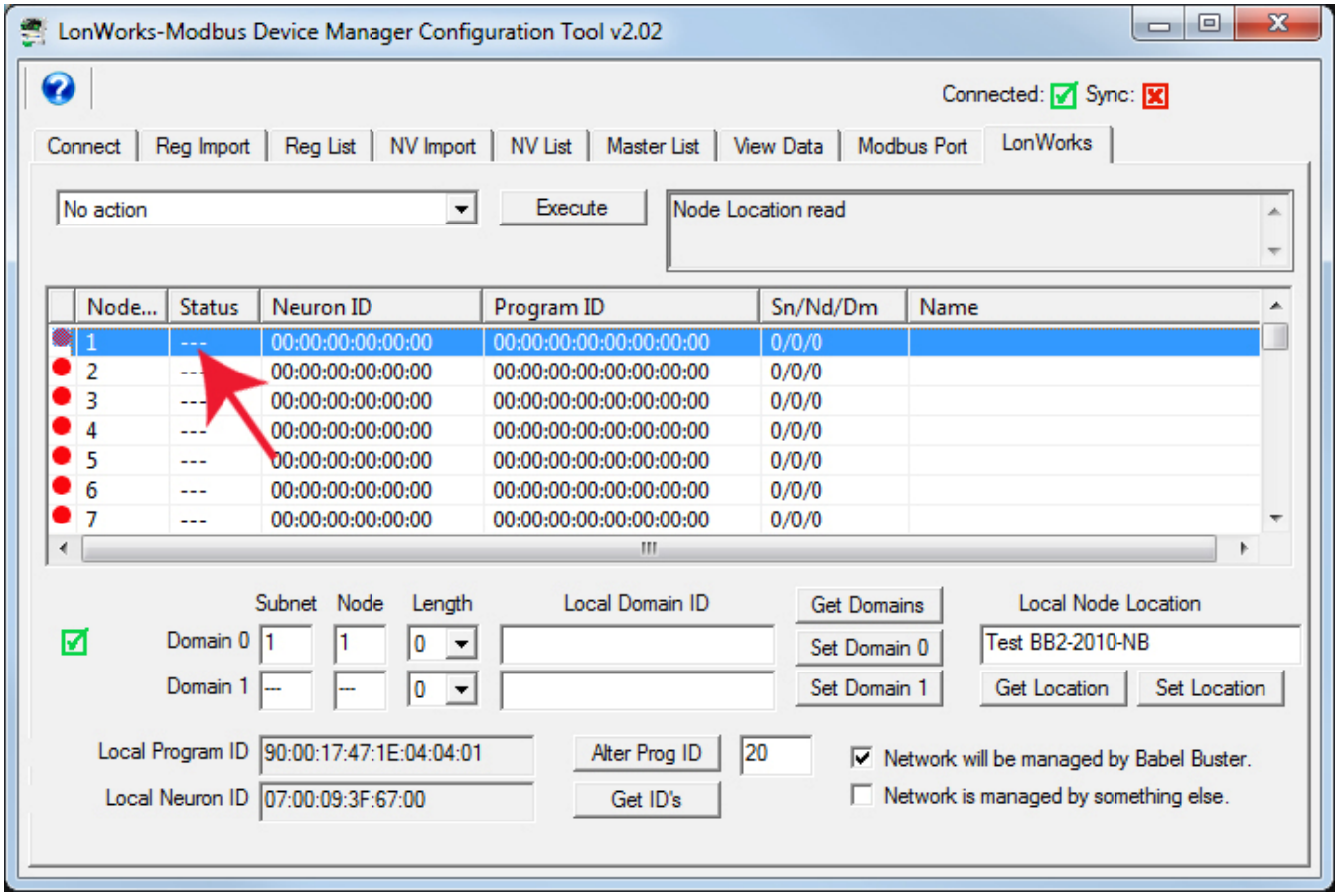
The meaning of the various letters that can potentially show up are as follows. Not all of these are error indications, some are merely progress indicators.

S - Service pin, means this node table entry has a valid Neuron ID
D - Domain set was successful
Q - Domain query was successful
R - Ready - will normally be replaced by "Ready" indication
W - Domain query came back with wrong subnet/node
F - Failed to set domain table in the LonWorks device at the Neuron ID shown
N - Domain query failed (almost always implied by 'F')
E - API error - there has been a miscommunication between the two processors in the gateway (contact tech support)

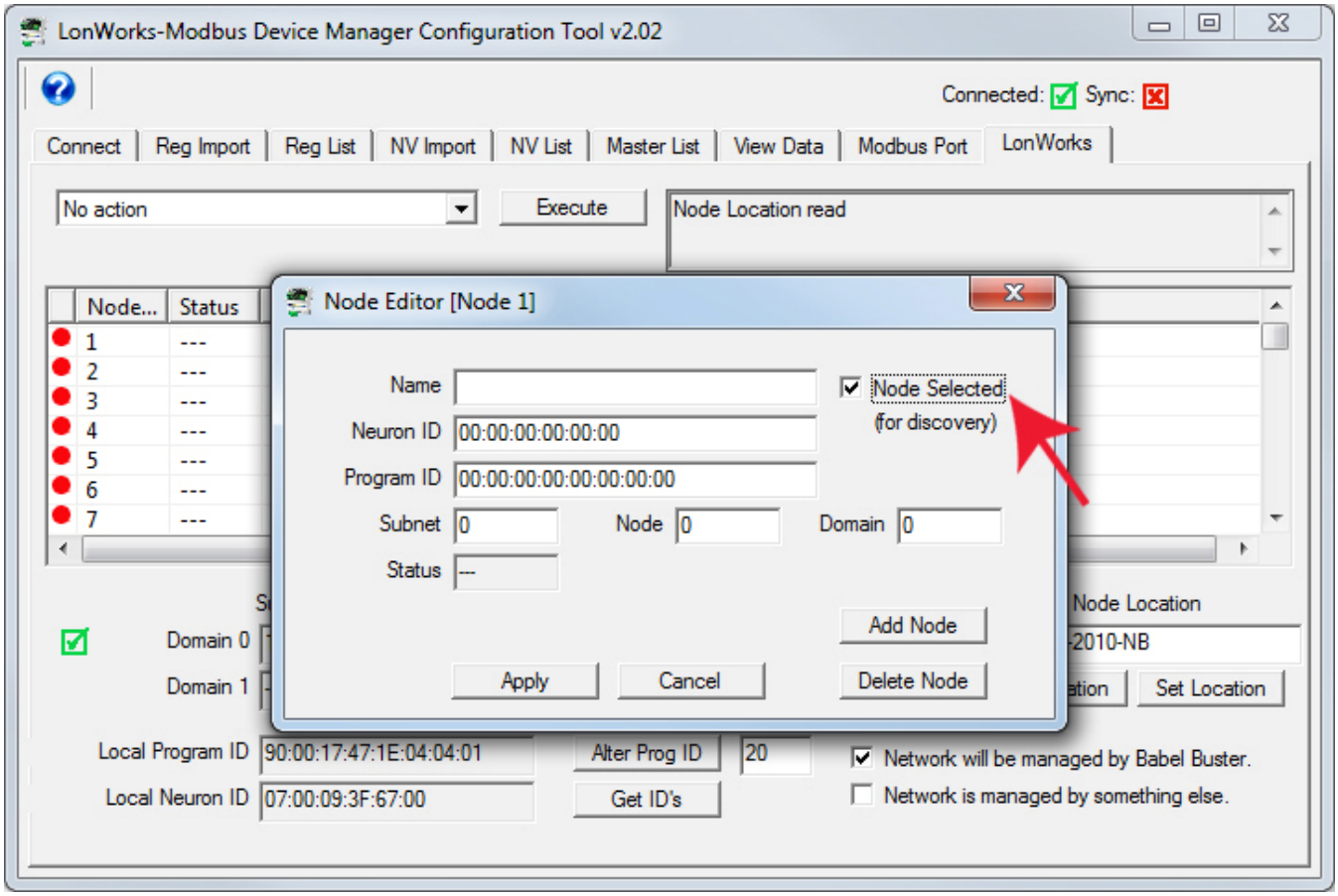
12.4 Node Discovery Using Service Pin

Virtually all LonWorks devices have a "Service Pin", or service button, also sometimes denoted as an 'Install' button. A node that cannot be discovered via network query due to the fact that its domain table is set to something unknown to (and incompatible with) the Babel Buster gateway can still be discovered using the service pin/button method.

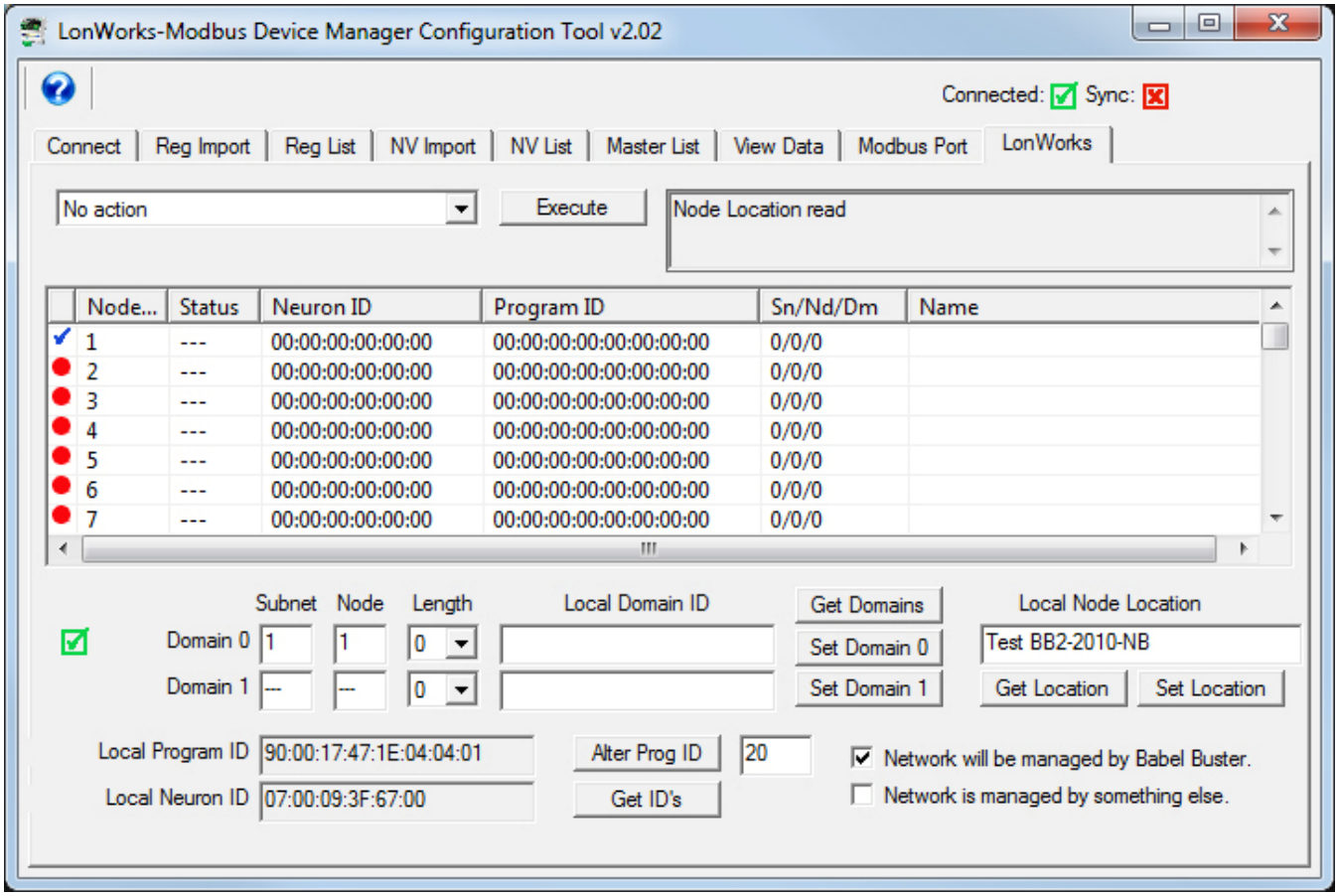
To discover a node using the service pin method, start by double clicking on an unused entry in the node table to open the Node Editor dialog.



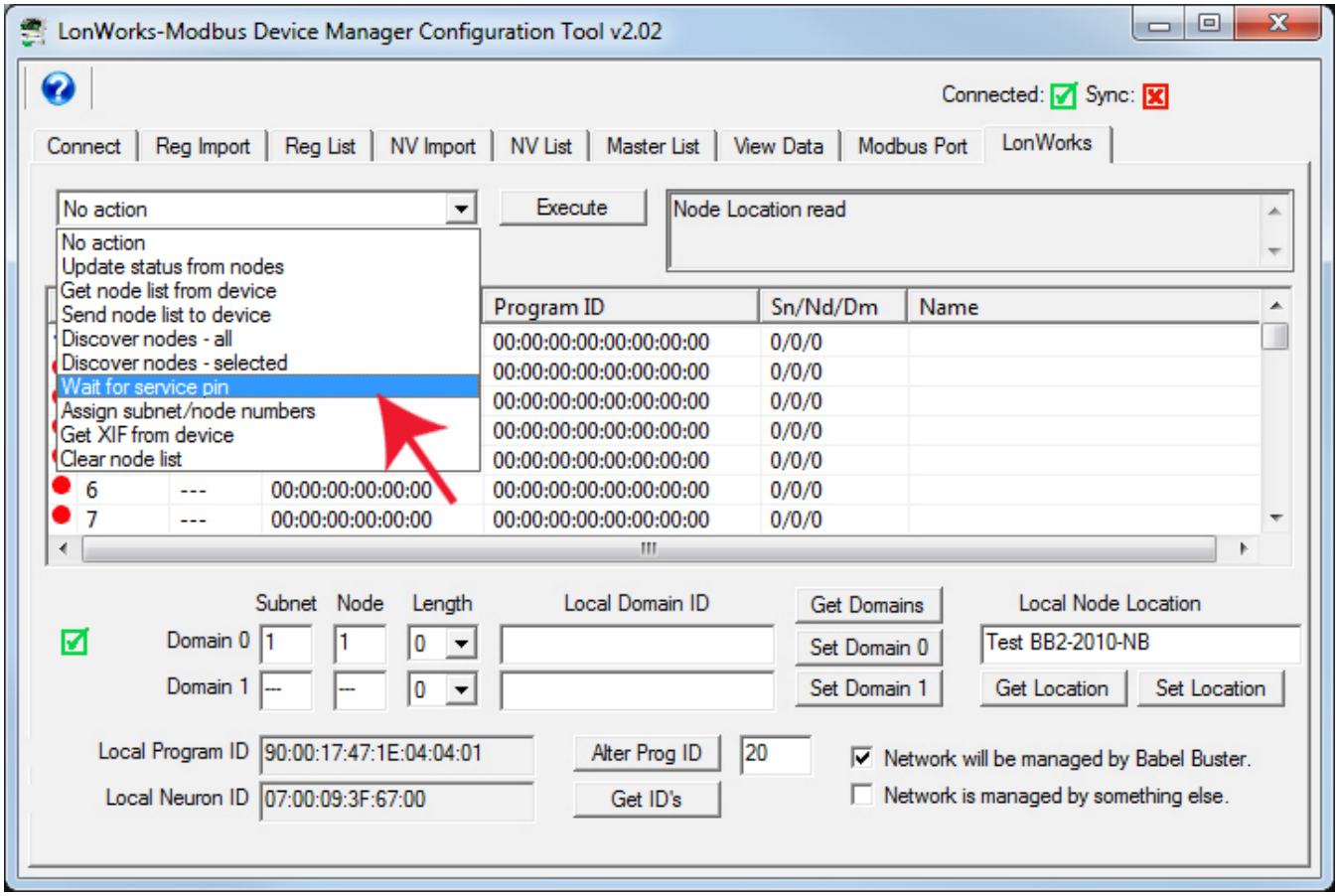
Click the 'Node Selected' box. This tells the tool that we are going to put whatever we find in this spot.



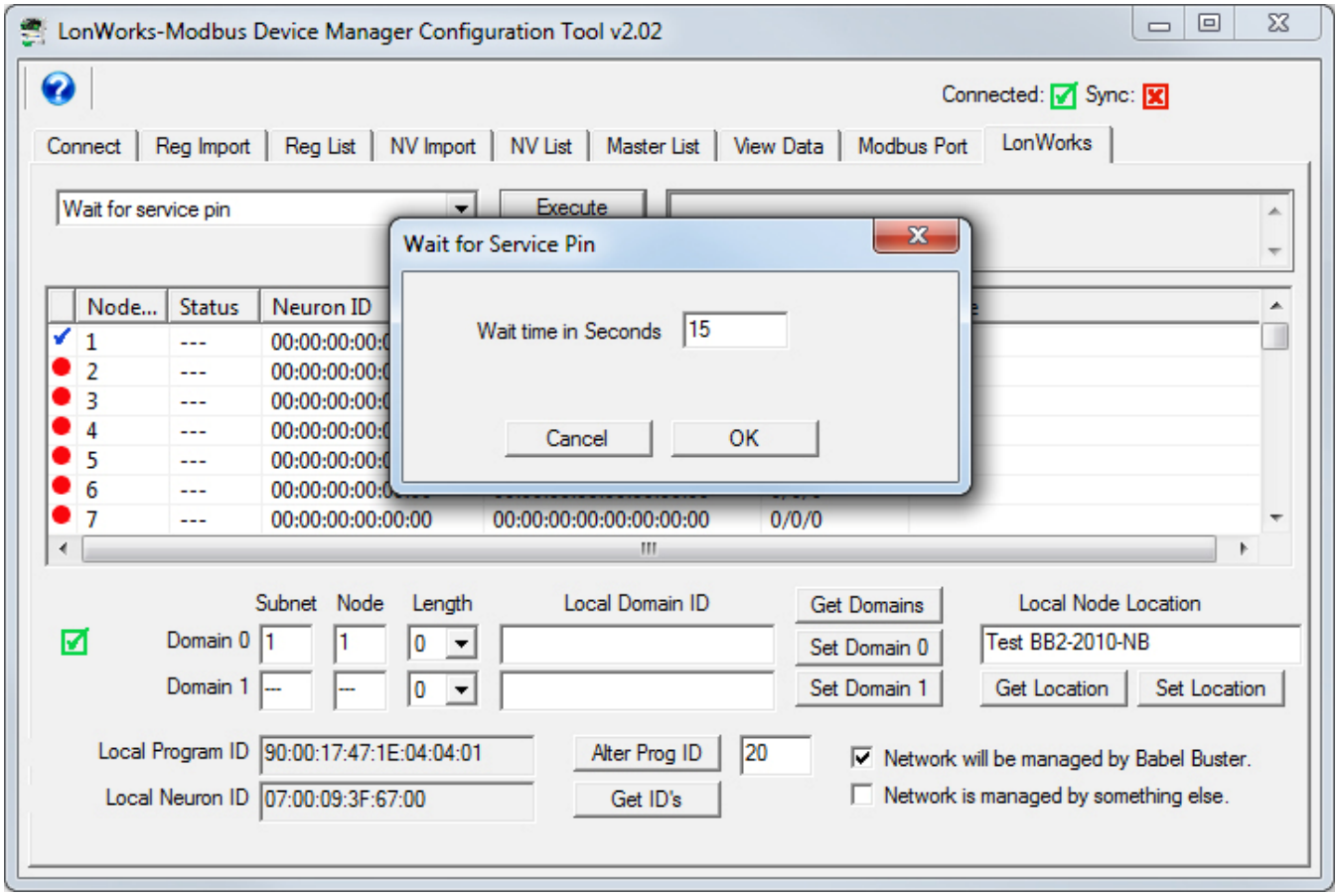
The selected node table entry will be indicated by a check mark in the icon column as illustrated here. (For discovery via network query, you can have multiple nodes showing as selected.)



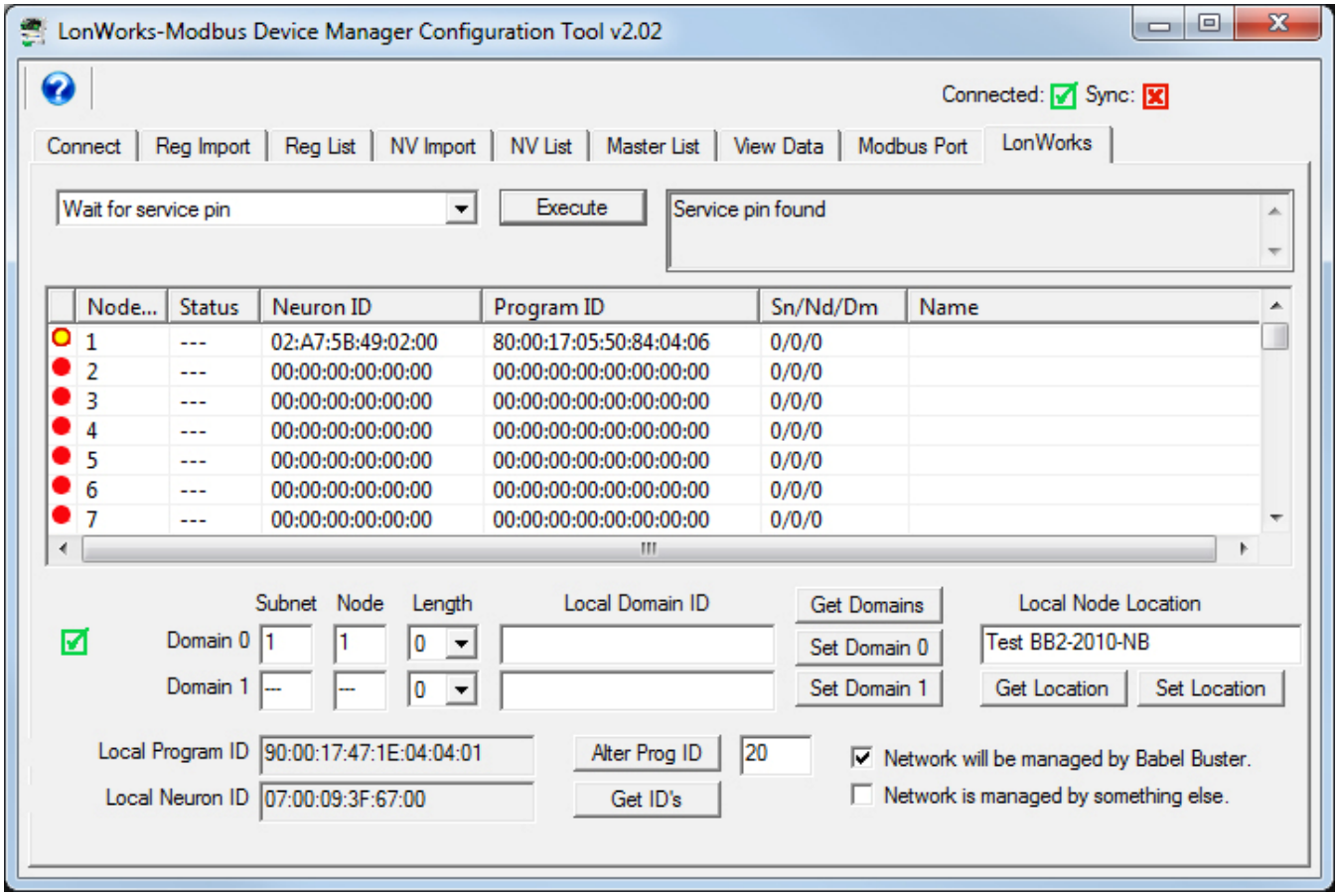
Provided you have a table entry selected, choose 'Wait for service pin' from the list and click Execute.



You will be given a dialog allowing you to specify a time period to wait for the service pin message to arrive on the LonWorks network (which is of course assumed to be connected to the device of interest). The wait time will default to 15 seconds, but if you have to hike a distance to get to the device, you will want to set this timeout much higher. Discovery via network query is preferred to avoid having to physically access the remote device, but sometimes the remote device has been previously configured in an incompatible manner and you have no choice.

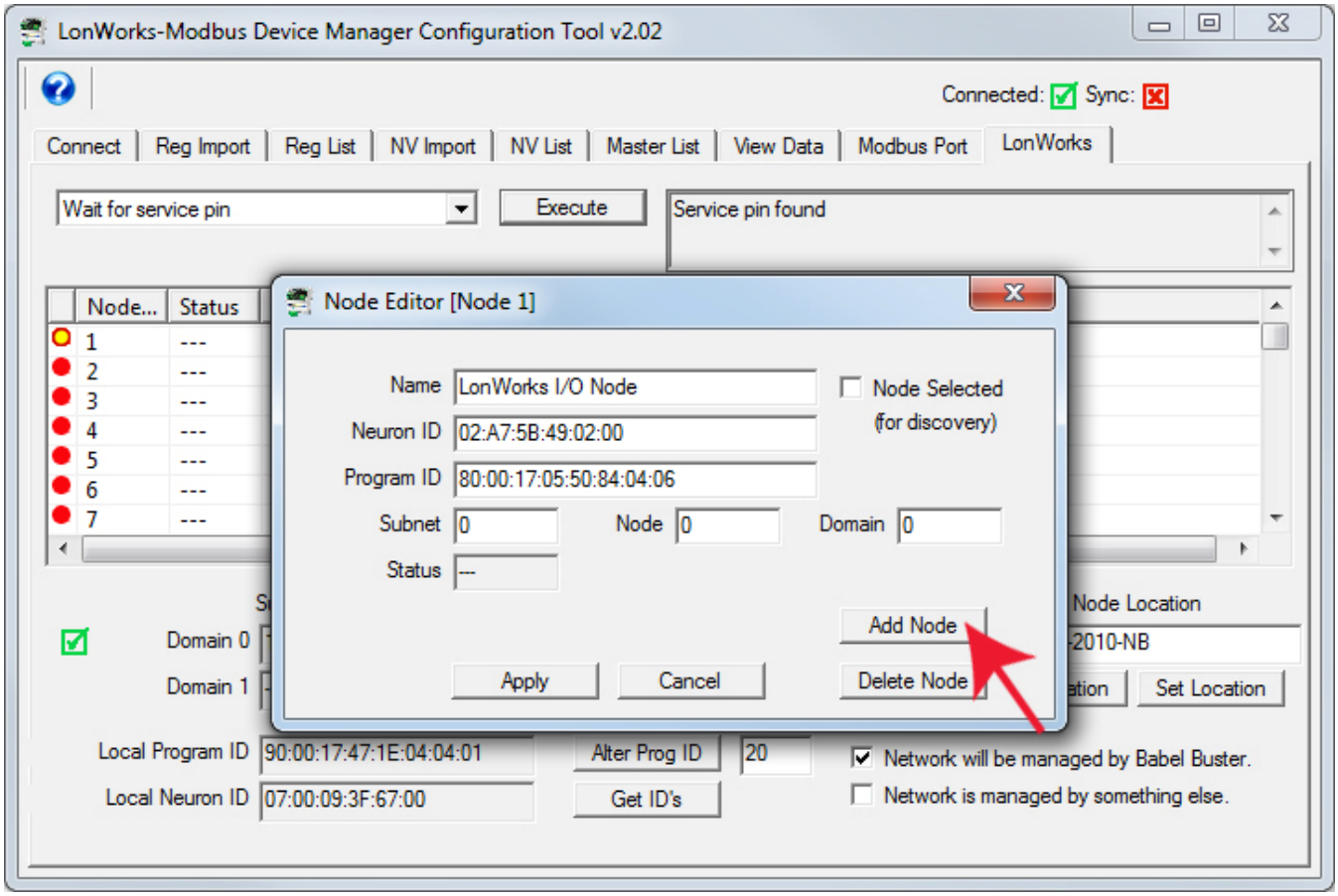


After clicking OK in the wait time dialog, proceed to press the button on the remote LonWorks device. When the service pin message is received, it will be indicated by an icon that is yellow with a red circle around it.

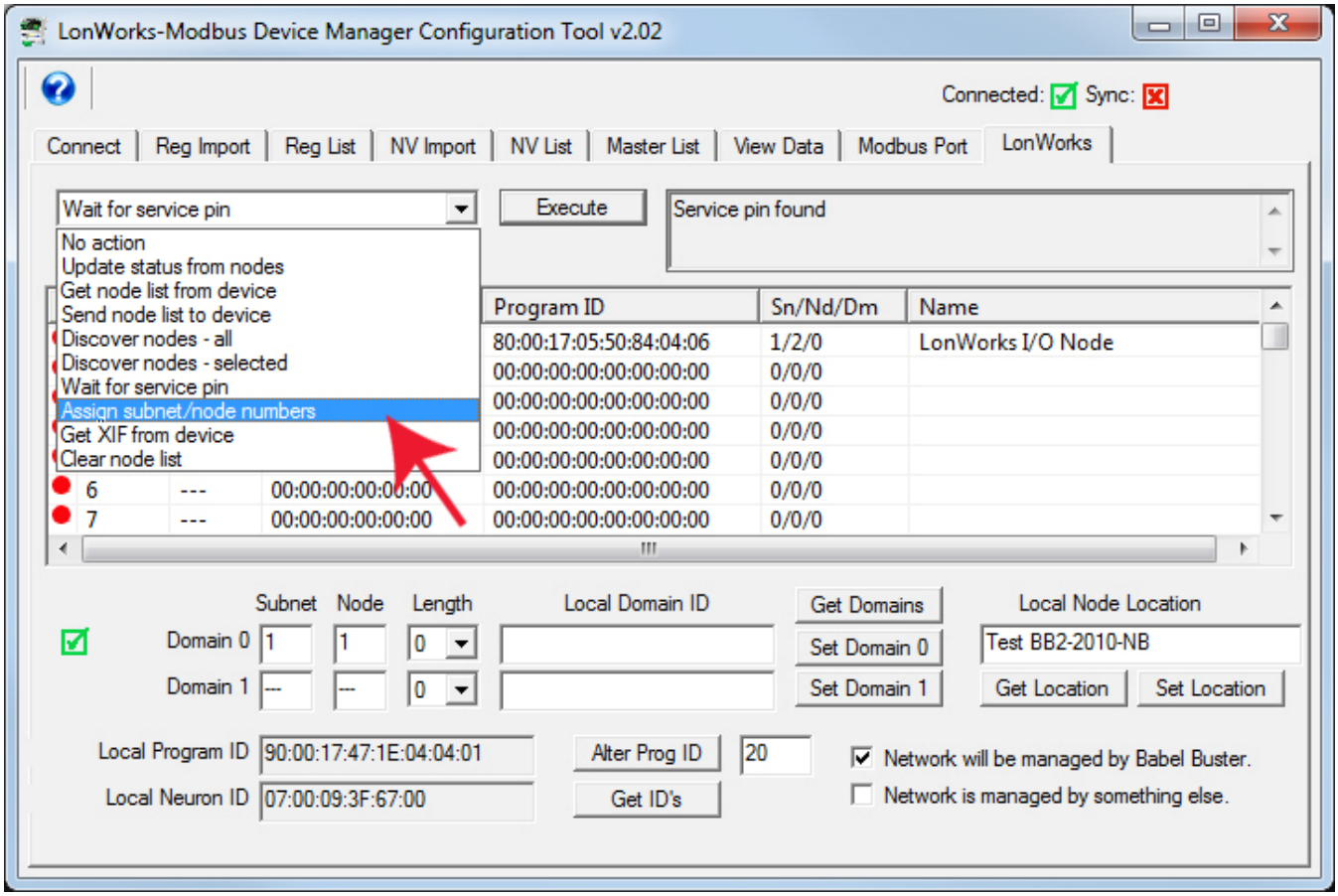


Next, you want to tell the tool 'yes, keep this one' by double clicking the node entry in the table to open the Node Editor dialog, and then clicking Add Node. You may also want to give the node a name at this point. Simply enter a name in the Name window before clicking Add Node. You can always come back to the Node Editor dialog later and add a name.

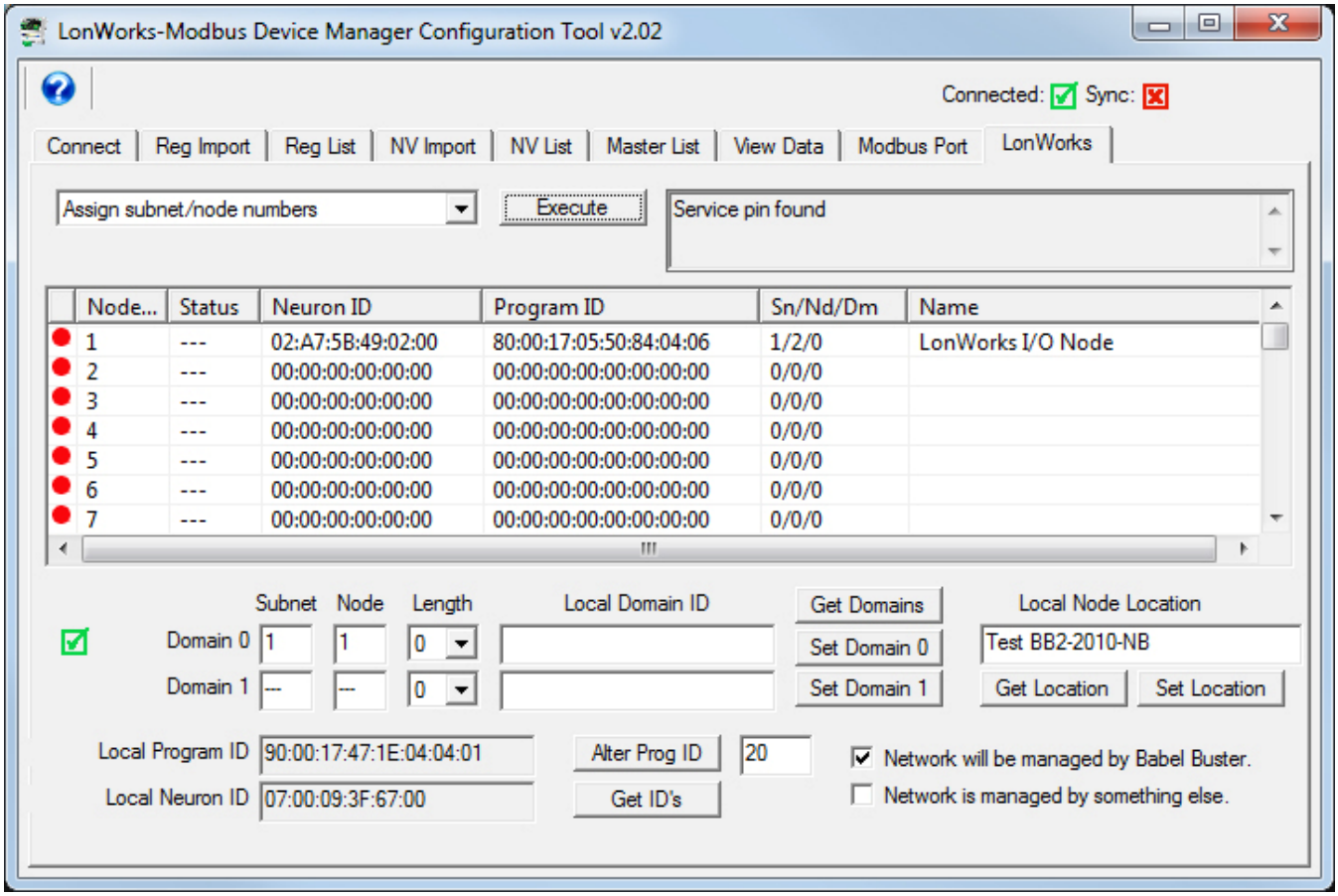
If you know exactly what subnet/node you want this device assigned to, you can also enter those numbers at this point. Be sure you know what you are doing here. If you use a subnet different than the gateway itself, you will not be able to communicate with it.



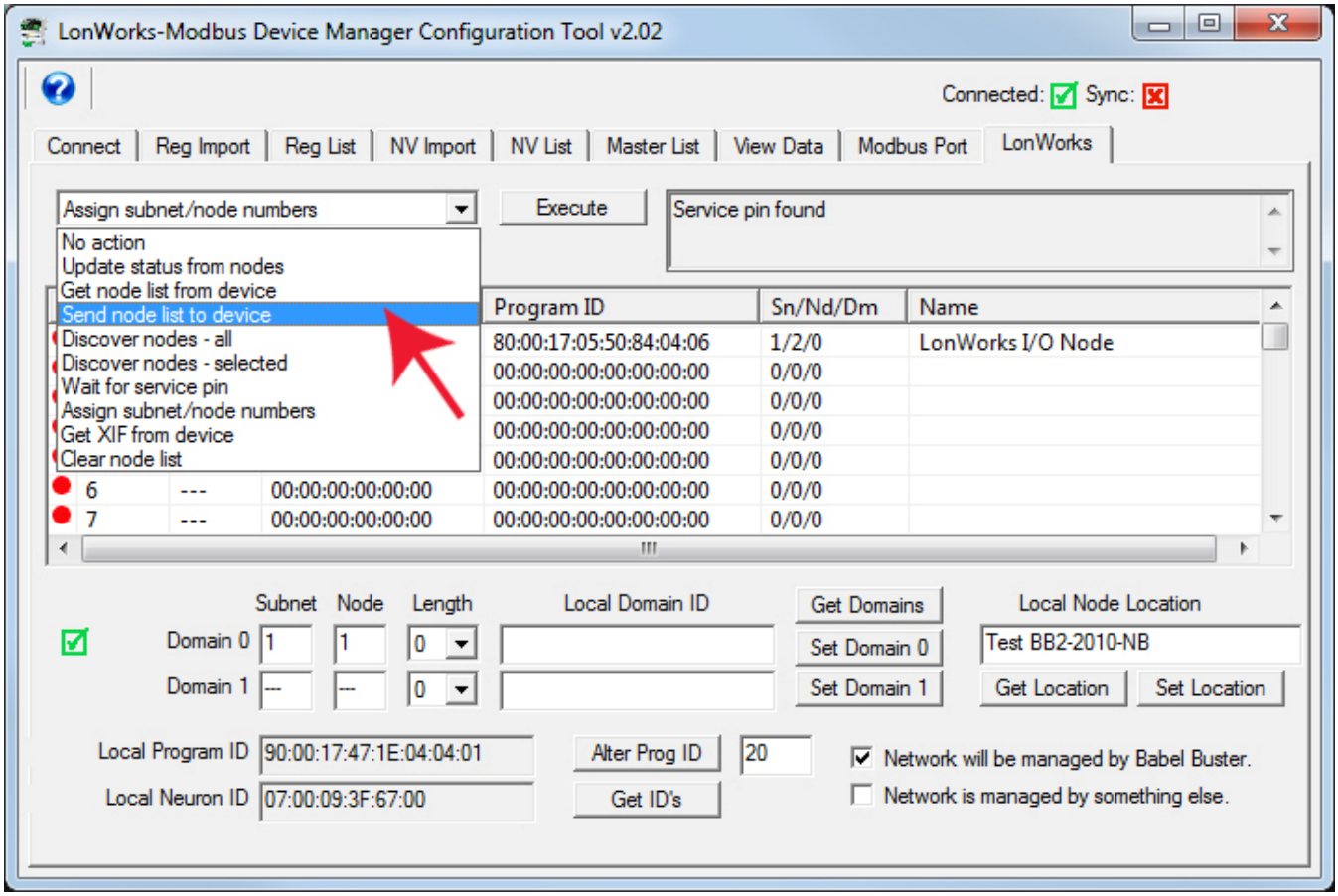
The best bet for picking subnet/node numbers is to let the configuration tool do it for you. Select 'Assign subnet/node numbers' from the list and click Execute.



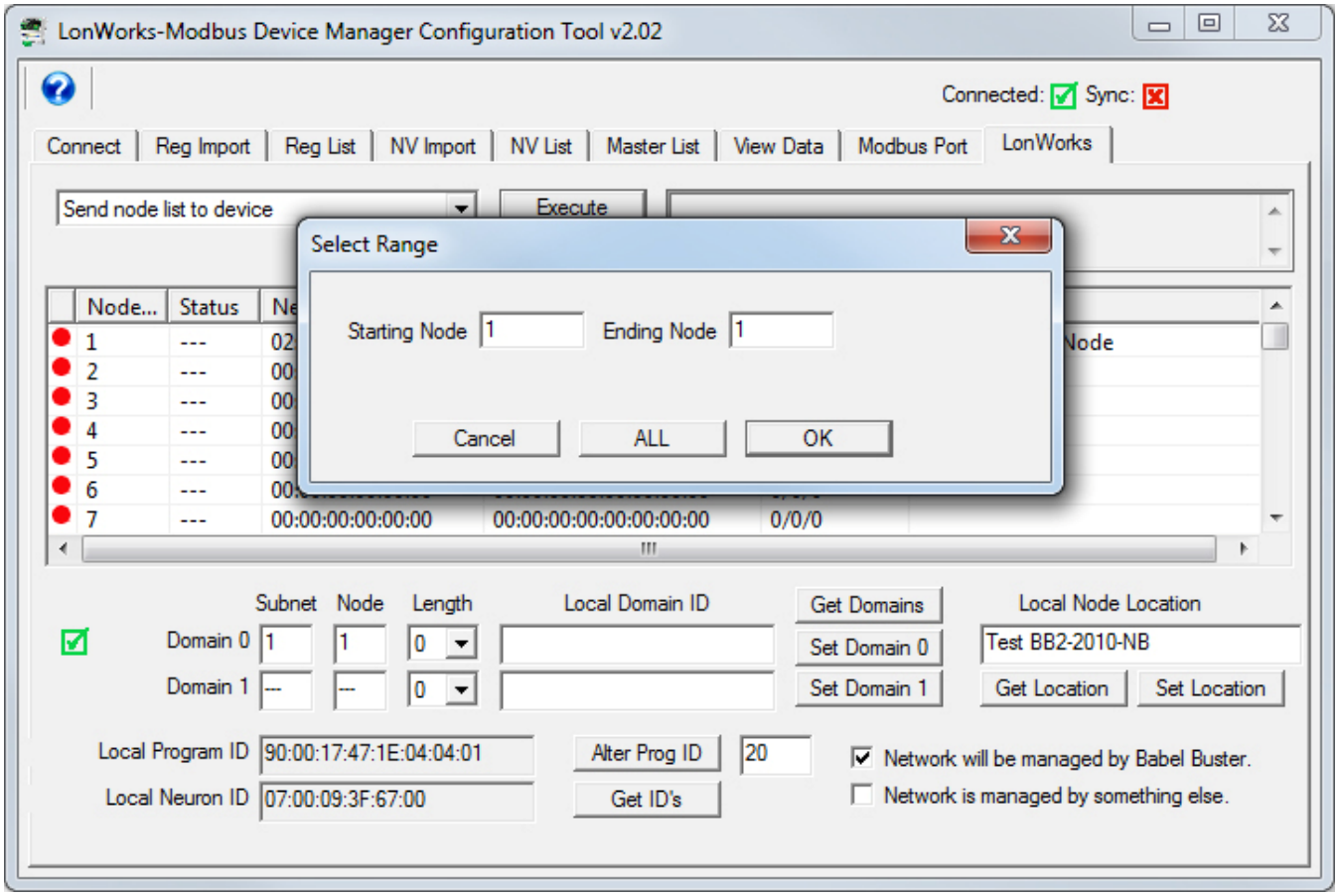
The gateway will always assign its own subnet, and will assign node numbers incrementing upward from its own node number plus one. Once you have 'added' the node and assigned a subnet/node, the icon in the first column will change to solid red indicating this is now a valid node table entry that has not yet been sent to the gateway. (Receiving a service pin message does not populate the node table in the gateway device, it only populates the table in the configuration tool.)



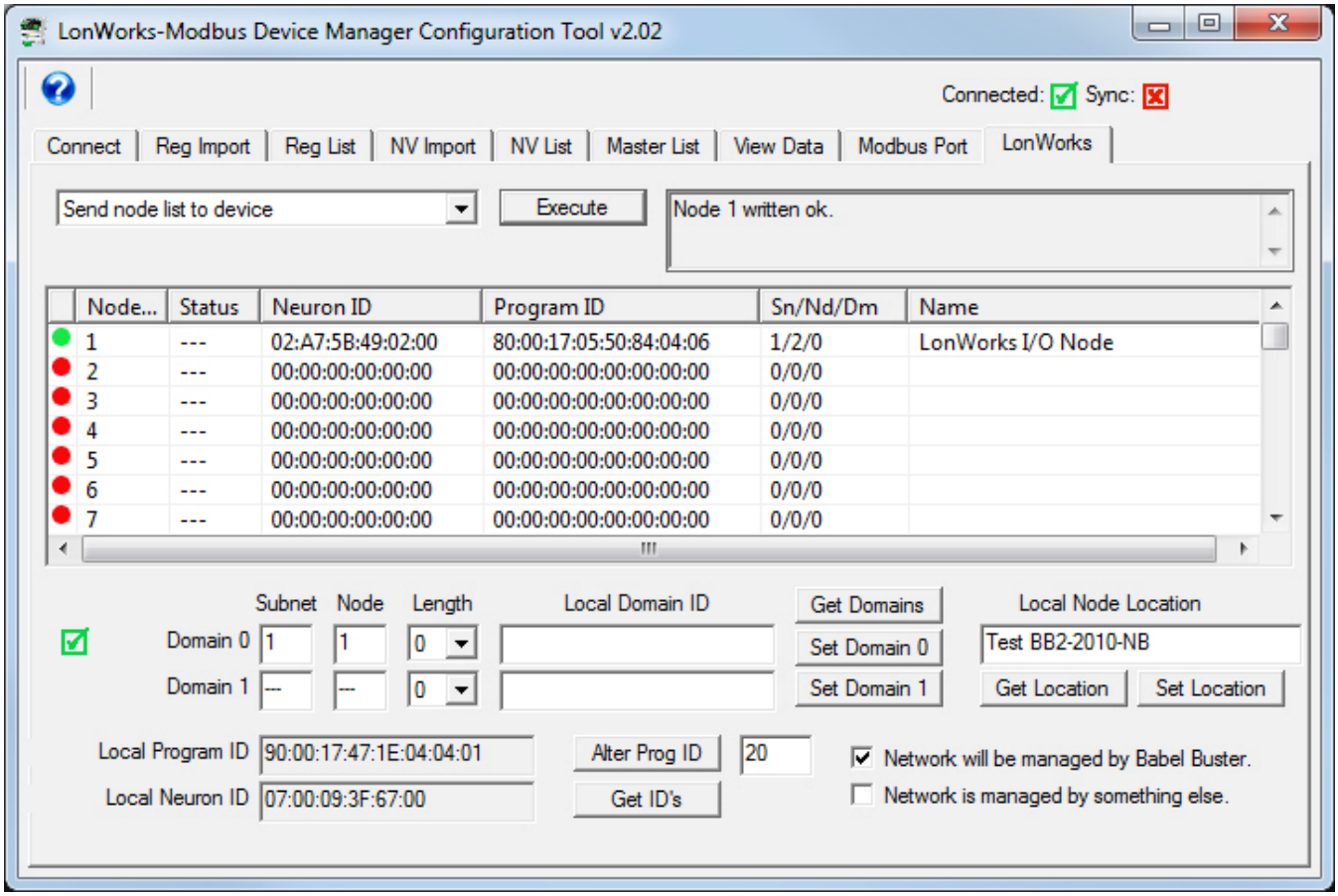
To add this node to the node table in the gateway device, select 'Send node list to device' and click Execute.



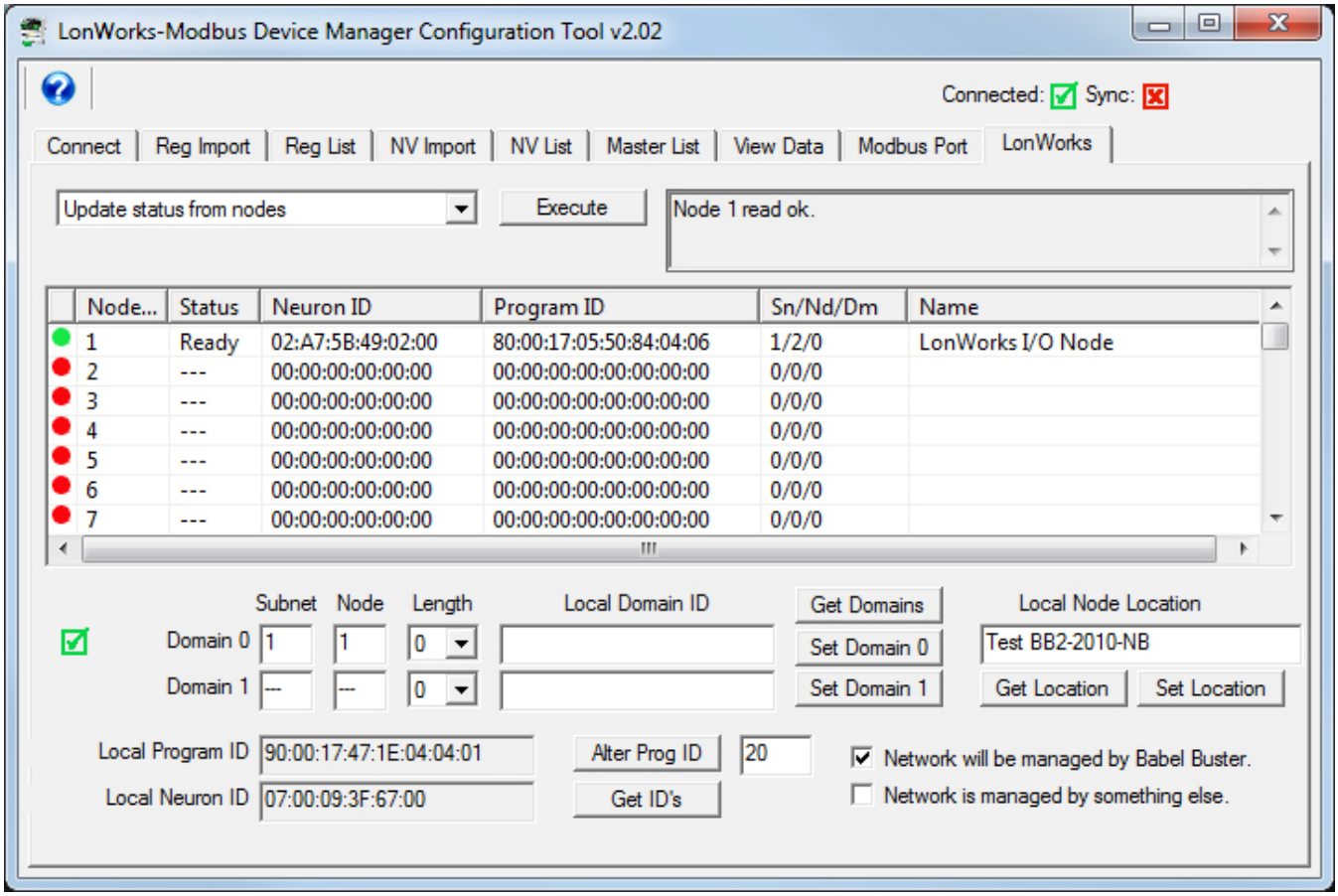
Any time you send the node list to the device (or get it from the device), you will be given the option of selecting a range. When sending node table entries to the gateway, it will default to a range of those nodes that are configured in the tool but not configured yet in the device.



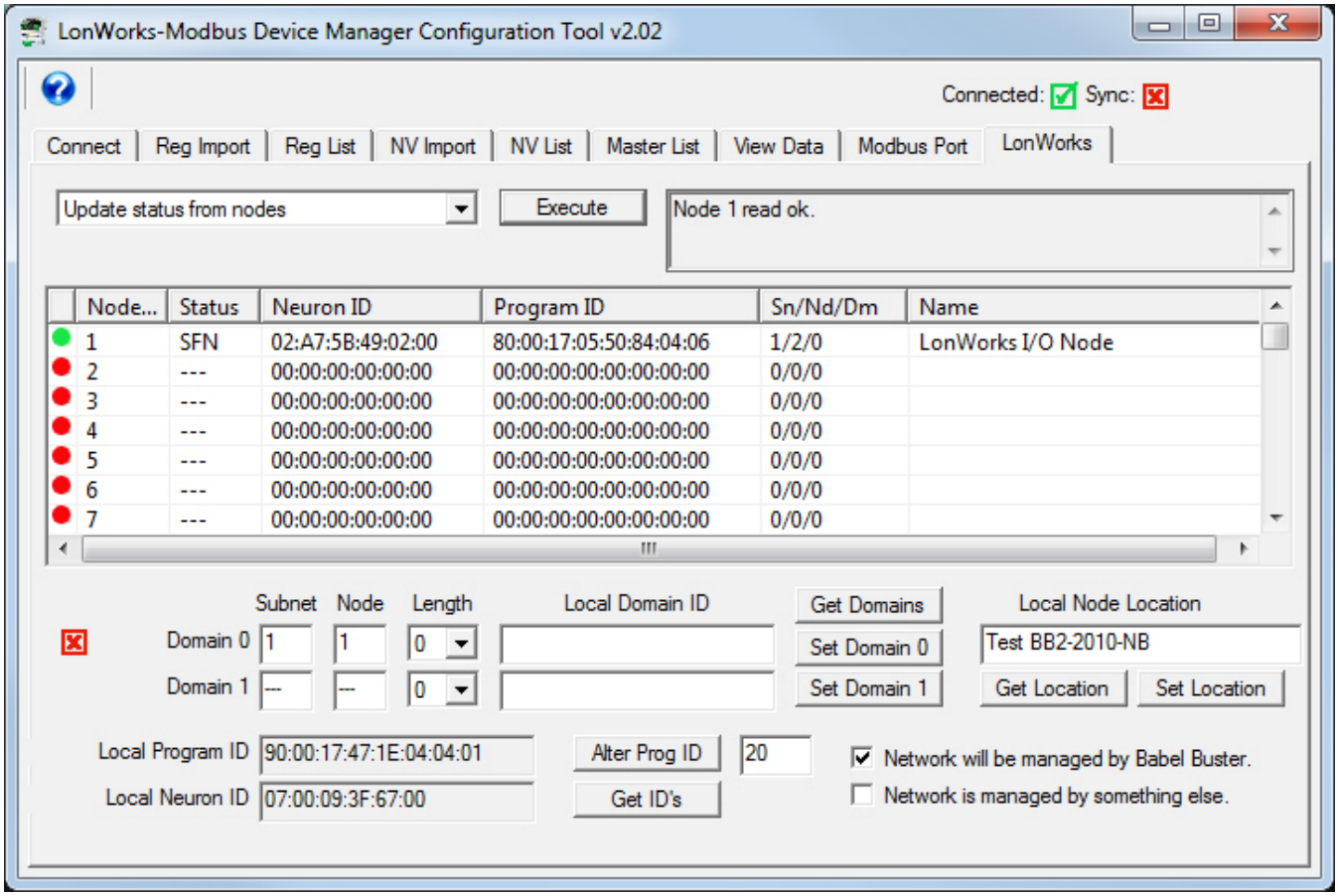
Once the node table entry is sent to the device, its icon will become green.



Next, you will want to select 'Update status from nodes' and click Execute (and provide range or just click OK in range dialog). If the node was just recently sent to the gateway, its status might not be "Ready" immediately. Try again in a few moments if not ready. If status remains not ready for an extended period, the status will indicate where it is stalled.



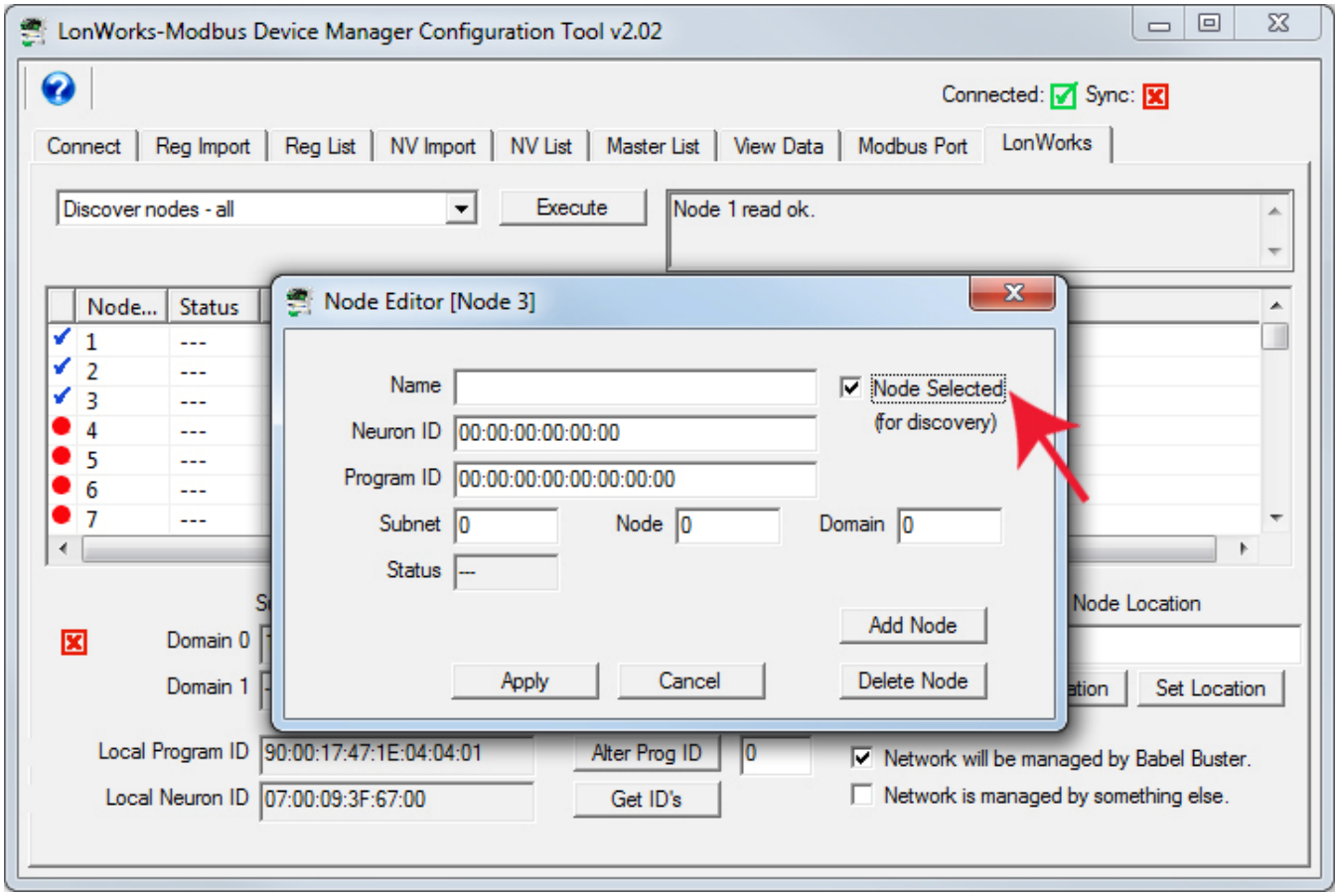
The most common non-ready status will be SFN. This means it was unable to force the remote LonWorks device onto the same domain as the gateway. It will retry every 15 seconds until successful, or forever if not.



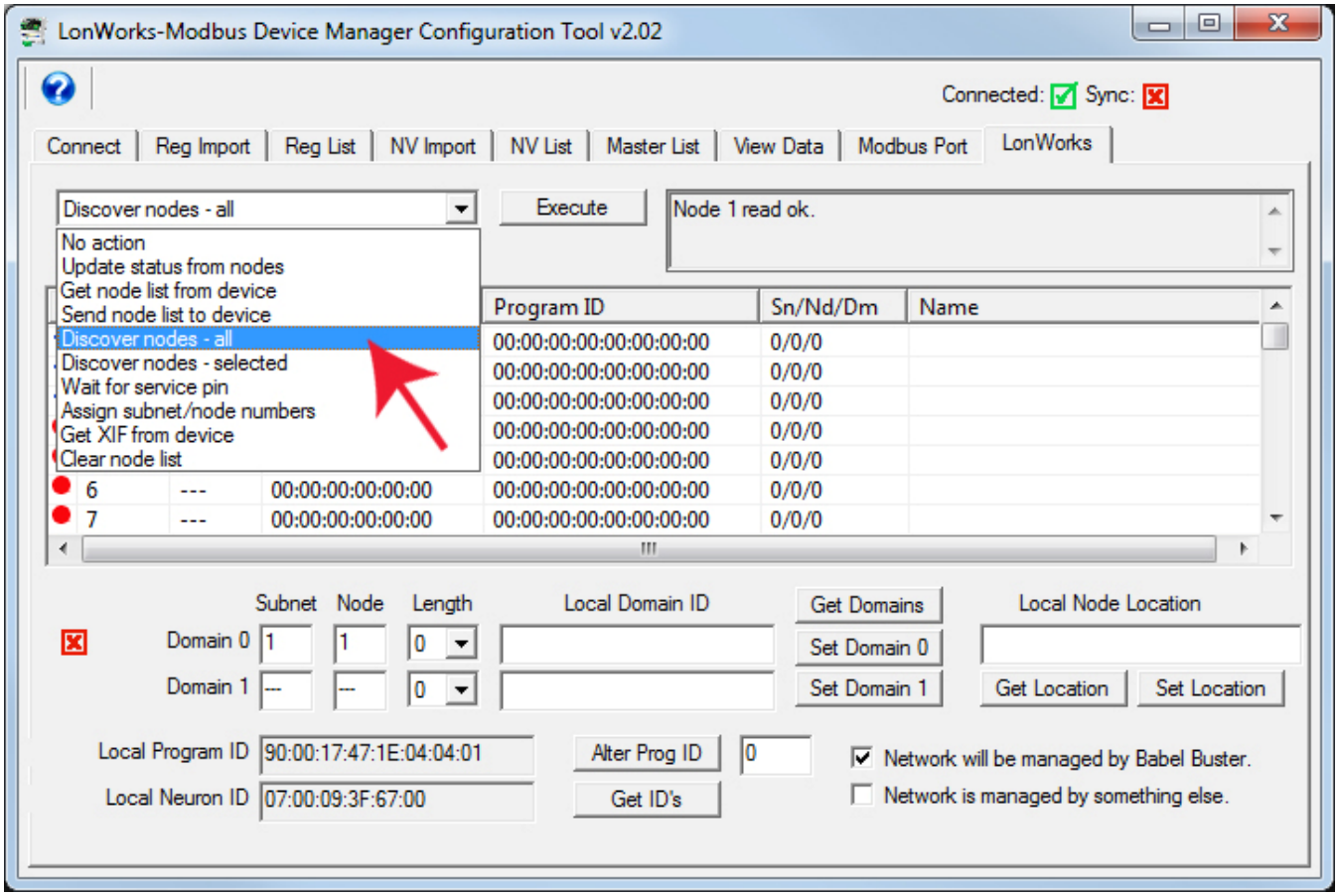
12.5 Discovery of All Nodes via Network Query

Discovery via network query is easy in that you do not need to physically access the remote LonWorks device. However, it does require that the remote device's domain table is configured such that the gateway is able to query it. If the domain length is zero, then the ID is effectively a wild card. LonWorks protocol recommends that all LonWorks products be manufactured with a default domain length of zero. But if the device had been previously commissioned on a managed network, the network management tool most likely changed the domain to something unknown to (and incompatible with) the Babel Buster gateway. You then have two options: (a) Use the service pin method outlined above, or (b) Find out what domain the prevous network management tool used and change the gateway's domain to match it.

To begin the discovery process. double click a node table entry to open the Node Editor dialog, and click Node Selected. Do this for as many nodes as you want to discover so that multiple unused entries in the table show a check mark in the icon column.

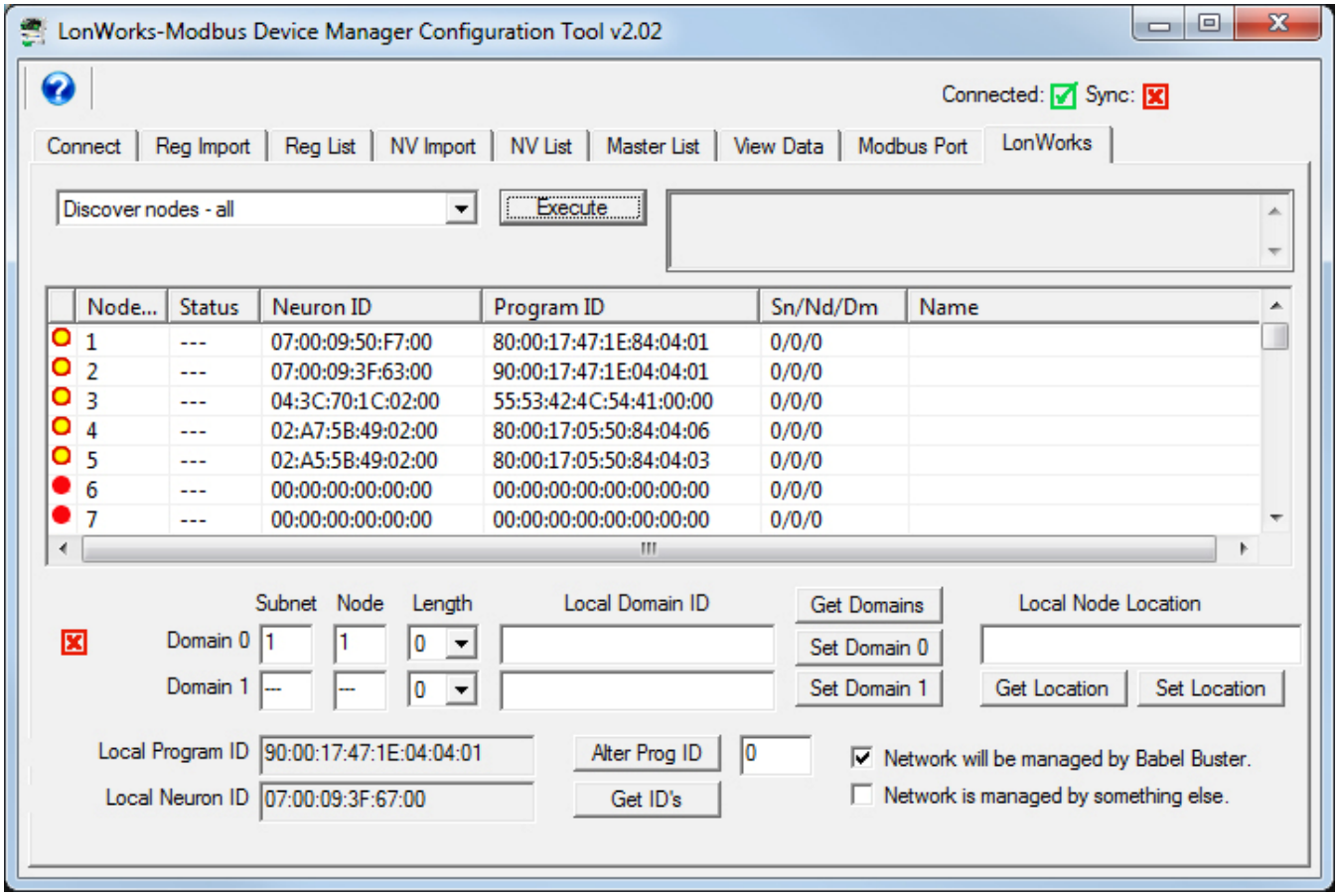


Select 'Discover nodes - all' from the list and click Execute.



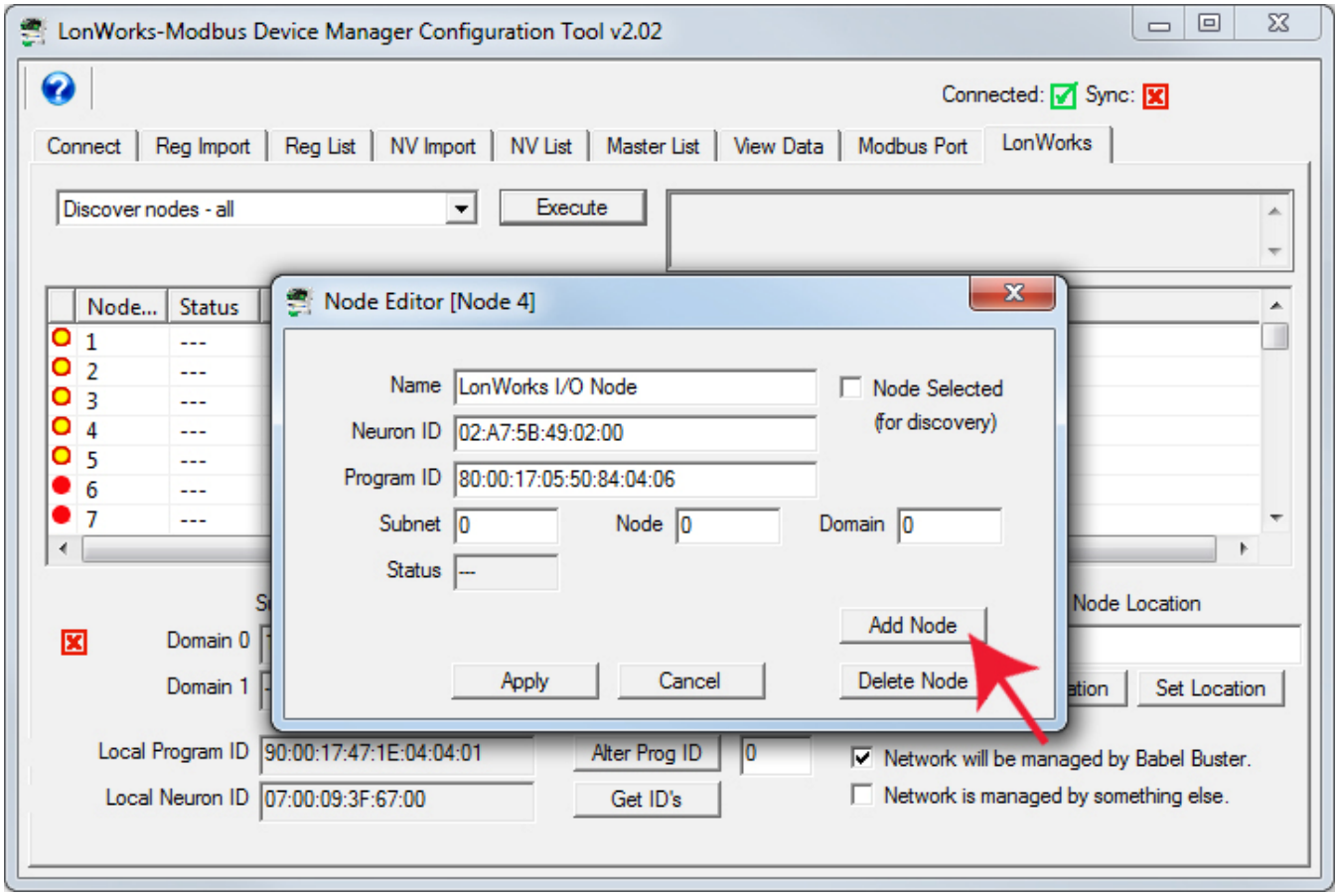
The gateway now sends out a query to the network. This process has a very specific technical definition, but in general terms, what is happening is this: The gateway announces to the world, "Hey, if you're out there, talk to me, and tell me who you are." If anything on the LonWorks network was able to hear that announcement, it attempts to reply with "Here I am, and this is my ID".

For each response received by the gateway, it will begin to fill in the selected node table entries in the tool. The icon will change to yellow with a red circle indicating that this is a discovered node. The node table in the gateway itself is not yet populated. Only the table in the configuration tool is being populated by this process.



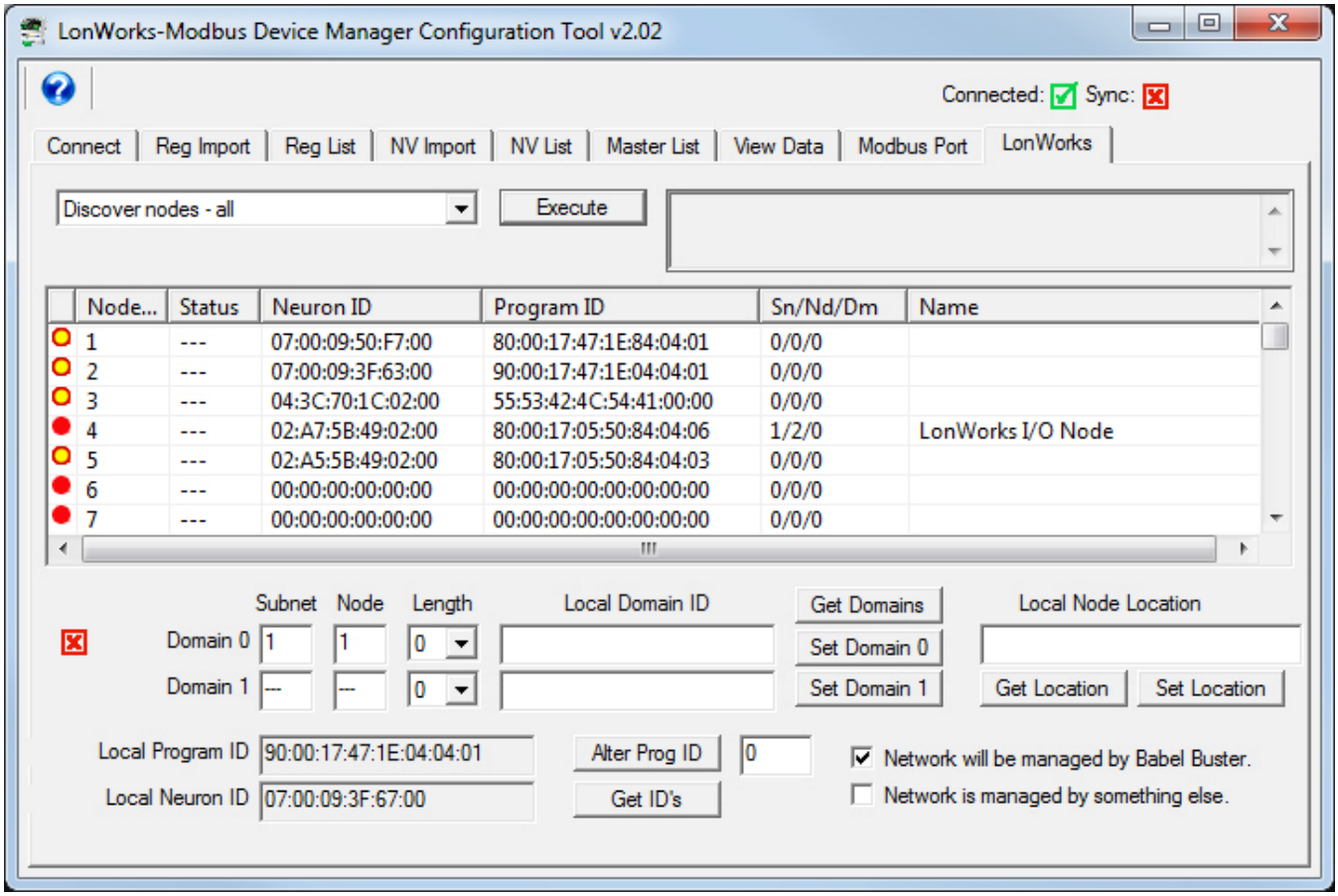
To "keep" the discovered node, double click on it, and click Add Node in the Node Editor. You may also want to give the node a name at this point. Simply enter a name in the Name window before clicking Add Node. You can always come back to the Node Editor dialog later and add a name.

If you know exactly what subnet/node you want this device assigned to, you can also enter those numbers at this point. Be sure you know what you are doing here. If you use a subnet different than the gateway itself, you will not be able to communicate with it.

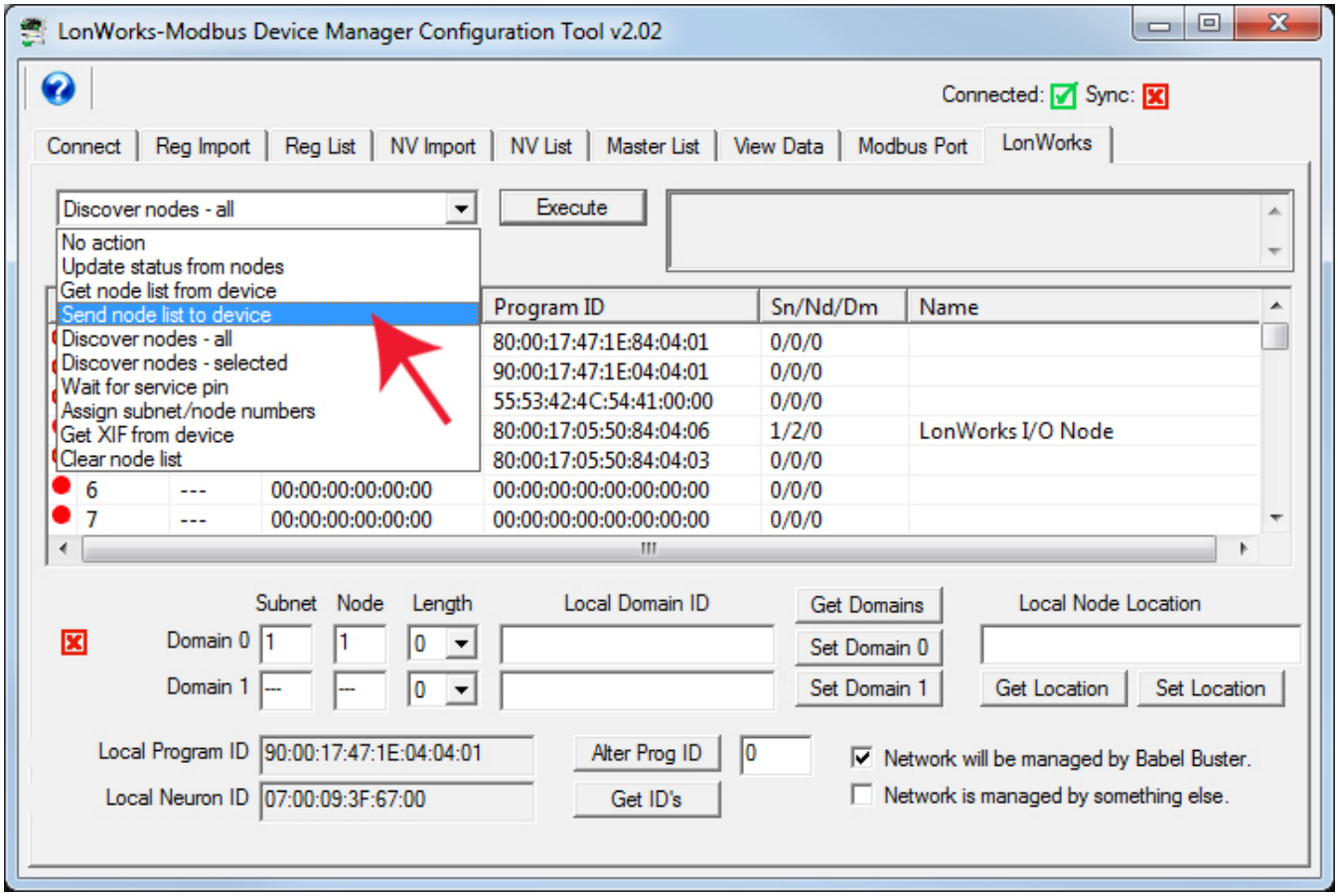


In this example, we have manually entered subnet 1, node 1, prior to clicking Add Node. You should use the same subnet as the gateway itself, and any node number in the range of 1 to 127 that is not used by the gateway itself, or any other node already in the table. Better yet, simply select 'Assign subnet/node numbers' from the list and click Execute - but wait! Don't do this until you have either added all nodes, and/or deleted those you do not care about.

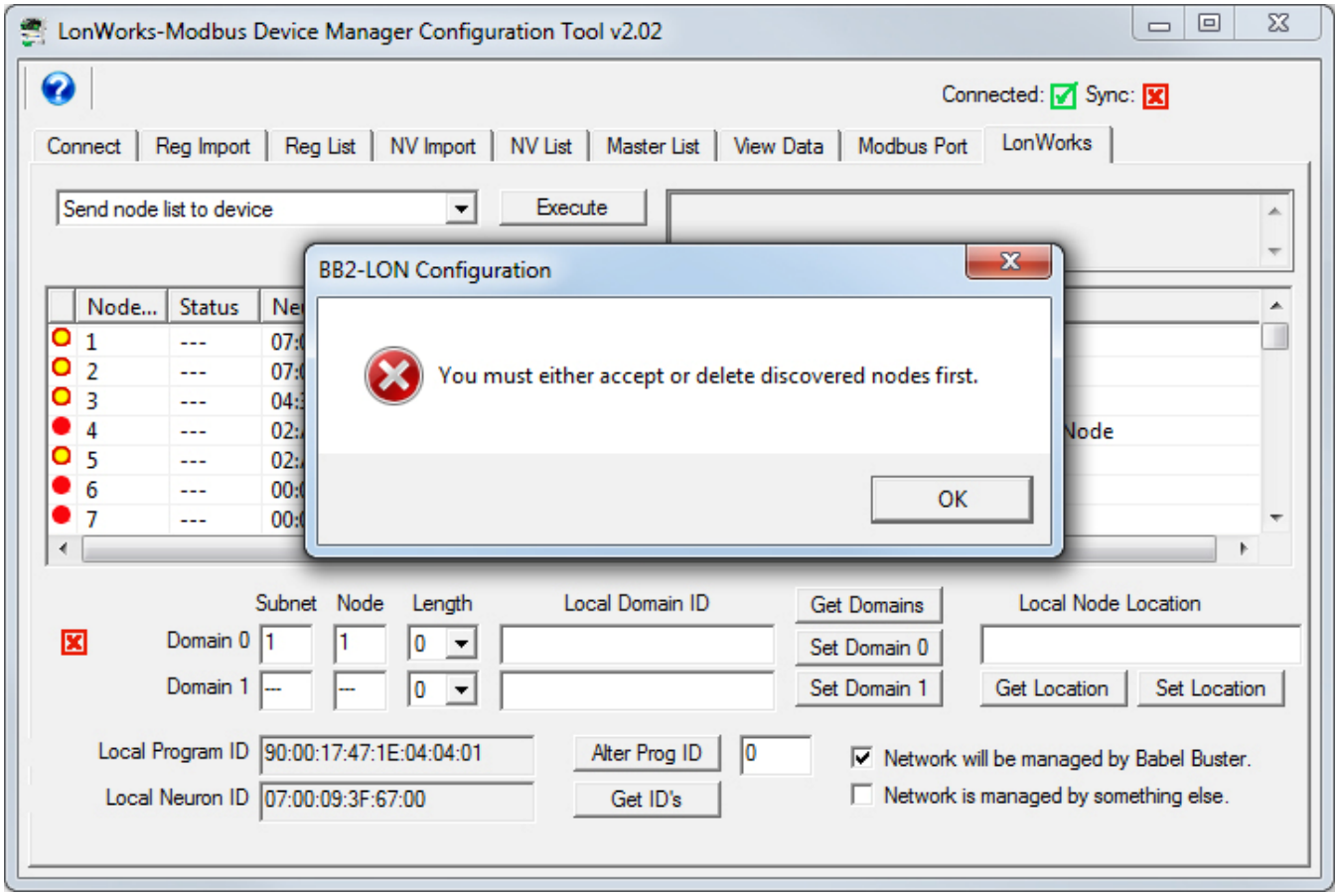
After entering a name and subnet/node, and then clicking Add Node, our table now looks like this.



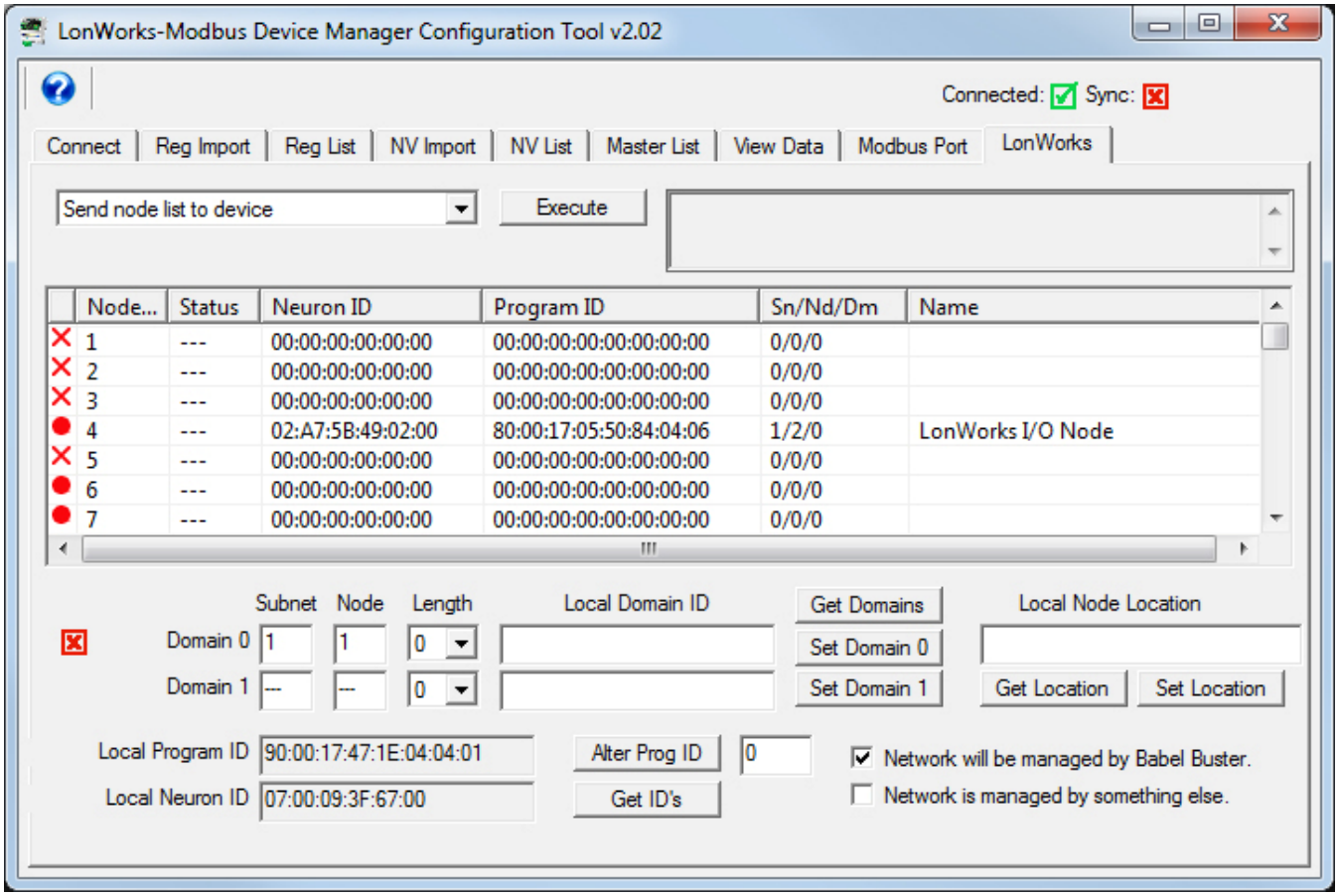
The node table is only populated in the configuration tool at this point. To populate the node table in the Babel Buster gateway, select 'Send node list to device' and click Execute.



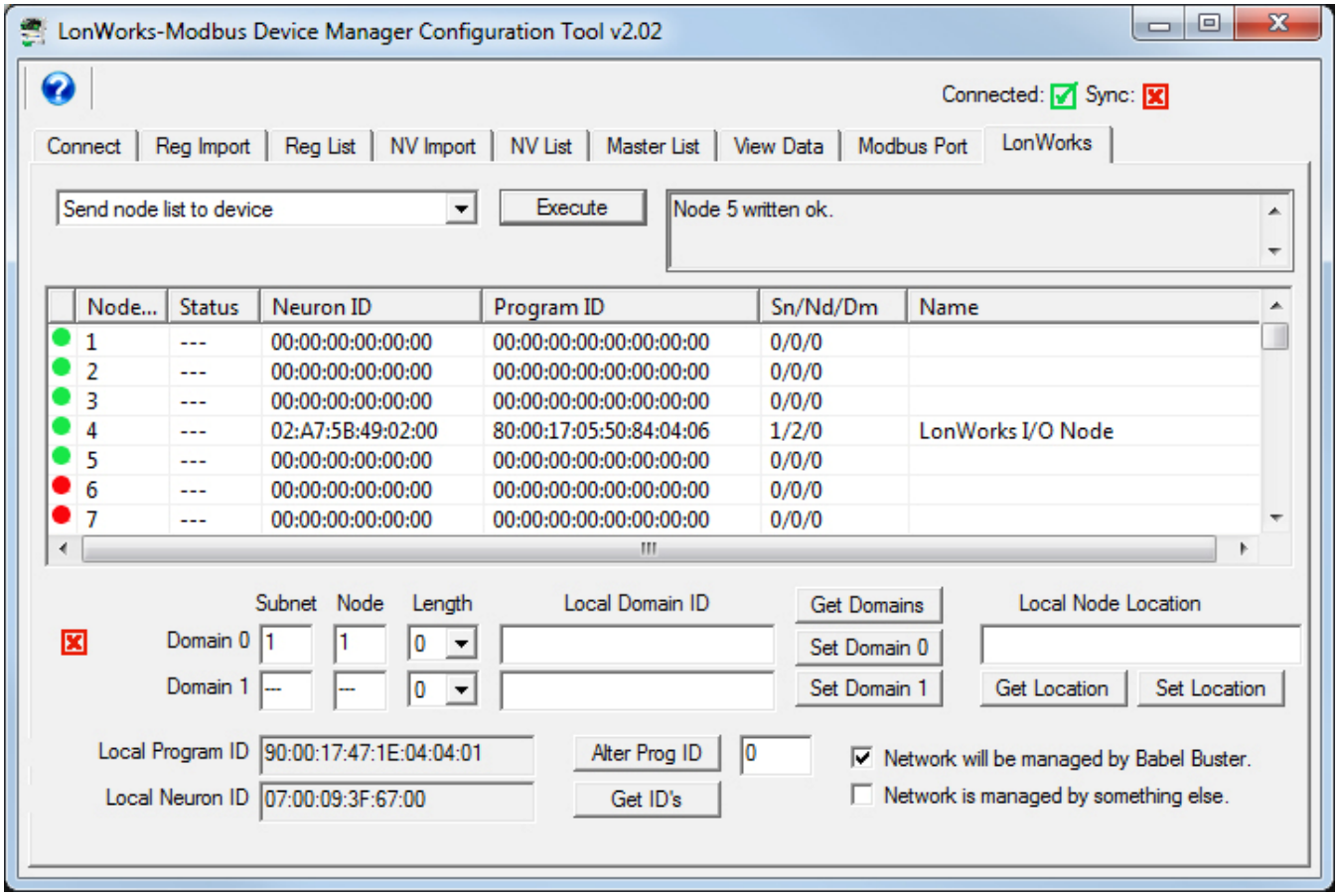
But there's a problem we deliberately left in place here to demonstrate. You need to either accept or reject all of the discovered nodes before you can send the new node table to the gateway device.



To get rid of a node table entry, double click that entry, and instead of clicking Add Node, click Delete Node. The deleted node will be indicated by a red X in the icon column.



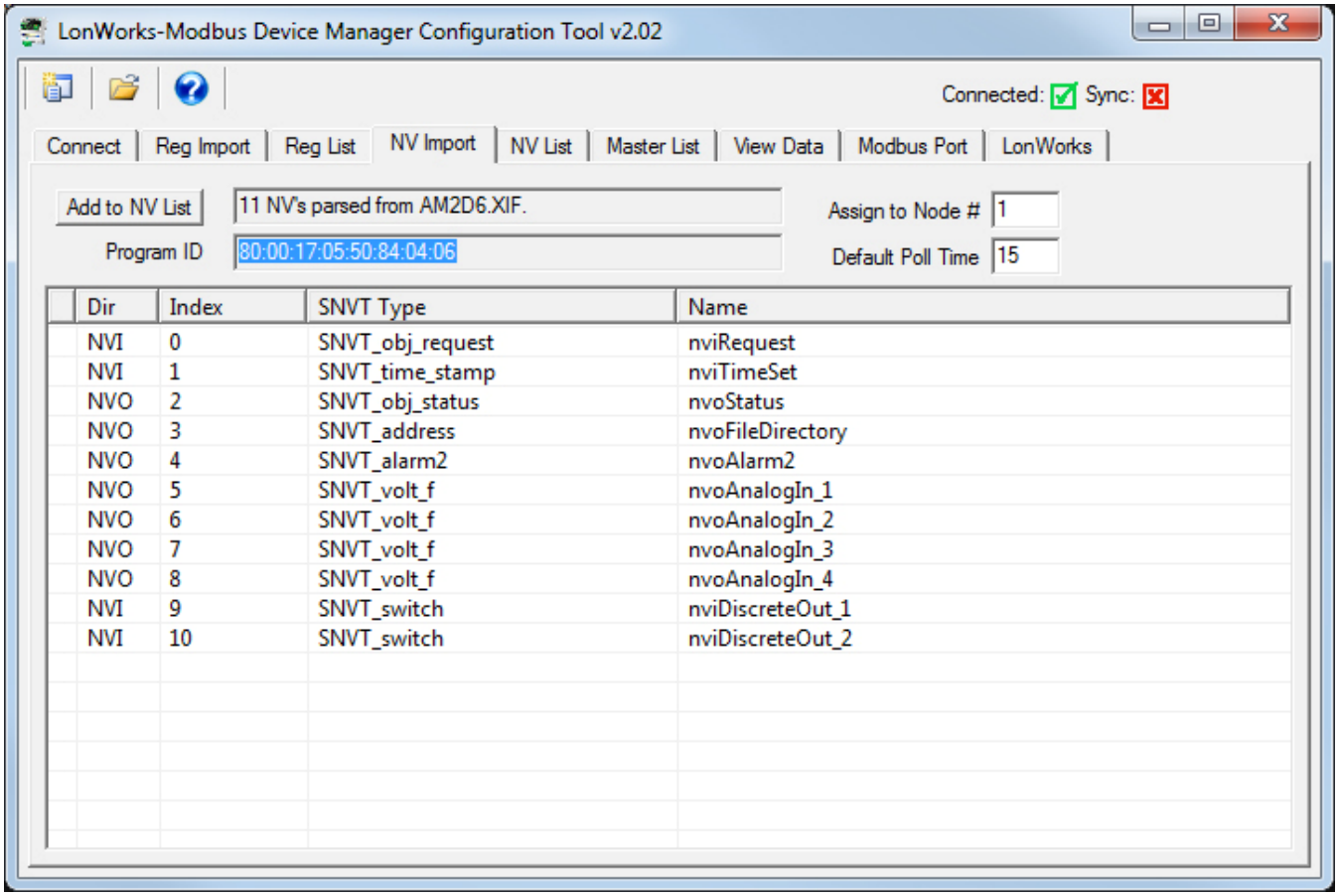
You can now send the cleaned up list to the gateway device. The green icon indicates that this node table entry now exists in the Babel Buster gateway device, not just in the configuration tool.



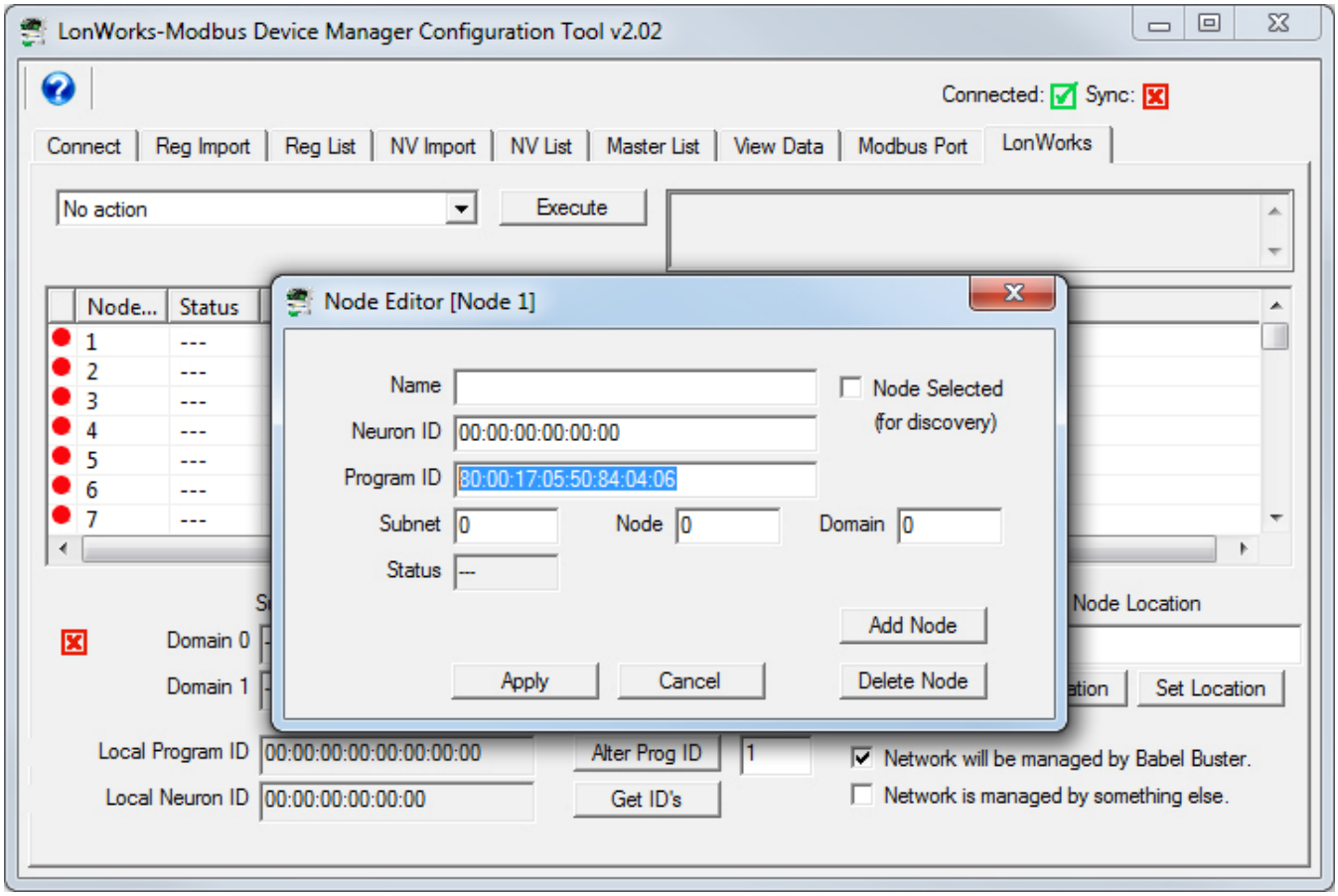
12.6 Discovery of Selected Node via Network Query

If you know a little bit about the remote LonWorks device you wish to discover and want to be sure you only discover that type of device, there is a method of accomplishing this. This process is especially useful if there is a large number of LonWorks devices on the network and you don't care about most of them.

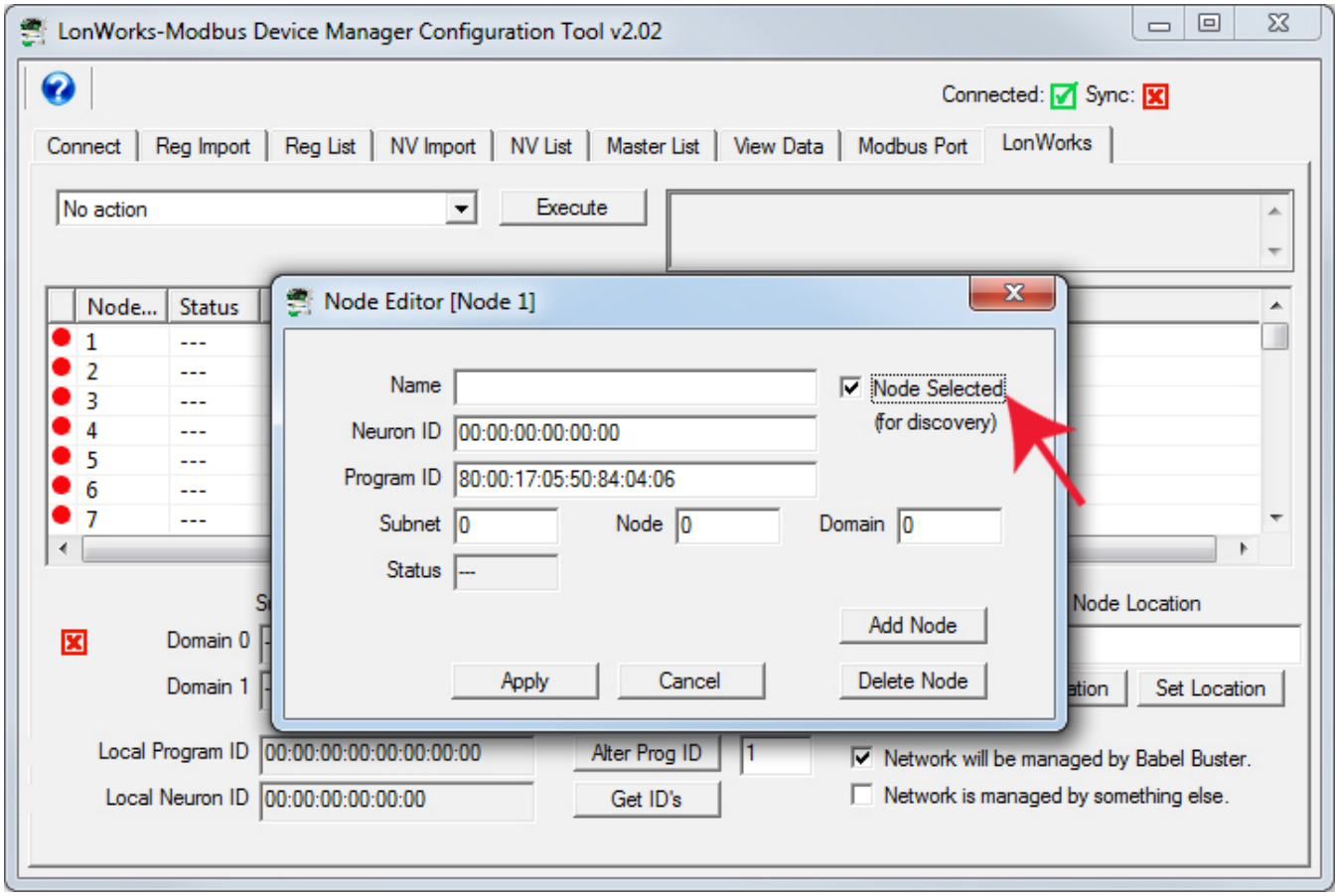
You need to obtain the Program ID of the device(s) of interest. If you imported an XIF file earlier, the program ID found in that file appears on the NV Import page. Simply copy that text from the NV Import page.



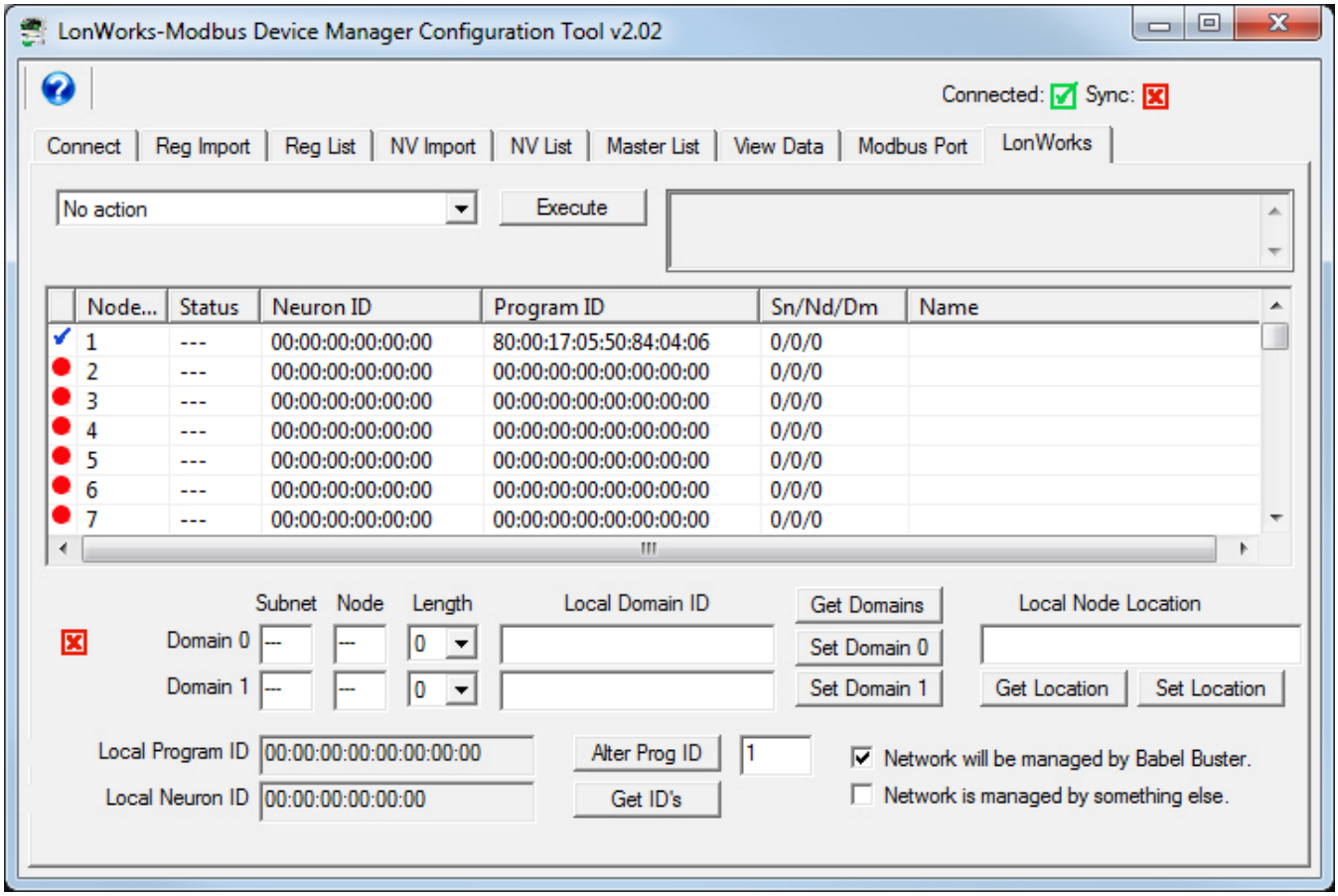
Double click an unused entry in the node table, and paste the Program ID into the Program ID window.



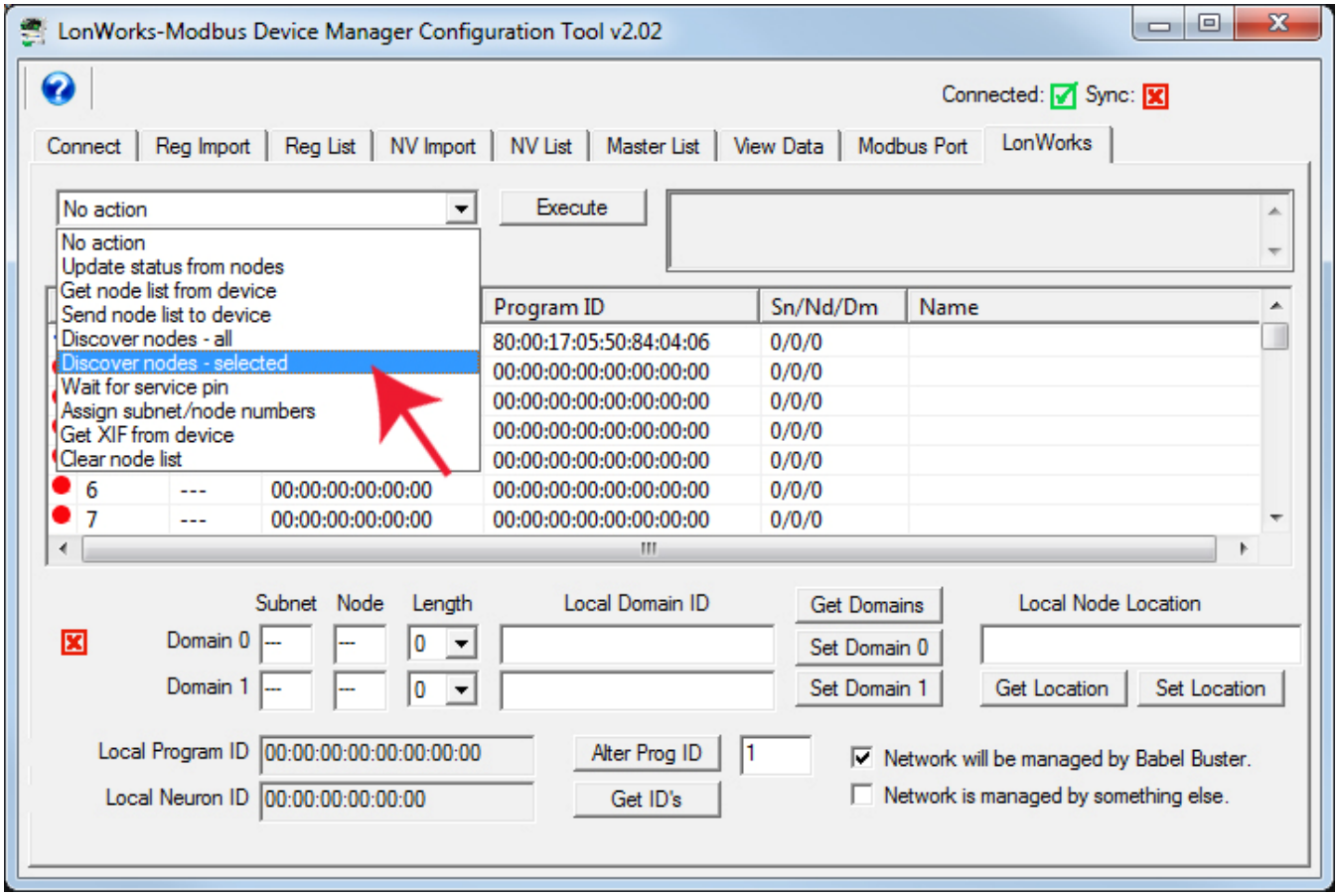
As with the other discovery methods, you also need to check 'Node Selected'. Then click Apply (not Add Node).



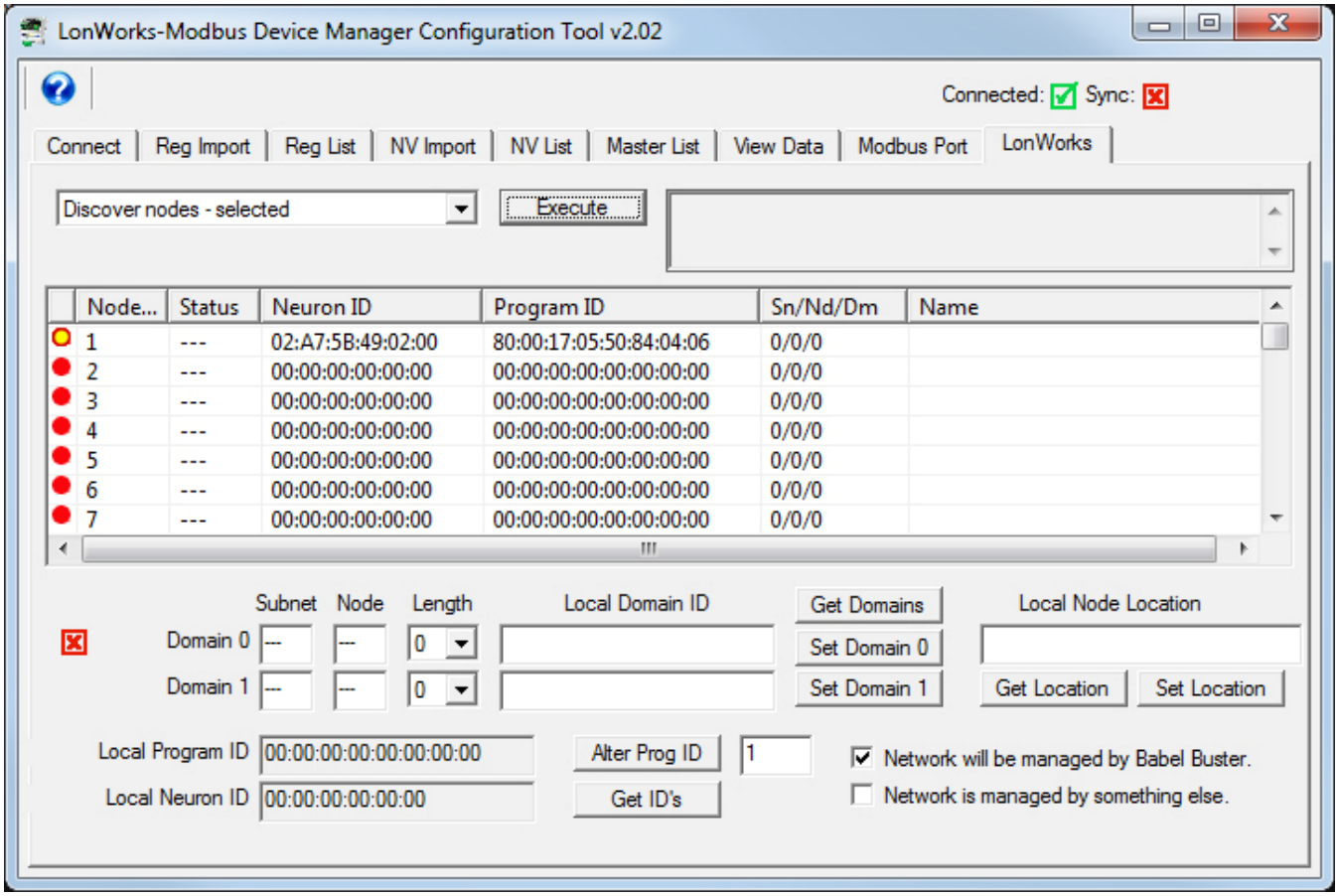
The node table now shows the check mark in the icon column as for other discover methods above, but now also shows a Program ID.



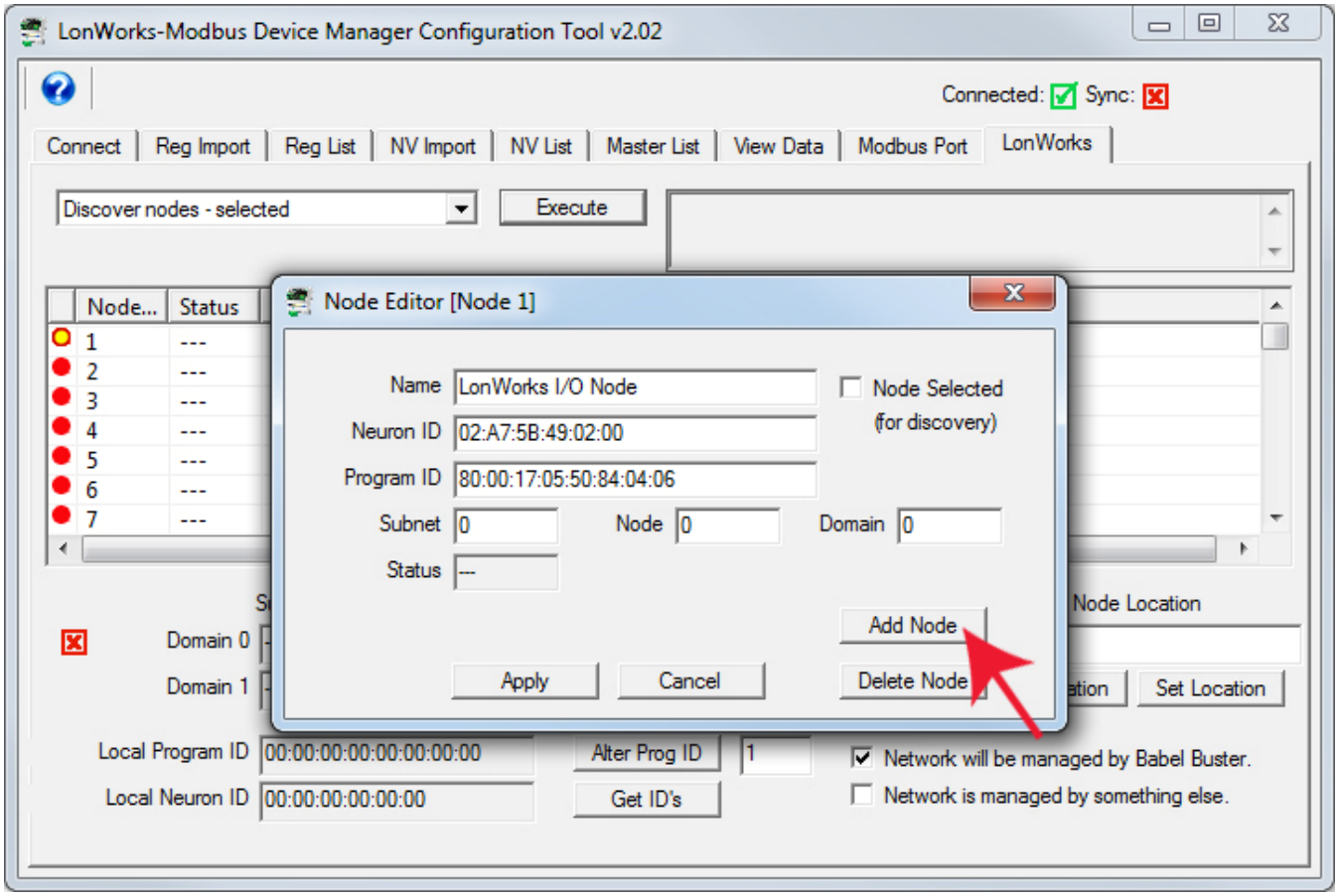
Next, select 'Discover nodes - selected' from the list and click Execute. Note that you can discover more than one node using this method. Simply use the Node Editor to enter the same Program ID in multiple table entries, and check 'Node Selected' for each.



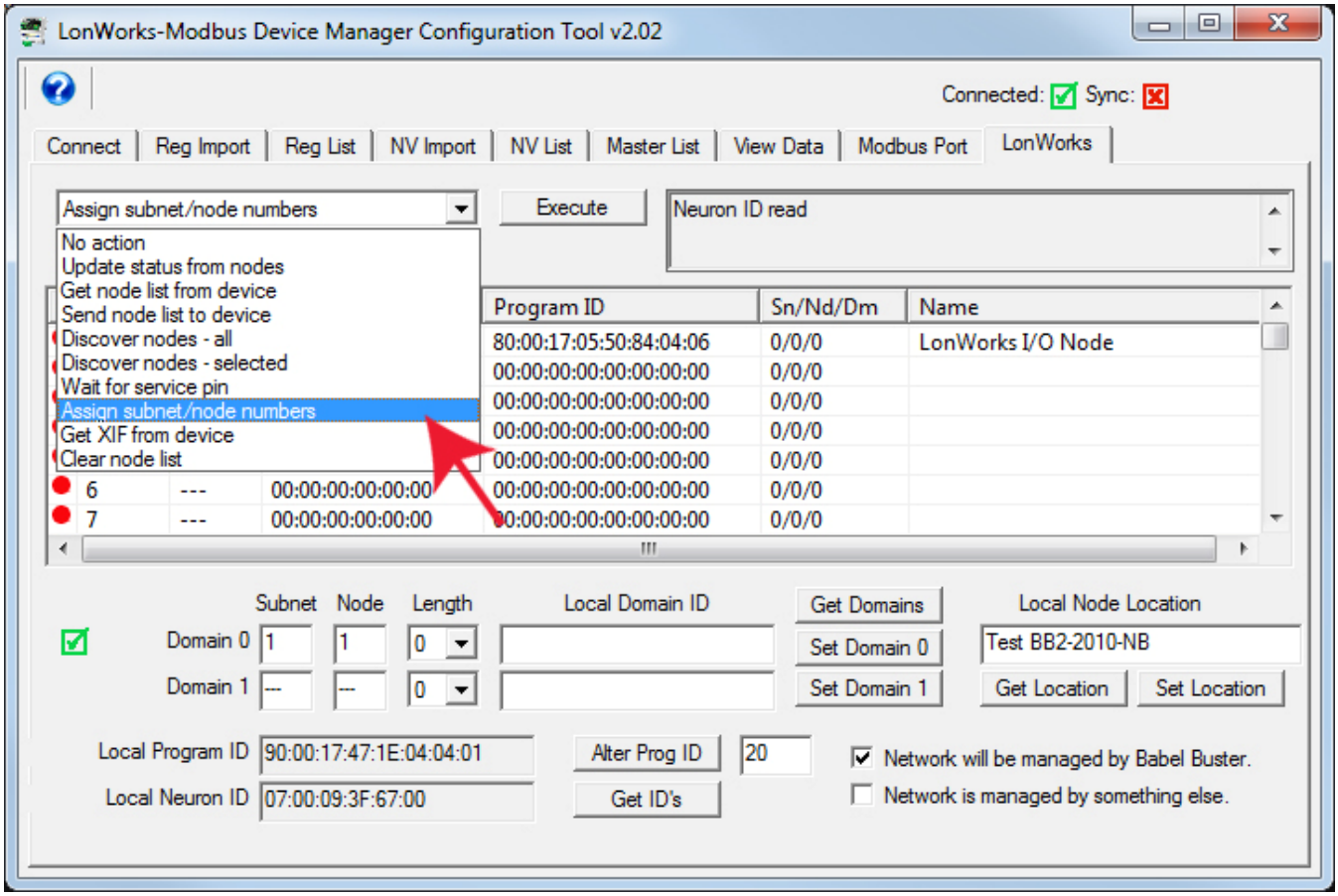
Refer to the discussion above about discovering all nodes if you're wondering what is going on at this point regarding how it discovers nodes. The only difference here is that the configuration tool is simply discarding replies from any nodes that do not match the Program ID provided. When the node is discovered, its icon will change to yellow with a red circle.



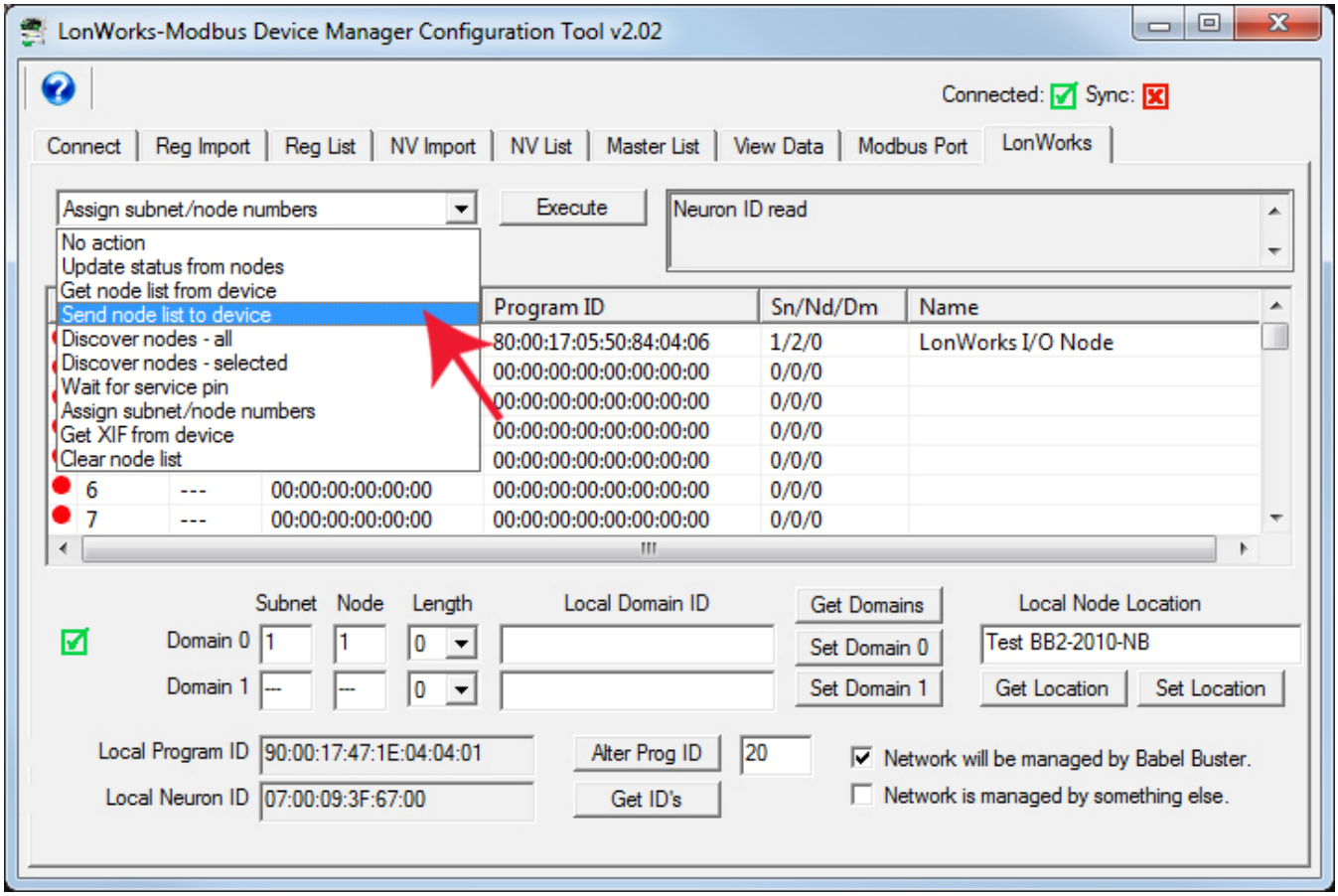
At this point, double click the node table entry to open the Node Editor. Provide a name and subnet/node if desired (see comments about this under discussion of all nodes above). Click Add Node.



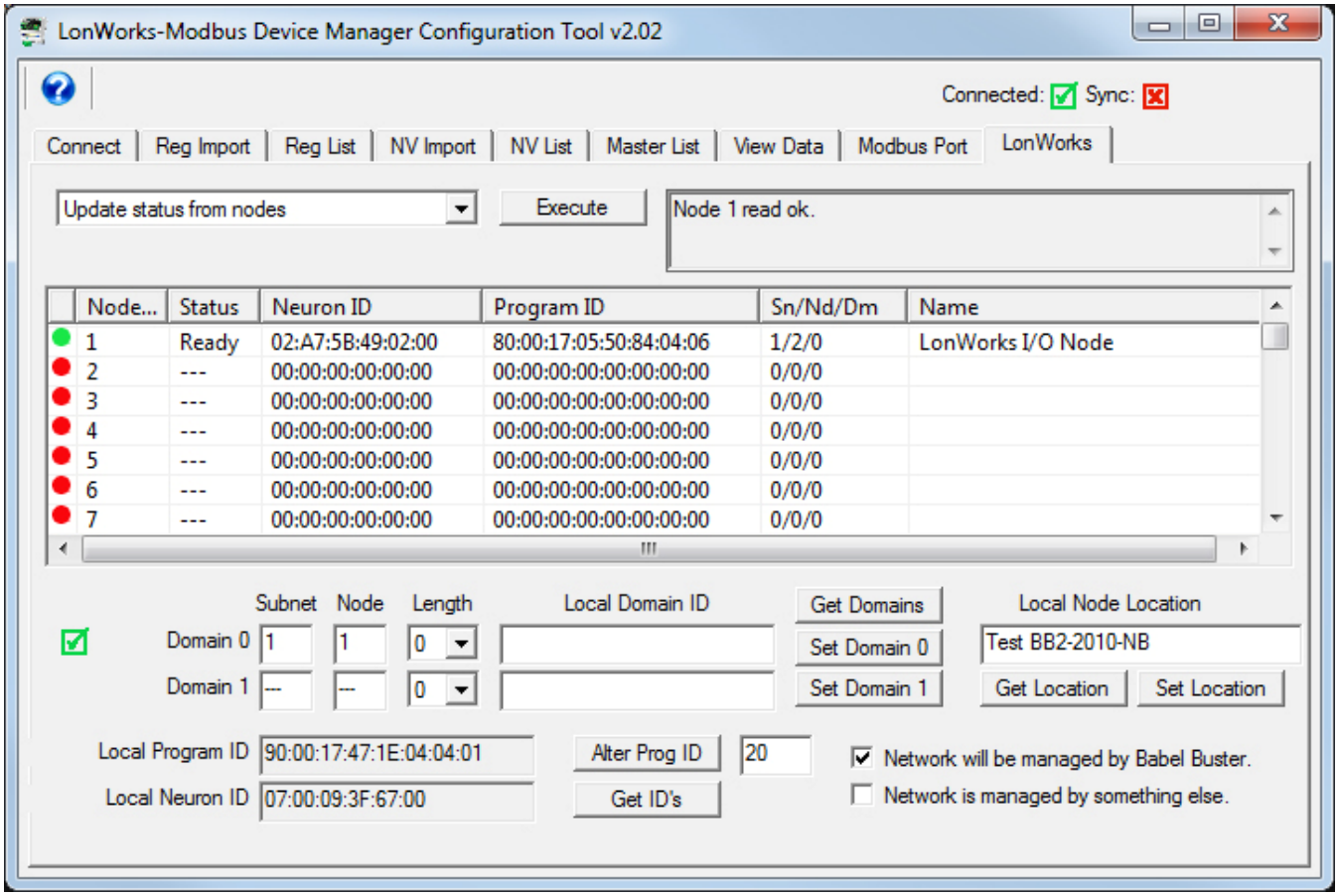
Most often, you will want to let the gateway assign suitable subnet/node numbers.



Send the node table to the gateway device as for the other methods above. Until you do this, the node is only known to the configuration tool. You need to populate the table in the device before the gateway will start to communicate with it.



Next, select 'Update status from nodes' and click Execute to confirm that this newly discovered node is now communicating successfully. At this point, you are ready to send the rest of the configuration from the NV List and Master List to the device.


LonWorks-Modbus Device Manager Configuration Tool v2.02
Connected:
Sync:
Connect
Reg Import
Reg List
NV Import
NV List
Master List
View Data
Modbus Port
LonWorks
Update status from nodes
Execute
Node 1 read ok.
Node...
Status
Neuron ID
Program ID
Sn/Nd/Dm
Name
1 Ready 02:A7:5B:49:02:00 80:00:17:05:50:84:04:06 1/2/0 LonWorks I/O Node
2 --- 00:00:00:00:00:00 00:00:00:00:00:00:00:00 0/0/0
3 --- 00:00:00:00:00:00 00:00:00:00:00:00:00:00 0/0/0
4 --- 00:00:00:00:00:00 00:00:00:00:00:00:00:00 0/0/0
5 --- 00:00:00:00:00:00 00:00:00:00:00:00:00:00 0/0/0
6 --- 00:00:00:00:00:00 00:00:00:00:00:00:00:00 0/0/0
7 --- 00:00:00:00:00:00 00:00:00:00:00:00:00:00 0/0/0
Subnet
Node
Length
Local Domain ID
Get Domains
Local Node Location
Domain 0
1
1
0
Set Domain 0
Test BB2-2010-NB
Domain 1

0
Set Domain 1
Get Location
Set Location
Local Program ID
90:00:17:47:1E:04:04:01
Alter Prog ID
20
Network will be managed by Babel Buster.
Local Neuron ID
07:00:09:3F:67:00
Get ID's
Network is managed by something else.

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'

usage - displays message indicating counts of objects and network variables currently used.
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

nf <n> - fetches whatever NV is defined at NV table map entry #n. View data using nvs <n>.
nu <n> <x1 … xN> - updates whatever NV is defined at NV table map entry #n using hex bytes x1..xN.

nvx <n> - display as XML the configuration found in NV table map entry #n.
ndevx <n> - displays node configuration information, 0=local/self, 1..max=remote nodes, XML

format

nvs <n> - display status, NV selector, and any pending diagnostic data for NV at table entry #n.
Display will be "S[] Sel=... Data: ..."
Sel= will be followed by 4-digit hex selector if it has been retrieved from the node
Data: will be followed by the data from the last "nf" command, cleared after displayed
Contents of S[] may be:
T - if tag timeout or message failed completion
N - if device not ready
E - if API returned an error code
F - if NV fetch is pending
U - if NV update is pending

nds <n> - shows node status for node table entry #n.
Display will be " [y] N: ... P ..." where Neuron ID follows N, program ID follows P.
Contents of [] may be:
S - if node has Neuron ID
D - set domain successful
Q - query domain successful
R - if node is considered 'ready'
W - if domain query came back with wrong subnet/node
F - failed domain set
N - failed domain query
E - if API reported an error

nid - displays Neuron ID
npid - displays device's Program ID
ninfo - displays node location configuration
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

Appendix B LonWorks Trouble Shooting

B.1 General Practice, LED Indicators

The LED indicators on the gateway device provide certain clues when things are not working right.

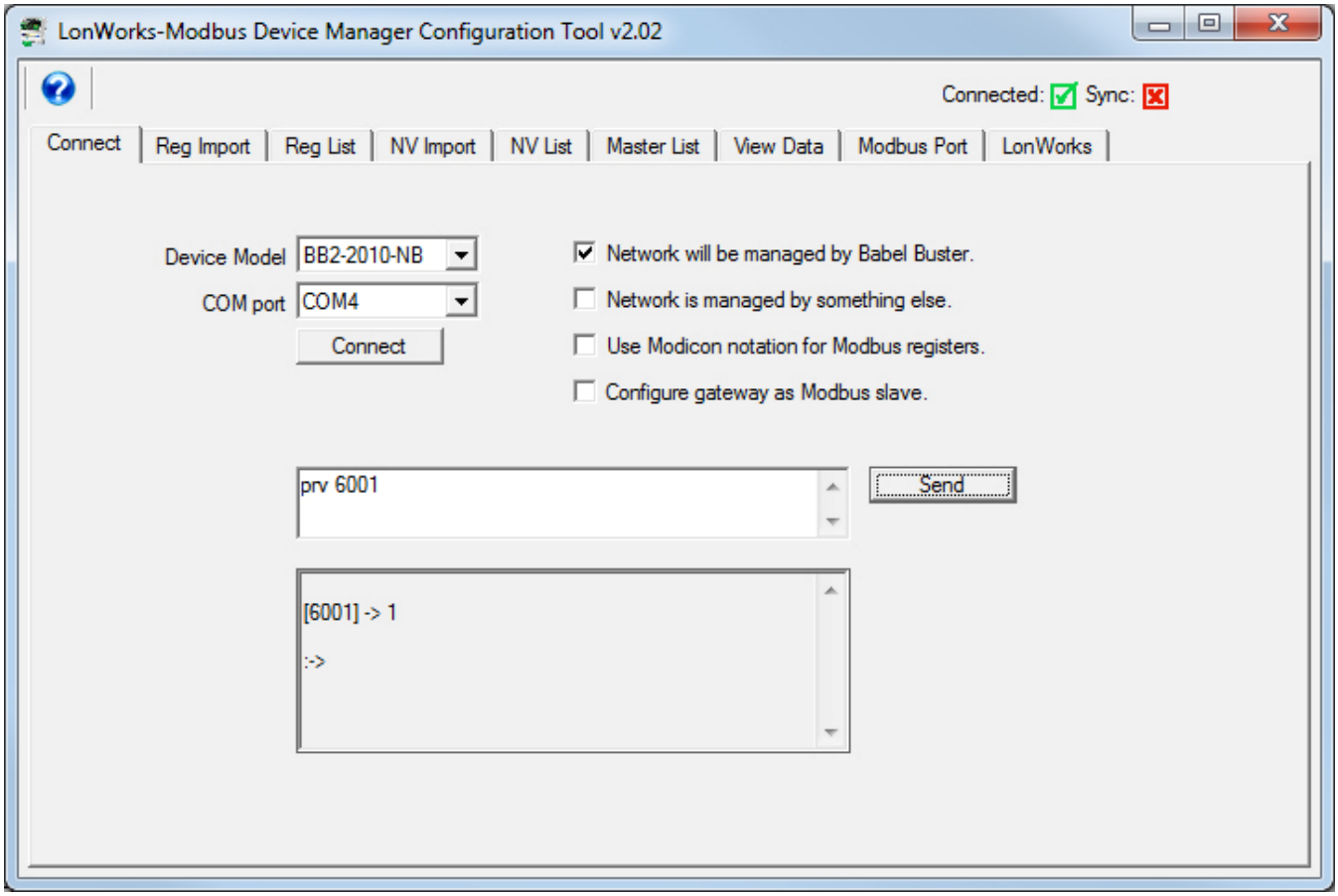
The LonWorks indicator LEDs display LonWorks network activity. Note, however, that activity of these LEDs is not only affected by configuration of the device, but by whether other devices are also communicating.

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	Yellow flash indicates NV update was sent by gateway, or node management message was sent. Green flash indicates NV fetch response was received by gateway or other action completed successfully.	Red solid on indicates Neuron chip is not running. Brief flash of red indicates error in processing NV request or node management request. Green indicates gateway's host processor is communicating with LonWorks Neuron chip.

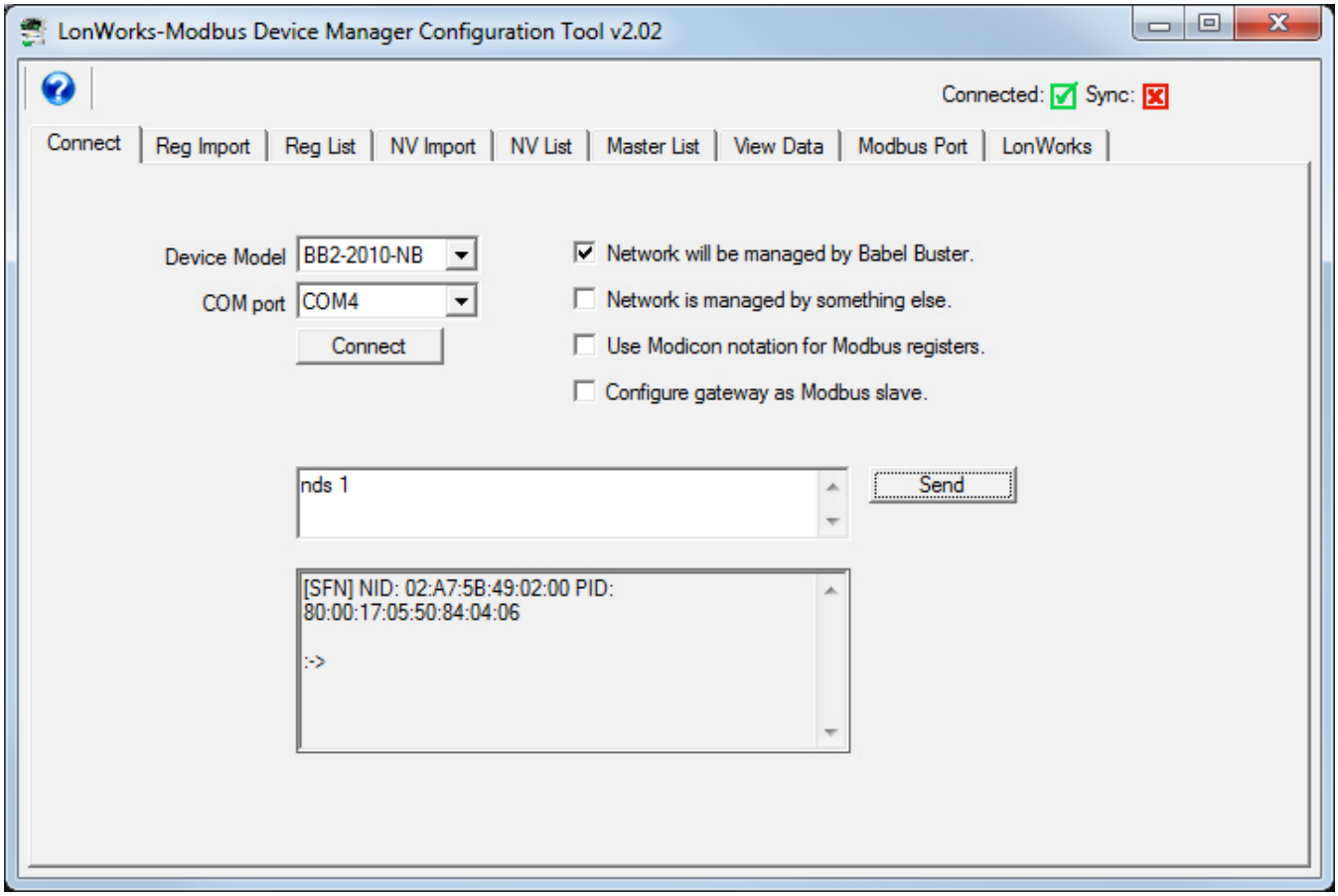
The "wink" behavior is invoked by sending a wink command to the BB2 gateway via the LonWorks network. This is generally just a diagnostic to see if you are successfully communicating with the device via LonWorks. Other than the few seconds it takes to execute the wink, the device will always be in "normal"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

B.2 Diagnostic Support

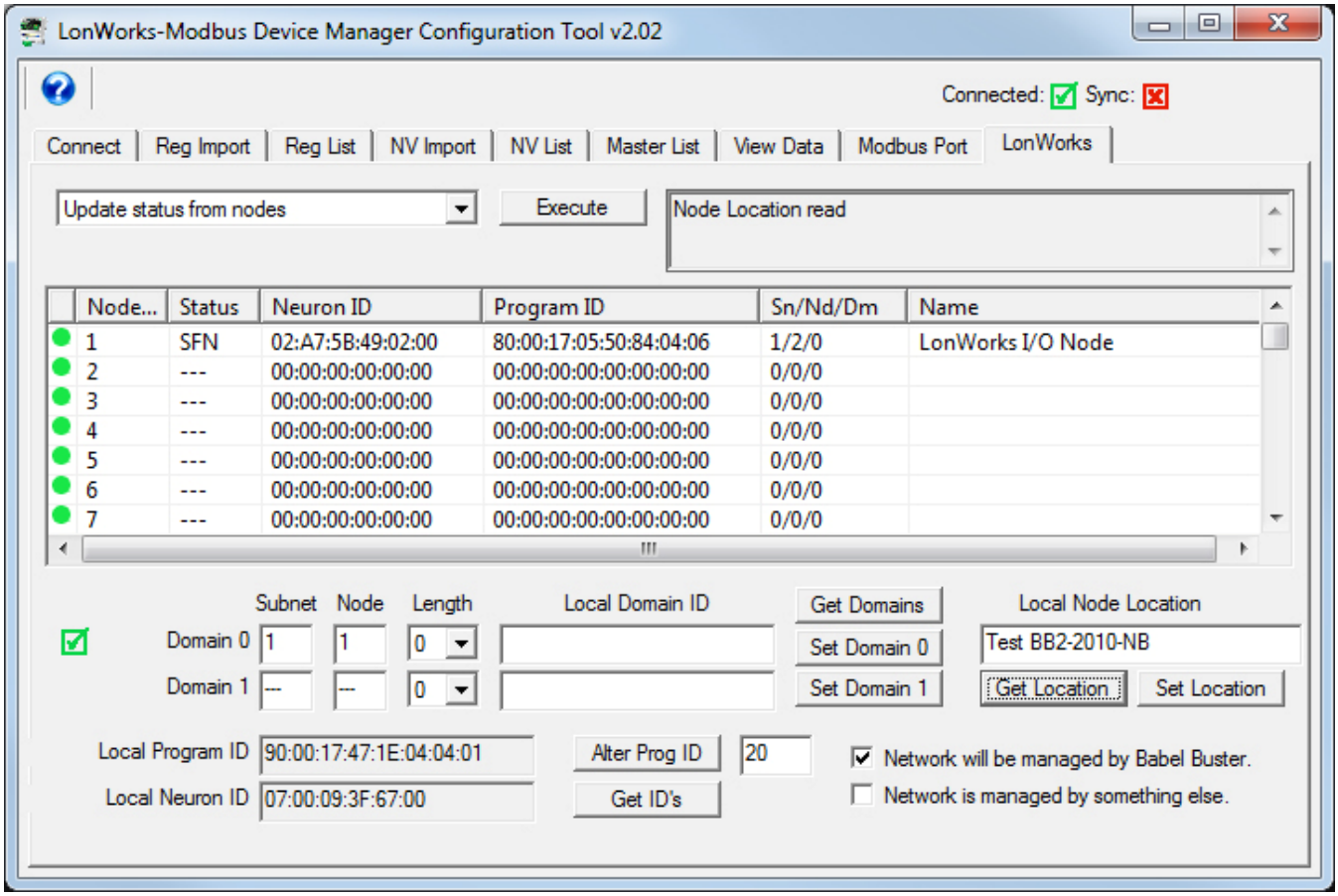
Diagnostic registers are available which allow Modbus to see if or when there are problems with a LonWorks device, or with specific Network Variables in that LonWorks device. Appendix E.2 shows the diagnostic registers available, which are also referred to as "virtual" registers since their content is constructed dynamically when you read them. You can also query these registers via the USB console commands (Appendix A). For example, virtual register 6001 will give you the status of LonWorks Node #1 (the #1 referring to list position on the LonWorks page). In the screen shot below, we can see that that gateway's attempt to set the domain for Node #1 has failed. If your Modbus master was reading holding register 6001, it would also see a value of 1, indicating this problem. A value of zero means no errors or problems.



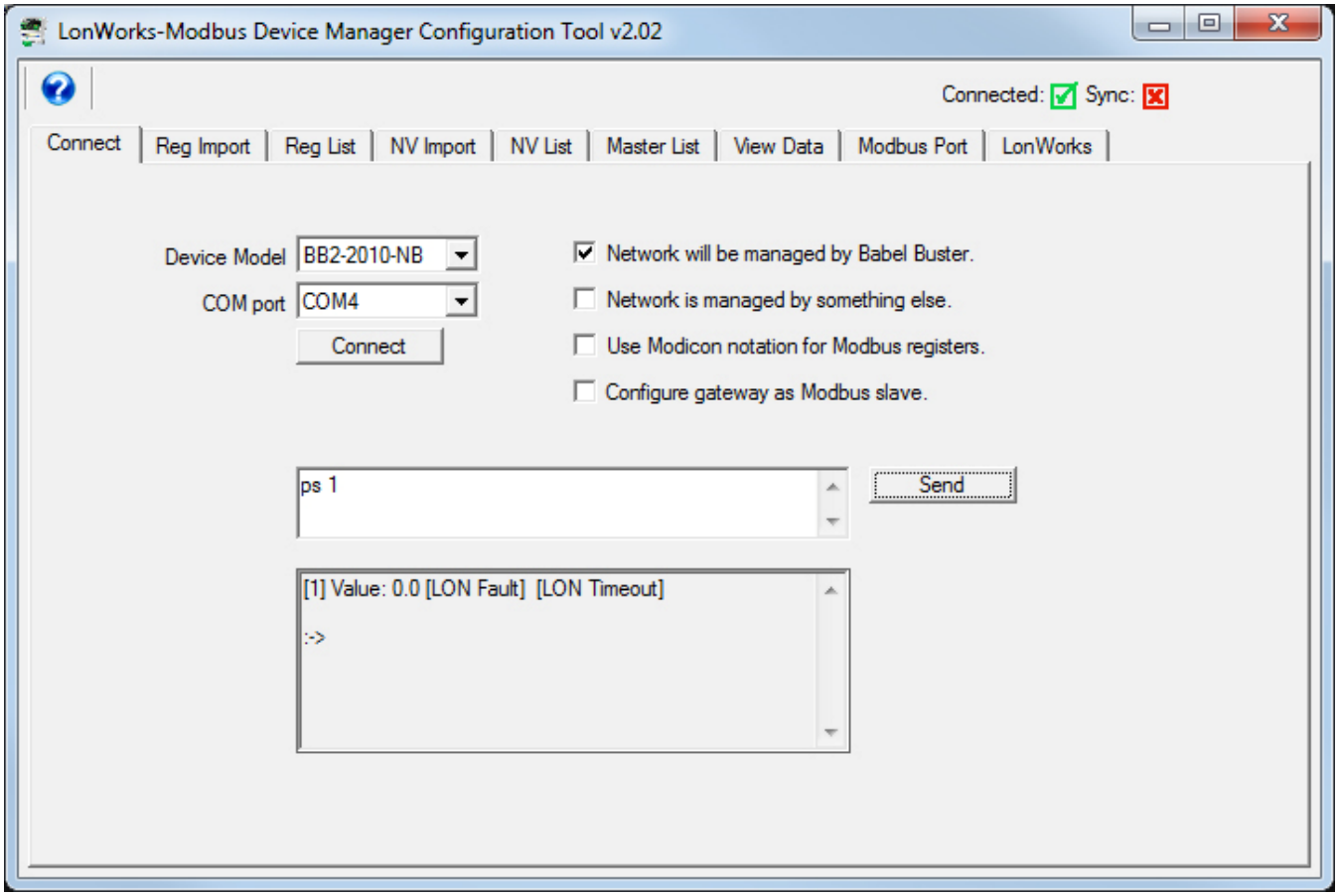
You can also use the "nds" command to query the status of device #1 as illustrated below.



Either of the above indications are displayed on the LonWorks page as illustrated below. Normally you would just use the "Update status from nodes" action on the LonWorks page. The diagnostic support is illustrated here to show that this same information is available to your Modbus master.

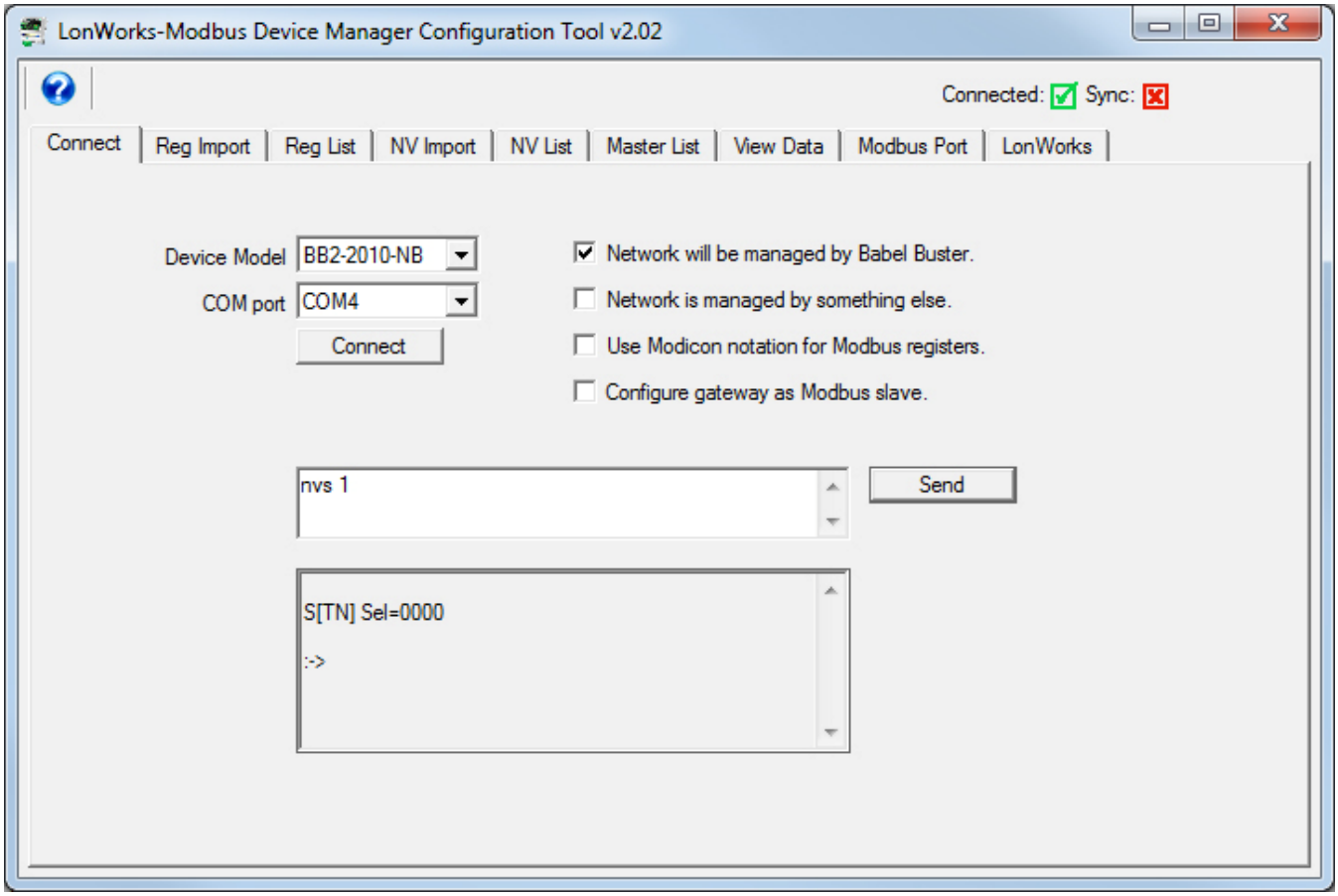


If the node status seems to be OK, indicating that the remote LonWorks device is responding, but a specific data point is in question, you can use the "ps" (point status) command to check the status of a specific data object as illustrated below.



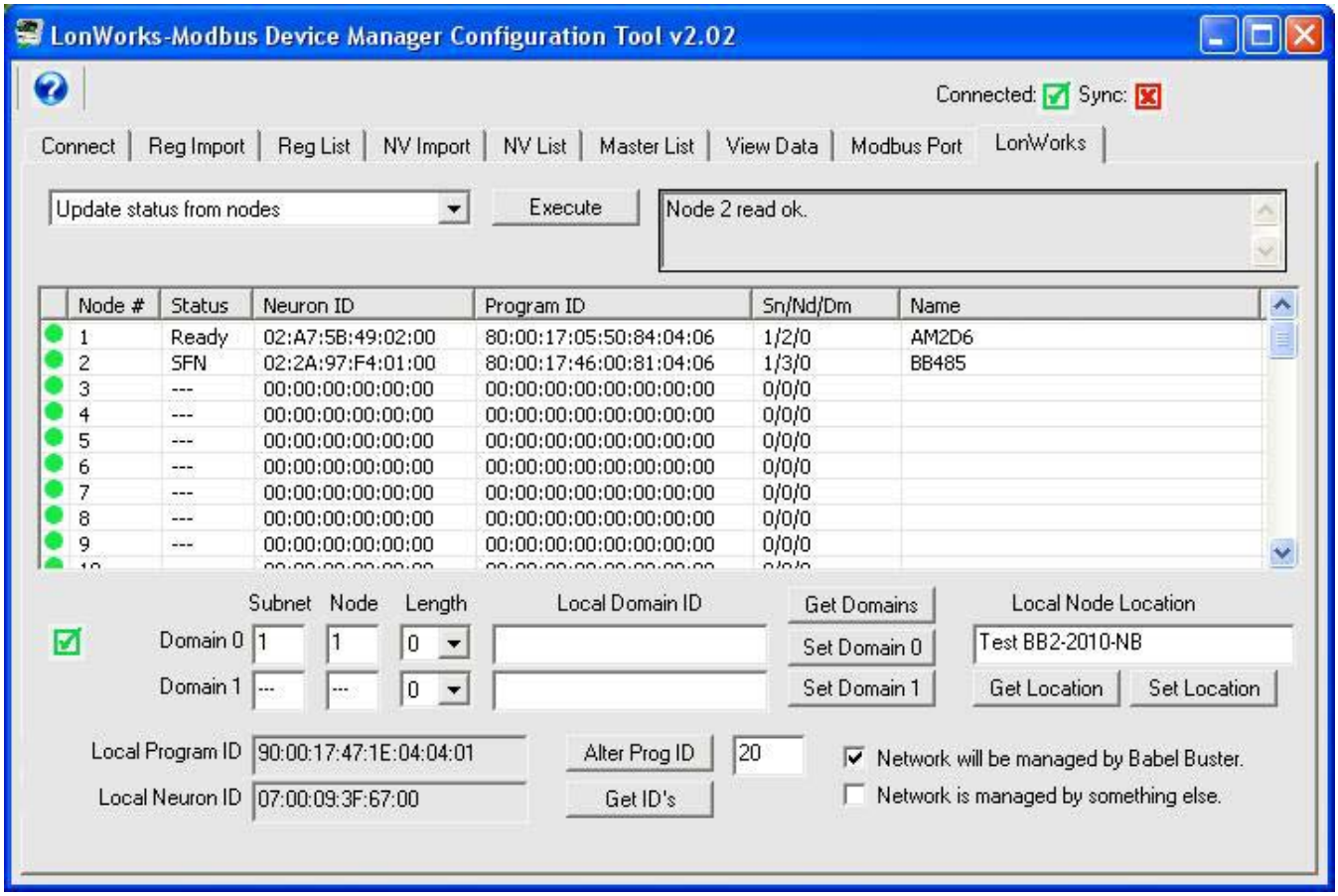
You can also query the status of a specific Network Variable map in the gateway as illustrated below. Refer to Appendix A for further description of these results.

Note that the status of individual Network Variables is accessible as Modbus registers (virtual registers) 7001-7300. These are described in Appendix E. This allows your Modbus master to check on specific variables. Note, however, that if your configuration is valid and the system has been initially operable, then the only status you would need to monitor longer term is device status. Once Network Variables have been found to be functional, the only latent failure will show up at the device level, and therefore checking individual Network Variables would be not only redundant, but more cumbersome since there are typically more variables than devices.



B.3 Node Status Not "Ready"

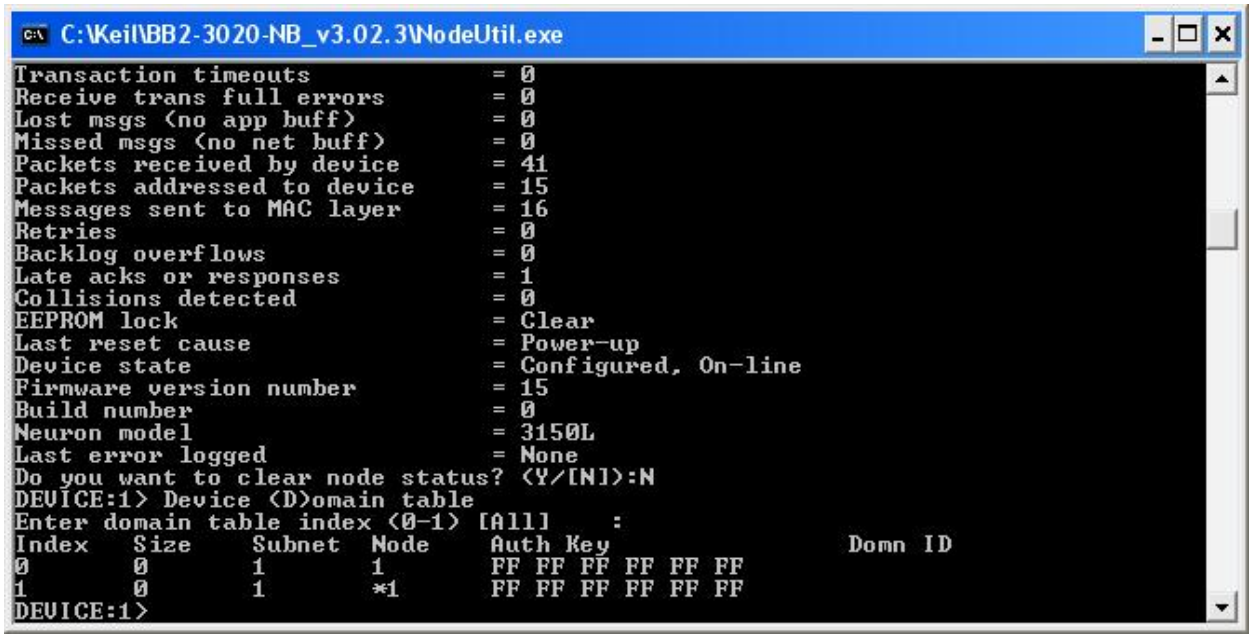
The following screen shot shows what you will typically see if a node on the network (in this example, node #2) is not "Ready" for the Babel Buster gateway to poll its network variables:



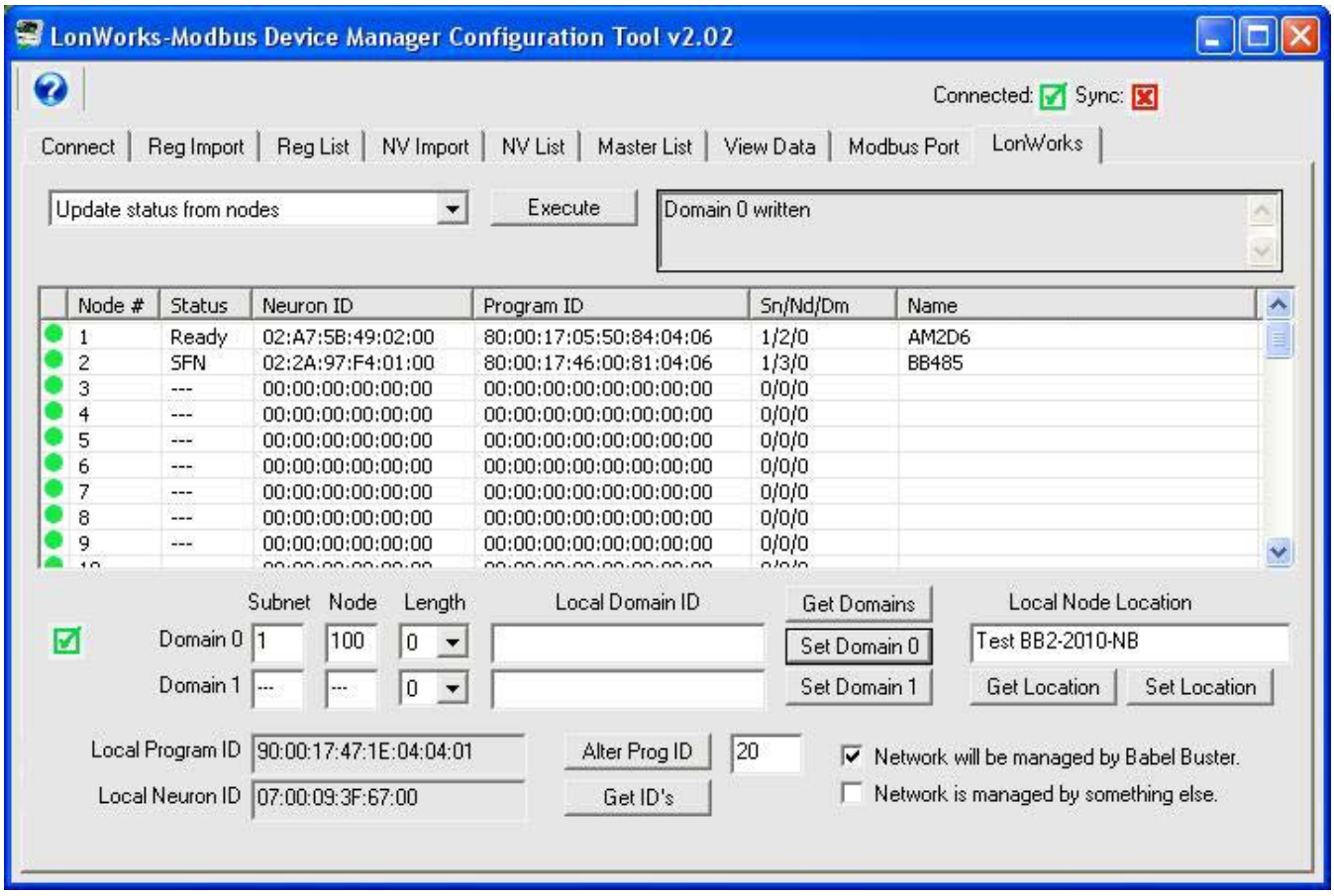
The above instance shows that node #2 has reported its Neuron ID and Program ID. The configuration tool wants to set this node to subnet 1, node number 3. However, it has not been successful as indicated by the "SFN" status.

The above instance occurred when the device at node #2 in the table happened to already be assigned to the same subnet and node number as the Babel Buster gateway itself. As a result, responses from node #2 will not be received. In fact, they will not be sent by node #2 because it thinks it should be sending the response to subnet 1, node 1, which is its own address, and it will therefore discard the transmission to itself.

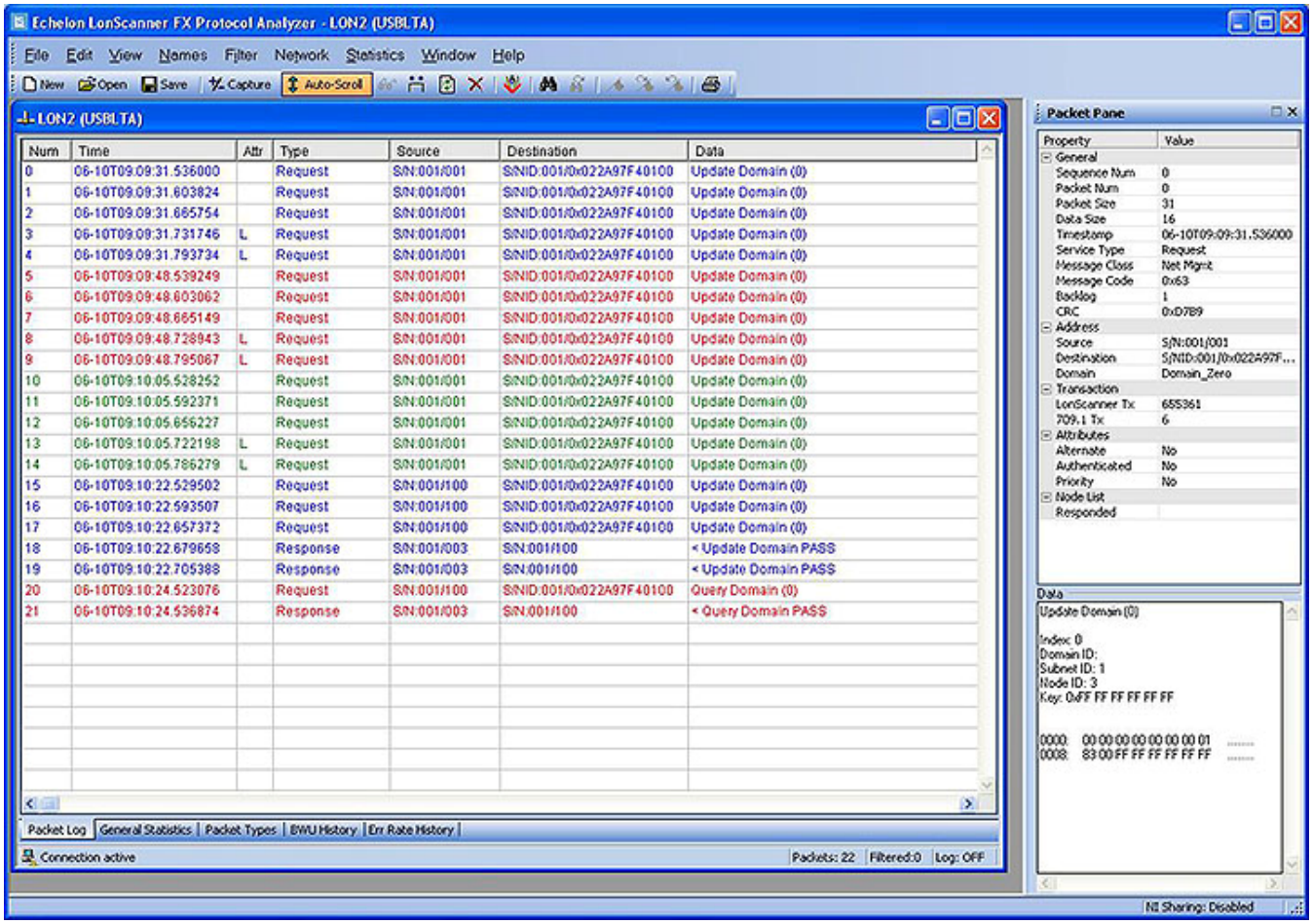
If you happened to have a LonWorks network interface and the Nodeutil program handy, connecting to node #2 (from table above) and checking its domain table will provide the results illustrated here:



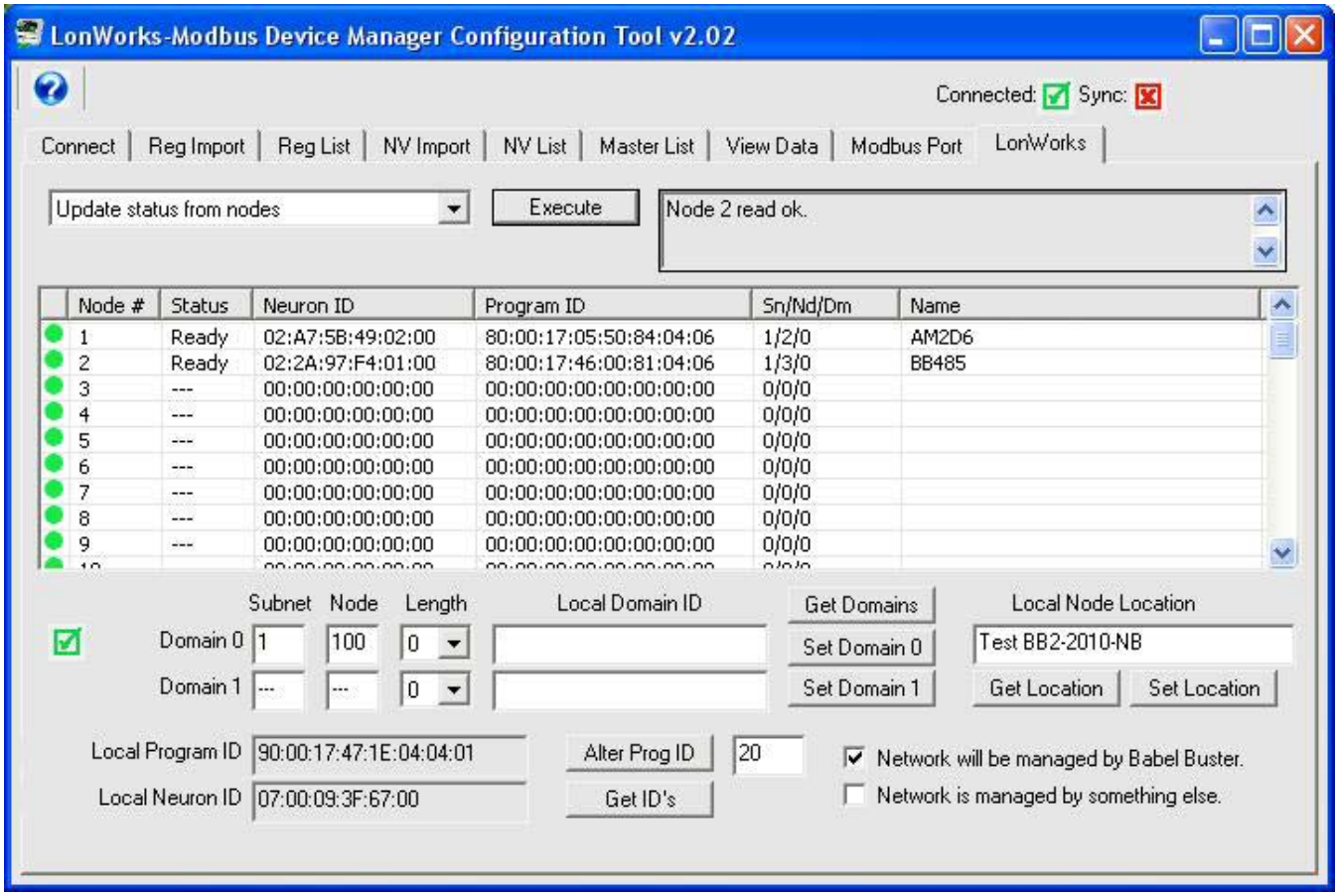
The resolution in this case is to simply change the node number of the Babel Buster gateway to something else as illustrated below.



The activity on the network during this exercise using LonScanner is illustrated here:



The LonScanner network capature illustrates the non-communicating state through record 14, then the node number was changed, and communication is successful. The resulting screen shot of the LonWorks page is shown below.



Nobody expects you to be using Nodeutil and LonScanner to decipher this problem. It is illustrated here to demonstrate the problem and solution. Therefore, in practice, if the Babel Buster gateway is managing the network ("Network will be managed by Babel Buster") but you still cannot communicate with the node, then simply try changing the node number of the gateway itself. Do this by changing the node number on the LonWorks page, and then click Set Domain.

B.4 Cannot Discover Node

Node discovery requires being able to receive responses from devices on the same domain as the gateway. The Babel Buster gateway will broadcast a message telling all devices to "respond to query". Devices will report their identity. However, if the device is on a domain different than the gateway, the responses will not be received by the gateway.

You have two choices for resolving this matter: (a) Use other tools (e.g. Nodeutil or other network management tool) to find out what the domain of the device is; (b) Use the service pin method to locate the device and make it known to the gateway.

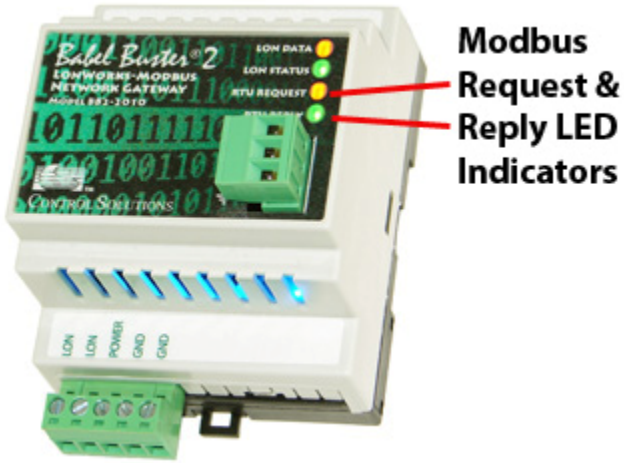
If the gateway is being used on a managed network, i.e., network managed by something other than Babel Buster, then your only choice is to find out (from the network management tool) what domain the devices are set to, and change the gateway's domain to match that domain. The service pin method will locate the device, but the gateway will still be unable to communicate because it has been instructed to not make any changes to device domain tables (implied in configuring the gateway for use on a network managed by something other than Babel Buster).

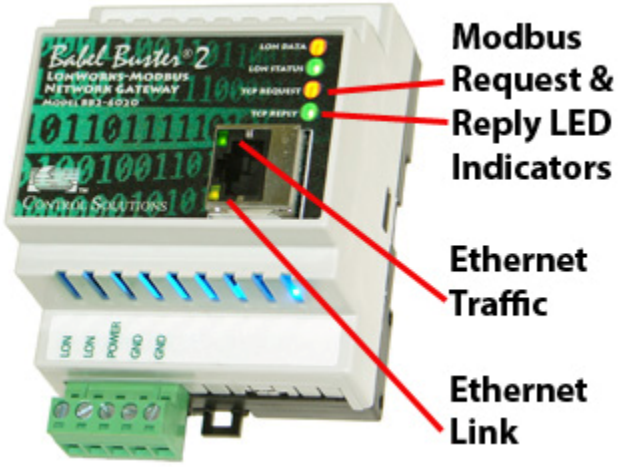
Appendix C Modbus Trouble Shooting

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 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.	

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.

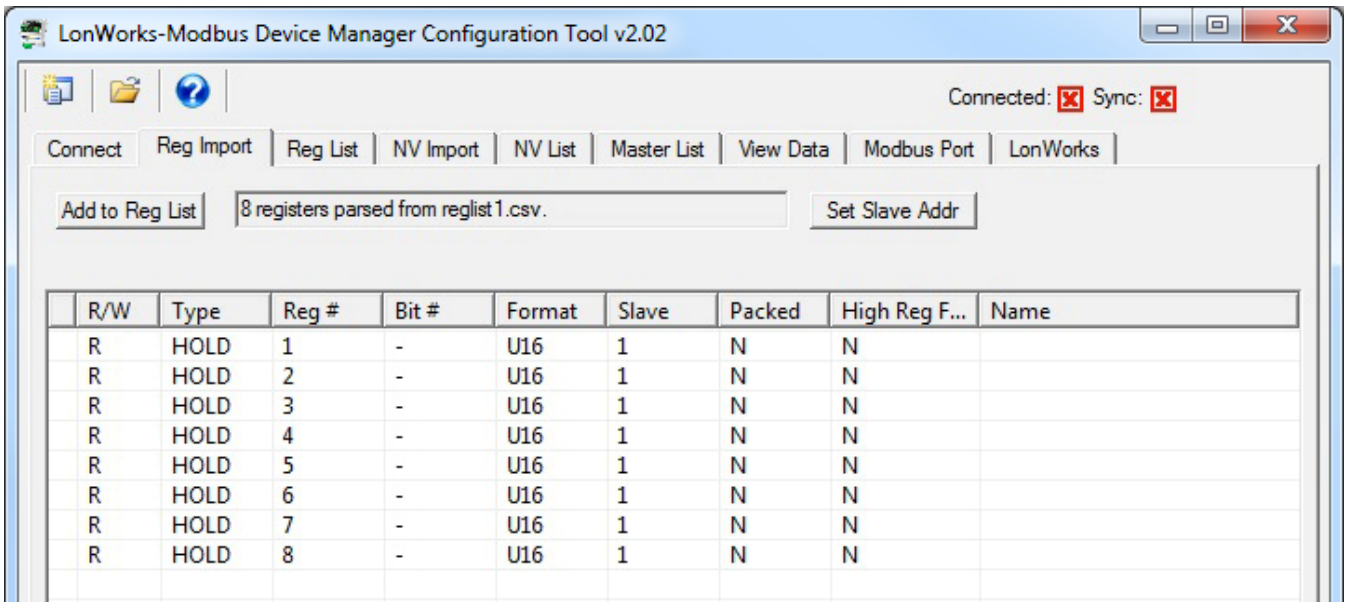
Label	Content	What it means
MODICON	Register number e.g. 40001	If MODICON is used, then do not include REG or TYPE. Register numbers should use Modicon notation, such as 40001 for holding register number 1.
REG	Register number e.g. 1	REG and TYPE are used together instead of MODICON when register numbers 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.

REG 1 2 3 4 5 6 7 8	The simplest CSV file for register import is just a list of numbers, optionally having a header line with just REG. 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). Importing this CSV results in the following screen shot.



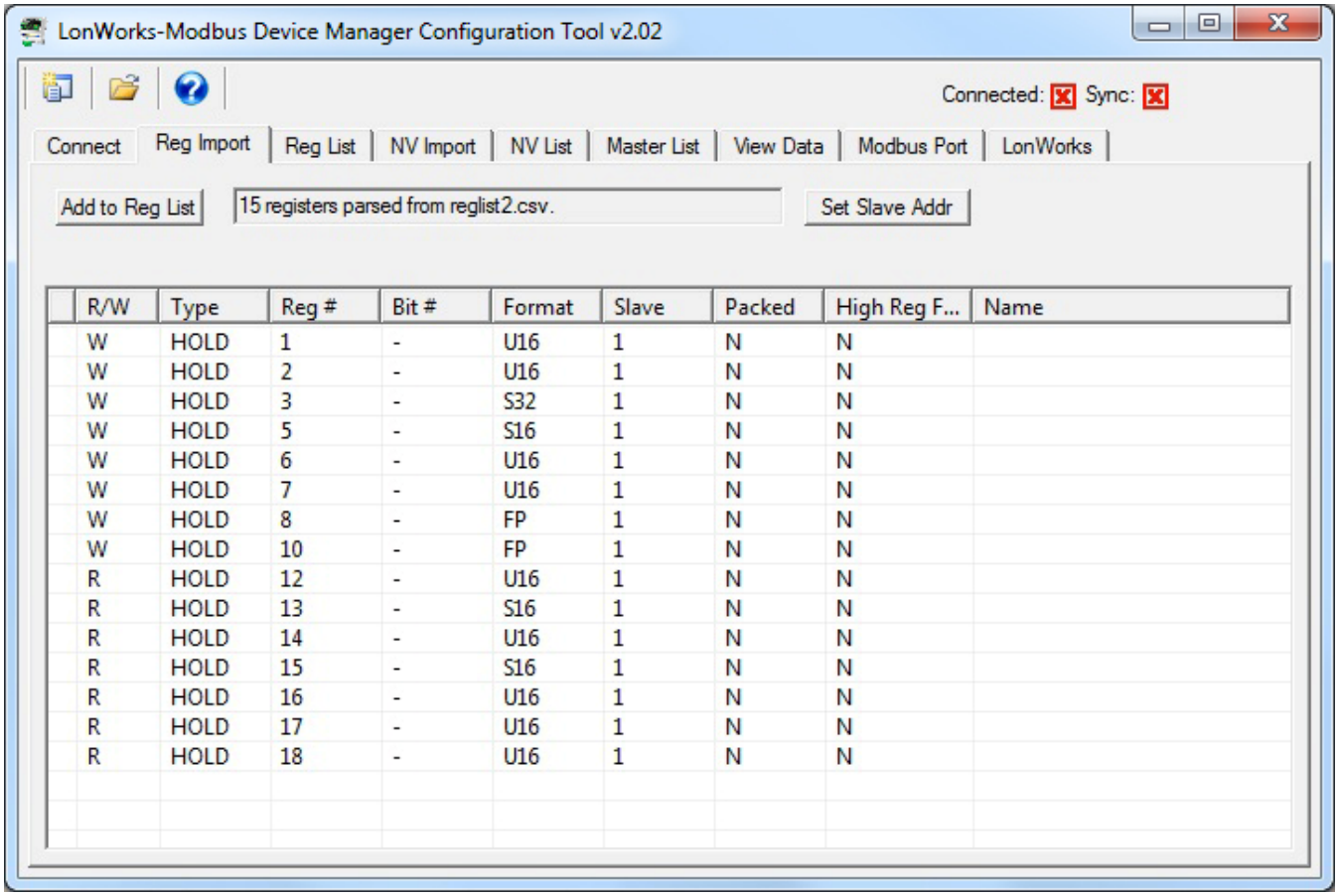
RW,REG,TYPE,FORMAT,SLAVE,BITNUMBER,PACKED,HIGHREGFIRST	A more complex register list using

```
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
```

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.



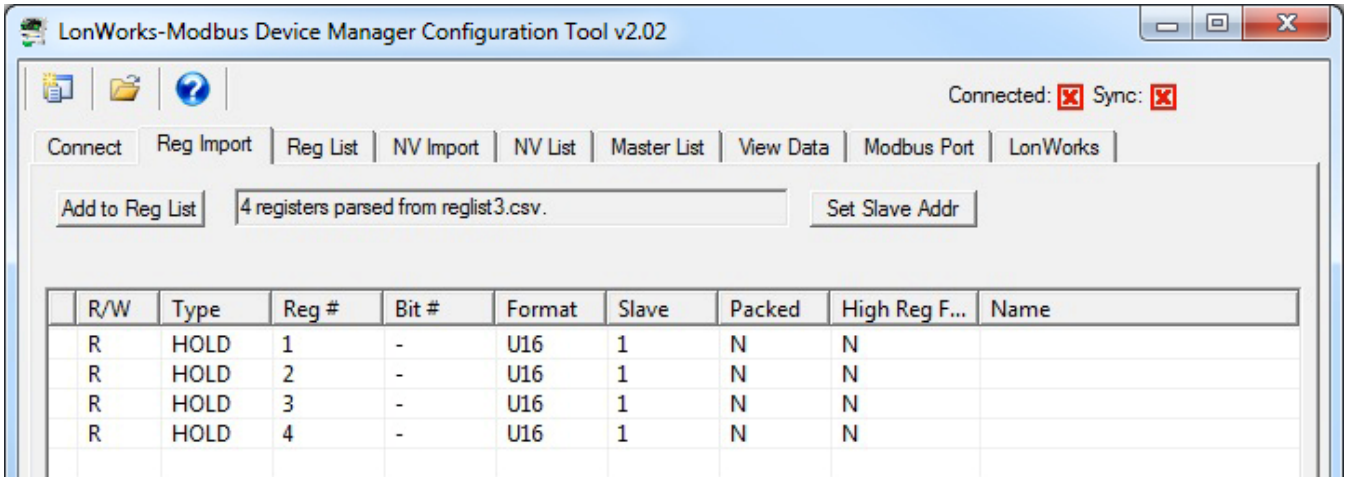
R/W	Type	Reg #	Bit #	Format	Slave	Packed	High Reg F...	Name
W	HOLD	1	-	U16	1	N	N	
W	HOLD	2	-	U16	1	N	N	
W	HOLD	3	-	S32	1	N	N	
W	HOLD	5	-	S16	1	N	N	
W	HOLD	6	-	U16	1	N	N	
W	HOLD	7	-	U16	1	N	N	
W	HOLD	8	-	FP	1	N	N	
W	HOLD	10	-	FP	1	N	N	
R	HOLD	12	-	U16	1	N	N	
R	HOLD	13	-	S16	1	N	N	
R	HOLD	14	-	U16	1	N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

```
MODICON
40001
40002
40003
40004
```

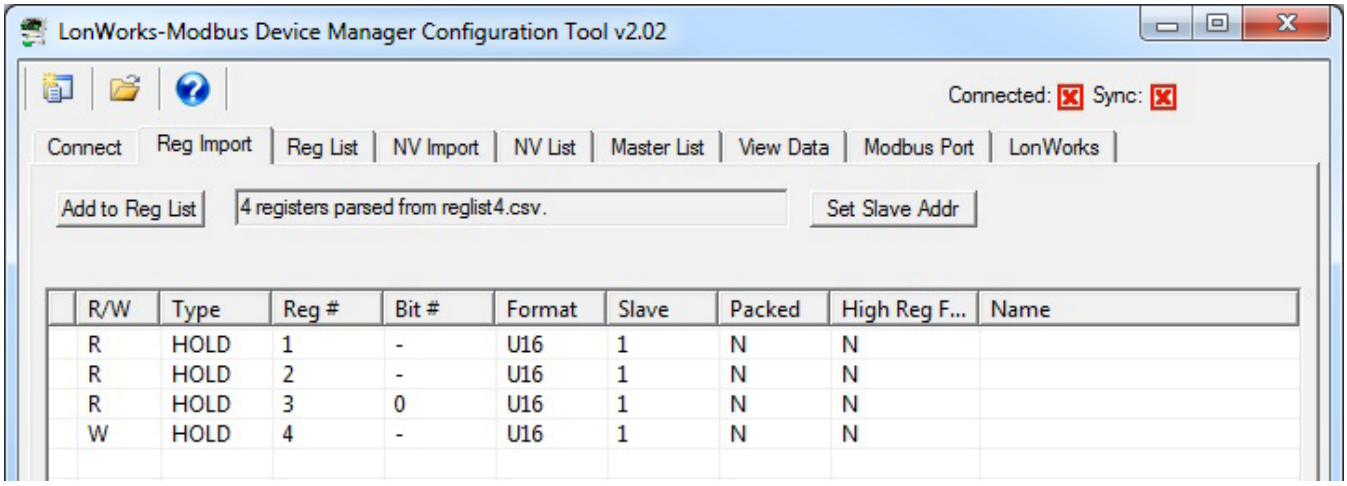
The other form of simplest CSV file for register import is just a list of numbers, optionally having a header line with just MODICON.

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.

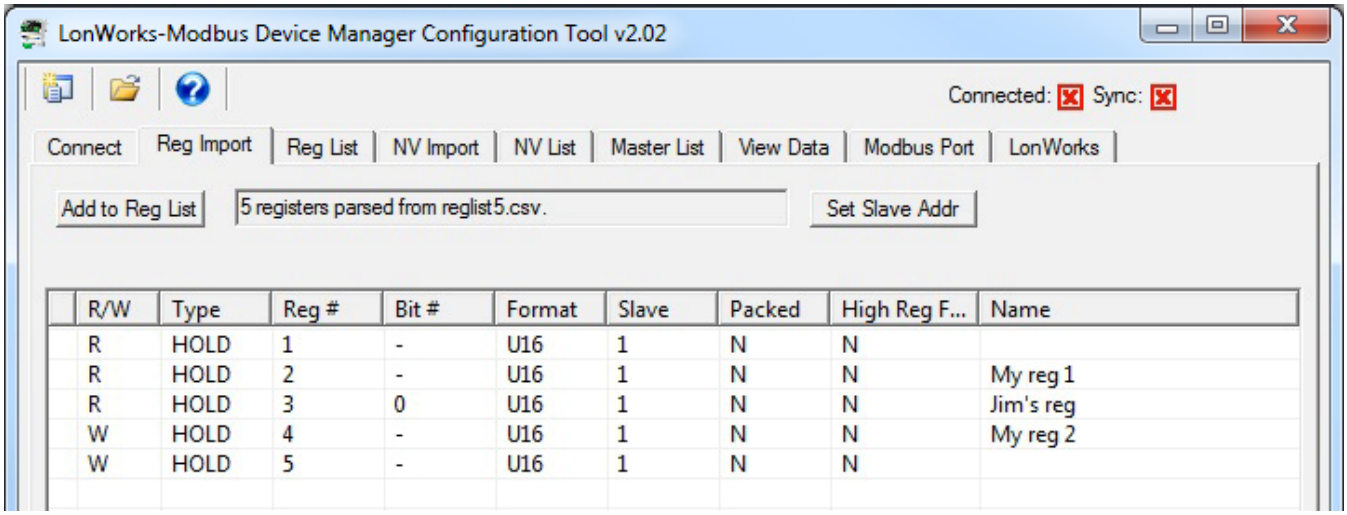
	Importing this CSV results in the following screen shot.



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 me 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.



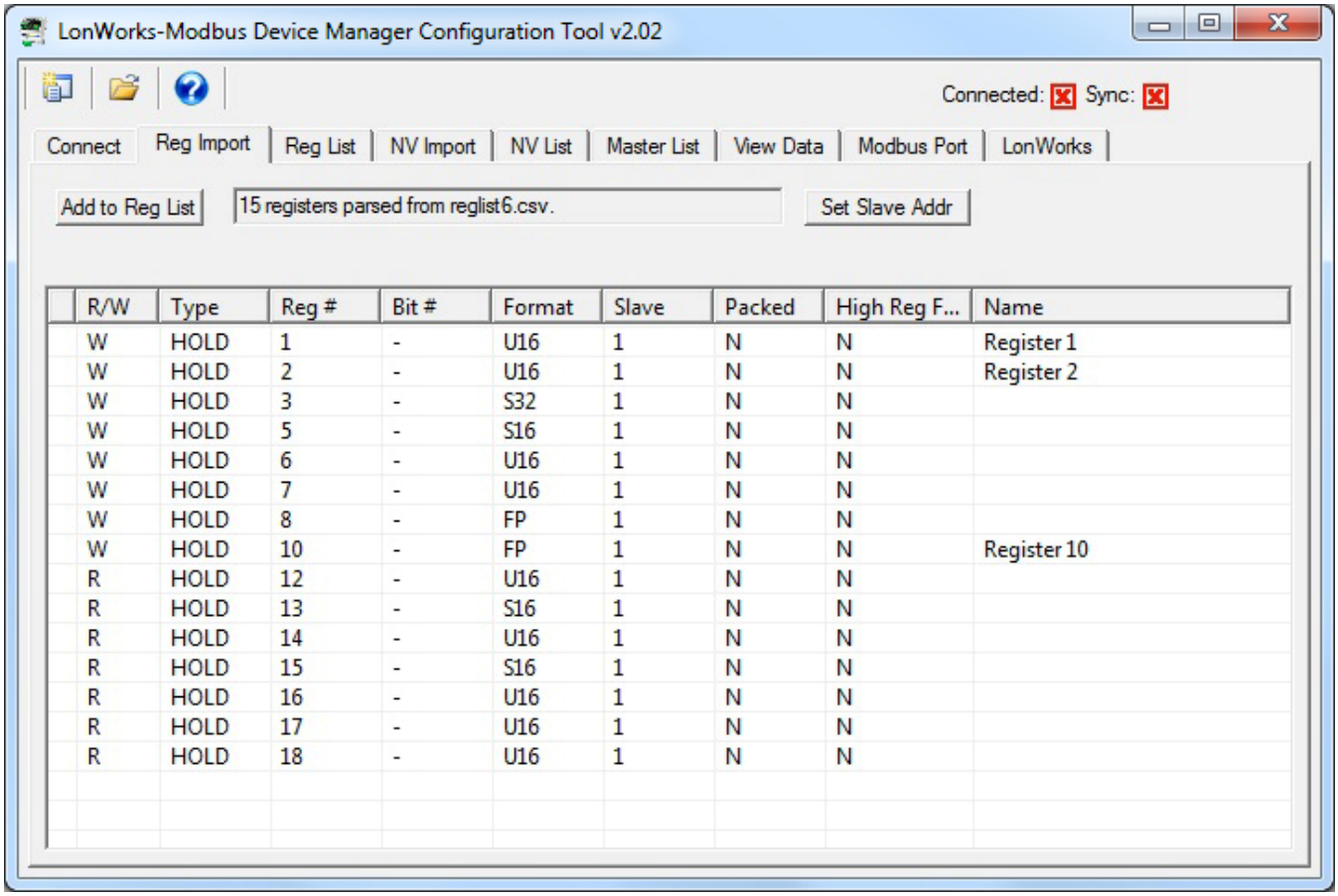
MODICON,BITNUMBER,RW,NAME 40001,,R 40002,,R,"My reg 1" 40003,0,R,Jim's reg 40004,,W,'My reg 2' 40005,,W	An optional register name column is added to the CSV in this example. Importing this CSV results in the following screen shot.




```
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.



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 - Data Objects

The following chart shows the available Modbus registers. Data objects are typically treated as holding registers. The same objects are also accessible as input registers, discrete inputs, and coils.

Data objects accessed as holding registers registers				
Modicon Std	Modicon Extd	Type	Std. Reg. No.	Description
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
Data objects accessed via Modbus function codes other than holding registers				
Modicon Std	Modicon Extd	Type	Std. Reg. No.	Description
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

E.2 Modbus Registers - Diagnostic Support

The following chart shows Modbus registers available for diagnostic support.

Diagnostic registers				
Modicon Std	Modicon Extd	Type	Std. Reg. No.	Description
				LonWorks device status code: 0 = No errors 1 = Failed to set domain in remote

46001 - 46050	406001 - 406050	Holding	6001 - 6050	LonWorks device 2 = Failed to query domain in remote LonWorks device 3 = Failed to poll/update NV in remote LonWorks device 4 = LonWorks device responded with wrong subnet/node 5 = Gateway detected a LonWorks API error
47001 - 47300	407001 - 407300	Holding	7001 - 7300	LonWorks Network Variable status code: 0 = No errors 1 = No response from remote LonWorks device 2 = LonWorks device is not ready (check device status) 3 = Gateway detected a LonWorks API error
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.

Appendix F LonWorks CSV Network Variable List Format

F.1 Data Labels on Header Line

The required format for importing LonWorks network variable 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.

The available column labels are outlined in the table below. Unlike the BACnet object CSV import where most columns are optional, the NV import requires all but the name in order to be functional. 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. Each line of data must contain as many elements as there are in the header line.

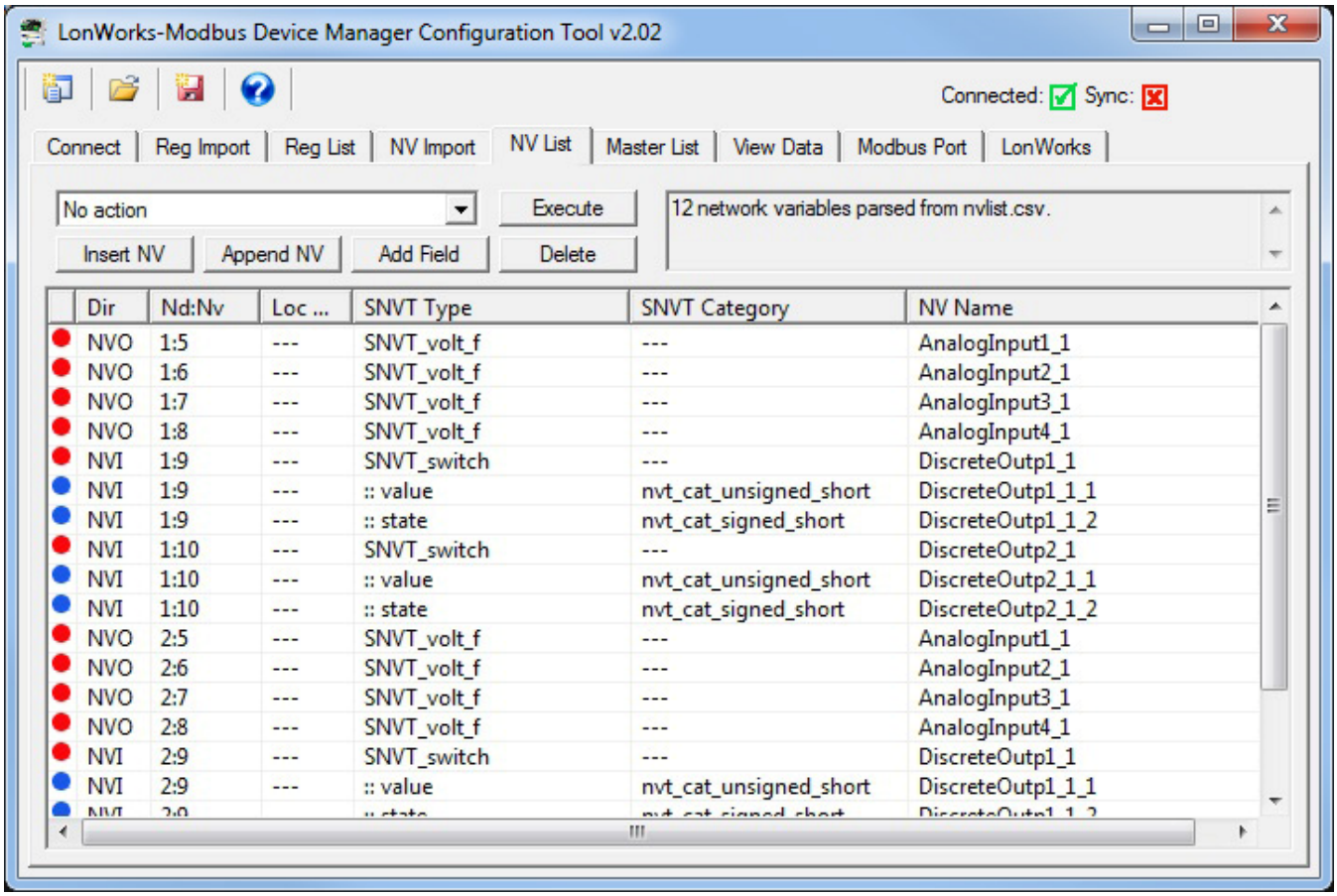
Label	Content	What it means
NODE	1..50	This number will correspond to the node number in the gateway's node table. The node table will contain addressing information such as the target device's Neuron ID, subnet and node numbers, and domain table index.
NVINDEX	0..16383	The network variable index identifies the network variable in the target device. You would obtain this information from the manufacturer of the device, or from the XIF file for that device.
DIR	1=NVI at target, 0=NVO at target	Identifies whether the network variable of interest is an NVI (network variable input) or NVO (network variable output) at the target device. The gateway can read from an NVO, and optionally write to an NVI. The gateway cannot write to an NVO in the target device.
SERVICE	Either R or W (or W+) (default is R)	Enter R to read the object, or W to write it. If omitted, defaults to R. Use W+ for write on Update.
SNVT	SNVT index 1..177 (zero if UNVT)	Enter the LonMark SNVT Index if the NV is a standard LonMark variable type. If it is manufacturer defined (UNVT), enter zero here.
UNVTSIZE	UNVT size 1..31 (zero if SNVT is valid non-zero index)	If SNVT is zero, then a non-zero number from 1 to 31 must be entered here to tell the gateway what size variable to expect.
NAME	Any unique character string up to 20 characters	Enter an alphanumeric name for the NV map. This name is for reference only, and does not affect the names of BACnet objects.

F.2 Example CSV File and Import

The following example illustrates the CSV file format and resulting screen screen shot following import.

```
NODE,NVINDEX,DIR,SERVICE,SNVT,UNVTSIZE,NAME
1,5,0,R,66,0,AnalogInput1_1
1,6,0,R,66,0,AnalogInput2_1
1,7,0,R,66,0,AnalogInput3_1
1,8,0,R,66,0,AnalogInput4_1
1,9,1,W+,95,0,DiscreteOutp1_1
1,10,1,W+,95,0,DiscreteOutp2_1
2,5,0,R,66,0,AnalogInput1_1
2,6,0,R,66,0,AnalogInput2_1
2,7,0,R,66,0,AnalogInput3_1
2,8,0,R,66,0,AnalogInput4_1
2,9,1,W+,95,0,DiscreteOutp1_1
2,10,1,W+,95,0,DiscreteOutp2_1
```

Importing this CSV, reflecting network variables in two different LonWorks nodes, results in the following screen shot.



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 object 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 (user interface) 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. Starting with Windows 7, that driver was no longer included, and Windows 8 complicated matters even further.

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".

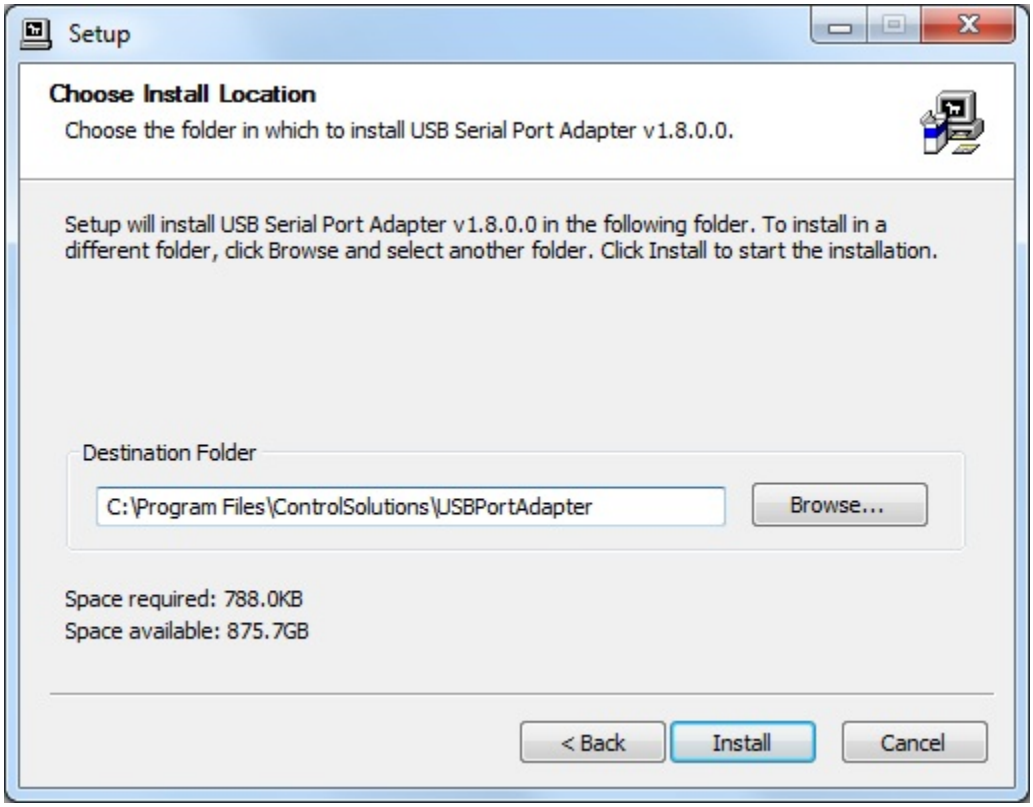
Unzip the contents of this file into a directory somewhere on your PC. The contents will look like this:

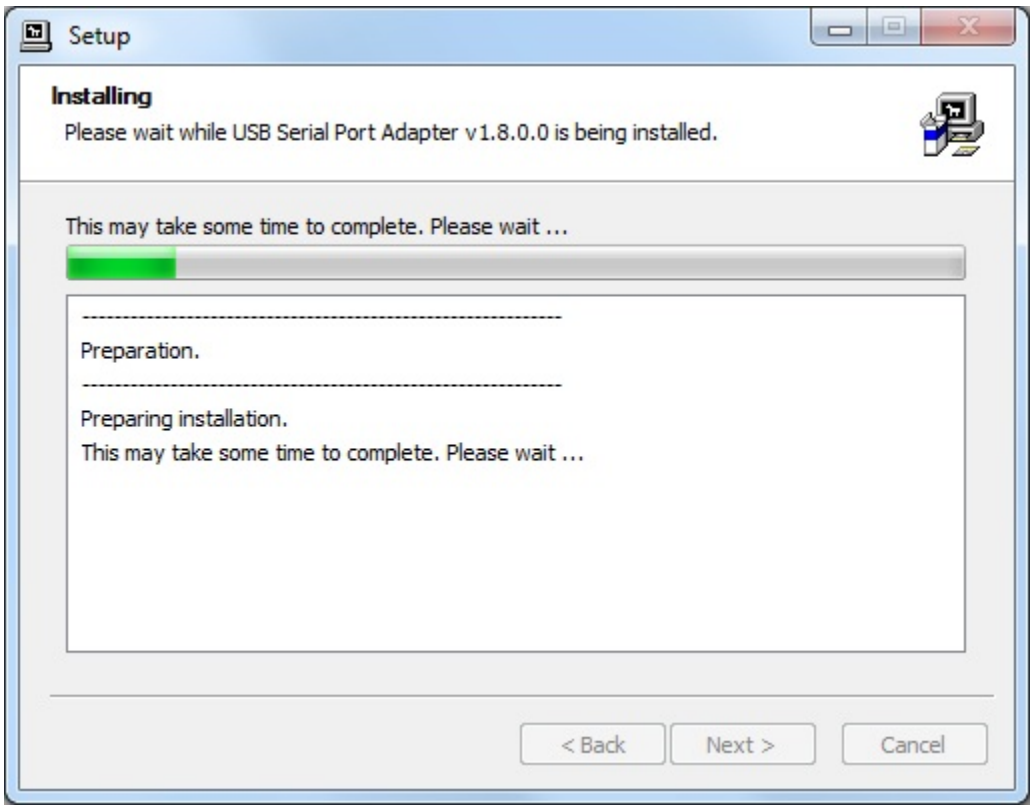
Double click "setup.exe". Say "yes" to any questions about whether to trust this software. Also, for Windows 8, you should right click on the setup file and "Run as administrator" - you will need to be logged in with administrator privileges.

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.

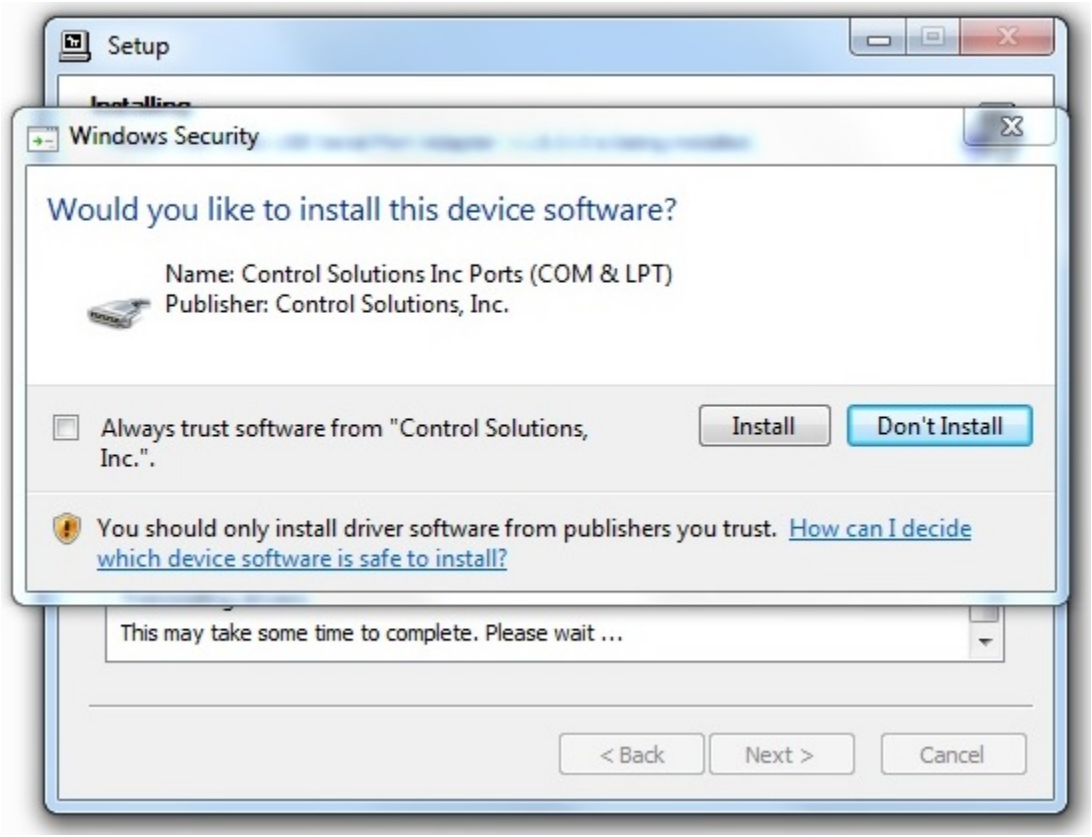
Technically, what you are doing in the process illustrated in the screen shots below is "driver pre-installation". After initial installation of this package, the device will automatically find the right driver when you plug it in, and driver installation will be finalized. Windows 8 will install the driver

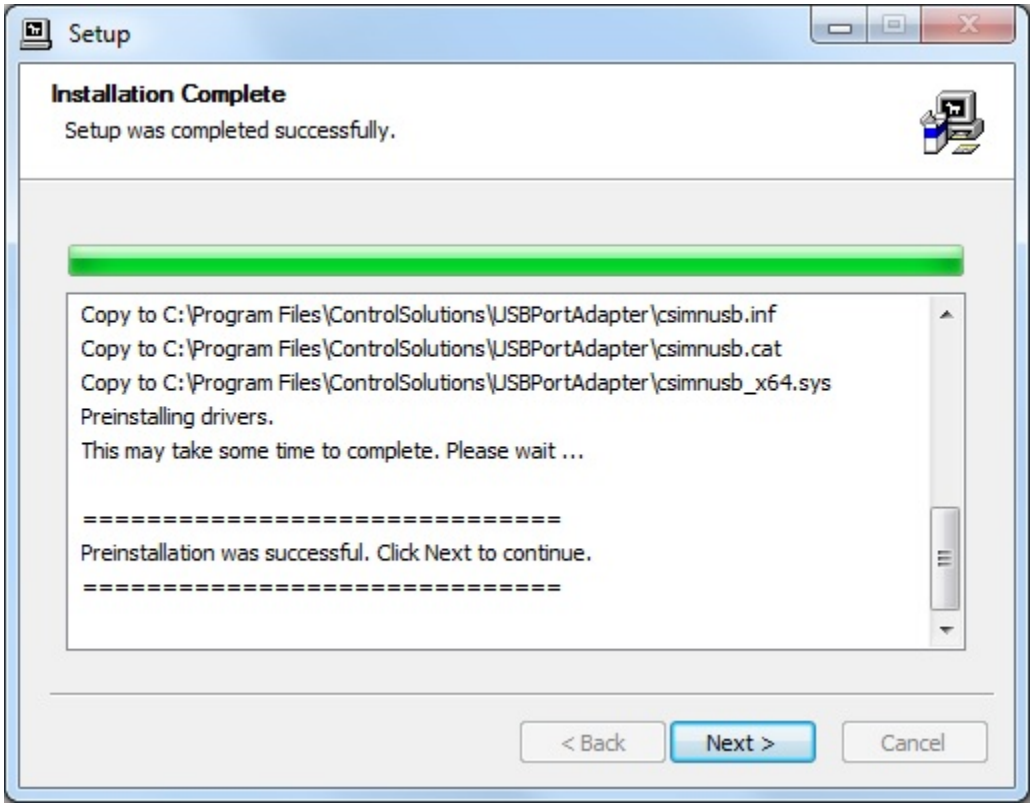
quietly and usually say nothing about it. Windows 7 will display a prompt telling you the new device was installed, but will not require responding to any prompts. Windows XP will go through the characteristic "Found new hardware" routine with a series of dialogs and prompts the first time you plug the 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

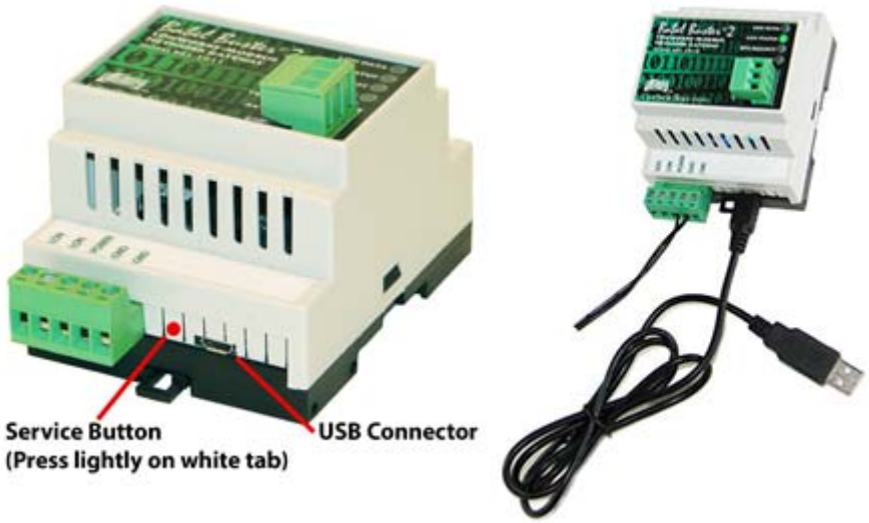
Solutions configuration tool's "Connect" page.

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.

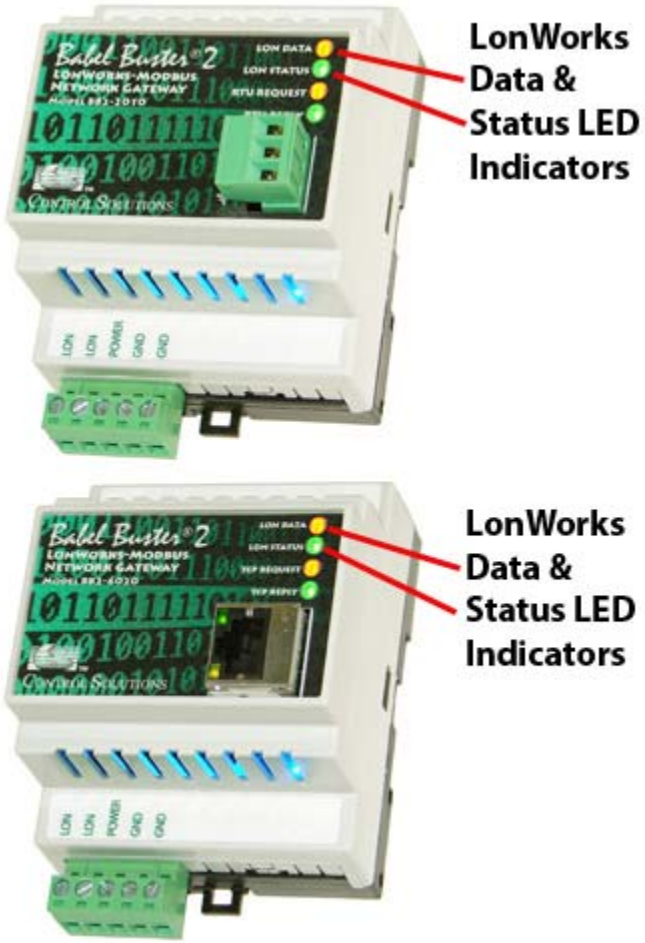
The service button is hidden behind the tab illustrated below. Press lightly on the white tab indicated. You will feel a slight clicking action when the button is pressed. ***It is not necessary to remove the cover to press the button*** *- it is intended to be actuated by the plastic tab that is part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



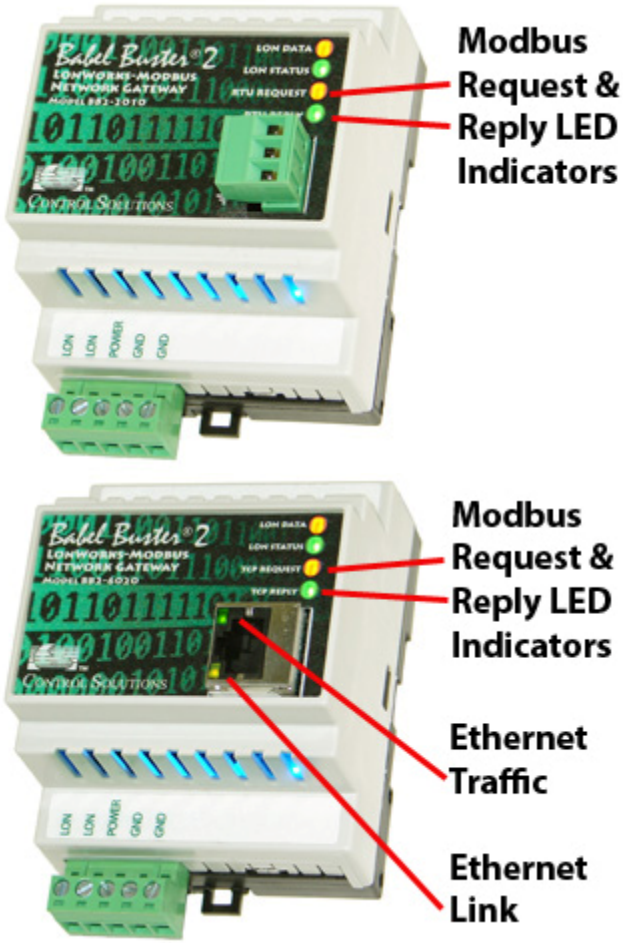
The LonWorks indicator LEDs display LonWorks network activity. Note, however, that activity of these LEDs is not only affected by configuration of the device, but by whether other devices are also communicating.

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	Yellow flash indicates NV update was sent by gateway, or node management message was sent. Green flash indicates NV fetch response was received by gateway or other action completed successfully.	Red solid on indicates Neuron chip is not running. Brief flash of red indicates error in processing NV request or node management request. Green indicates gateway's host processor is communicating with LonWorks Neuron chip.

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.	

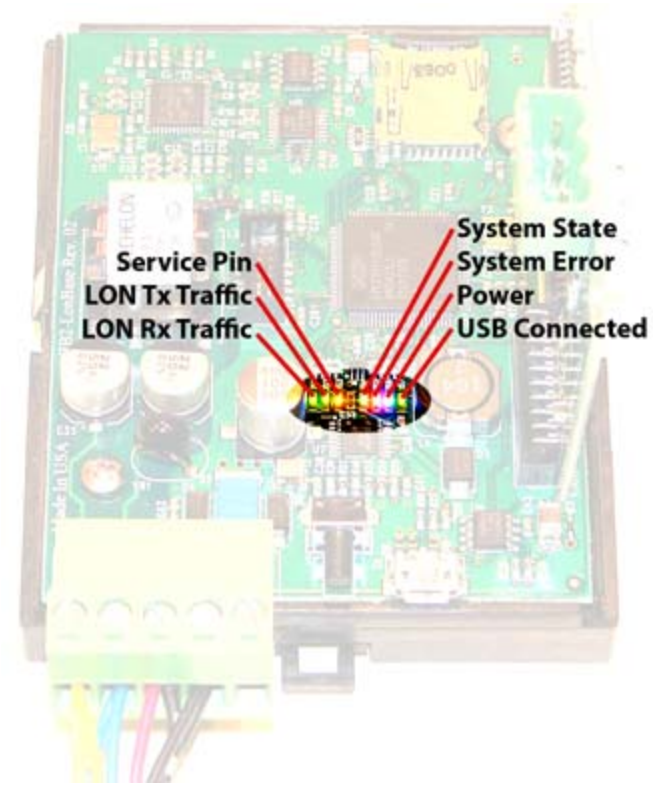
An MS/TP device that is functioning normally will always be at least passing the token, and usually polling for master periodically. The only time a device will not poll for master is if another device exists on the network with a MAC address only 1 greater than the gateway itself. It then just passes the token to that device without any intermediate polling for master.

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.

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.

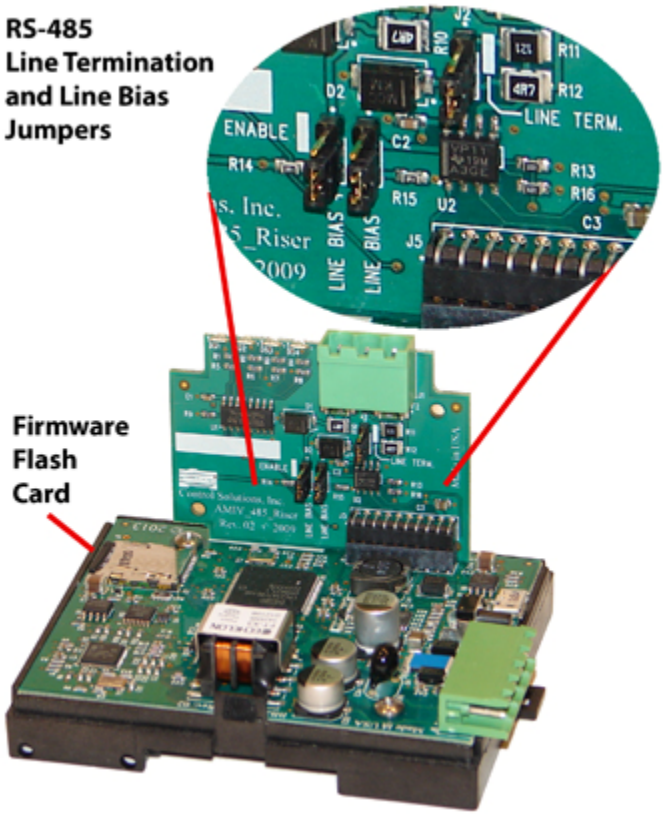
J.4 RS-485 Line Termination & Bias

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

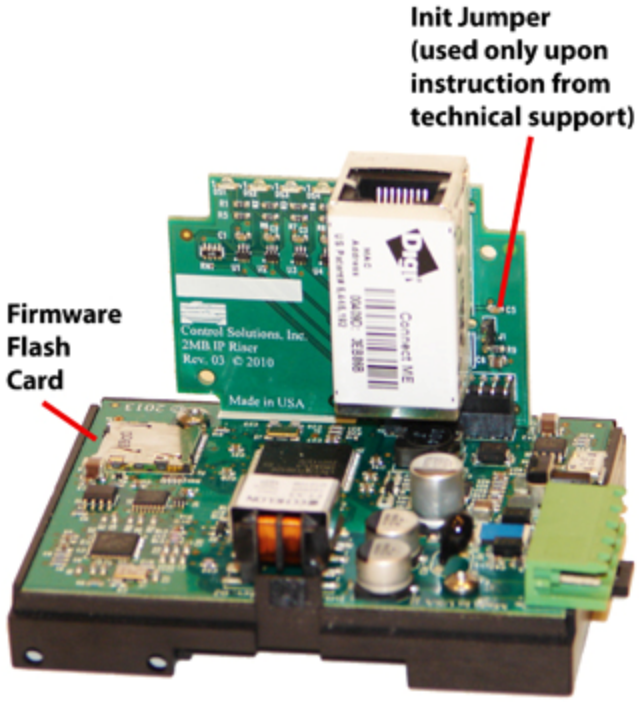
does not have to be the end point. Line bias holds the line in a known neutral state when no devices are transmitting. Without bias, the transition from offline to online by a transmitter can look like a false start bit and cause loss of communication.

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.


Init Jumper
(used only upon
instruction from
technical support)
Firmware
Flash
Card
Control Solutions, Inc.
2MB IP Riser
Rev. 03 © 2010
Made in USA

Appendix K LonWorks Terminology

K.1 Definition of NV, SNVT, etc.

Binding - A process of connecting inputs on one LonWorks node to outputs on another, typically done by a network management tool such as Echelon's LonMaker. If you need to bind the gateway into your LonWorks network, then the -NB version is the wrong gateway to be using. Use the standard LonMark certified gateway instead. The -NB gateway does not require any binding since it is able to simply poll the variables of interest in other LonWorks devices.

Echelon - The company (Echelon Corp.) that created LonWorks.

Functional Block - A logical element for organizing inputs and outputs in a LonWorks node that will be bound into a managed network. There are no functional blocks in the -NB gateway. These only exist in the standard LonMark certified version of the gateway.

LonMark - The certifying body responsible for testing and listing certified devices for purposes of assuring interoperability of LonWorks devices. The -NB gateway is not LonMark certified because by LonMark standards, you should not be using Modbus as your network in the first place, and putting LonWorks devices on a non-LonWorks network is not something they would certify.

LonWorks - The trademarked name of the protocol itself.

LonWorks Object - these can come by a wide variety of names, such as Sensor Object, Actuator Object, and a long list of other objects that are officially recognized as universal object types by LonMark. The definition of objects exists primarily for standardizing documentation and supporting interoperability of standard LonWorks devices.

Node Object - A logical element that exists in any LonWorks device for purposes of interoperability on a LonWorks network. When putting a LonWorks device on a Modbus network, you will have little use for this object and can generally disregard it.

NV - Network Variable. This is the data element transmitted over the LonWorks network. The closest analogy in Modbus is the register; however, an NV can contain multiple elements of data equivalent to multiple Modbus registers in a single variable. The gateway has the ability to disassemble the structured NV into multiple Modbus registers when reading LonWorks data, and vice versa when writing.

NV Index - This is effectively the "address" of the Network Variable inside the LonWorks device. If you do not know the NV Index of a data element you wish to query, then you can't query it.

NVI - Network Variable Input. This means "input" to the LonWorks device, from the network. An actuator such as a relay will have an NVI to turn it on and off. You will most often just write to an NVI, but you can also read an NVI, or both.

NVO - Network Variable Output. This means "output" from the LonWorks device, to the network. A sensor such as a temperature probe will have an NVO to send its readings to the rest of the network. You can only read an NVO, there is no way to write an NVO (and that is dictated by LonWorks protocol, not by any particular manufacturer's implementation).

SNVT - Standard Network Variable Type. This refers to the format of the data contained in a Network Variable (NV). There are close to 200 different NV types. For example, temperature and pressure each have their own types. In fact, temperature and pressure each have multiple types,

some integer and some floating point. Conversion from LonWorks to Modbus requires knowing the SNVT so that the raw data can be properly scaled to make it useful on the Modbus side.

SNVT index - This is the code defining the SNVT (see above). The NV Index is the address of the variable, and the SNVT index is the code that tells you what's found at that address in terms of data content and format.