



BB3-3101
BACnet-Modbus Gateway
Rev. 1.0 – May 2022

User Guide

Babel Buster 3

© 2022 Control Solutions, Inc.

User Guide Contents

[1 Introduction](#)

- 1.1 How to Use This Guide
- 1.2 Important Safety Notice
- 1.3 Warranty
- 1.4 Required License Information

[2 Overview of Gateway Functions](#)

- 2.1 What is New in BB3-3101
- 2.2 Object Server Model for a Gateway
- 2.3 Data Flow in the Gateway

[3 Overview of How to Configure Gateway](#)

- 3.1 Build from CSV List of Modbus Registers
- 3.2 Build from CSV List of Modicon Registers
- 3.3 Build from CSV List of BACnet Objects
- 3.4 Build Configuration from Scratch
- 3.5 Verifying Configuration in the Gateway Device
- 3.6 Changing Object Counts
- 3.7 Converting Register to Multiple Objects

[4 Tool 'Local Port' Page](#)

- 4.1 Connecting Configuration Tool to Gateway Device
- 4.2 Special Setting for Server-Only Applications

[5 Tool 'Who-Is' Page](#)

- 5.1 Finding Devices
- 5.2 Select Target
- 5.3 Getting Device Information
- 5.4 Clear Who-Is Cache
- 5.5 Read All Configuration
- 5.6 Targeted Who-Is

[6 Tool 'Read/Write' Page](#)

- 6.1 Read Property
- 6.2 Write Property

[7 Tool 'Device' Page](#)

- 7.1 Retrieve Device Object Information from Device
- 7.2 Change Device Object Configuration
- 7.3 Reassign Device Instance
- 7.4 Get Object Counts from Device
- 7.5 Reconfigure Object Counts
- 7.6 Fixing "Invalid Configuration"
- 7.7 Export and Import Configuration with XML File

[8 Tool 'Object Map' Page](#)

- 8.1 Internal BACnet Object Configuration Parameters
- 8.2 Modbus Master Configuration Parameters
- 8.3 BACnet Client Configuration Parameters
- 8.4 Insert, Add, Delete Objects
- 8.5 Read/Write Object Map in Device
- 8.6 Export and Import Configuration as XML File
- 8.7 Using Default on Comm Fail
- 8.8 Data Conversion Examples

[9 Tool 'Object Text' Page](#)

- 9.1 Read/Write Object Text
- 9.2 Edit Object Text

- 9.3 Export and Import Text as CSV file
- 9.4 Clearing Object Text

[10 Tool 'Object List' Page](#)

- 10.1 Read/Write All Object Maps in Device
- 10.2 Select an Object Map for Editing
- 10.3 Export and Import Configuration as CSV file
- 10.4 Clearing Object Maps, Counts, and Device

[11 Tool 'Modbus' Page](#)

- 11.1 Set Modbus Port Parameters
- 11.2 Check/Reset Errors
- 11.3 Diagnostic Modbus Read/Write

[12 Tool 'Data List' Page](#)

- 12.1 Read All Object Data
- 12.2 Error Codes
- 12.3 Multistate Value Offset

[13 Tool 'Firmware' Page](#)

- 13.1 Firmware Organization
- 13.2 Verifying Primary Application Image
- 13.3 Firmware Update Process

[Appendix A – CSV File Format](#)

- A.1 Object Maps
- A.2 Object Text

[Appendix B – Modbus Trouble Shooting](#)

- B.1 Observing Modbus Errors, LED Indicators
- B.2 Error Codes, Reliability Codes
- B.3 Auto-Reset Errors
- B.4 Trouble Shooting Tips
- B.5 Modbus Reference Information

[Appendix C – BACnet Trouble Shooting](#)

- C.1 Observing BACnet Errors, LED Indicators
- C.2 Error Codes, Reliability Codes
- C.3 Auto-Reset Errors
- C.4 Trouble Shooting Tips, USB Adapter
- C.5 Trouble Shooting Tips, Gateway Behavior
- C.6 System Fault Indications

[Appendix D - Modbus Slave Register Map](#)

- D.1 Modbus Slave Map - Calculated Registers
- D.2 Modbus Slave Map - Alternate Map
- D.3 Relinquishing Commandable Objects

[Appendix E - BACnet Object Properties](#)

- E.1 Data Object Properties
- E.2 Device Object Properties

[Appendix F - BACnet Codes](#)

- F.1 BACnet Object Property Codes
- F.2 BACnet Engineering Units Codes
- F.3 BACnet Error Codes

[Appendix G - Configuration XML File Format](#)

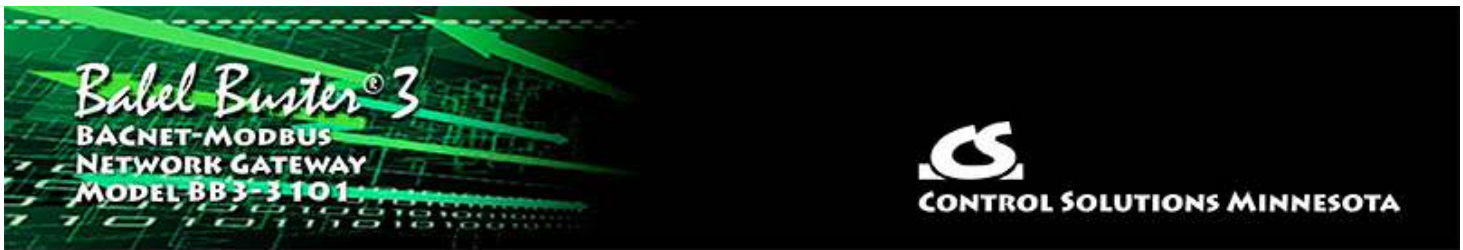
- G.1 Configuration File Editing

[Appendix H - USB Driver Installation](#)

- H.1 Driver Installation

[Appendix J - Hardware Details](#)

- J.1 Wiring
- J.2 Front Panel LED Indicators
- J.3 MTX002 USB to MS/TP Adapter
- J.4 RS-485 Line Termination & Bias
- J.5 Restore Factory Defaults



1 Introduction

1.1 How to Use This Guide

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

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

1.2 Important Safety Notice

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

1.3 Warranty

This 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.

1.4 Required License Information

The BB3-3101 configuration tool includes the SmartWin library (<http://smartwinlib.org>) under the following terms:

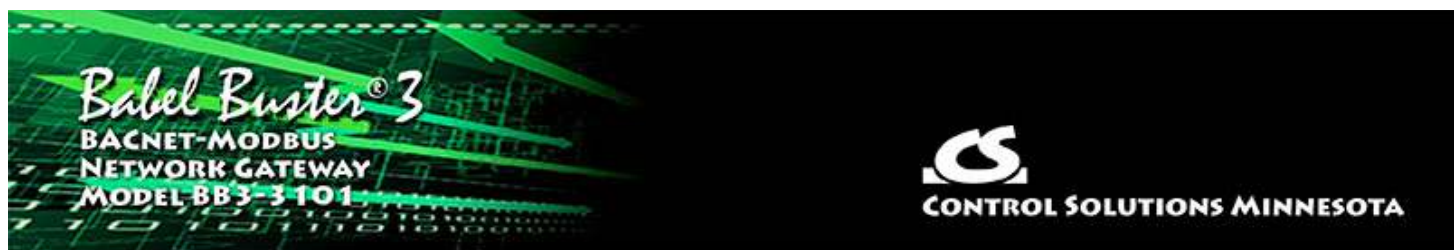
License agreement for SmartWin++ (BSD license)

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.



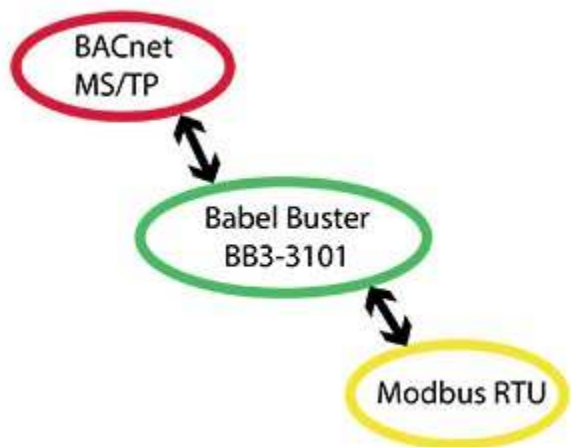
2 Overview of Gateway Devices

2.1 What is New in BB3-3101

The BB3-3101 is an upgrade to the BB2-3010 Modbus RTU to BACnet MS/TP gateway. The BB3-3101 features a faster processor and more memory, which leads to support for a larger number of BACnet objects. In addition, support for state strings for both Binary and Multi-state objects is added in the BB3-3101.

Key Differences	BB2-3010	BB3-3101
Maximum number of BACnet objects	315	1024
Maximum number of State Strings	(none)	4096
MS/TP Port Isolated	No	Yes

Upgrading: If you have an XML configuration file saved from a BB2-3010, you can directly import that into your BB3-3101 by simply opening that file in the configuration tool (see sections 7 and 8).



2.2 Object Server Model for a Gateway

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

The central data element in every Control Solutions gateway is an “object”. Each object has rules for accessing that object which are specific to the protocol of the network. Each object has at least two sets of rules, one set for each of the two (or more) networks that may access the object. The object model is often optimized to cater to a specific protocol, and will most often favor the more complex protocol. For example, all Control Solutions gateways that include BACnet connectivity will have objects that are defined primarily as BACnet objects, with auxiliary rules for accessing the objects via Modbus (or some other protocol).

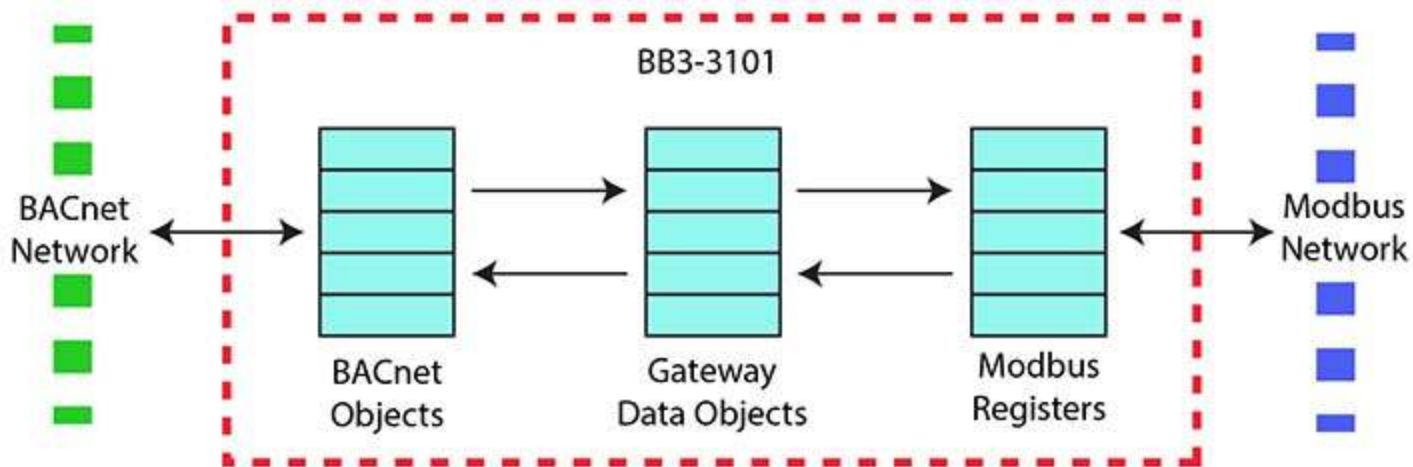
Control Solutions gateways will function as servers, providing a copy of the most recent data found in its data base when a client requests that data. In master/slave terms, the server is a slave while the client is a master. Some applications will treat the gateway as a server from both (all) networks connected. But most applications will want the gateway to be a server on one side, and a client on the other side. The most frequent application of the BB3-3101 will have it functioning as a Modbus master (client) and BACnet slave (server).

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). BACnet 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.

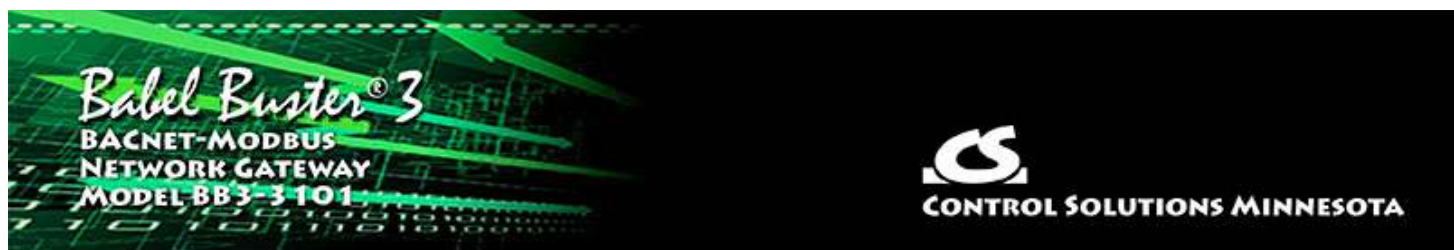
The BB3-3101 can be configured as a Modbus RTU master or slave (client or server), but can never be both at the same time (as specified by Modbus protocol). The BB3-3101 defaults to being a BACnet server (slave), but can also function, at the same time, as a BACnet client (master). Slave functionality requires no special configuration, but client functionality requires specifying a list of remote devices and data points to read or write. The entries in this list are referred to in BB3-3101 vocabulary as “object maps”.

2.3 Data Flow in the Gateway

There are three different realizations of the same data within the gateway. The direct connection to the Modbus network is the Modbus register. The direct connection to the BACnet network is the BACnet object. The translating connection in between is a gateway data object. The internal gateway data object very closely mirrors the BACnet object. The primary difference is that the gateway object can be reassigned to different BACnet object types. The same pool of gateway objects could be assigned to all be BACnet Analog Inputs, or assigned to all be BACnet Binary Outputs, just as an example.



The BB3-3101 gateway will most often be a Modbus master. However, it can also operate as a Modbus slave. When the gateway is operating as a slave, Modbus register numbers and function codes are used to decode which BACnet object is being requested. Any necessary data conversion is done automatically, on the fly. Normally, BACnet Analog objects will be read from the Modbus side as IEEE-754 floating point (register pair). But if the "Alt Map" option is being used, then you can read Analog objects as 16-bit integer Modbus registers.



3 Overview of How to Configure the Gateway

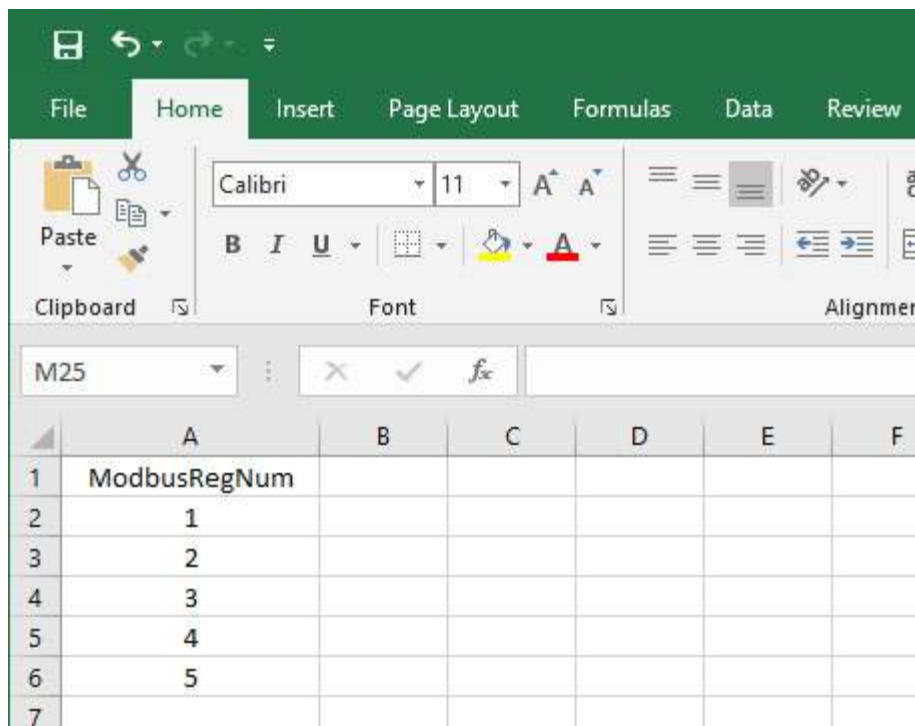
You need to do these three things in general to configure the gateway:

- (a) Configure the gateway as a BACnet device.
- (b) Configure some object mappings.
- (c) Configure the Modbus port.

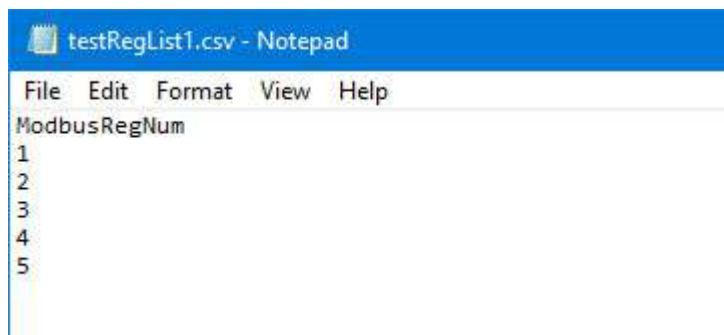
There are a few options for getting the objects and object mappings defined, and these are discussed in the following parts of this section of the user guide. Configuring the gateway as a BACnet device and setting up the Modbus port are discussed in sections later in this user guide. Later sections will also go into more detail about the features of each of the object mapping pages talked about here. The majority of your time spent configuring the gateway will be spent on the object mappings, so we want to start by taking a look at that process, and the options you have for approaching that task.

3.1 Building Configuration from CSV List of Modbus Registers

This configuration tool attempts to simplify configuration by recognizing the most minimal CSV file. If you only need to read a list of Modbus holding registers, then the CSV file can be as simple as a list of register numbers, with the heading "ModbusRegNum" as illustrated below. Note that these are proper register numbers, not Modicon style numbers. If your documentation shows register numbers starting at 40001, then either use the Modicon form of CSV illustrated in the next section, or drop the 40000 part and just enter 1 where the documentation shows 40001. If your Modbus documentation shows holding register numbers starting at 0, then add 1 to every register number when creating the CSV file.

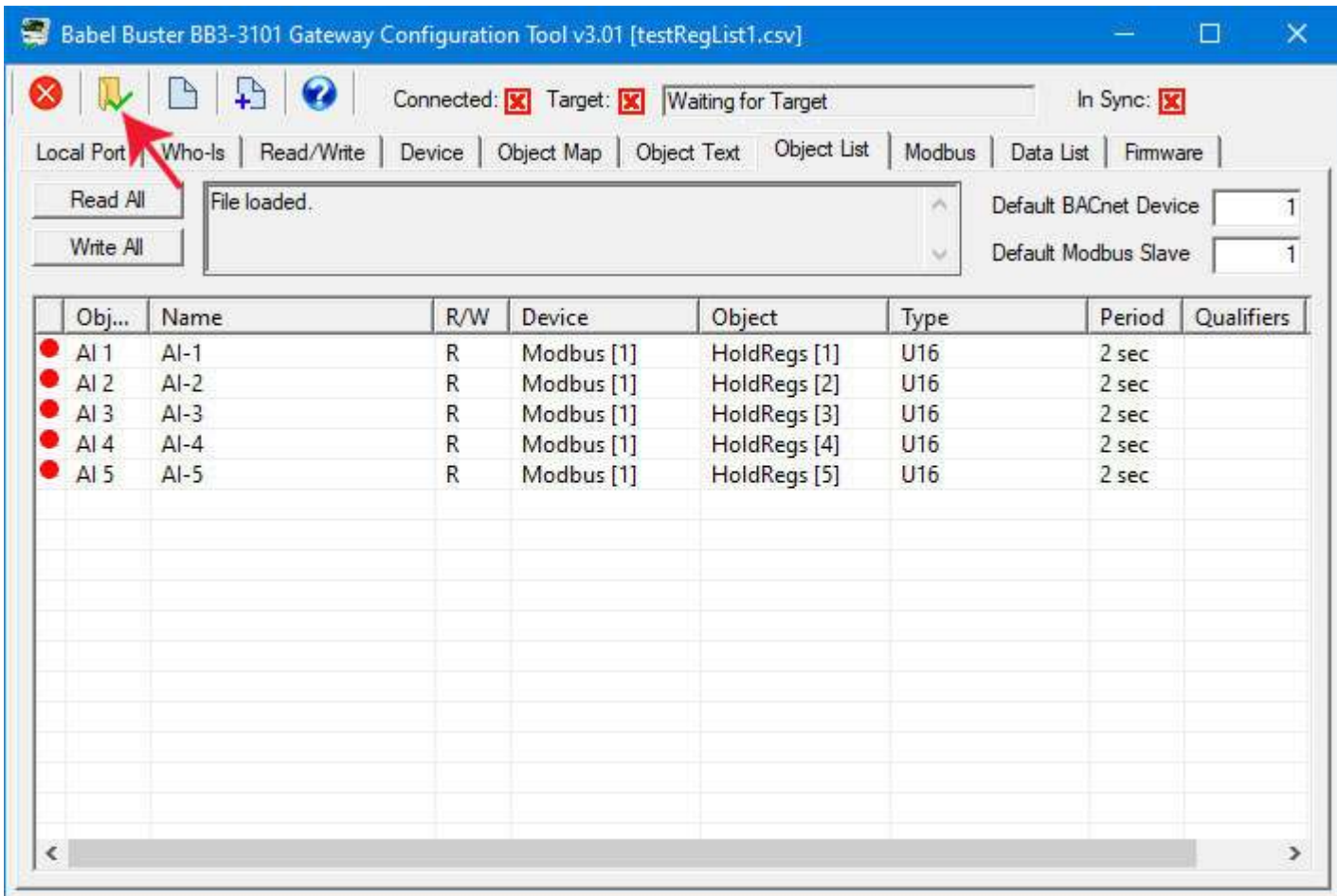


If you are using a spreadsheet program to enter the numbers, then be sure to "save as CSV" when saving the file. You can also use a simple text editor such as Notepad to enter the CSV as plain text, as illustrated below. Be sure to save your text as a .csv file, not .txt. The configuration tool needs to see a file name ending in .csv.

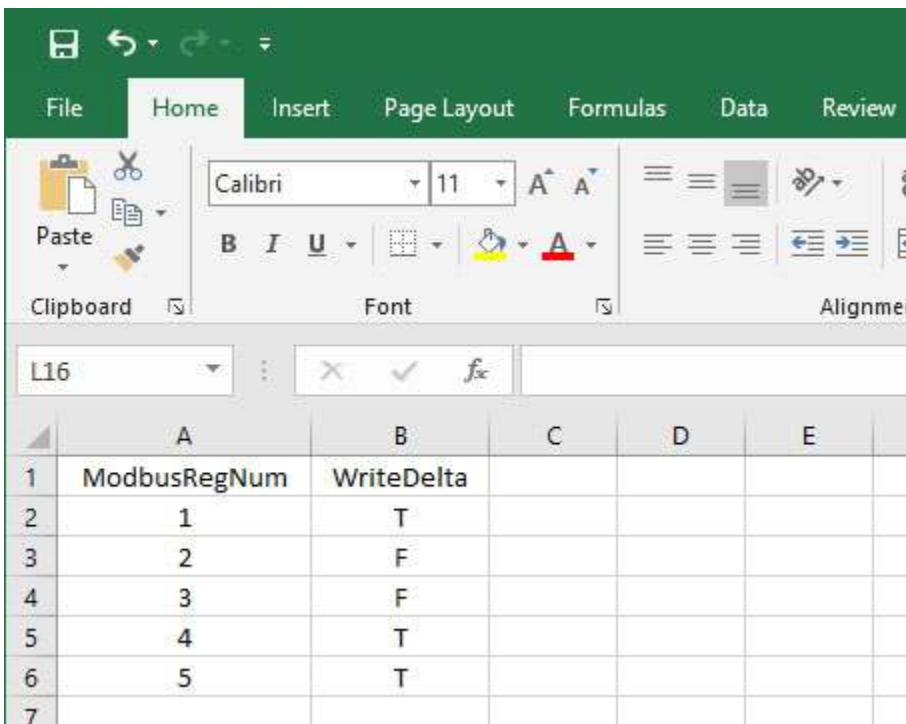


Once you have your file ready to go, open the configuration tool, go to the "Object List" page, and click the file open icon. Then find your *.csv file and open it. Given only a list of register numbers, they will default to reading as unsigned 16-bit holding registers, and the values will be assigned to BACnet Analog Input objects. All of the objects will be configured to read the Modbus registers. If you need to write some of the registers, continue on to the next example.

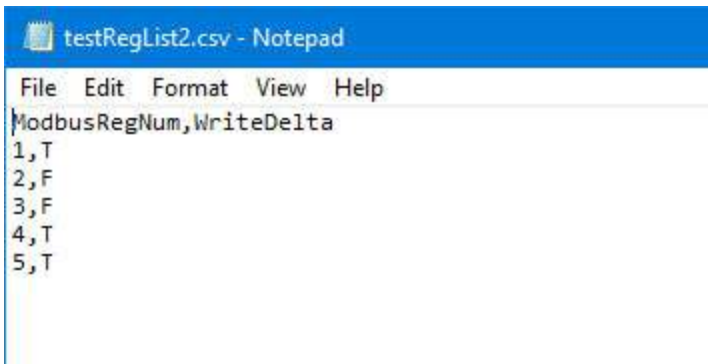
When you load configuration from a file, object counts on the Device page are automatically allocated, provided no objects had been previously allocated. If you load a file, then add some lines to the file, and try to load the new file, you will get an error message saying something about "object out of range". This means the configuration tool retained your previous object counts, but you now need more. To remedy this problem, go to the Device page, and click the "new" icon (that would be the left-most icon in the top toolbar). Clicking the "new" icon clears everything so you can start over with a new number of objects.



The above example shows reading a set of Modbus holding registers. Suppose you want to write some of them, and you only want the Modbus register to be written to (or updated) when a new value is written to the corresponding BACnet object. To accomplish this, add a second column labeled "WriteDelta" (no spaces between words). Enter T if you want to write the Modbus register, or F to read it instead.



The plain text version of this example looks like this:

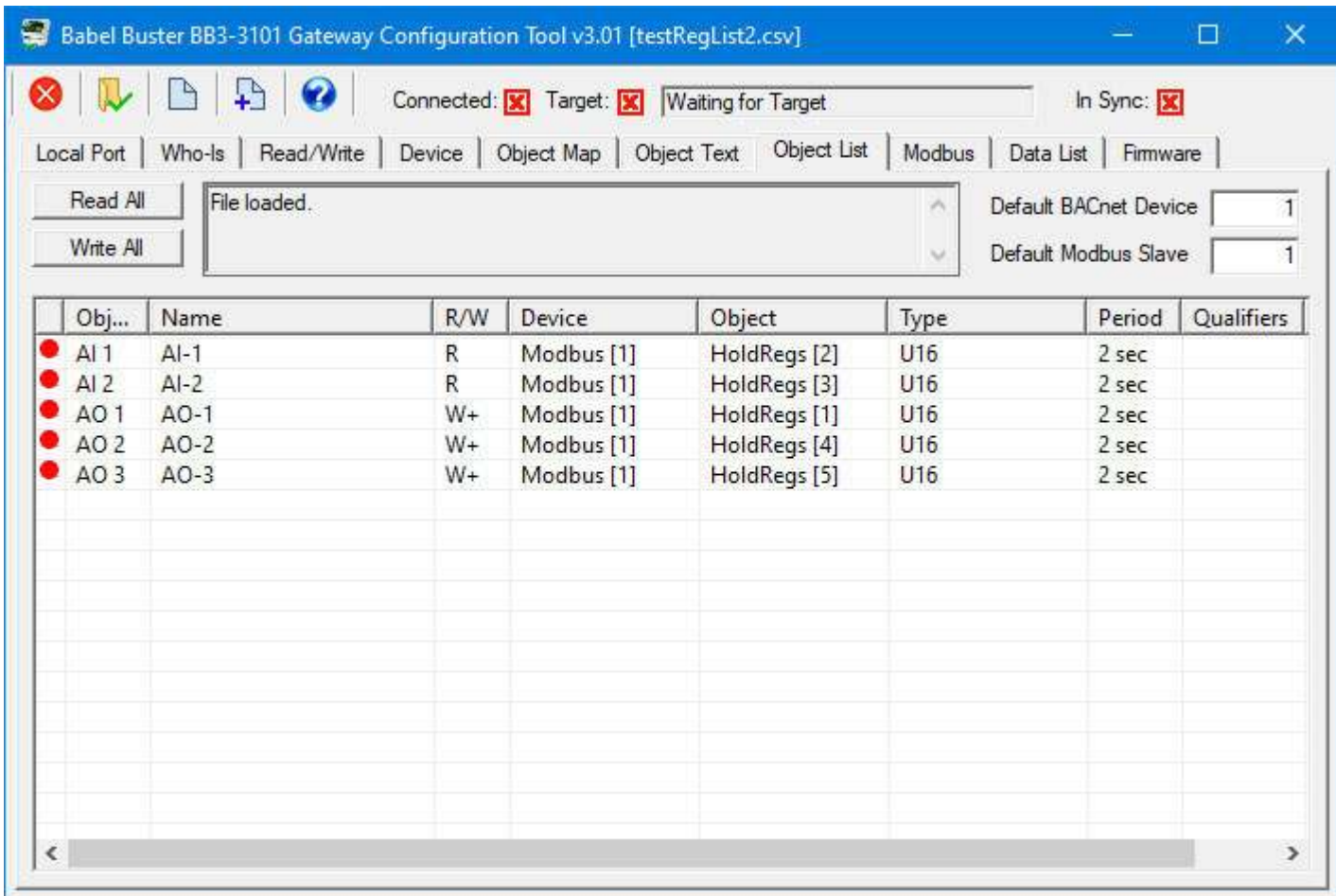


```
testRegList2.csv - Notepad
File Edit Format View Help
ModbusRegNum,WriteDelta
1,T
2,F
3,F
4,T
5,T
```

Upon loading this version of CSV file, note that the registers flagged for writing are assigned to Analog Output objects rather than inputs. The Analog Output object is what BACnet expects to write to in the gateway. When the respective Analog Output object is written from BACnet, the gateway will write the new value to its assigned Modbus holding register.

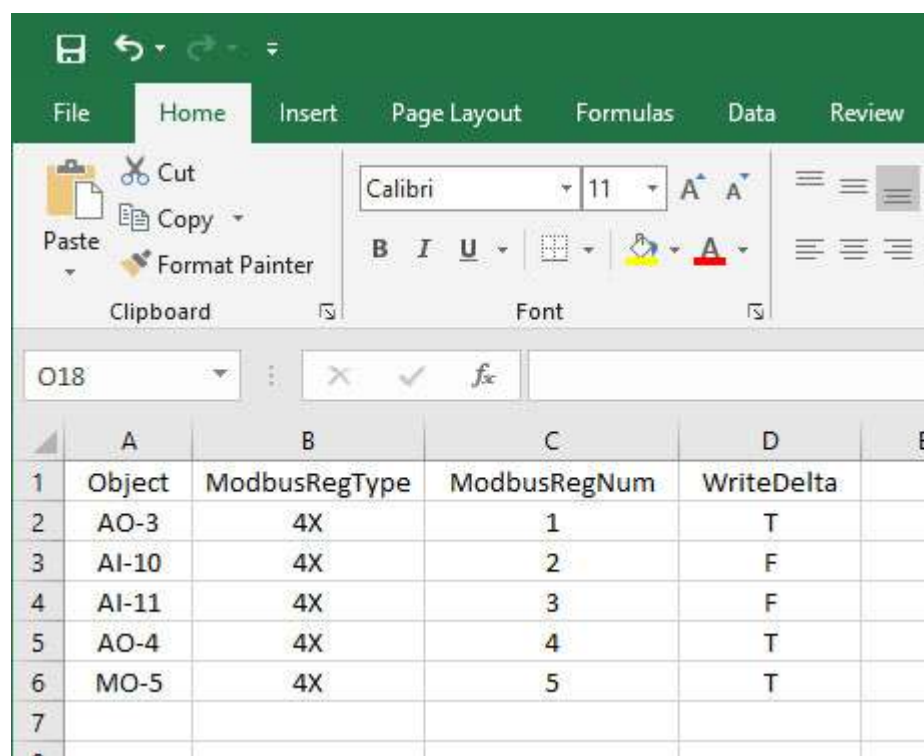
Also note the number in brackets after "Modbus" in the Device column. This is the Modbus RTU slave address. The value shown in the "Default Modbus Slave" window will be used when loading the CSV file when that column is not explicitly provided in the CSV file.

To see a complete listing of all possible CSV columns, refer to Appendix A.



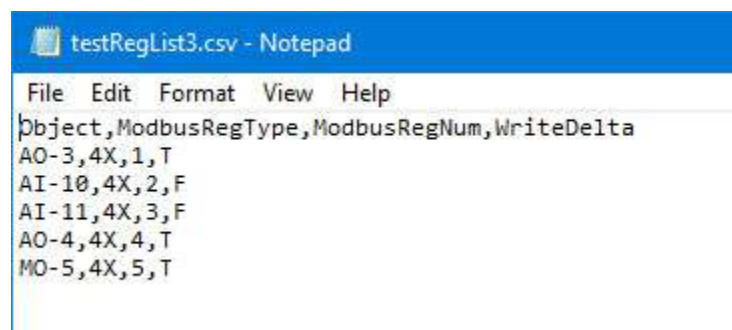
The next example is simply for additional illustration purposes. By default, loading the CSV file will start with object 1 and count up sequentially in assigning objects. If you have some reason to pick specific objects, or want to use objects other than the default Analog type, you can explicitly select BACnet objects to be assigned as illustrated here. In

addition, if you want to specify Modbus registers other than holding registers, you can do that with the ModbusRegType column. This example still results in all holding registers because they all indicate "4X", but if you wanted to change some of the registers to read input registers instead, then you would replace "4X" with "3X".



	A	B	C	D	E
1	Object	ModbusRegType	ModbusRegNum	WriteDelta	
2	AO-3	4X	1	T	
3	AI-10	4X	2	F	
4	AI-11	4X	3	F	
5	AO-4	4X	4	T	
6	MO-5	4X	5	T	
7					

The plain text version of this CSV file would look like this:

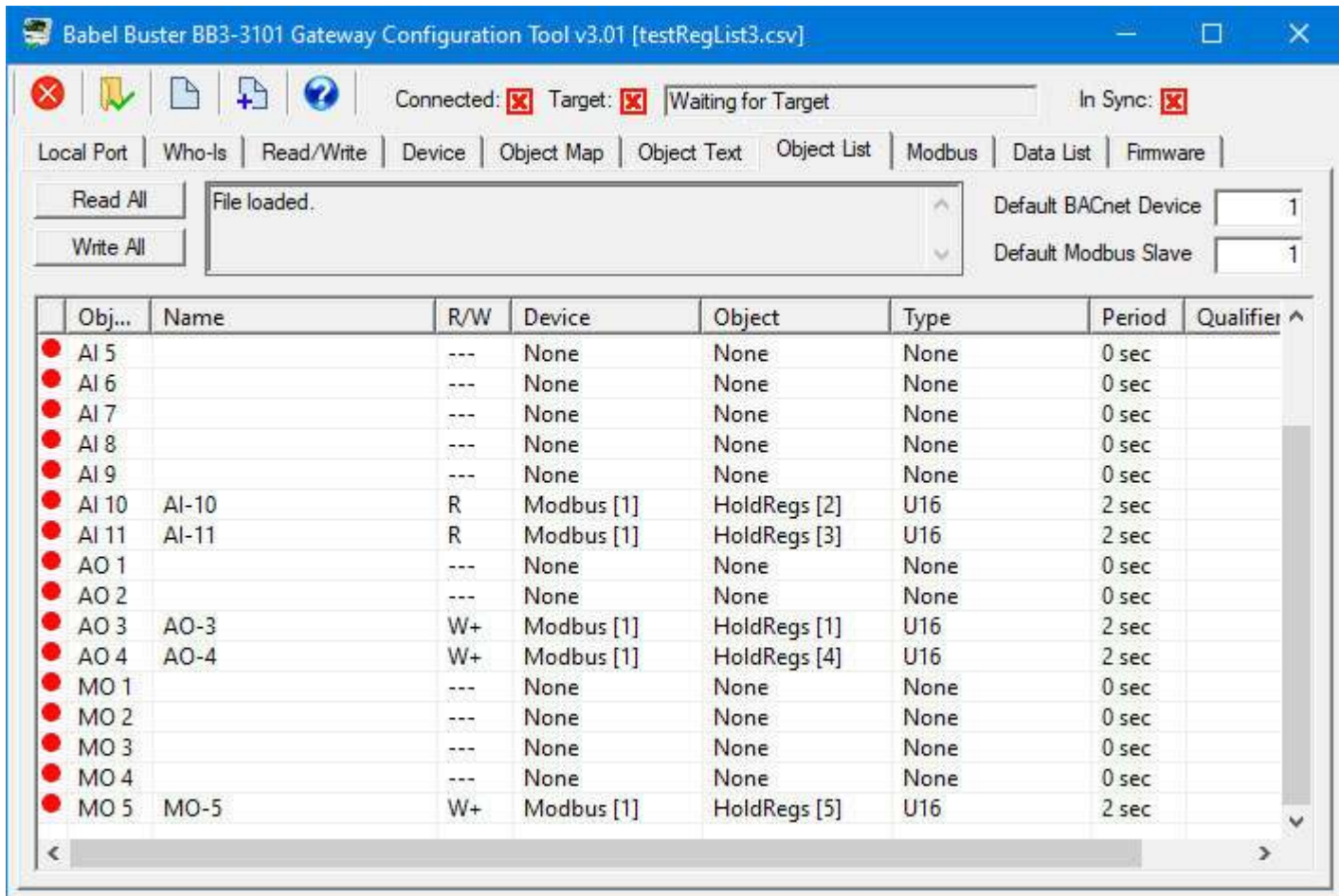


```

Object,ModbusRegType,ModbusRegNum,WriteDelta
AO-3,4X,1,T
AI-10,4X,2,F
AI-11,4X,3,F
AO-4,4X,4,T
MO-5,4X,5,T

```

After loading this CSV file, the Object List page would look like this, assuming you scroll down a bit. With Analog Input objects explicitly starting at 10 in this example, the first 9 objects will be present but unused at this point.

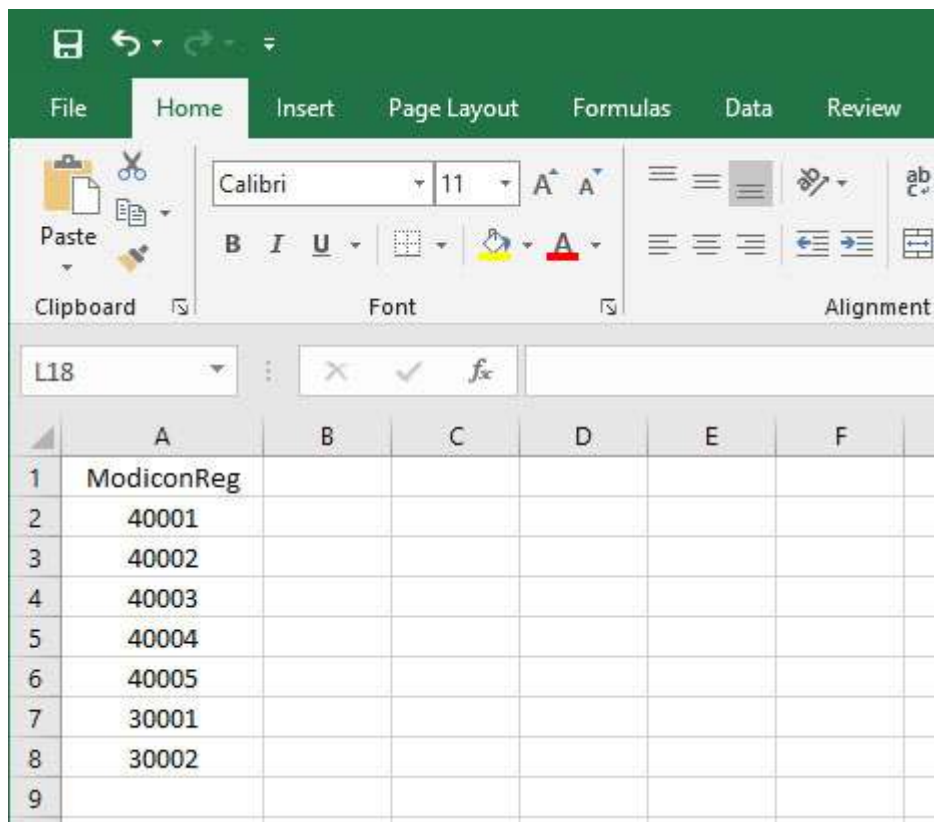


3.2 Building Configuration from CSV List of Modicon Registers

Older Modbus documentation, and some newer documentation using older conventions, will refer to the device's first holding register as number 40001. This is short hand for saying "holding register 1". To get beyond the limit of 9,999 registers using this numbering scheme, some documentation will use 400001 instead for the first holding register. This style of Modbus register documentation is a convention first introduced by Modicon back in the 1970's when Modbus was first created. The current Modbus protocol specification does not recognize Modicon notation. Actually, it does not recognize register "numbering" either, it recognizes raw addresses where the first holding register is 0000.

Some Modbus documentation will use 40001 to represent the first holding register. Some will document 0 as the first register. A majority of documentation will reference the first holding register as 1. You need to spend some time studying the documentation to try to determine which convention has been used. If you see a lot of numbers in the 40,000 range, it is Modicon. But otherwise, you have to look for evidence as to whether register numbers (or addresses) start at 0 or 1. If they start at 0, add 1 to all numbers when entering configuration parameters in the Babel Buster configuration tool.

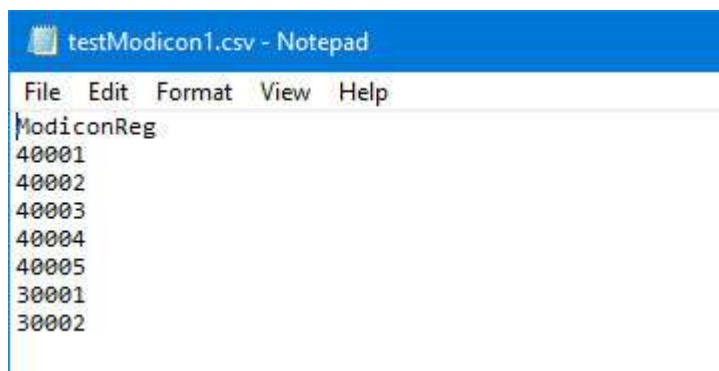
The following spread sheet shows holding registers 1 through 5, and input registers 1 and 2, expressed in Modicon notation.



The screenshot shows the Microsoft Excel interface with the 'Home' tab selected. The ribbon includes options for Clipboard, Font, and Alignment. The font is set to Calibri, size 11. The alignment is set to left. The active cell is L18. The table data is as follows:

	A	B	C	D	E	F
1	ModiconReg					
2	40001					
3	40002					
4	40003					
5	40004					
6	40005					
7	30001					
8	30002					
9						

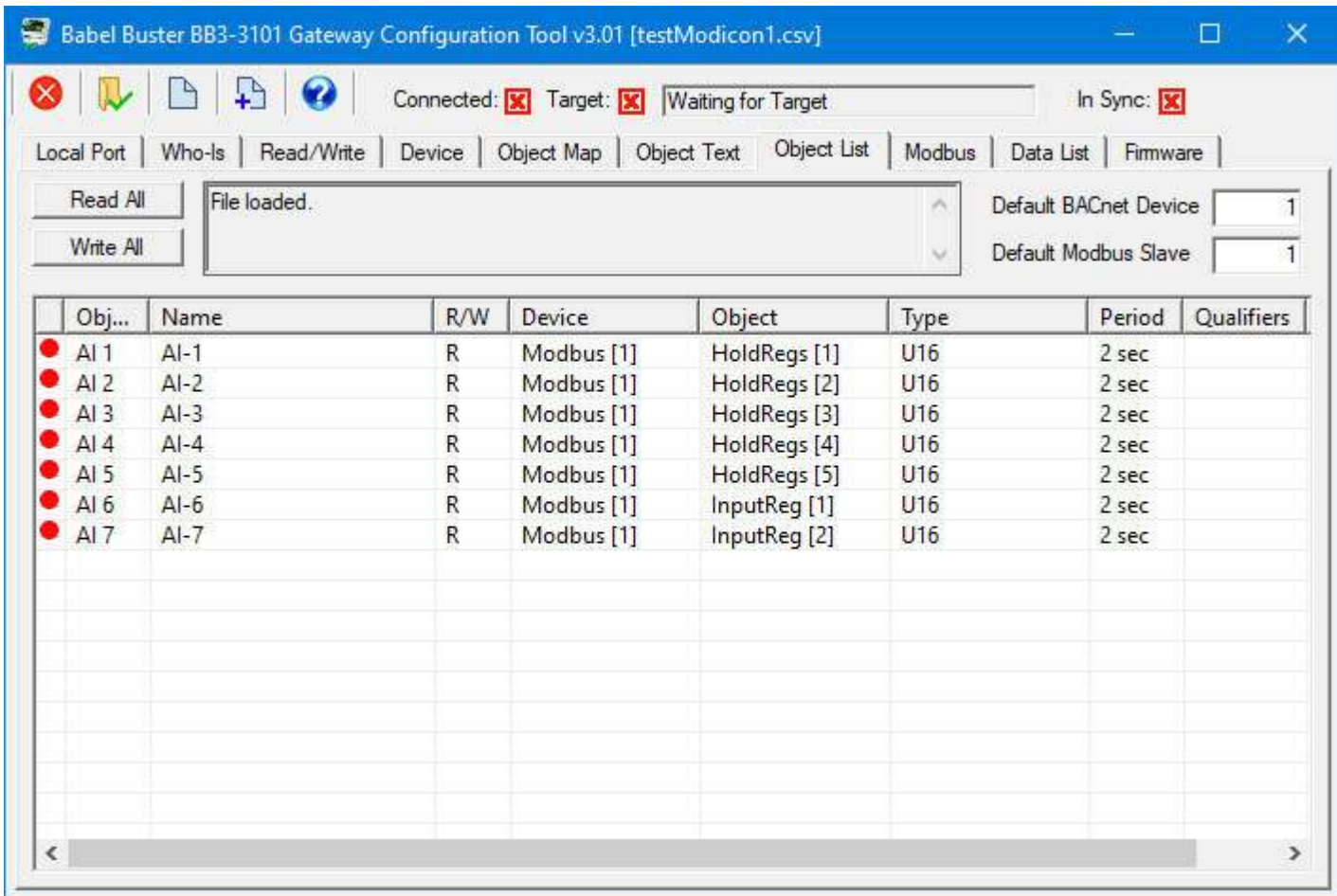
This would be the raw text form of the CSV file:



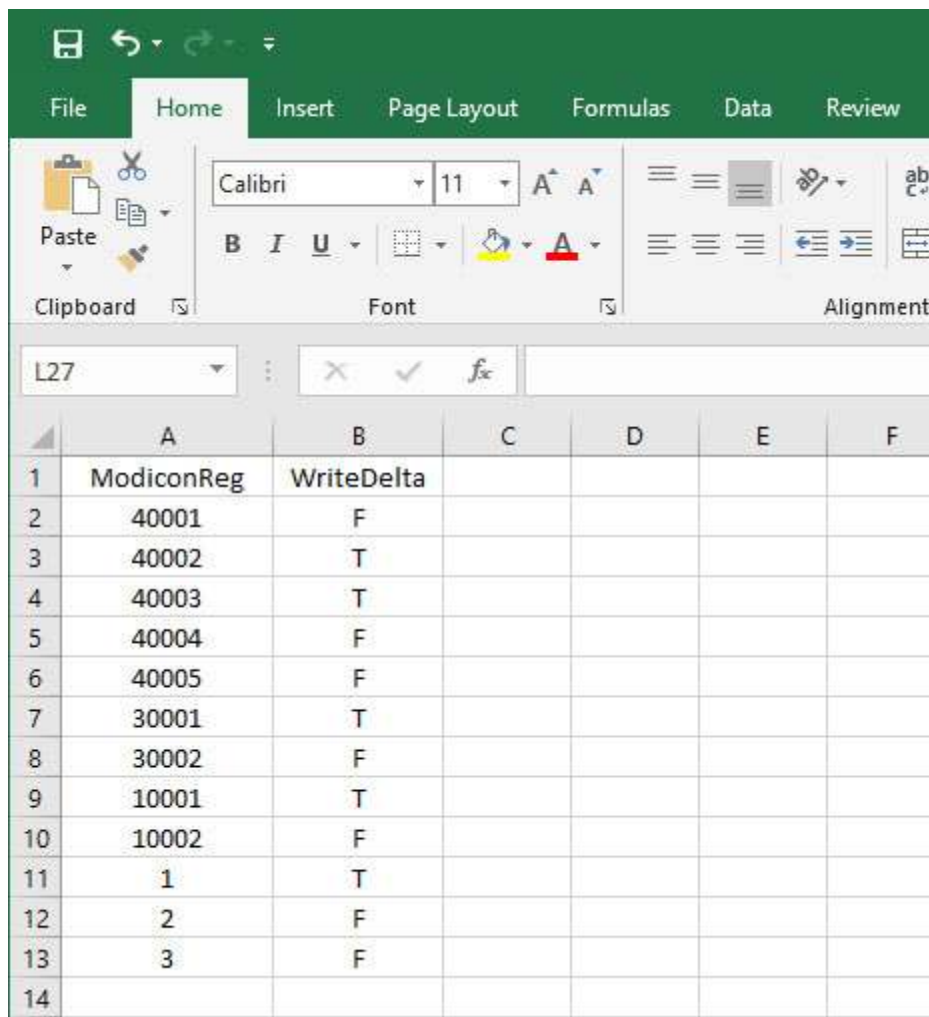
The screenshot shows a Notepad window titled 'testModicon1.csv'. The text content is as follows:

```
ModiconReg  
40001  
40002  
40003  
40004  
40005  
30001  
30002
```

Upon loading this CSV file, the Modicon shorthand numbers are converted to register type and register number as illustrated here.

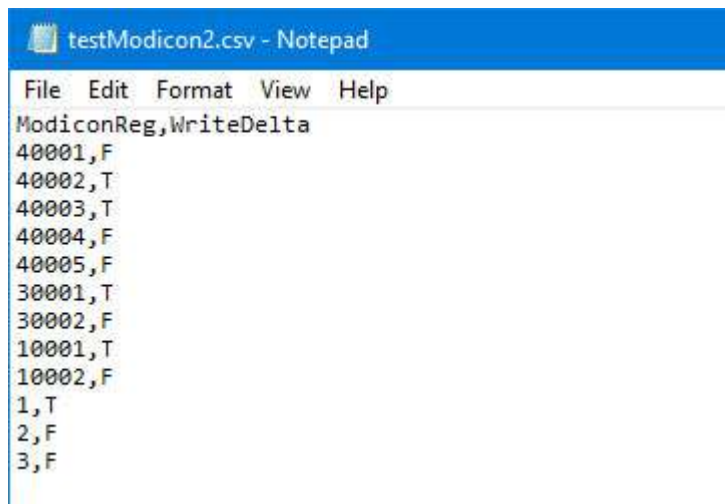


The same options for additional columns in the CSV file noted above are available for Modicon register notation. In the following example, we add some discrete inputs and coils, and mix up the reading and writing a bit.



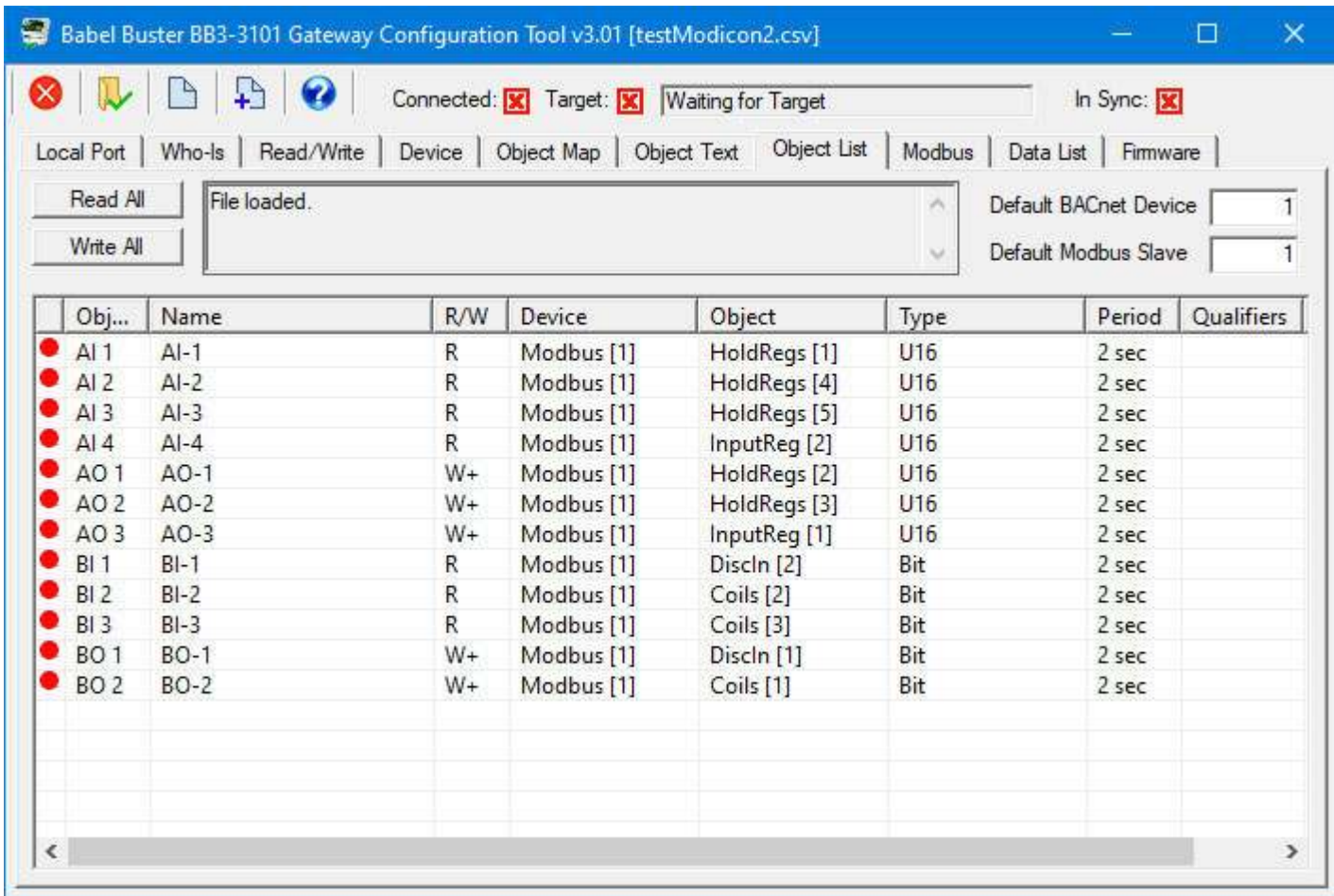
	A	B	C	D	E	F
1	ModiconReg	WriteDelta				
2	40001	F				
3	40002	T				
4	40003	T				
5	40004	F				
6	40005	F				
7	30001	T				
8	30002	F				
9	10001	T				
10	10002	F				
11	1	T				
12	2	F				
13	3	F				
14						

The plain text version of this CSV file would look like this:



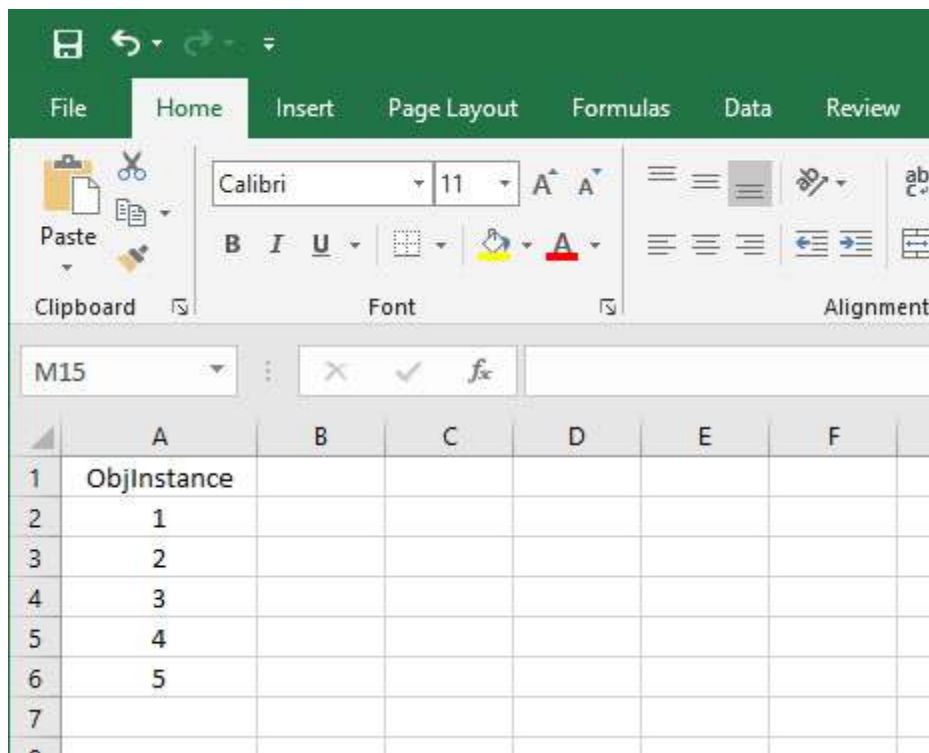
```
testModicon2.csv - Notepad
File Edit Format View Help
ModiconReg,WriteDelta
40001,F
40002,T
40003,T
40004,F
40005,F
30001,T
30002,F
10001,T
10002,F
1,T
2,F
3,F
```

After loading this CSV file into the configuration tool, the BACnet object assignments and Modbus mappings look like this:



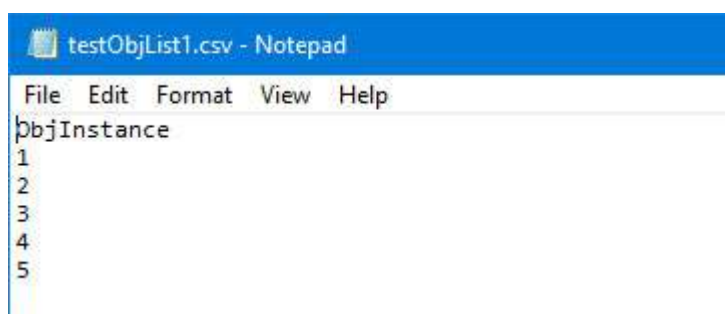
3.3 Building Configuration from CSV List of BACnet Objects

The majority of applications for a BB3-3101 gateway are for putting a Modbus device on a BACnet network. But suppose you have a Modbus PLC, running as a Modbus master, that wants to read data from one or more BACnet devices. These gateways are equally capable of going this direction, and the configuration tool provides CSV import flexibility to simplify that type of configuration as well. In its simplest form, you need only list object numbers, provided they are all Analog Input type BACnet objects.



	A	B	C	D	E	F
1	ObjInstance					
2	1					
3	2					
4	3					
5	4					
6	5					
7						

The plain text would look like the following. Be sure to save your text as a .csv file, not .txt. The configuration tool needs to see a file name ending in .csv.

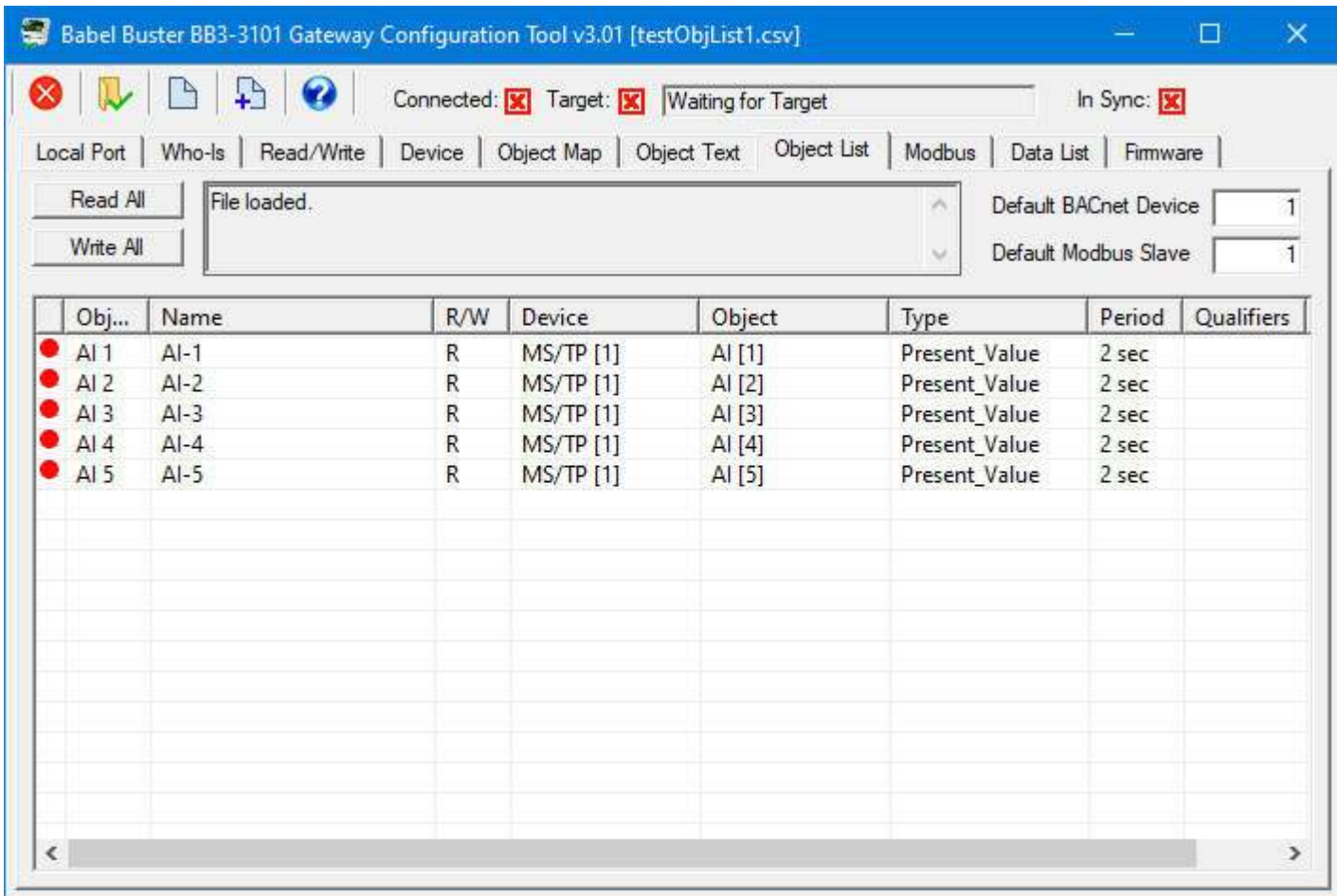


```
testObjList1.csv - Notepad
File Edit Format View Help
ObjInstance
1
2
3
4
5
```

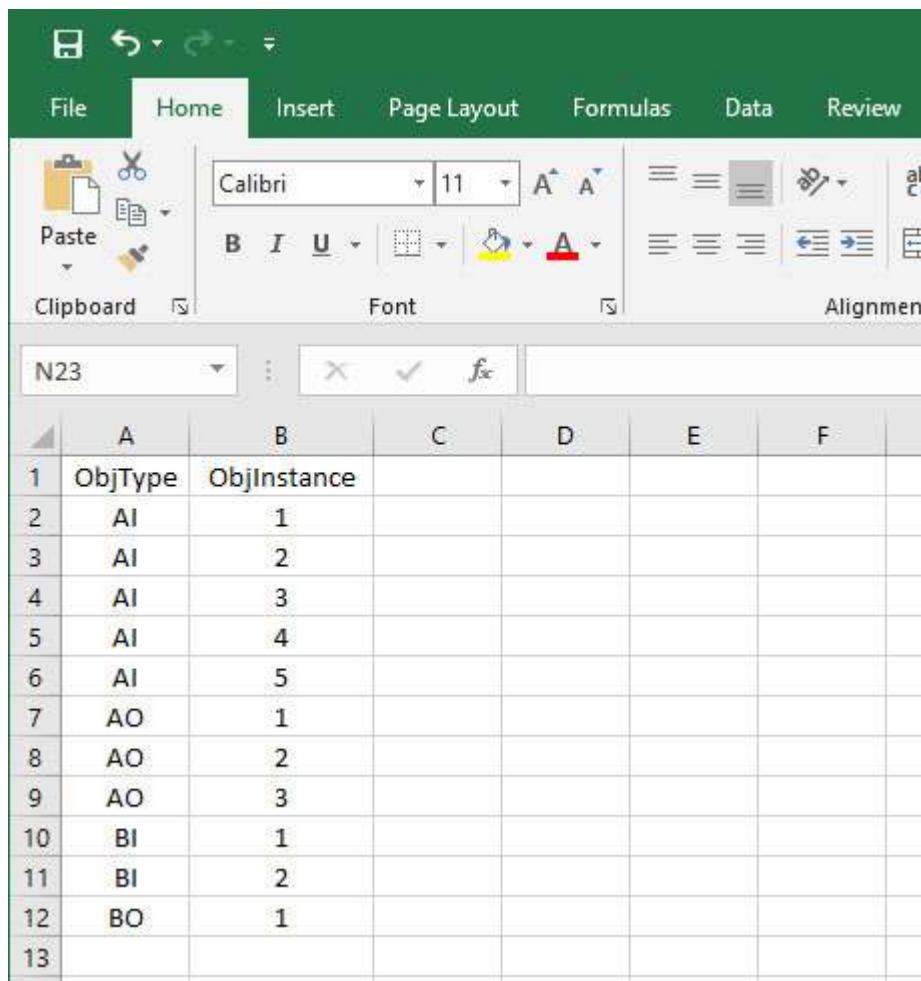
After loading this CSV file into the configuration tool, the Object List screen looks a bit different compared to the Modbus examples above. The Device column now shows "MS/TP" instead of "Modbus" because this is now what the gateway is going to be polling instead. The number in brackets after "MS/TP" is the device instance, and will default to whatever is entered in the Default BACnet Device window. The objects that will be read from that other MS/TP device will be the object numbers listed in the CSV file, and will by default be assigned as Analog Input types both in that device and in the gateway. By adding additional columns to the CSV file, you can specify different object types. Of course, you can also go to the Object Map and make changes after loading the CSV file.

Although a bit more complicated to set up as a CSV file, you can have the same gateway polling BACnet devices on one side, and polling Modbus devices (with gateway configured as Modbus master) on the other side. This means the gateway would actively read data from BACnet devices and write it to Modbus devices, and/or vice versa.

And just in case you were wondering, no, you cannot load multiple CSV files on top of each other. You need to combine them into one consistent CSV file first, and then load. By consistent, we mean the file must have one and only one header line, and it must be the first line in the file. Each following line must have as many entries on the line as the header line had.

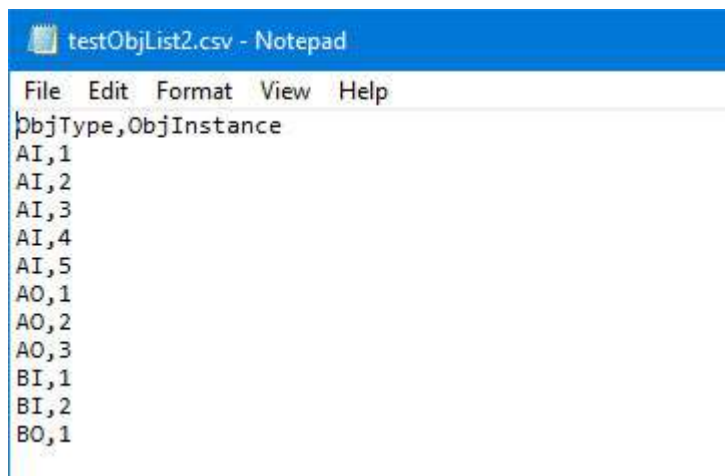


When only a list of object numbers are provided in the CSV file, they will default to being all Analog Input objects. To specify different types of objects in the target MS/TP device to be read (and/or written), you would add the ObjType column.



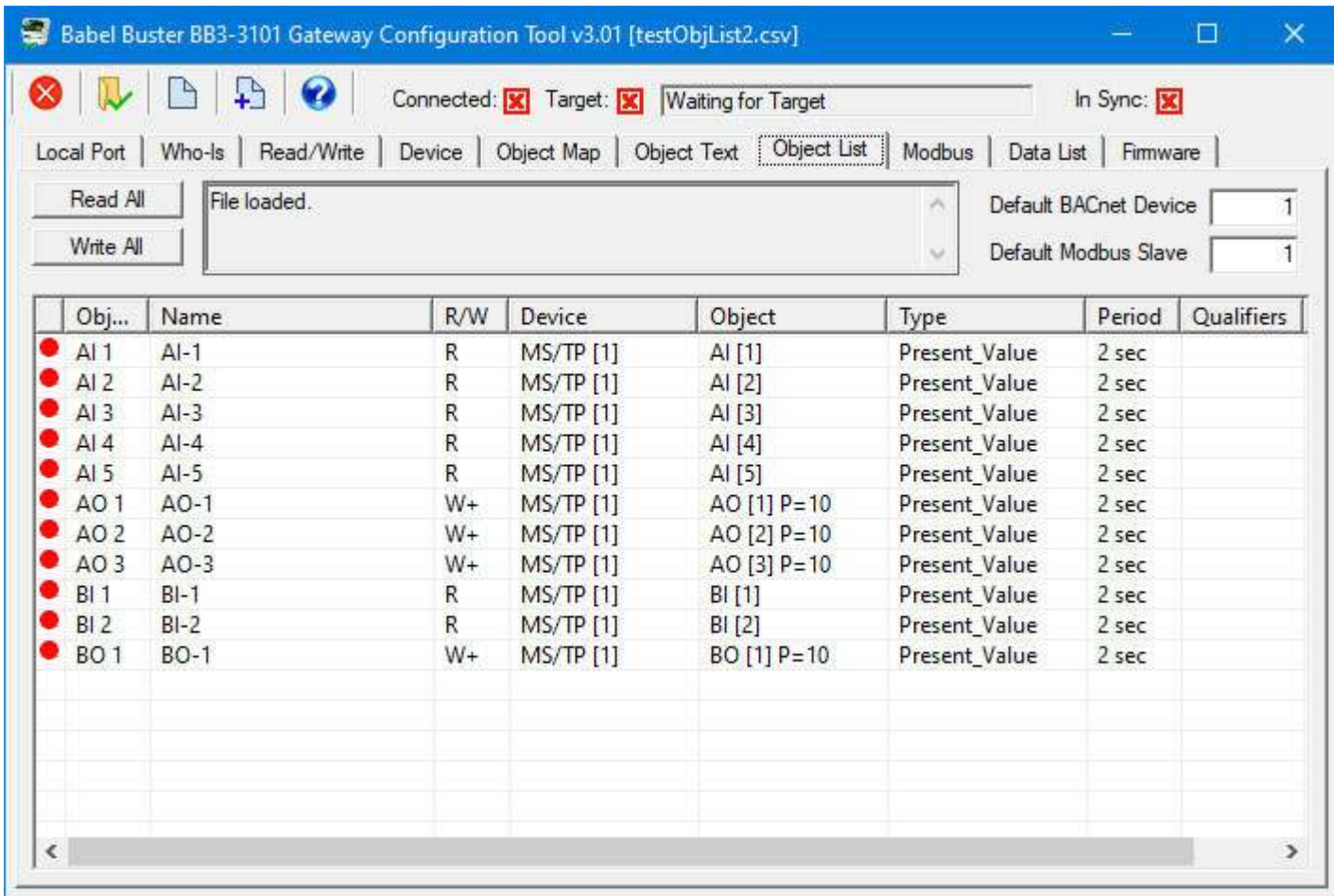
	A	B	C	D	E	F
1	ObjType	ObjInstance				
2	AI	1				
3	AI	2				
4	AI	3				
5	AI	4				
6	AI	5				
7	AO	1				
8	AO	2				
9	AO	3				
10	BI	1				
11	BI	2				
12	BO	1				
13						

The plain text looks like this:



```
testObjList2.csv - Notepad
File Edit Format View Help
ObjType,ObjInstance
AI,1
AI,2
AI,3
AI,4
AI,5
AO,1
AO,2
AO,3
BI,1
BI,2
BO,1
```

When this CSV file is loaded into the configuration tool, the resulting object map looks like the following. Note that the objects allocated inside the gateway are selected by the configuration tool to correspond as appropriately as possible to the objects in the target device. If you want to explicitly assign objects in the gateway rather than let the tool select defaults, then add the Object column in which you specify objects such as "AO-3" and "AI-10" as seen in a previous example.



3.4 Building Configuration from Scratch

When you built a configuration starting with a CSV file, object counts were calculated for you as the file was loaded. When you start from scratch and build a configuration manually, you need to start by deciding how many objects to allocate and what types of objects you will need. The gateway has a finite amount of memory available for buffering data for the BACnet objects. To provide maximum flexibility, that available memory is not divided up among a fixed set of objects. That resource pool can be reallocated any way you like, up to a total of 1024 objects.

Assuming the configuration tool is offline, not connected to any gateway device, you can allocate objects by entering numbers in the second column to the right of the list of object types on the Device page. Then click the 'Get Counts' button to accept those numbers as your configured set of objects.

If you the configuration tool is online, connected to a gateway via the MS/TP network, then clicking the 'Get Counts' button will query the gateway to ask it how many objects it currently has in its resource pool. If you need to change the number of objects in the gateway device, the procedure for doing that is outlined in section 7 of this user guide.

In the example illustrated below, we have allocated 10 Analog Input objects.

Babel Buster BB3-3101 Gateway Configuration Tool v3.01

Connected: ☒ Target: ☒ Waiting for Target In Sync: ☒

Local Port | Who-Is | Read/Write | Device | Object Map | Object Text | Object List | Modbus | Data List | Firmware

Object ID (device instance)

Device Name

Device Description

Device Location

☒ ☐ Auto-Reset Errors

Max Master

MS/TP Baud Rate

MS/TP Station/MAC Addr

New Password

Old Password

Object Allocation	Present	Pending
Analog Input Count	<input type="text" value="10"/>	<input type="text" value="10"/>
Analog Output Count	<input type="text" value="0"/>	<input type="text" value="0"/>
Analog Value Count	<input type="text" value="0"/>	<input type="text" value="0"/>
Binary Input Count	<input type="text" value="0"/>	<input type="text" value="0"/>
Binary Output Count	<input type="text" value="0"/>	<input type="text" value="0"/>
Binary Value Count	<input type="text" value="0"/>	<input type="text" value="0"/>
Multistate Input Count	<input type="text" value="0"/>	<input type="text" value="0"/>
Multistate Output Count	<input type="text" value="0"/>	<input type="text" value="0"/>
Multistate Value Count	<input type="text" value="0"/>	<input type="text" value="0"/>
Multistate Max State	<input type="text" value="0"/>	<input type="text" value="0"/>

Once you have allocated some objects (or retrieved object counts from a connected online gateway device), you may proceed to the Object Map page where you can enter all of the parameters for mapping that BACnet object in the gateway. Regardless of what the object is going to be used for, you need to provide some basic information to allow that object to really behave as a BACnet object. One of the requirements is that each object in a BACnet device is required to have a unique object name. Description is optional; however, some BACnet clients will expect to see a description in addition to object name.

After defining the minimum requirements for existing as an object, you then have the option of deciding what that object is going to do for you in the gateway. For the gateways in question in this user guide, the most common application is going to be reading or writing a register in a Modbus device. To map this object to a Modbus register, click on 'Map Modbus Object'. When you do that, additional windows will appear which allow you to select register number, register type, data format, slave address or TCP device number, and so on.

Babel Buster BB3-3101 Gateway Configuration Tool v3.01

Connected: ✗ Target: ✗ Waiting for Target In Sync: ✗

Local Port | Who-Is | Read/Write | Device | **Object Map** | Object Text | Object List | Modbus | Data List | Firmware

Object Type/Instance: Analog Input 1 Units: no_units

Object Name: Object 1

Description: My first scratch built object

☒ Read Periodic ☐ Write Periodic ☐ Set Default on Power-Up ☐ Enable Max Quite Time ✗

☐ Write on Delta ☐ Set Default on Comm Fail

Poll Rate (Sec): 0 Slope/Scale Factor: 0 Timeout (Sec): 0

Initial COV Increment: 0 Intercept/Offset: 0 Max. Quiet Time (Sec): 0

Initial COV Period: 0 Delta for Send: 0 Default Value: 0

Initial Relinquish Default: Read Fails before Fault: 0

☒ Map Modbus Object ☐ Map BACnet Object

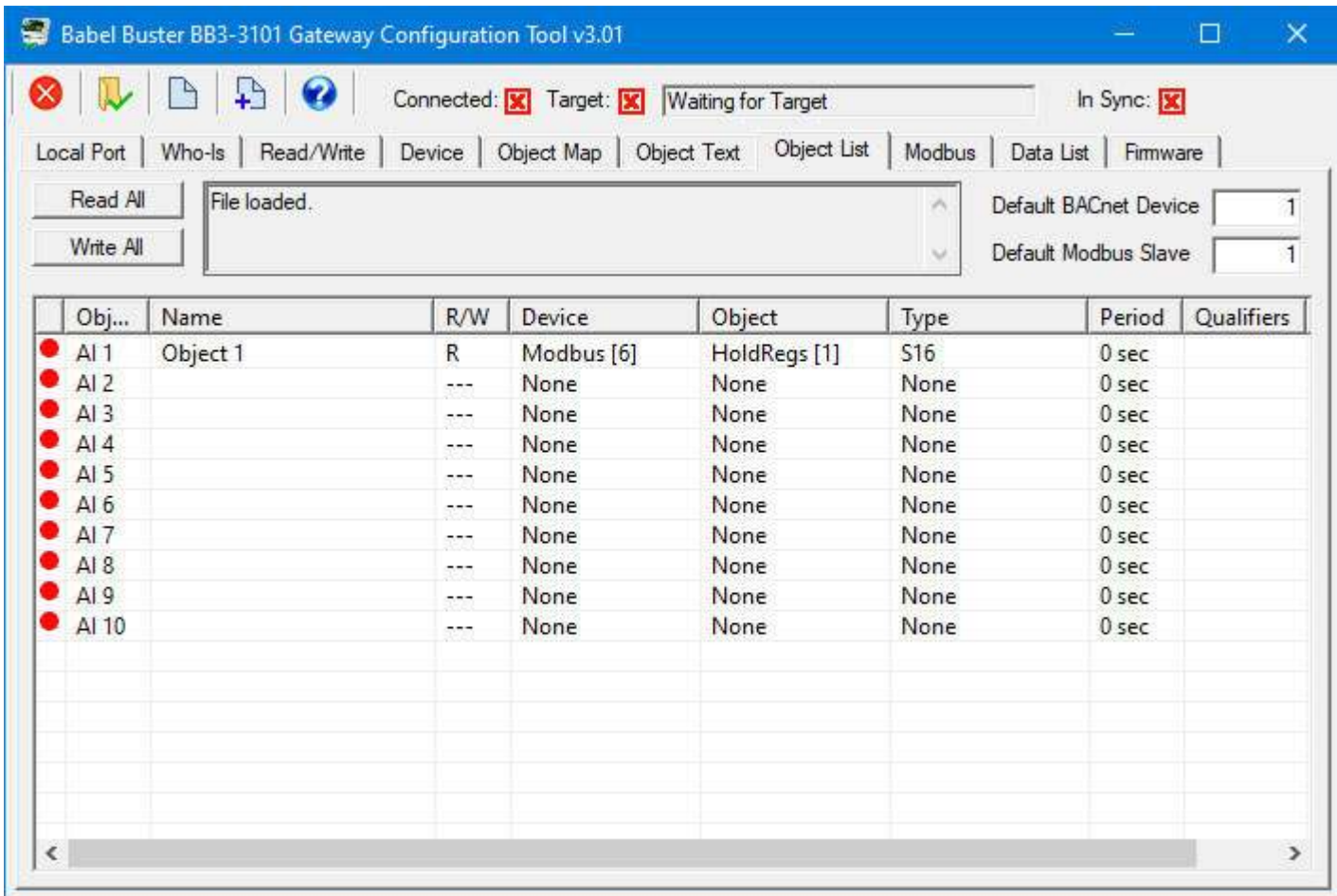
Register Number 1..N: 1 Unit/Slave Addr: 6 ☐ Member of Packed Register

Register Type: Holding Register (fc 16) Mask (Hex): 0000 ☐ Pack Mixed Object Types

Register Format: Signed 16-bit Fill (Hex): 0000

Insert Add Delete Read Device Write Device

Once you have entered object information and mapping information, this new object will now appear on the list on the 'Object List' page in the same manner as if you had imported a CSV file to configure the gateway.



In order to have a completely functioning BACnet device on the MS/TP network, you do need to define a few additional parameters on the Device page. From the BACnet perspective, these are properties found in the BACnet Device object for the gateway. As with all other objects in the gateway, the Device object needs to have a unique device name. Description and location are optional. You also need to assign an Object ID or device instance to the Device object. This number needs to be unique on your BACnet network, and this is the number that will be returned by the gateway in response to any Who-Is request on the network.

You must enter port parameters for the MS/TP port. Max Master will default to 127, and you should leave that number alone unless you understand what this number does and you understand your reasons for changing it. Set the baud rate to match the baud rate of your network. Set the MAC address to a number that is not already used on the network. Note that incorrect baud rate, incorrect max master, duplicate MAC address, or duplicate object instance will all cause communication problems, not only for the gateway but for other devices on the network.

Babel Buster BB3-3101 Gateway Configuration Tool v3.01

Connected: Target: Waiting for Target In Sync:

Local Port | Who-Is | Read/Write | Device | Object Map | Object Text | Object List | Modbus | Data List | Firmware

Object ID (device instance)

Device Name

Device Description

Device Location

☐ Auto-Reset Errors

Max Master

MS/TP Baud Rate

MS/TP Station/MAC Addr

New Password

Old Password

Object Allocation	Present	Pending
Analog Input Count	<input type="text" value="10"/>	<input type="text" value="10"/>
Analog Output Count	<input type="text" value="0"/>	<input type="text" value="0"/>
Analog Value Count	<input type="text" value="0"/>	<input type="text" value="0"/>
Binary Input Count	<input type="text" value="0"/>	<input type="text" value="0"/>
Binary Output Count	<input type="text" value="0"/>	<input type="text" value="0"/>
Binary Value Count	<input type="text" value="0"/>	<input type="text" value="0"/>
Multistate Input Count	<input type="text" value="0"/>	<input type="text" value="0"/>
Multistate Output Count	<input type="text" value="0"/>	<input type="text" value="0"/>
Multistate Value Count	<input type="text" value="0"/>	<input type="text" value="0"/>
Multistate Max State	<input type="text" value="0"/>	<input type="text" value="0"/>

3.5 Verifying Configuration in the Gateway Device

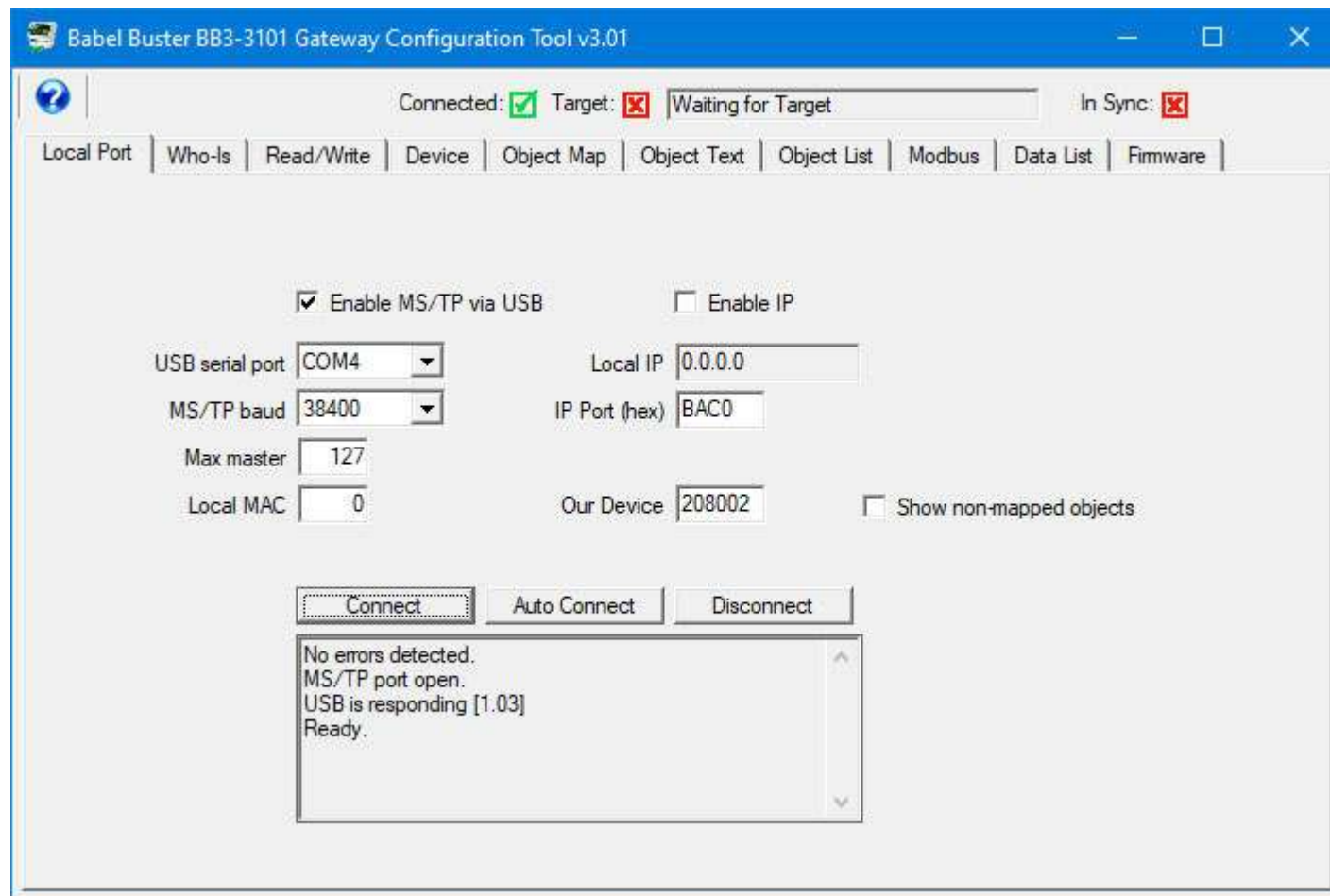
One of the most common problems in the case of a gateway not functioning as desired is entering configuration changes into the configuration tool software but forgetting to actually transmit that information to the gateway device. The preceding discussions about building configurations talked about the available methods for creating a configuration. You do need to refer to later sections in this user guide to review the full details of everything configured on each of the pages in the configuration tool, and in particular, how you send that information from the tool software to the actual physical device. We are getting a little ahead of ourselves by talking about how to verify configuration in the device at this point in the user guide, but it is a topic general in nature, so it is provided here for future reference.

If you think the gateway is configured, but it is not functioning as desired, read the configuration from the device to be sure what is actually in the device is the configuration you thought was in the device. You will notice throughout the configuration tool pages that there are a number red 'X' boxes or green check boxes (red and green dots on the list pages have the same meaning). Pay attention to these. If any boxes are the red X, it means what you see on the screen may not be in sync with what is in the device. You have two options to get the red X to become a green check mark: Either read from the device or write to the device. If you read from the device, the tool will display whatever it found in the device (potentially replacing any changes you had made on the screen). If you write to the device, you will replace the configuration previously in the device with whatever is displayed on the screen.

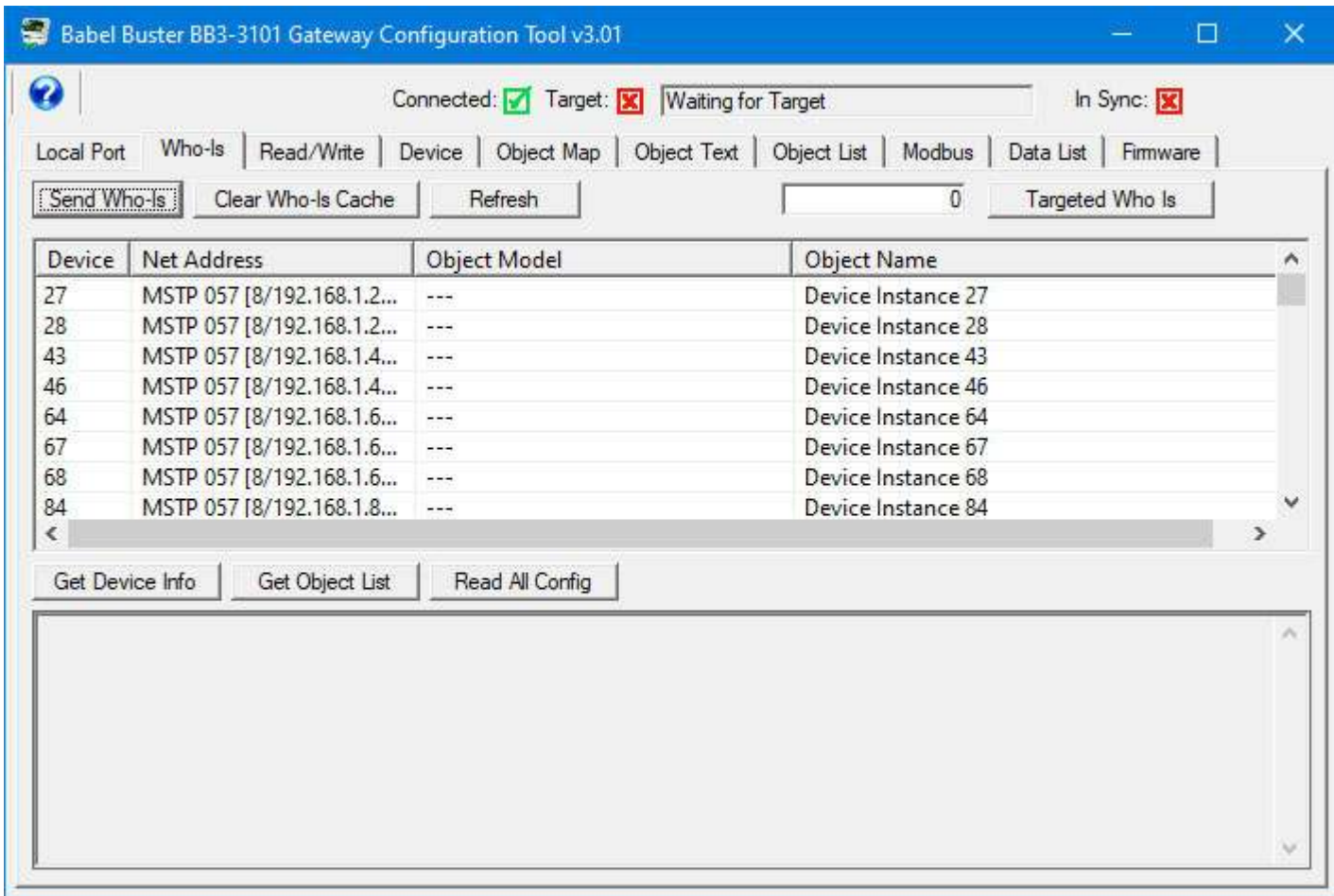
Until all of the red icons are replaced with green icons in the configuration tool, you have

no assurance that the configuration in the device actually is what you think it is. If you are having problems with a gateway not doing what you want, make sure you are looking at all green icons first, then review the configuration to verify that the mappings are asking for the registers you intended, and so on.

Start by connecting the tool to the network using the Local Port page illustrated below. Refer to section 4 for full detail on what all you need to do on this page and what your options are for connecting to the physical network.

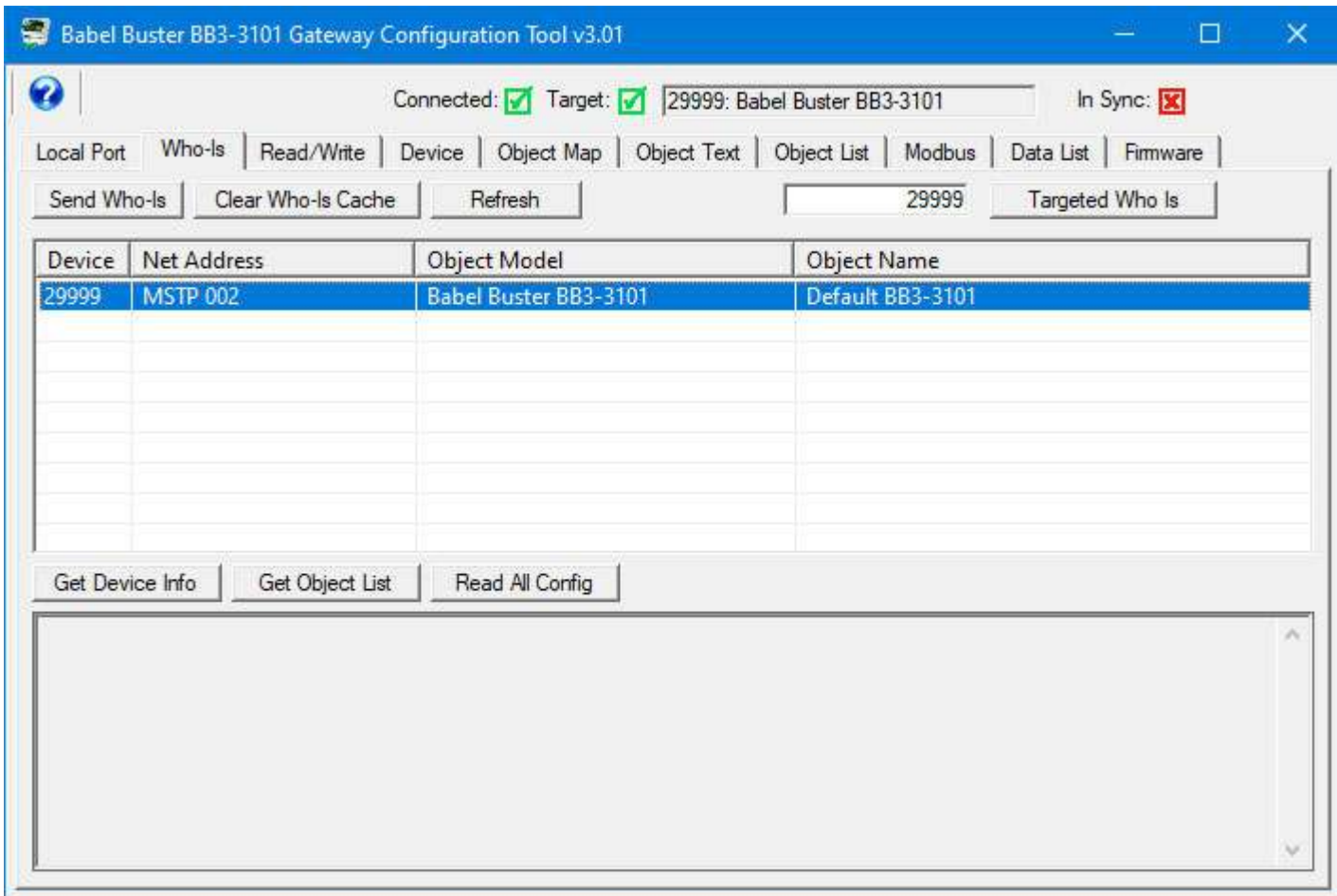


The configuration tool will transmit the BACnet "Who-Is" message on the network when you click "Send Who-Is". The list will begin to populate, and may be a long busy list on a large network. After you click the Refresh button a time or two, the tool will show a list of all BACnet devices found on the network. Select your gateway from this list. Refer to section 5 in this user guide for more about the Who-Is page.

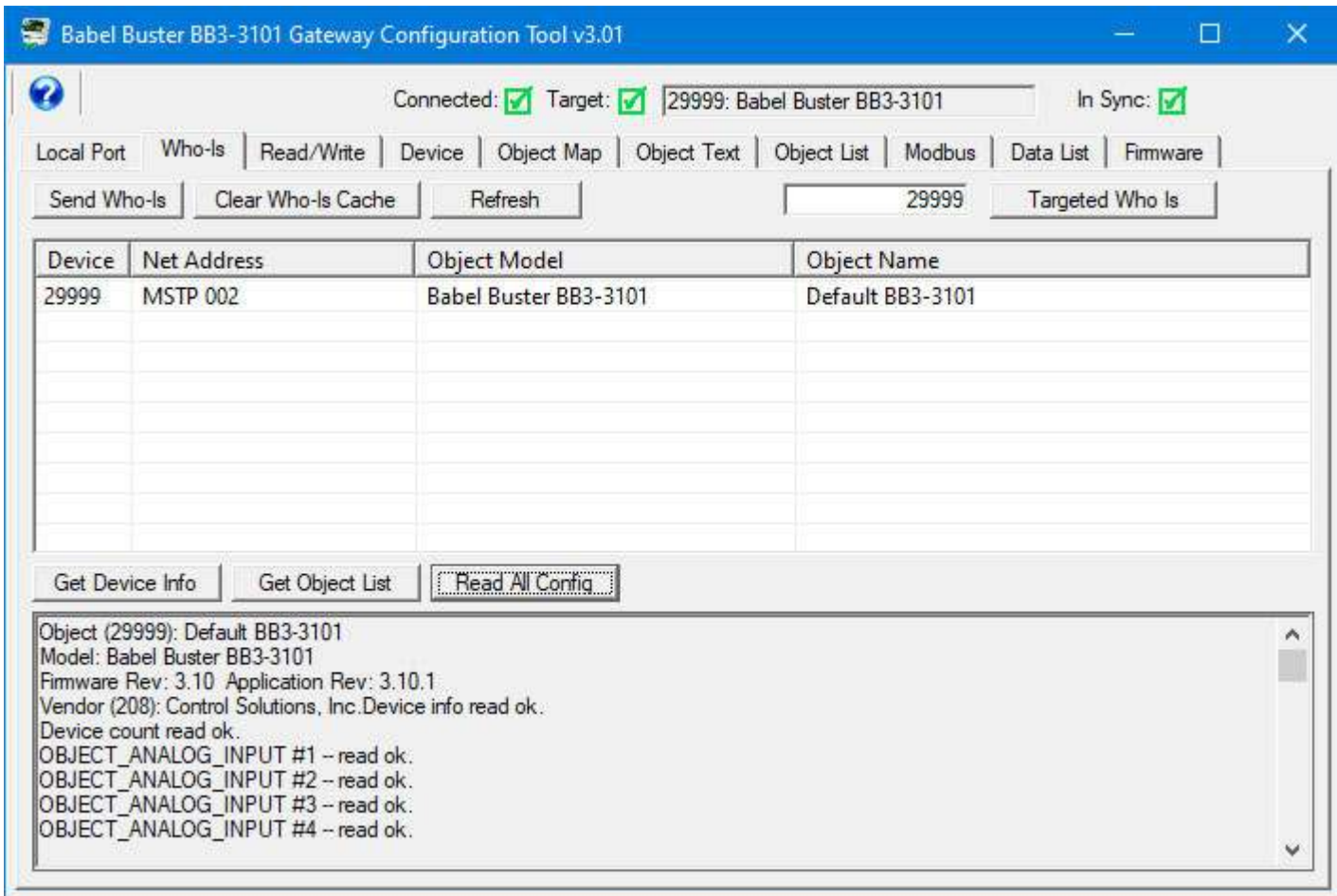


If you are on a busy network, you can enter the device instance nexted to the "Targed Who Is" button, then click that button. Once the device instance is found, you may click Refresh to also retriive the model and name.

If you have not previously set the device instance in a brand new BB3-3101, its device instance will be 20800. Use that default instance instead of the 29999 shown in this example.



You can go to each of 5 different pages in the tool and click the 'Read Device' button on each page to get the information from the device that will appear on that page. The short cut method is to simply click the "Read All Config" button on the Who-Is page. While reading all of the configuration information from the connected gateway, progress will continue to update in the log window at the bottom of the Who-Is page. Wait for this activity to stop, and then proceed to review the information on other pages in the tool.



Following the "read all" process, the Device page will now show the BACnet Device object properties as well the counts of different types of objects currently found in the gateway.

Babel Buster BB3-3101 Gateway Configuration Tool v3.01

Connected: ☒ Target: ☒ 29999: Babel Buster BB3-3101 In Sync: ☒

Local Port | Who-Is | Read/Write | Device | **Object Map** | Object Text | Object List | Modbus | Data List | Firmware

Object ID (device instance) 29999

Device Name
Default BB3-3101

Device Description

Device Location
Unassigned

☒ ☐ Auto-Reset Errors

Max Master 127

MS/TP Baud Rate 38400

MS/TP Station/MAC Addr 2

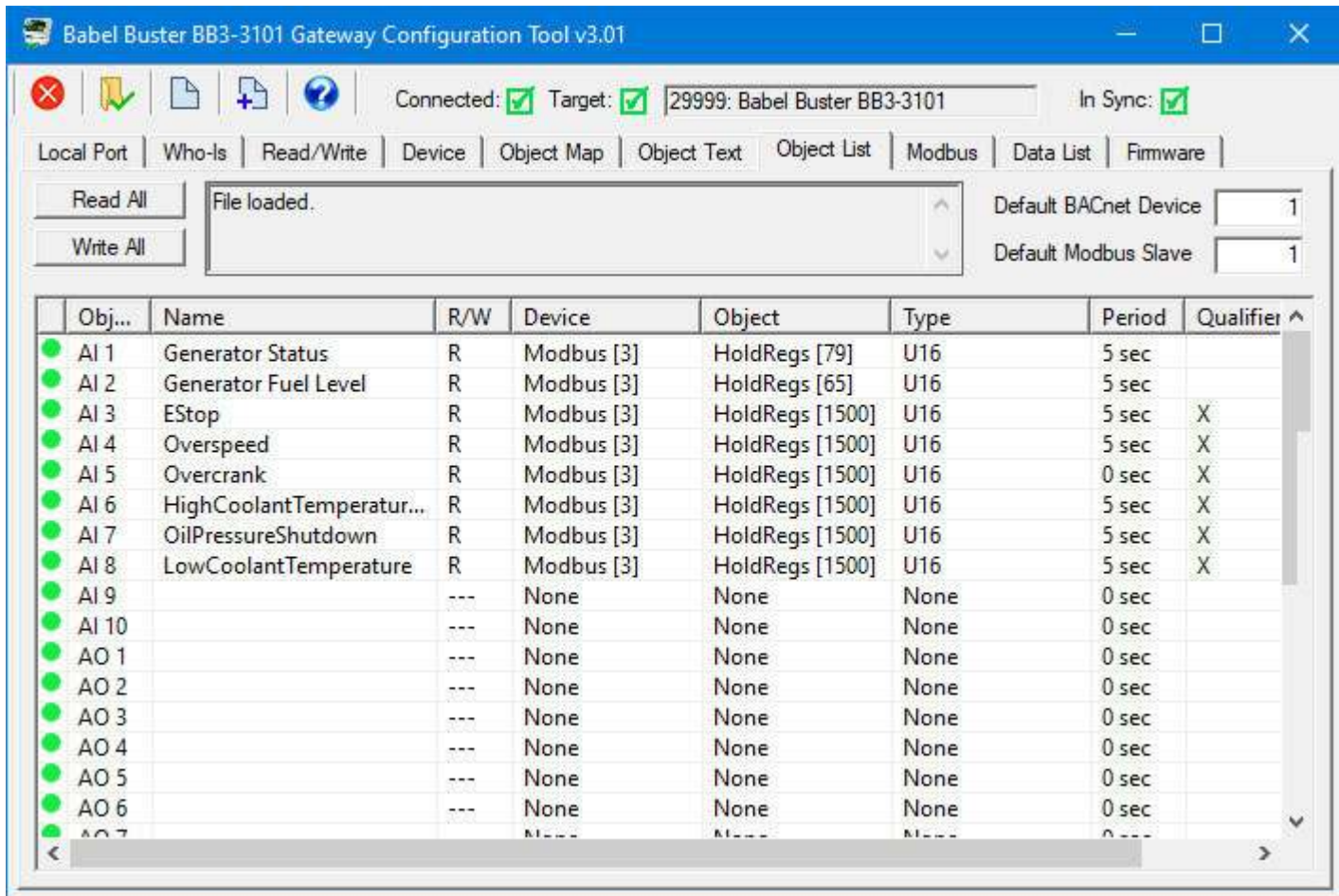
New Password

Old Password

Object Allocation	Present	Pending
Analog Input Count	10	10
Analog Output Count	10	10
Analog Value Count	10	10
Binary Input Count	10	10
Binary Output Count	10	10
Binary Value Count	10	10
Multistate Input Count	5	5
Multistate Output Count	5	5
Multistate Value Count	5	5
Multistate Max State	10	10

Device count read ok.

The Object List page will show the list of objects in the device and display mapping information. You can also go to the Object Map page and review (and optionally change) the details of the object mapping.

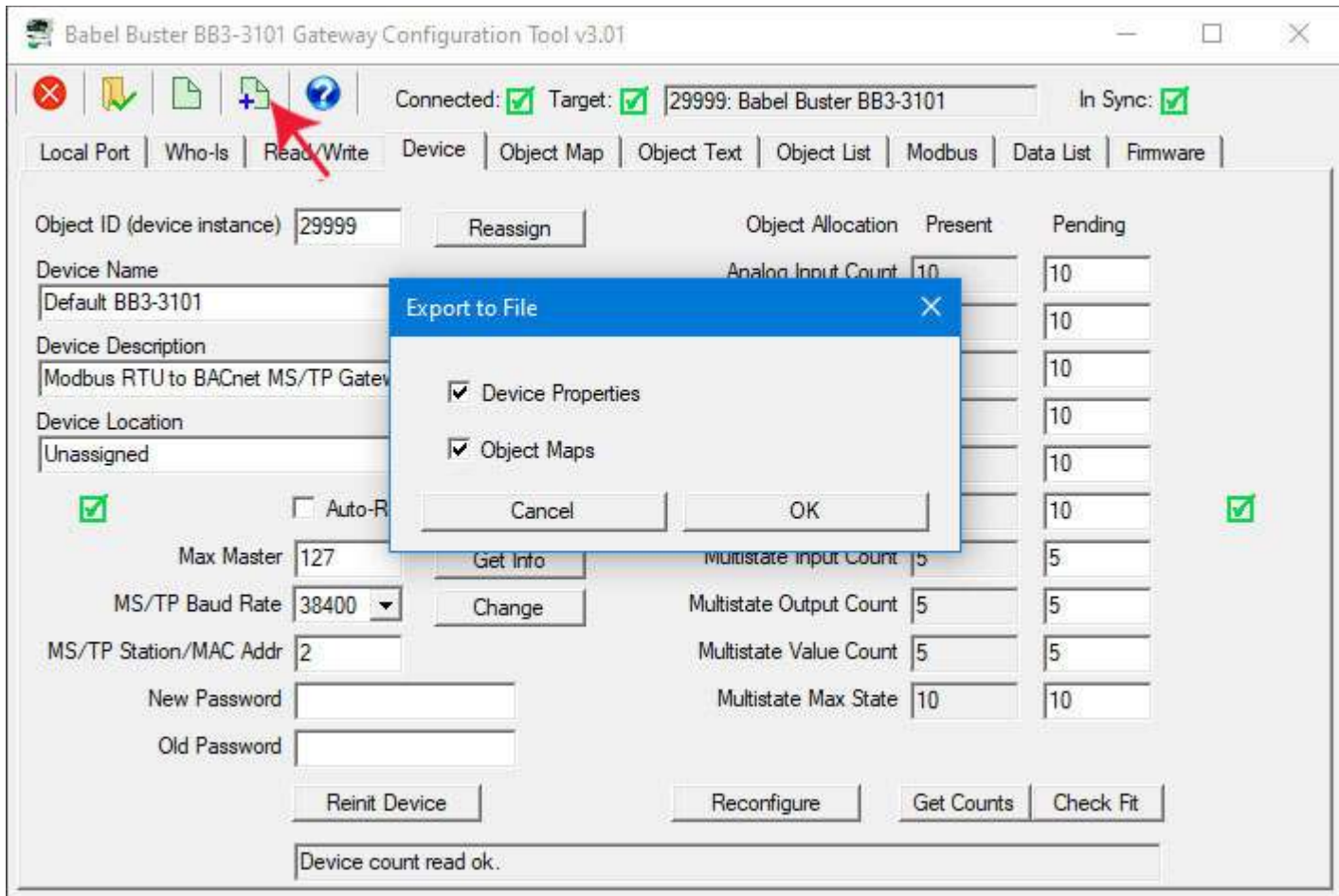


3.6 How to Change Object Counts While Retaining Object Mappings

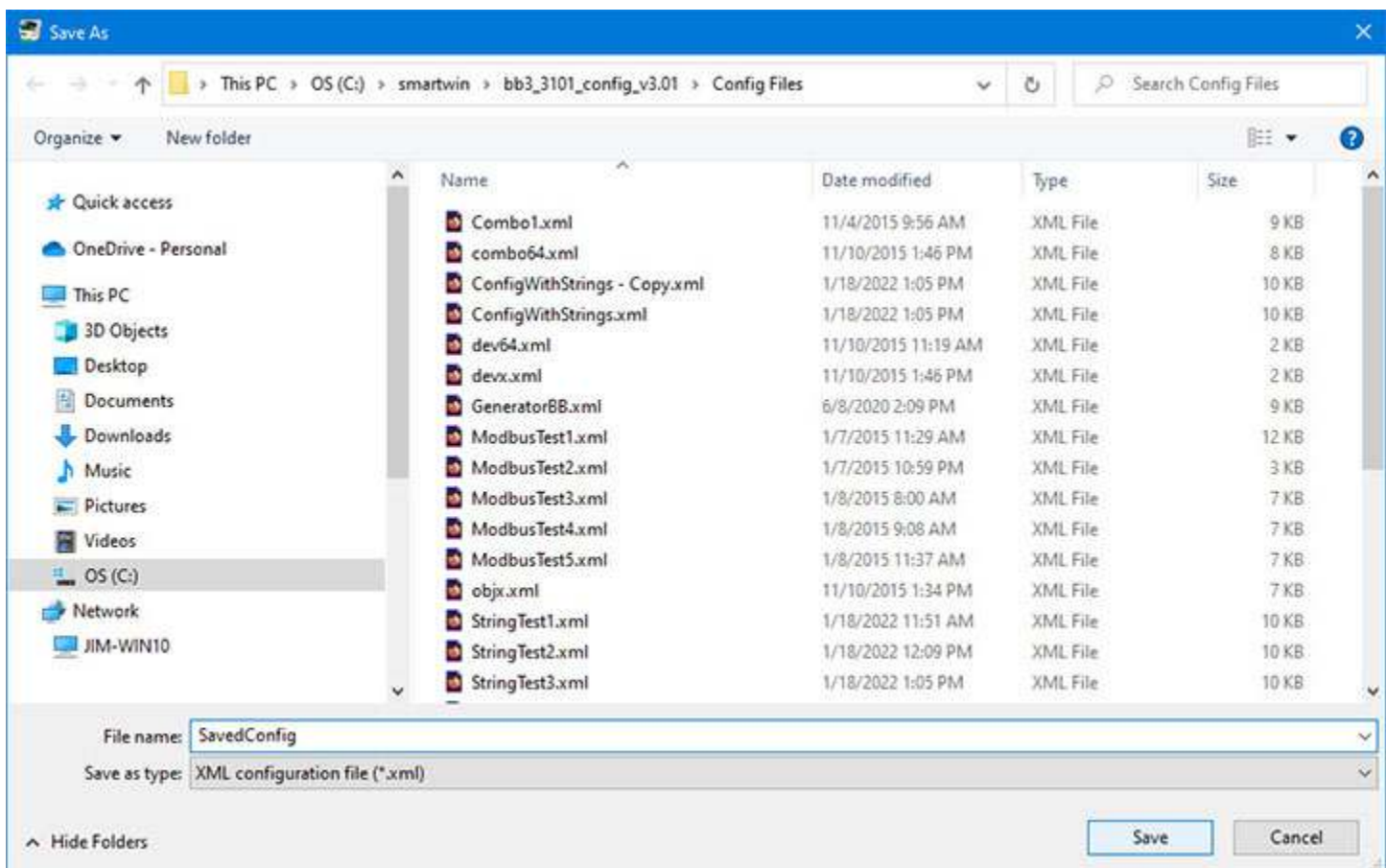
This topic is provided here for future reference. Until you have configured the gateway for the first time, you can disregard this procedure. But at some point, you may find that you need to add some objects to a gateway that had been previously configured. The details for establishing the initial configuration are talked about both above and in the next several sections of this user guide. However, the process for modifying the object counts in a device that has already been configured requires going back and forth between more than one page.

It is important to note that any time you change the object count allocation in the physical gateway device, you will erase whatever configuration had been in the previously allocated objects. You are effectively wiping the slate clean. While starting from a clean slate can be advantageous at times, you may not want to wipe the slate clean, just change counts. The procedure for effectively doing this is to save your objects, wipe the slate clean, then put the previous objects back in place but within the newly allocated set of objects.

Start by connecting to the device and reading all configuration from the device as outlined in the above discussion (3.5) on verifying your configuration. After that process is complete, go to the Device page and click the new file icon. You have the option of exporting device properties, object maps, or both. Although you will only be needing the object maps for this procedure, it is a good idea to save everything in the event you want to completely replicate the gateway device. Click OK to save the file.



The familiar file dialog will appear. Find a directory where you want to put your file, and enter a file name for the configuration XML file about to be created.



Once you have saved the configuration, go to the Device page and enter the new counts in the column to the right of the list of available object types. The box that you are allowed to edit is the pending or proposed new count. The grayed out box is the counts as they actually exist prior to any changes.

Object Allocation	Present	Pending
Analog Input Count	10	100
Analog Output Count	10	10
Analog Value Count	10	10
Binary Input Count	10	100
Binary Output Count	10	10
Binary Value Count	10	10
Multistate Input Count	5	5
Multistate Output Count	5	5
Multistate Value Count	5	5
Multistate Max State	10	10

You are limited to a maximum of 1,024 total objects, but you may allocate them any way you wish. If you attempt to allocate more than the maximum number of objects as depicted in the screen shot below, then the counts will be forced within bounds.

Babel Buster BB3-3101 Gateway Configuration Tool v3.01 [SavedConfig.xml]

Connected: ☒ Target: ☒ 29999: Babel Buster BB3-3101 In Sync: ☒

Local Port | Who-Is | Read/Write | Device | Object Map | Object Text | Object List | Modbus | Data List | Firmware

Object ID (device instance) 29999 Reassign

Device Name
Default BB3-3101

Device Description
Modbus RTU to BACnet MS/TP Gateway

Device Location
Unassigned

☒ ☐ Auto-Reset Errors

Max Master 127 Get Info

MS/TP Baud Rate 38400 Change

MS/TP Station/MAC Addr 2

New Password

Old Password

Reinit Device

Reconfigure

Get Counts

Check Fit

File saved.

Object Allocation	Present	Pending
Analog Input Count	10	500
Analog Output Count	10	500
Analog Value Count	10	500
Binary Input Count	10	0
Binary Output Count	10	0
Binary Value Count	10	0
Multistate Input Count	5	0
Multistate Output Count	5	0
Multistate Value Count	5	0
Multistate Max State	10	10

It is recommended that you click the "Check Fit" button to be sure the object counts you have requested are within bounds. The example above is not. When you click Check Fit, counts will be limited and the example illustrated above will lead to the result illustrated below.

Babel Buster BB3-3101 Gateway Configuration Tool v3.01 [SavedConfig.xml]

Connected: ☒ Target: ☒ 29999: Babel Buster BB3-3101 In Sync: ☒

Local Port | Who-Is | Read/Write | Device | Object Map | Object Text | Object List | Modbus | Data List | Firmware

Object ID (device instance) 29999 Reassign

Device Name
Default BB3-3101

Device Description
Modbus RTU to BACnet MS/TP Gateway

Device Location
Unassigned

☒ ☐ Auto-Reset Errors

Max Master 127 Get Info

MS/TP Baud Rate 38400 Change

MS/TP Station/MAC Addr 2

New Password

Old Password

Reinit Device

Reconfigure

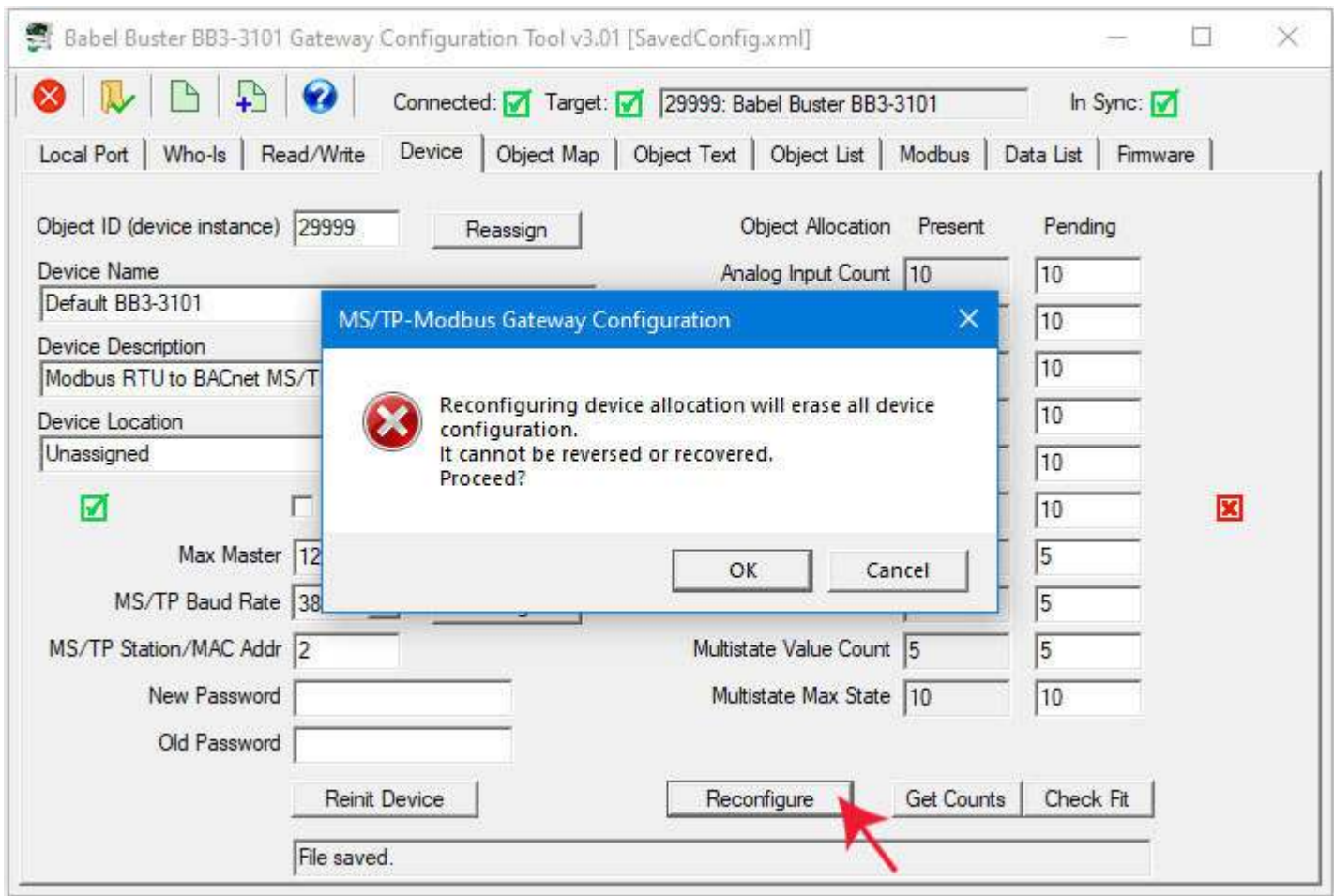
Get Counts

Check Fit

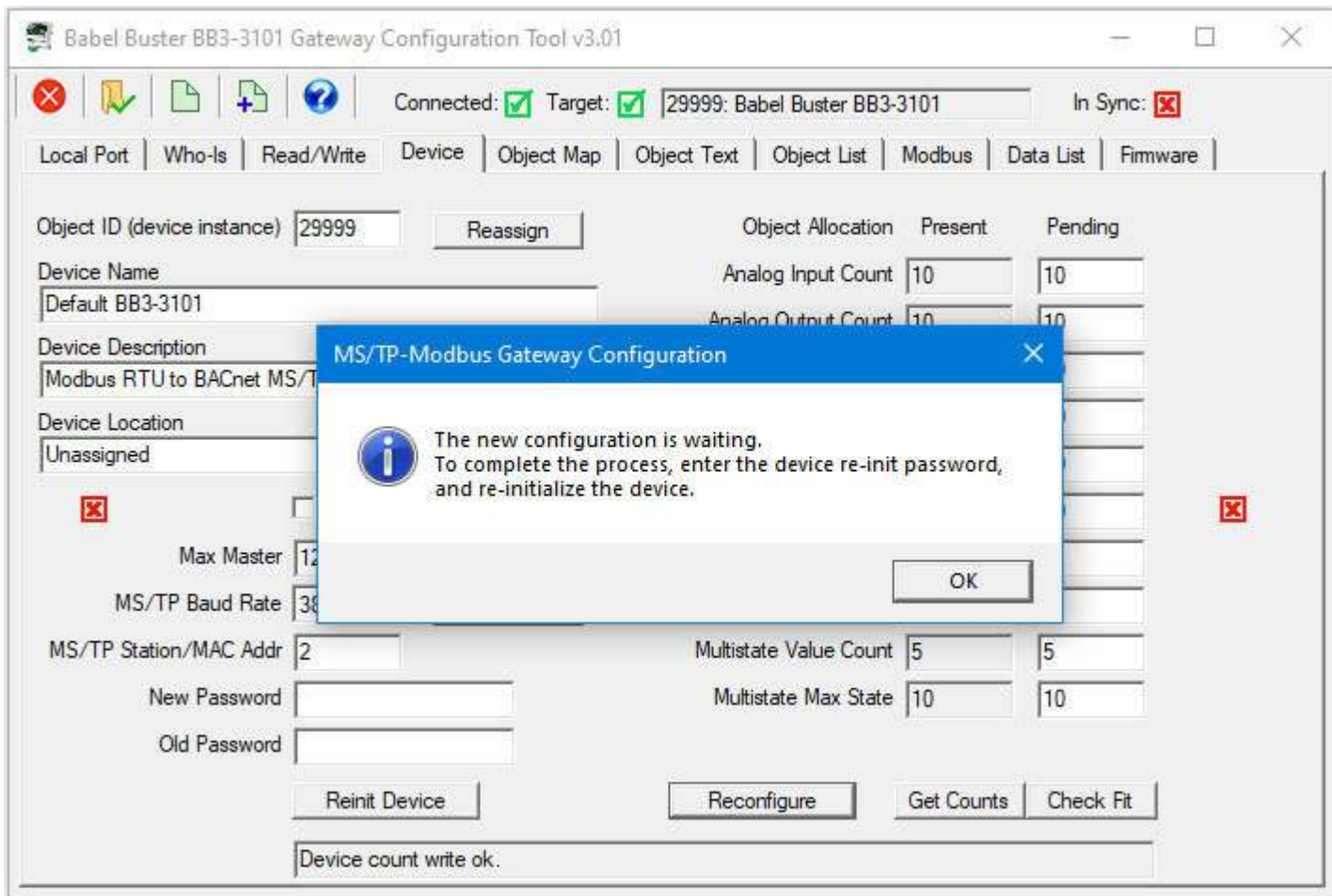
File saved.

Object Allocation	Present	Pending
Analog Input Count	10	500
Analog Output Count	10	500
Analog Value Count	10	24
Binary Input Count	10	0
Binary Output Count	10	0
Binary Value Count	10	0
Multistate Input Count	5	0
Multistate Output Count	5	0
Multistate Value Count	5	0
Multistate Max State	10	10

After entering the new counts and verifying them by clicking Check Fit, click the Reconfigure button.



A dialog appears reminding you of what you just read above about wiping the slate clean. If you click OK, then the new counts are sent to the gateway device.



You have one last opportunity to abandon your changes. The new counts are in the device but waiting for the final authorization to act on them. If you skip the next step of clicking the Reinit Device button, and power cycle the device instead, nothing will have changed in the device (and reconnecting and clicking Get Counts will retrieve the old counts and all configuration is still intact).

You must enter the reinitialize password in the Old Password window before the reinitialization will be accepted. This defaults to "buster" when initially shipped from the factory. Enter the password and then click the Reinit Device button.

Babel Buster BB3-3101 Gateway Configuration Tool v3.01 [SavedConfig.xml]

Connected: ☒ Target: ☒ 29999: Babel Buster BB3-3101 In Sync: ☒

Local Port | Who-Is | Read/Write | Device | Object Map | Object Text | Object List | Modbus | Data List | Firmware

Object ID (device instance) 29999 Reassign

Device Name
Default BB3-3101

Device Description
Modbus RTU to BACnet MS/TP Gateway

Device Location
Unassigned

☒ ☐ Auto-Reset Errors

Max Master 127 Get Info

MS/TP Baud Rate 38400 Change

MS/TP Station/MAC Addr 2

New Password

Old Password buster

Reinit Device

Reconfigure Get Counts Check Fit

Device re-init sent.

Object Allocation	Present	Pending
Analog Input Count	10	10
Analog Output Count	10	10
Analog Value Count	10	10
Binary Input Count	10	10
Binary Output Count	10	10
Binary Value Count	10	10
Multistate Input Count	5	5
Multistate Output Count	5	5
Multistate Value Count	5	5
Multistate Max State	10	10

If you are watching the device itself just after clicking Reinit Device, you will see lights inside change and become red for a short time. After the red lights go out, you may now continue. If you do not have the device in sight, just wait for 30 seconds or so, and attempt the Get Counts noted next. If you get a timeout message, wait a bit longer and try again.

Babel Buster BB3-3101 Gateway Configuration Tool v3.01

Connected: ☒ Target: ☒ 29999: Babel Buster BB3-3101 In Sync: ☒

Local Port | Who-Is | Read/Write | Device | Object Map | Object Text | Object List | Modbus | Data List | Firmware

Object ID (device instance) 29999 Reassign

Device Name
Default BB3-3101

Device Description
Modbus RTU to BACnet MS/TP Gateway

Device Location
Unassigned

☒ Auto-Reset Errors

Max Master 127 Get Info

MS/TP Baud Rate 38400 Change

MS/TP Station/MAC Addr 2

New Password

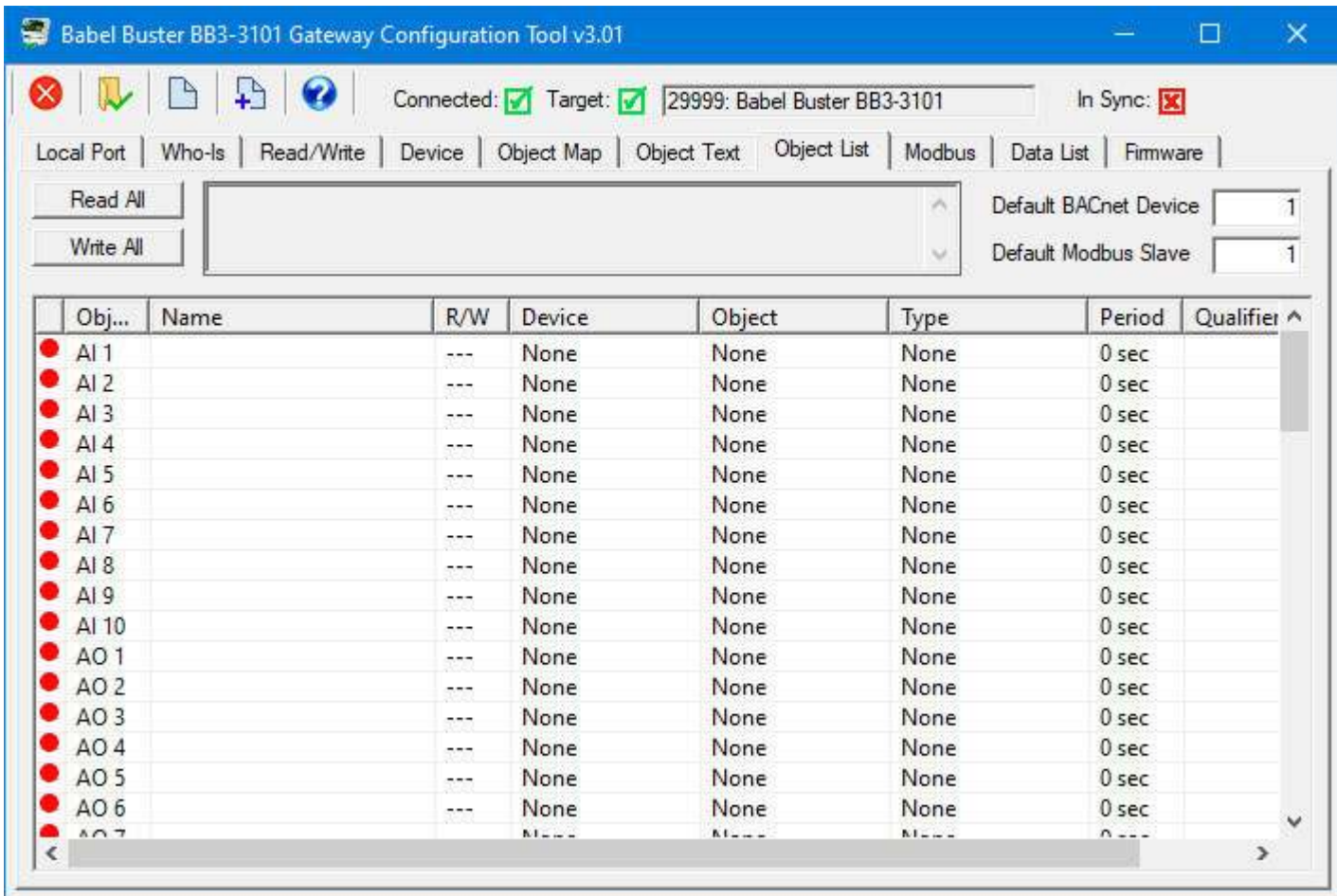
Old Password

Reinit Device Reconfigure Get Counts Check Fit

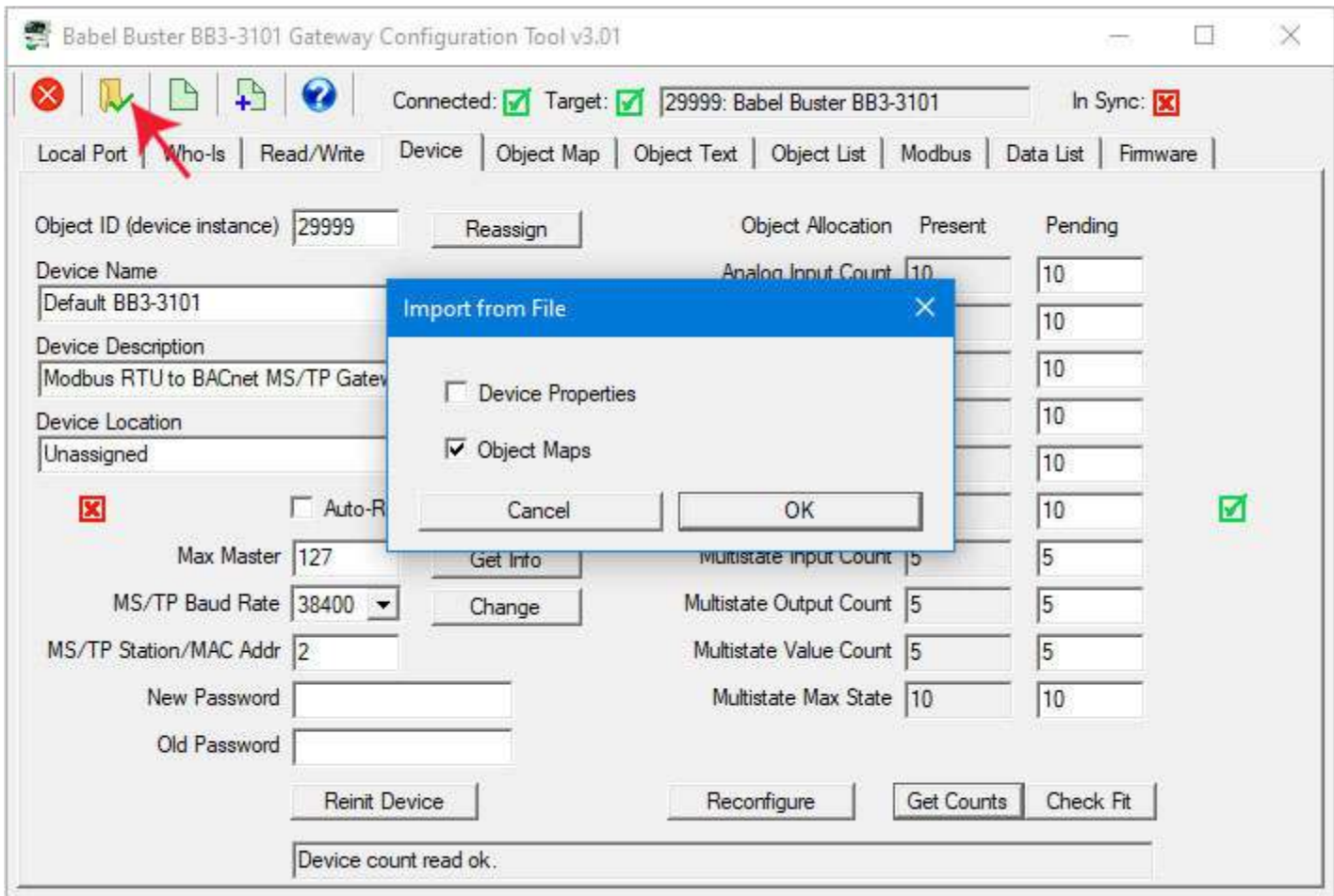
Device count read ok.

Object Allocation	Present	Pending
Analog Input Count	10	10
Analog Output Count	10	10
Analog Value Count	10	10
Binary Input Count	10	10
Binary Output Count	10	10
Binary Value Count	10	10
Multistate Input Count	5	5
Multistate Output Count	5	5
Multistate Value Count	5	5
Multistate Max State	10	10

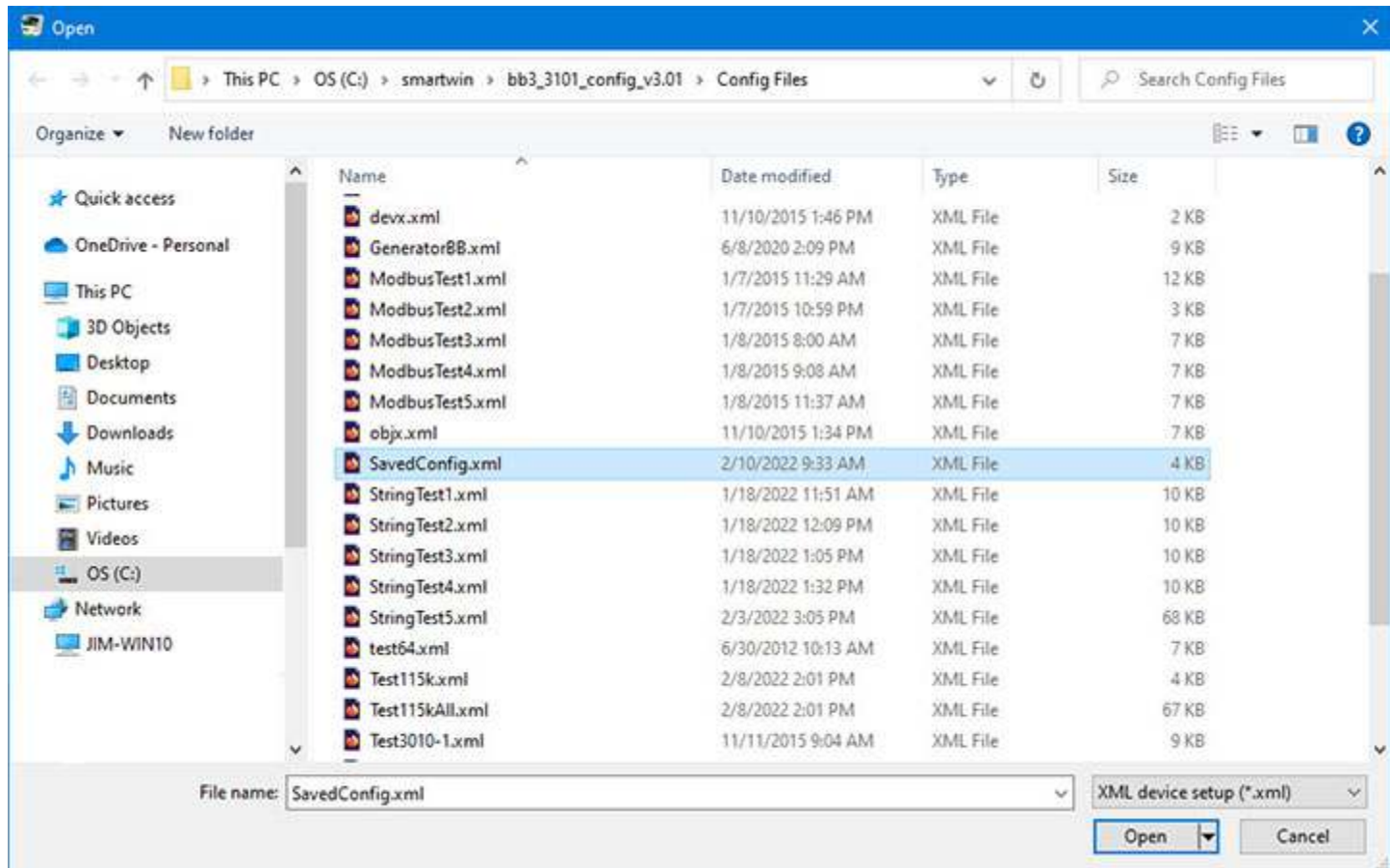
Following the reinitialization (which results in a BACnet warmstart execution), click get Counts to verify the updated object counts in the device. Although you probably didn't change any other Device object parameters, click the Get Info page anyway just to get the entire Device page in sync (all icons green).



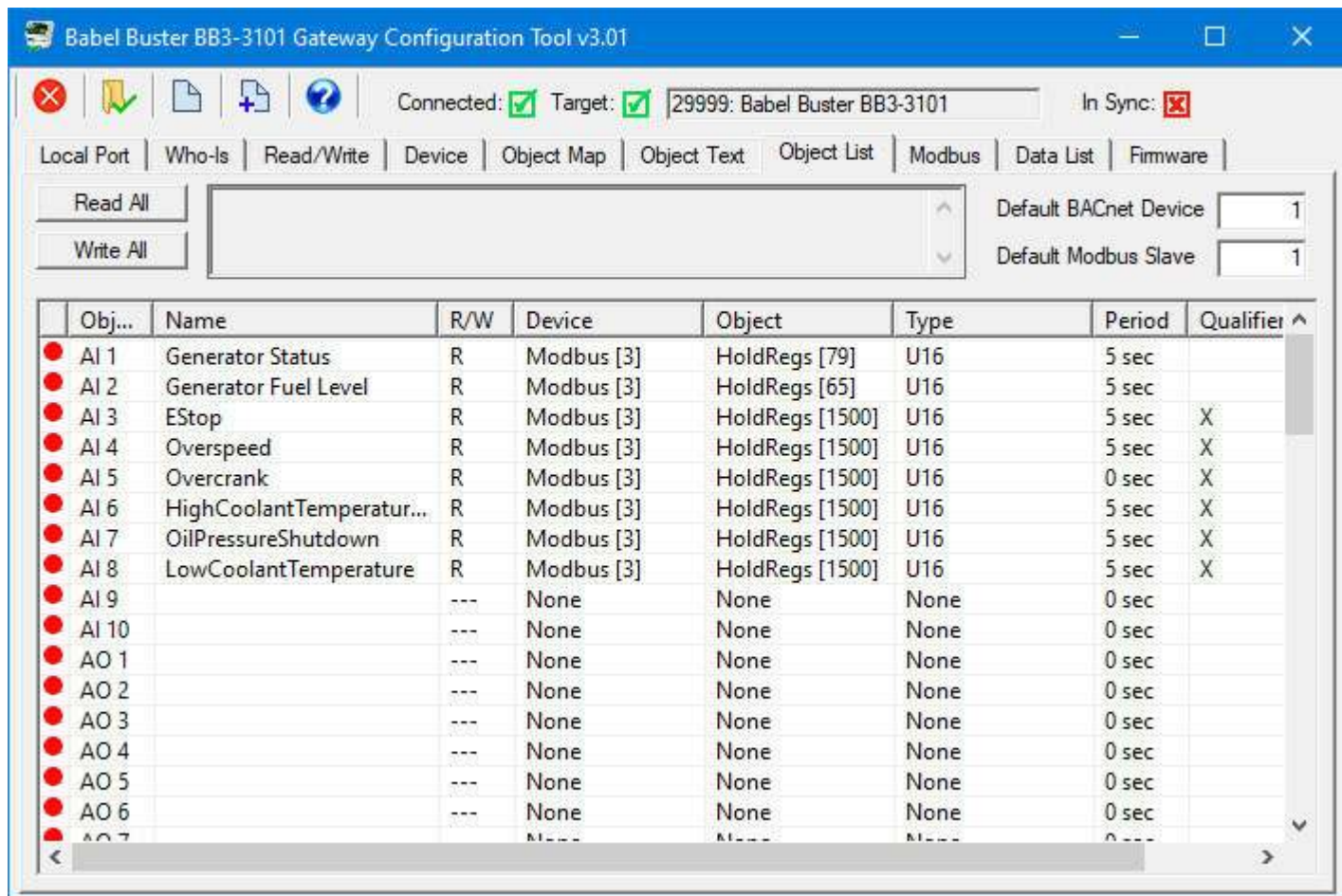
The configuration tool knows you just wiped the slate clean in the device, so it wipes its own slate clean as well. There will be no object mapping information displayed, only blank objects (with the number of objects corresponding to the object counts on the Device page).



Now it is time to recover the saved objects. Click on the file open icon. Select only the object maps. Un-check or deselect "device properties" as these will wipe out the object counts you just took time to change. Click OK only after making sure only the object maps are selected for importing.

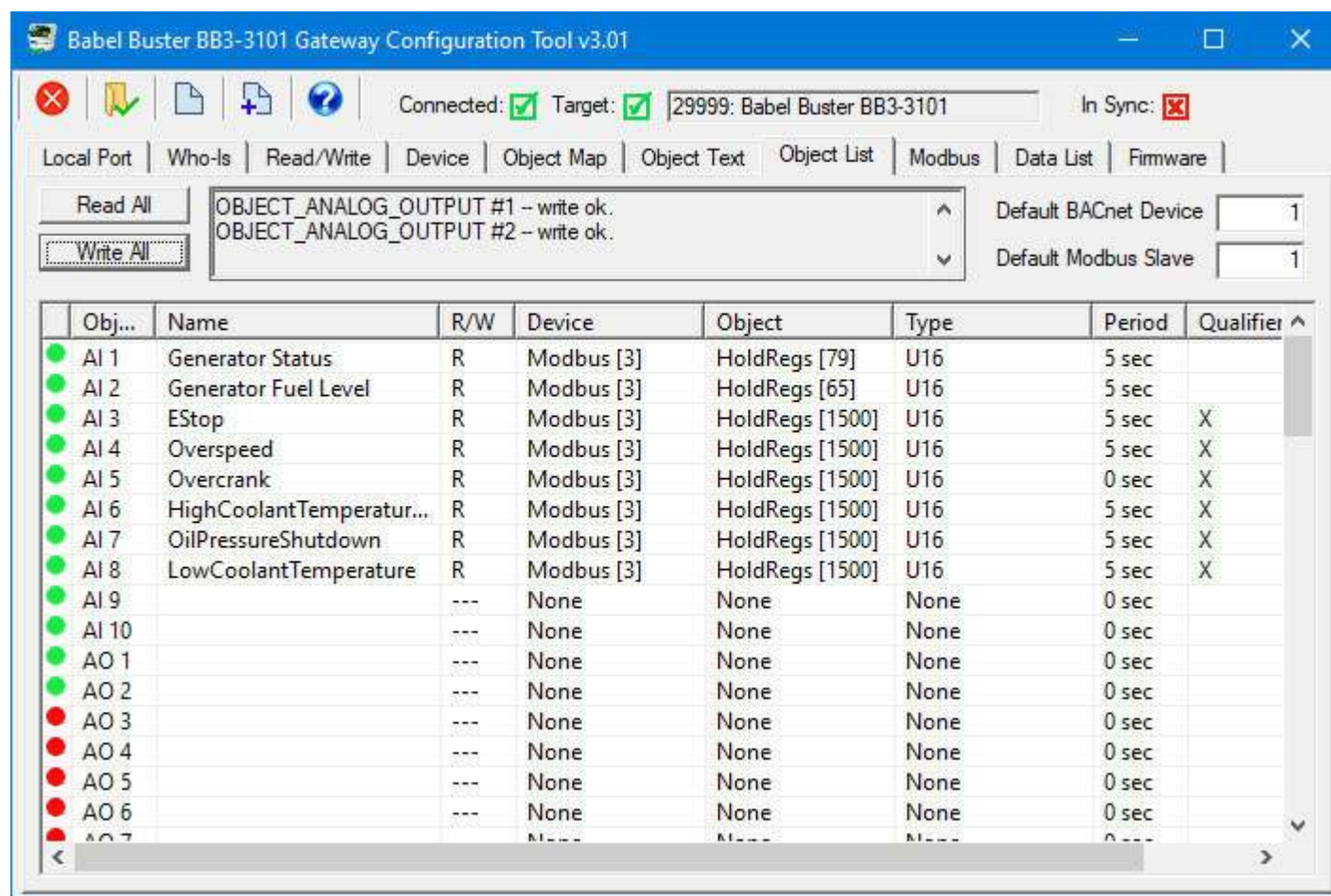


The same familiar file dialog will appear. Find the file you had just saved a few steps prior, select it and click Open.



Your saved configuration will now appear on the Object List page. This example is short - your actual configuration will likely fill the entire screen.

The red icons in the first column tell you that this information is not yet physically in the gateway. From the Object List page, click Write All. This will begin the process of sending the entire list of objects and their object mappings to the gateway device. As each object is sent, its icon turns green.



Upon completion of the Write All process, all of the icons on the Object List page will be green. The items on the Device page were already in sync from preceeding steps. The only remaining thing the tool doesn't know about is the Modbus page. If you go there and click the Read Device button on the far upper right, you should see the In Sync icon in the top right of the screen now turn green as well. The "In Sync" icon that appears in the upper right corner of every page is the global "everything is in sync". If red, it means there is at least one red icon on some page somewhere in the configuration tool.

3.7 Converting Modbus Register to Multiple BACnet Objects

Modbus is notorious for packing multiple data points into a single holding register, especially when a collection of status bits are just one bit each. When converting those status bits to some other protocol, especially BACnet, it is user unfriendly, if not downright non-functional, to put that collection of bits into something like a floating point Analog Input object. Thankfully, Babel Buster gateways can split a single holding register full of status bits into a collection of individual Binary objects.

The key to splitting a packed holding register into 16 different objects is to set up object maps that look like you are reading the same register 16 times. Internally, the Babel

Buster optimizes the query. It sees you are reading the same register over and over, and does an actual read only once and then shares the data with all 16 object maps. Each time, the original Modbus data is processed according to the map's criteria, and the result is placed in the given target object. Each holding register can map to as many as 16 Binary objects, but you do not need to map all 16 bits. You can choose any number up to 16 and selectively mask only those bits into the appropriate number of objects.

Each of the object maps are set up the same as you would for reading any other unsigned 16-bit holding register. The only parameter you need to set differently is to put a non-zero value in the "mask" field. This mask is logically AND-ed with the Modbus data, that result is right justified, and then the final result is placed into the BACnet object if reading from Modbus. Reverse the process if writing to Modbus. You also need to check the box that says "Member of Packed Register" in the configuration tool software. This "Member" status tells the gateway to check the next object map and see if it references the same register, and share data with that object if so. If the "Pack Mixed Object Types" is selected, the gateway scans all object maps looking for more matching maps to share data with. The gateway will operate far more efficiently if you arrange the "Member of Packed Register" maps in sequential order, but you can force it to scan all objects each time if there is some reason to, such as placing some bits in Binary objects but placing a multi-bit field in an Analog object.

The same masking tricks can be used to write a packed Modbus holding register. There is one additional parameter that can be useful when writing, namely the "fill" parameter. If the register contains a bit that should always be set no matter what, you use the same values that you would for mask, except put them in the "fill" window instead. Just like the "mask" is logically AND-ed with the data, the "fill" is logically OR-ed with the data before sending it out to the Modbus device.

Documentation tends to be slightly different for every Modbus device. But if your device packs multiple bits into a single holding register, the documentation will note up to 16 different items found at the same register number or address. The bits may be identified with "Bn" or "Dn" or just "bit n". Most of the time, the least significant bit will be called bit 0 and the most significant will be bit 15. It is possible you could find reference to bit 1 through bit 16, in which case just subtract one from the number to reference the table below. For each of the 16 bit positions, the mask (or fill) will be as follows. These mask values are shown in hexadecimal, as is expected by the BB3-3101/3060 configuration tool.

B0/D0/bit 0	mask = 0001
B1/D1/bit 1	mask = 0002
B2/D2/bit 2	mask = 0004
B3/D3/bit 3	mask = 0008
B4/D4/bit 4	mask = 0010
B5/D5/bit 5	mask = 0020
B6/D6/bit 6	mask = 0040
B7/D7/bit 7	mask = 0080
B8/D8/bit 8	mask = 0100
B9/D9/bit 9	mask = 0200
B10/D10/bit 10	mask = 0400
B11/D11/bit 11	mask = 0800
B12/D12/bit 12	mask = 1000
B13/D13/bit 13	mask = 2000

B14/D14/bit 14 mask = 4000

B15/D15/bit 15 mask = 8000

Some Modbus devices also pack two 8-bit values into a single 16-bit register. The two values will typically be documented as "high byte" and "low byte" or simply have "H" and "L" indicated. If you run into this scenario, the masking for bytes is as follows:

High byte mask = FF00

Low byte mask = 00FF

There have been a few instances of documenting packed bits in a 32-bit register. Although Modbus protocol is strictly 16-bit registers, some implementations force you to read pairs of registers. If your device documents 32 packed bits, then you would insert 0000 in front of each mask above, and the remainder of the list would be as follows:

B16/D16/bit 16 mask = 00010000

B17/D17/bit 17 mask = 00020000

B18/D18/bit 18 mask = 00040000

B19/D19/bit 19 mask = 00080000

B20/D20/bit 20 mask = 00100000

B21/D21/bit 21 mask = 00200000

B22/D22/bit 22 mask = 00400000

B23/D23/bit 23 mask = 00800000

B24/D24/bit 24 mask = 01000000

B25/D25/bit 25 mask = 02000000

B26/D26/bit 26 mask = 04000000

B27/D27/bit 27 mask = 08000000

B28/D28/bit 28 mask = 10000000

B29/D29/bit 29 mask = 20000000

B30/D30/bit 30 mask = 40000000

B31/D31/bit 31 mask = 80000000

The following example shows picking bit 7 from 16-bit holding register 55, and placing its state in Binary Input 1.

Babel Buster BB3-3101 Gateway Configuration Tool v3.01 [Test115kAll.xml]

Connected: ☒ Target: ☒ 29999: Babel Buster BB3-3101 In Sync: ☒

Local Port | Who-Is | Read/Write | Device | Object Map | Object Text | Object List | Modbus | Data List | Firmware

Object Type/Instance: Binary Input 1 Units: no_units

Object Name: Modbus Status Bit 7

Description: Example of splitting packed register

☒ Read Periodic ☐ Write Periodic ☐ Set Default on Power-Up ☐ Enable Max Quite Time ☒

☐ Write on Delta ☐ Set Default on Comm Fail

Poll Rate (Sec): 2 Slope/Scale Factor: 0 Timeout (Sec): 0

Initial COV Increment: Intercept/Offset: 0 Max. Quiet Time (Sec): 0

Initial COV Period: 0 Delta for Send: Default Value: 0

Initial Relinquish Default: Read Fails before Fault: 0

☒ Map Modbus Object ☐ Map BACnet Object

Register Number 1..N: 55 Unit/Slave Addr: 2 ☒ Member of Packed Register

Register Type: Holding Register (fc 16) Mask (Hex): 0080 ☐ Pack Mixed Object Types

Register Format: Unsigned 16-bit Fill (Hex): 0000

Insert Add Delete Read Device Write Device

If the Modbus register is a 32-bit register, then the mask and fill values expand to 32 bits each. The following example will pick out a 4-bit value from bits 12-15 of the 32-bit register at Modbus address 1 and place the value (which will be in the range of 0..15) in Analog Input 1.

Babel Buster BB3-3101 Gateway Configuration Tool v3.01 [MaskFill32Bit.xml]

Connected: ☒ Target: ☒ 29999: Babel Buster BB3-3101 In Sync: ☒

Local Port | Who-Is | Read/Write | Device | Object Map | Object Text | Object List | Modbus | Data List | Firmware

Object Type/Instance: Analog Input 1 Units: no_units

Object Name: Analog Input 1 Description:

☒ Read Periodic ☐ Write Periodic ☐ Set Default on Power-Up ☐ Enable Max Quite Time ☒
☐ Write on Delta ☐ Set Default on Comm Fail

Poll Rate (Sec): 2 Slope/Scale Factor: 0 Timeout (Sec): 0
Initial COV Increment: 0 Intercept/Offset: 0 Max. Quiet Time (Sec): 0
Initial COV Period: 0 Delta for Send: 0 Default Value: 0
Initial Relinquish Default: Read Fails before Fault: 0

☒ Map Modbus Object ☐ Map BACnet Object

Register Number 1..N: 1 Unit/Slave Addr: 1 ☒ Member of Packed Register
Register Type: Holding Register (fc 16) Mask (Hex): 0000F000 ☐ Pack Mixed Object Types
Register Format: Unsigned 32-bit (high reg first) Fill (Hex): 00000000

Insert Add Delete Read Device Write Device File saved.



4 Tool 'Local Port' Page

4.1 Connecting Configuration Tool to Gateway Device

All configuration of the BB3-3101 is done via the MS/TP network, and all configuration is accessible via BACnet object properties. The gateway may be connected to the configuration tool software through either a BACnet IP to MS/TP router, or via the Control Solutions MTX002 USB to MS/TP adapter. Use of the configuration tool software provides a convenient user interface for setting up the gateway, but its use is not required. Any BACnet client can be used to access the configuration properties, but it will be far more convenient to use the configuration tool software provided by Control Solutions at no cost.



If connecting to MS/TP without an IP router, you must use a Control Solutions MTX002 USB to MS/TP adapter with the configuration tool. Please note that this is not a generic RS485 adapter. The MTX002 is itself an MS/TP device with a USB connection. Windows does not always do an accurate and consistent job of MS/TP token passing, especially when the PC is busy with multiple tasks. The solution was to move the MS/TP token passing out of Windows, and this is what the MTX002 does. In addition, standard PC hardware is incapable of 76800 baud, but the MTX002 does support that baud rate.

The first page displayed when you open the configuration tool is the Local Port page. If you do not have a device connected and simply want to work offline to build a configuration file, you do not need to do anything here. However, if you will be communicating with a gateway device to configure it, then you need to tell the configuration tool to connect to the network.

Babel Buster BB3-3101 Gateway Configuration Tool v3.06

Connected: Target: Waiting for Target In Sync:

Local Port | Who-Is | Read/Write | Device | Object Map | Object Text | Object List | Modbus | Data List | Firmware

☐ Enable MS/TP via USB ☐ Enable IP

USB serial port: COM1 Local IP: 192.168.1.112

MS/TP baud: 38400 IP Port (hex): BAC0

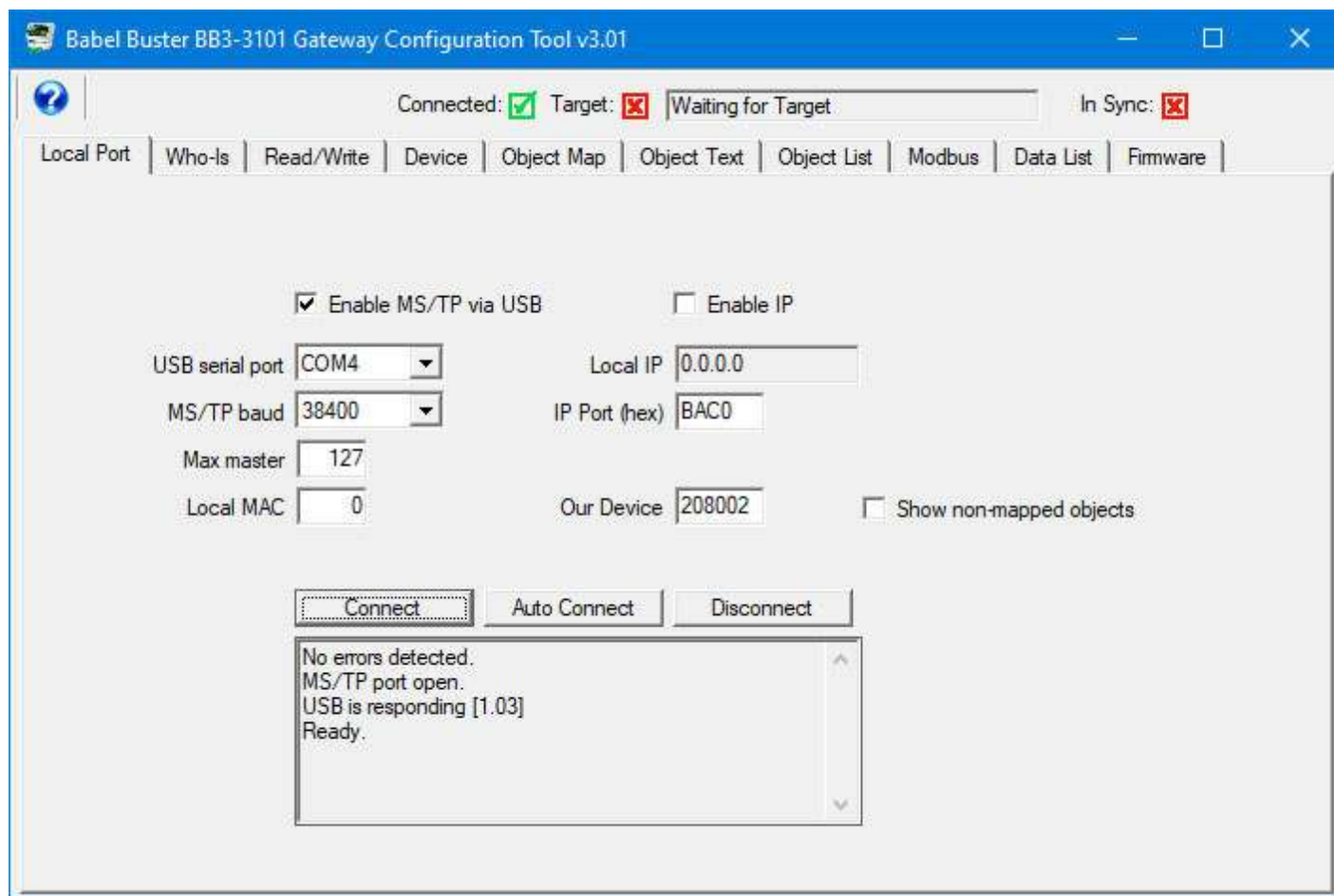
Max master: 127

Local MAC: 0 Our Device: 208002 ☐ Show non-mapped objects

Connect Auto Connect Disconnect

If you will be connecting to MS/TP via MTX002, you will need to have installed the USB driver first. Refer to Appendix H if you have not already done this. After installing the driver and connecting the MTX002, check your PC's device manager to see which COM port the MTX002 got assigned to. Then select that COM port from the list on the Local Port page of the tool.

Select the COM port that the MTX002 is on. Select the baud rate that your MS/TP device (or network) is running at. Leave the "Max master" set to 127 unless you know exactly why you are changing it. The Local MAC will default to zero - this is the MAC address that the MTX002 will use as its own address on the network. If you are aware of another device on the MS/TP network that already uses this MAC address, change the Local MAC to some address between 0 and 127 that is not already used. After making all of these selections, click the "Enable MS/TP via USB" box, and then click the Connect button.



Upon clicking the Connect button, you should see something like the above response displayed in the status window. Your PC very likely has more than one COM port. If you open a valid COM port, but one which does not have an MTX002 present, you will still get "No errors detected" and "MS/TP port open". This is because it did not detect errors, and it did in fact find a valid port to open. **But it is not talking to the MTX002 unless you see the two additional lines stating "USB is responding" and "Ready".** You will not be able to communicate with your MS/TP device if you do not see the last two lines.

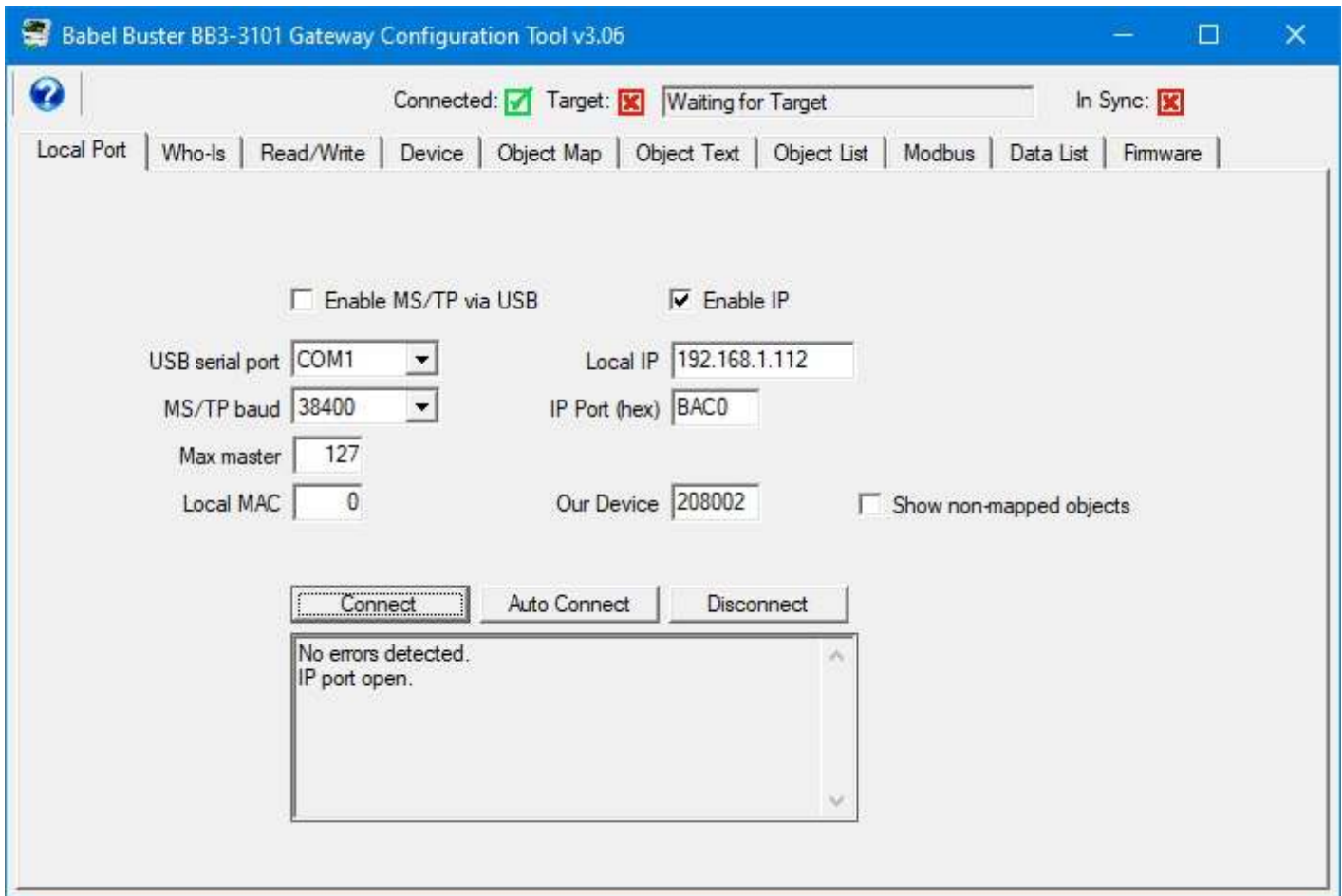
You may also connect the configuration tool to your BACnet IP network. This option is available if you have a BACnet IP to MS/TP router on your network. The configuration tool software will configure a gateway on the other side of a BACnet router the same as if connected directly to MS/TP.

To connect BACnet IP, enter the IP port number if it is not the standard BACnet number (which the tool will default to). Enter a device instance in the unlikely event there is already another device on the network with the number shown in the "Our Device" window. The IP address will become populated with your PC's IP address as soon as you open the program. Click the "Enable IP" box, and then click the Connect button.

The Local Port tab in the BB3-3101 configuration tool has been modified (as of v3.06) to allow selecting one of multiple IP interfaces on a PC. Upon opening the program, the Local IP window will display the IP address that Windows has chosen as its default.

However, if this is not the desired interface, enter the IP address of the interface you would rather use, then check Enable IP, and click Connect.

To change interfaces, should you discover the incorrect one was initially connected, start by clicking Disconnect. Then enter the correct IP address, and click Connect again.



Upon connecting to BACnet IP, you should see "No errors detected" and "IP port open" as illustrated above.

Do not be surprised if your firewall complains about the configuration tool trying to access the network the first time you connect via BACnet IP. Simply click "Allow access" or whatever is applicable in your current version of Windows.



4.2 Special Setting for Server-Only Applications

The BB3-3101 most commonly used as either a Modbus master while being a BACnet server, or BACnet client while being a Modbus slave. Sometimes the gateway is both

Modbus master and BACnet client. However, when you want the gateway to be only a Modbus slave and BACnet server (no active polling of anything on the part of the gateway), the configuration tool tends to think your objects are going unused, and by default will not display them or try to configure them.

There is an additional check box appears on the Local Port page. Check "Show non-mapped objects" to cause the tool to display all objects regardless of whether mapped to Modbus or BACnet as shown on the Object Map page. This selection will also allow the tool to write object name, description, and units properties to objects that are otherwise "unused".

The screenshot shows the 'Local Port' configuration page of the Babel Buster BB3-3101 Gateway Configuration Tool v3.01. The interface includes a status bar at the top with 'Connected: [checked]', 'Target: [X] Waiting for Target', and 'In Sync: [X]'. Below the status bar is a tabbed menu with 'Local Port' selected. The main configuration area contains several settings: 'Enable MS/TP via USB' is checked, while 'Enable IP' is unchecked. The 'USB serial port' is set to 'COM4', 'MS/TP baud' is '38400', 'Max master' is '127', and 'Local MAC' is '0'. On the right, 'Local IP' is '0.0.0.0', 'IP Port (hex)' is 'BAC0', and 'Our Device' is '208002'. A checkbox labeled 'Show non-mapped objects' is checked. At the bottom, there are three buttons: 'Connect', 'Auto Connect', and 'Disconnect'. Below these buttons is a text area displaying the following status messages: 'No errors detected.', 'MS/TP port open.', 'USB is responding [1.03]', and 'Ready.'.

Connected:	Target:	In Sync:
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/> Waiting for Target	<input checked="" type="checkbox"/>

Local Port | Who-Is | Read/Write | Device | Object Map | Object Text | Object List | Modbus | Data List | Firmware

☒ Enable MS/TP via USB ☐ Enable IP

USB serial port: COM4 Local IP: 0.0.0.0

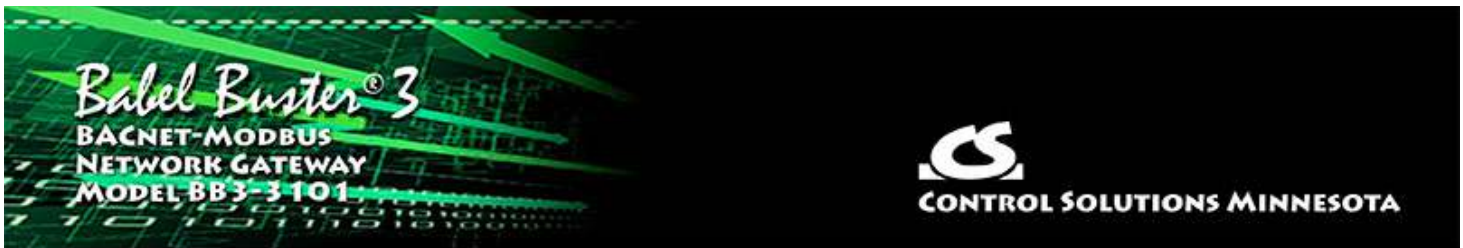
MS/TP baud: 38400 IP Port (hex): BAC0

Max master: 127 Our Device: 208002

Local MAC: 0 ☒ Show non-mapped objects

Connect Auto Connect Disconnect

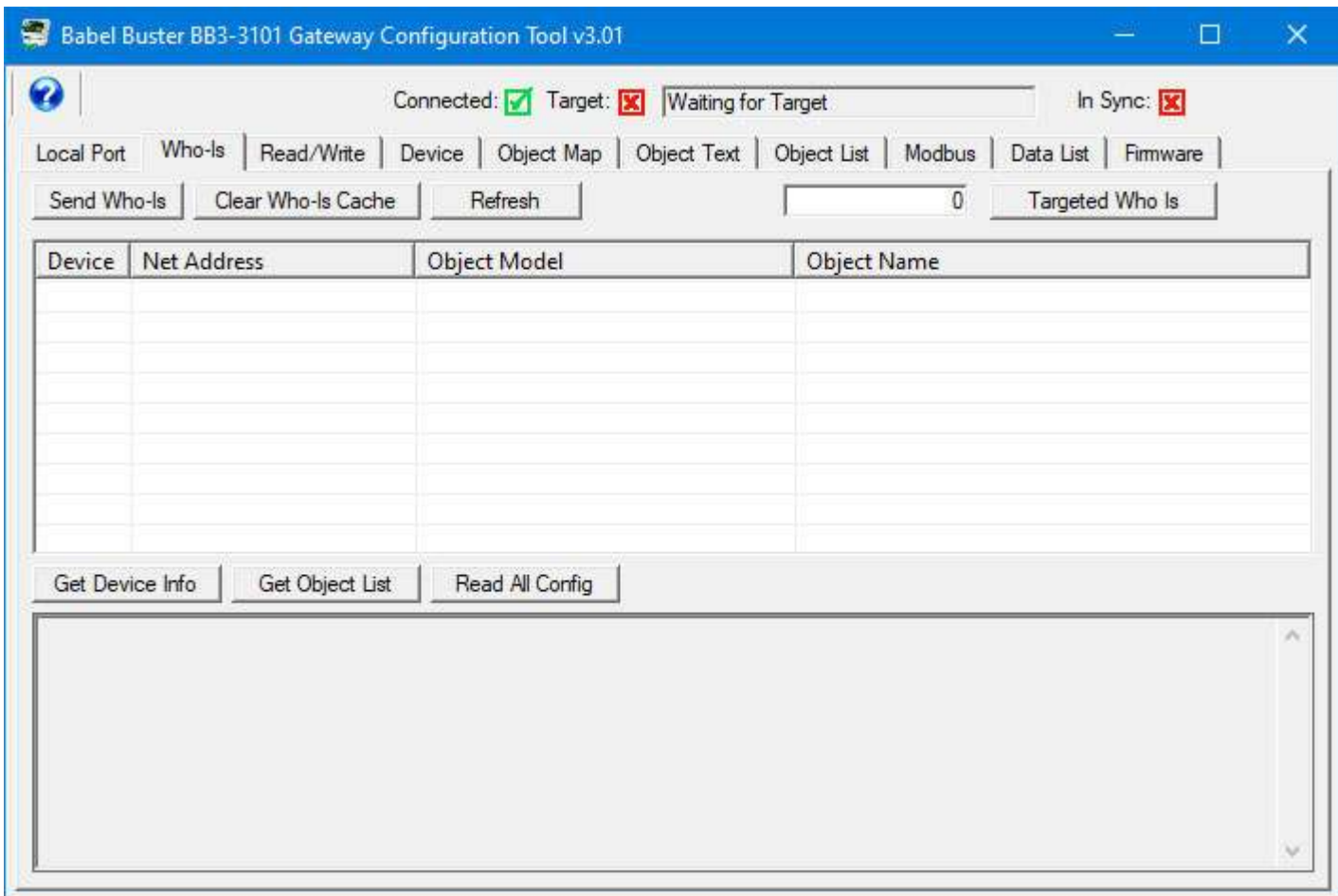
No errors detected.
MS/TP port open.
USB is responding [1.03]
Ready.



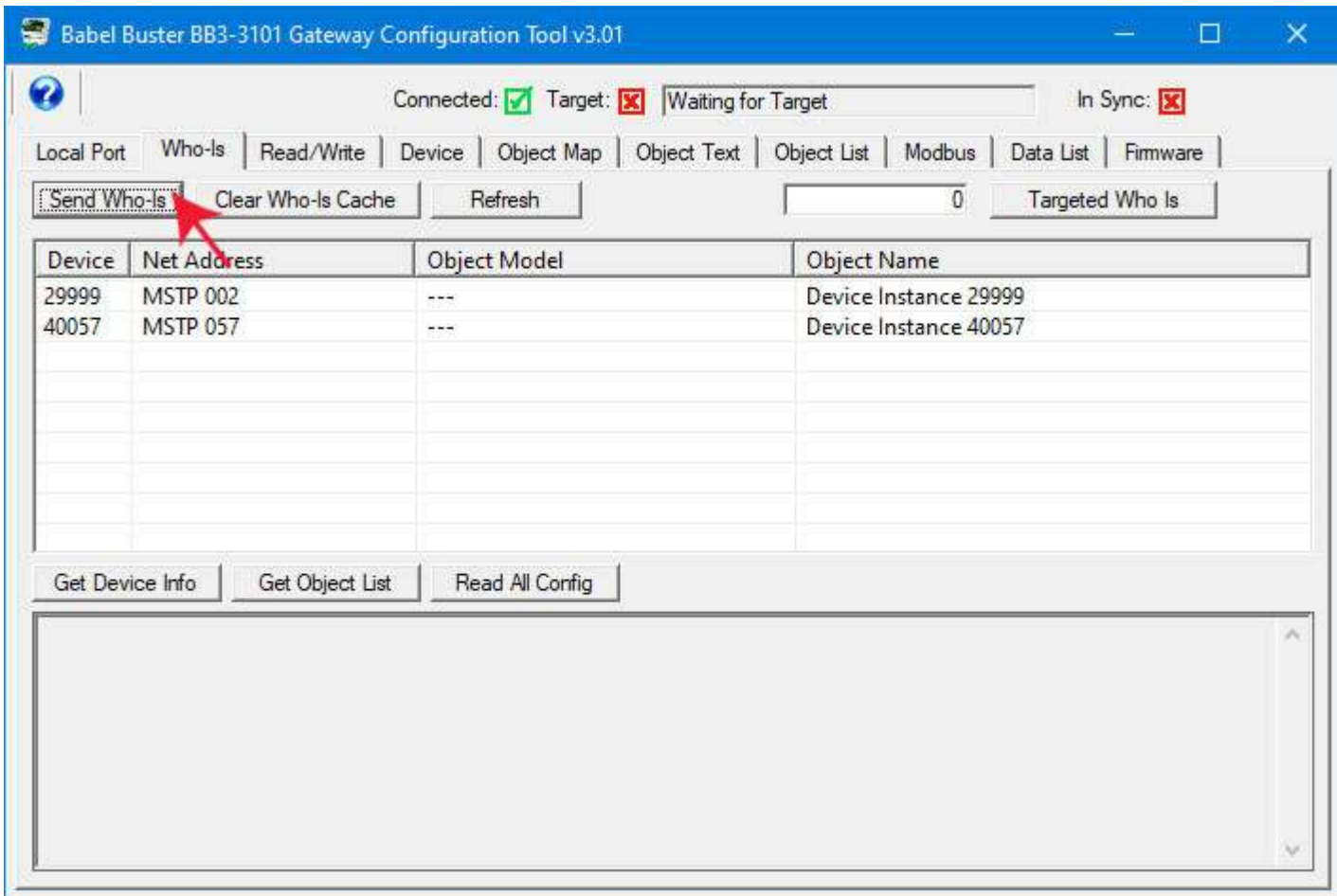
5 Tool 'Who-Is' Page

5.1 Finding Devices

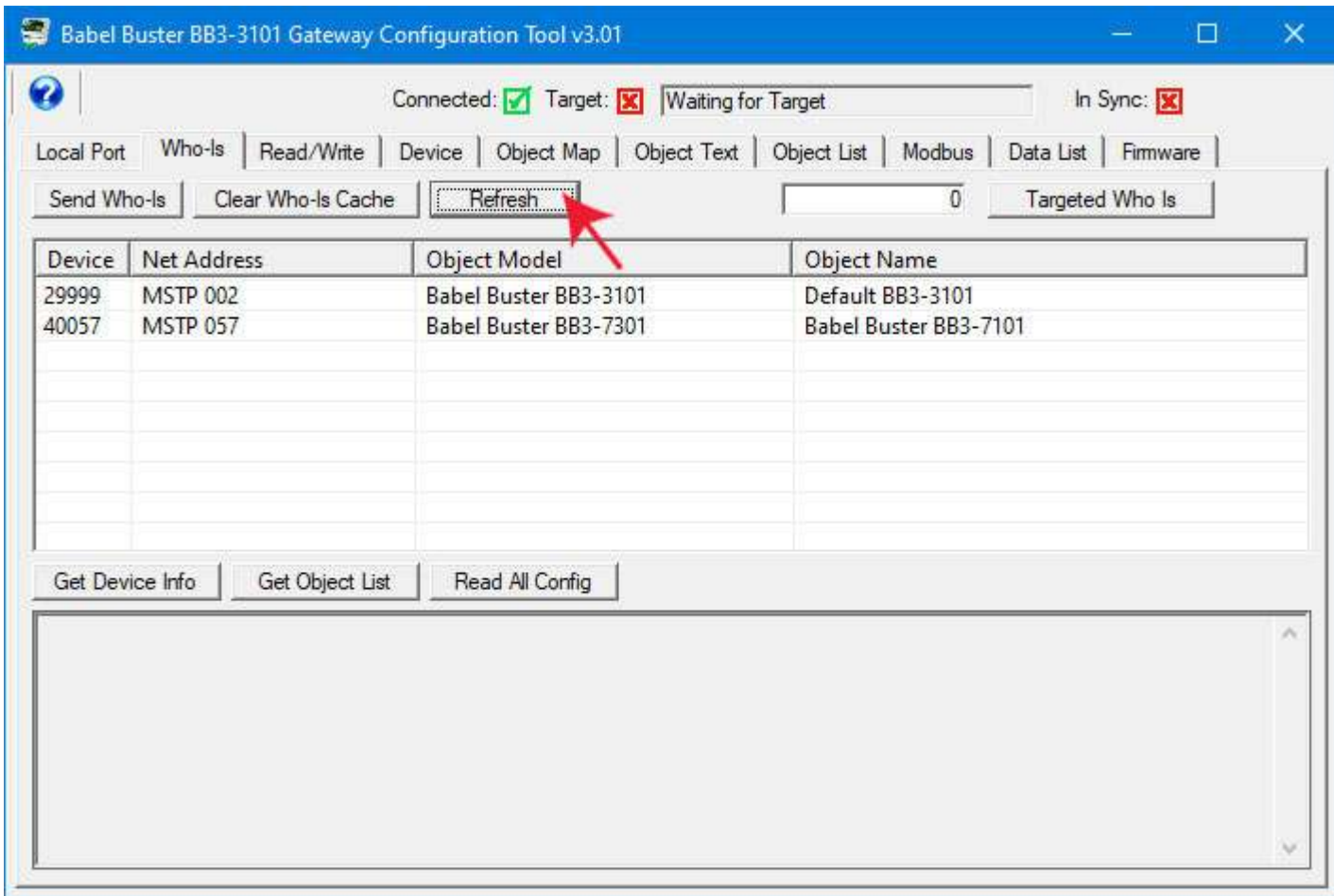
Start by connecting the configuration tool to the network as outlined in section 4 of this user guide. Then come here to the Who-Is page. At first, there will be no devices listed.



Click the "Send Who-Is" button to discover devices on the MS/TP network.

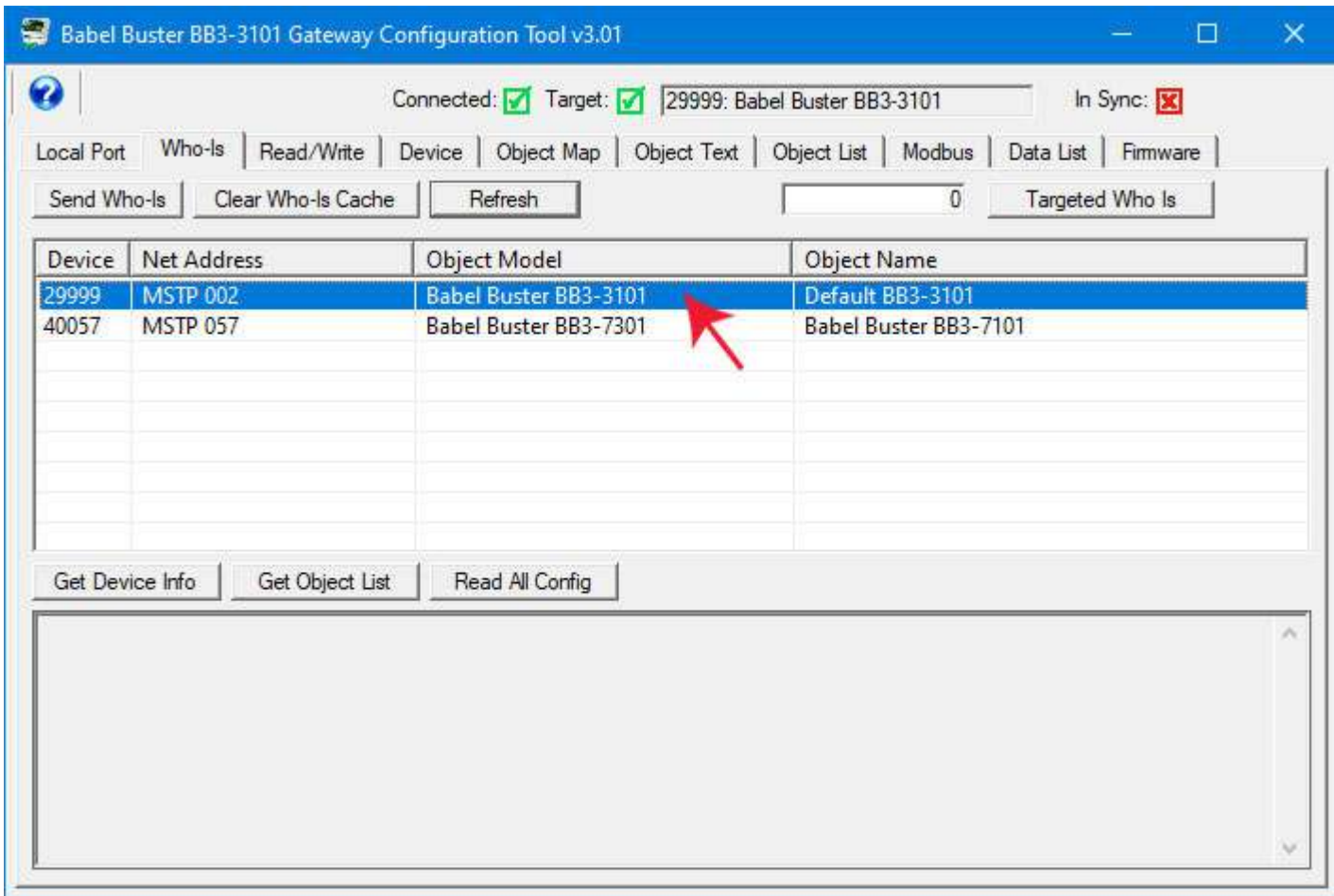


Click the Refresh button to cause the tool to request additional information from each device on the list, namely object model and object name as found in each device's BACnet Device object.



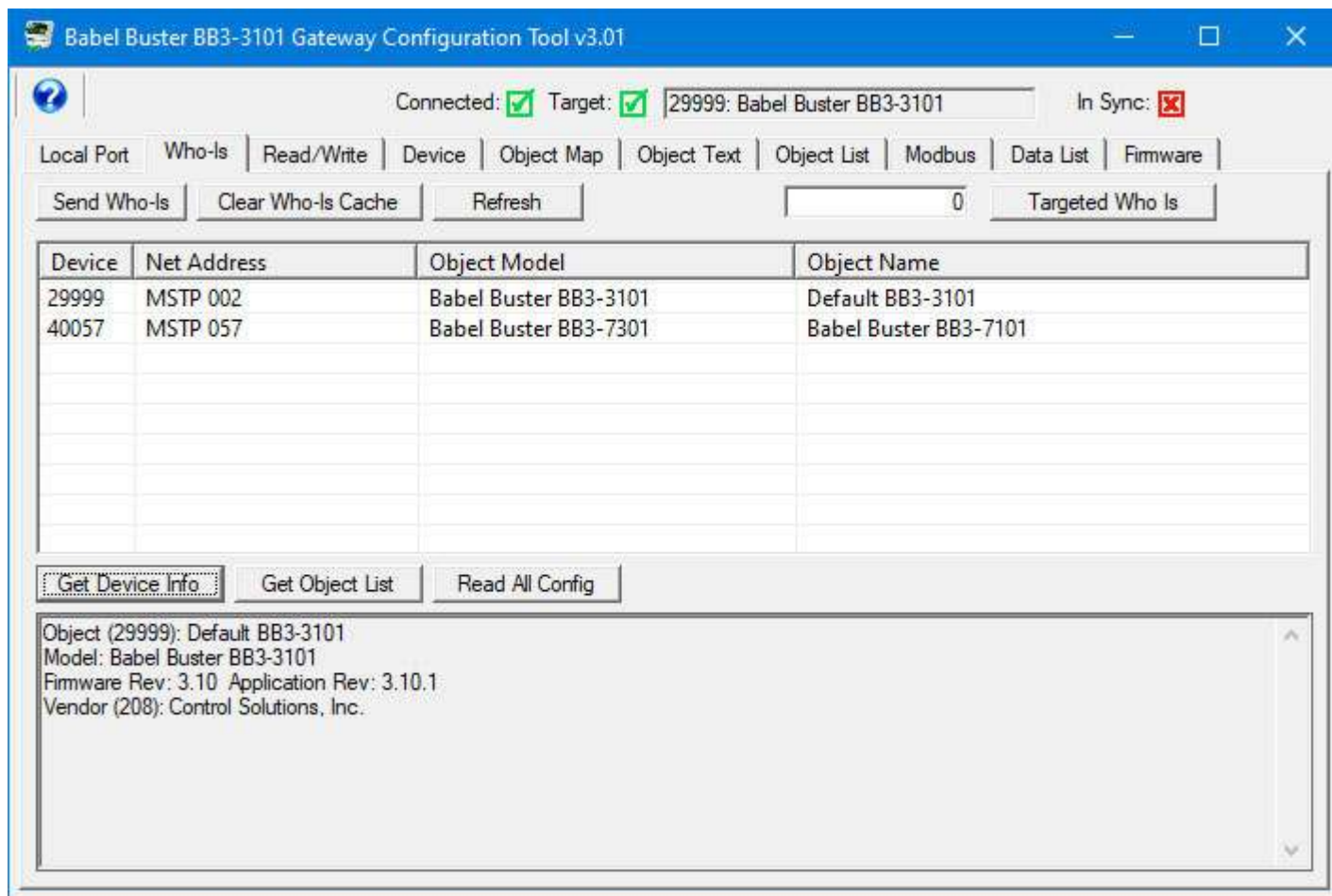
5.2 Select Target

In order to start interacting with a specific device, you need to double click that device on the list. Once selected, the icon to the right of "Target" will turn green, and the device instance and object model will appear in the target window at the top of the screen.



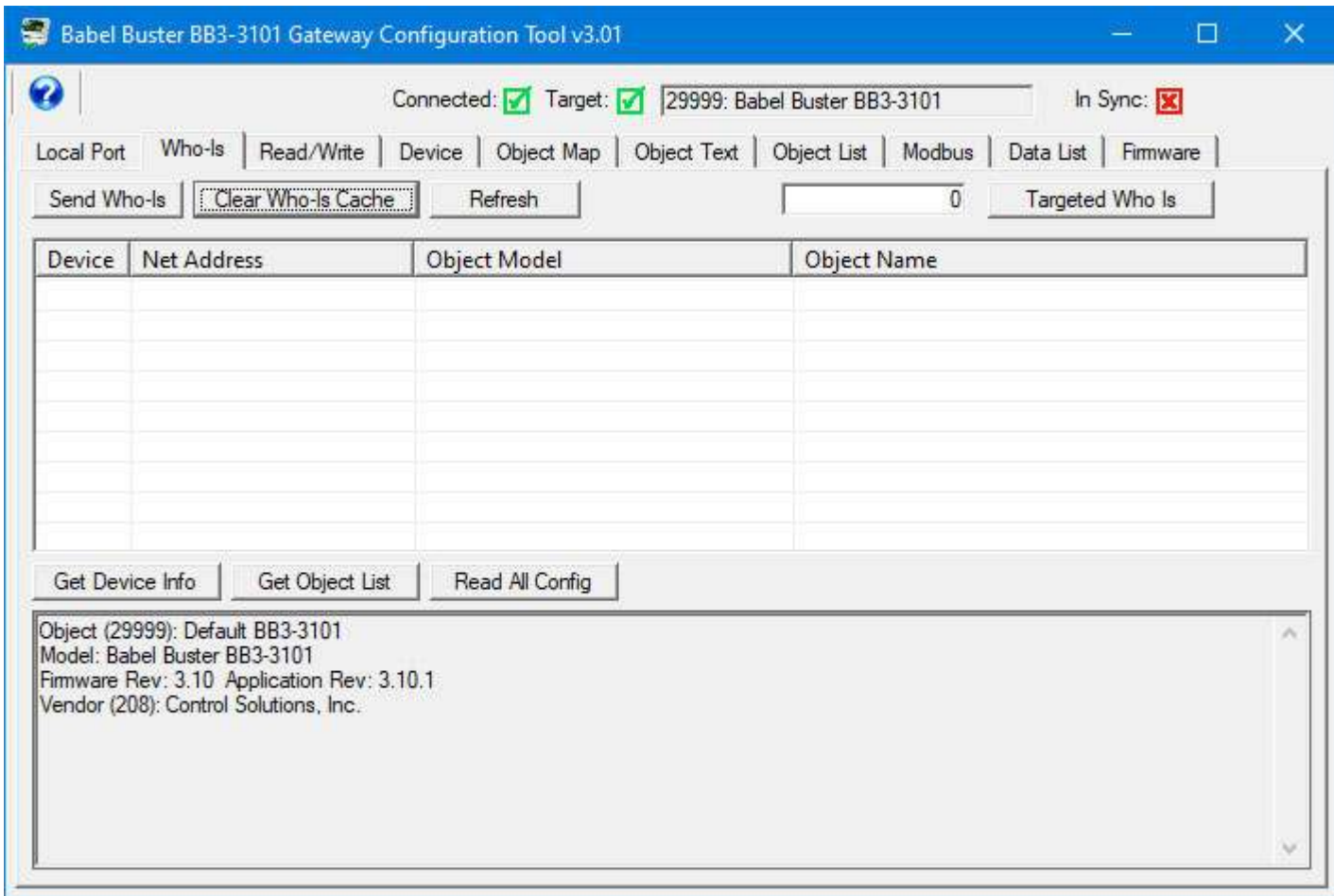
5.3 Getting Device Information

Click the Get Device Info button to get additional information about the selected device, including things like firmware revision. This step is optional and has no bearing on configuring the device. You can also retrieve a listing of all objects found in the device by clicking the Get Object List button. Both of these buttons will work on any BACnet device regardless of whether it is a Babel Buster gateway.



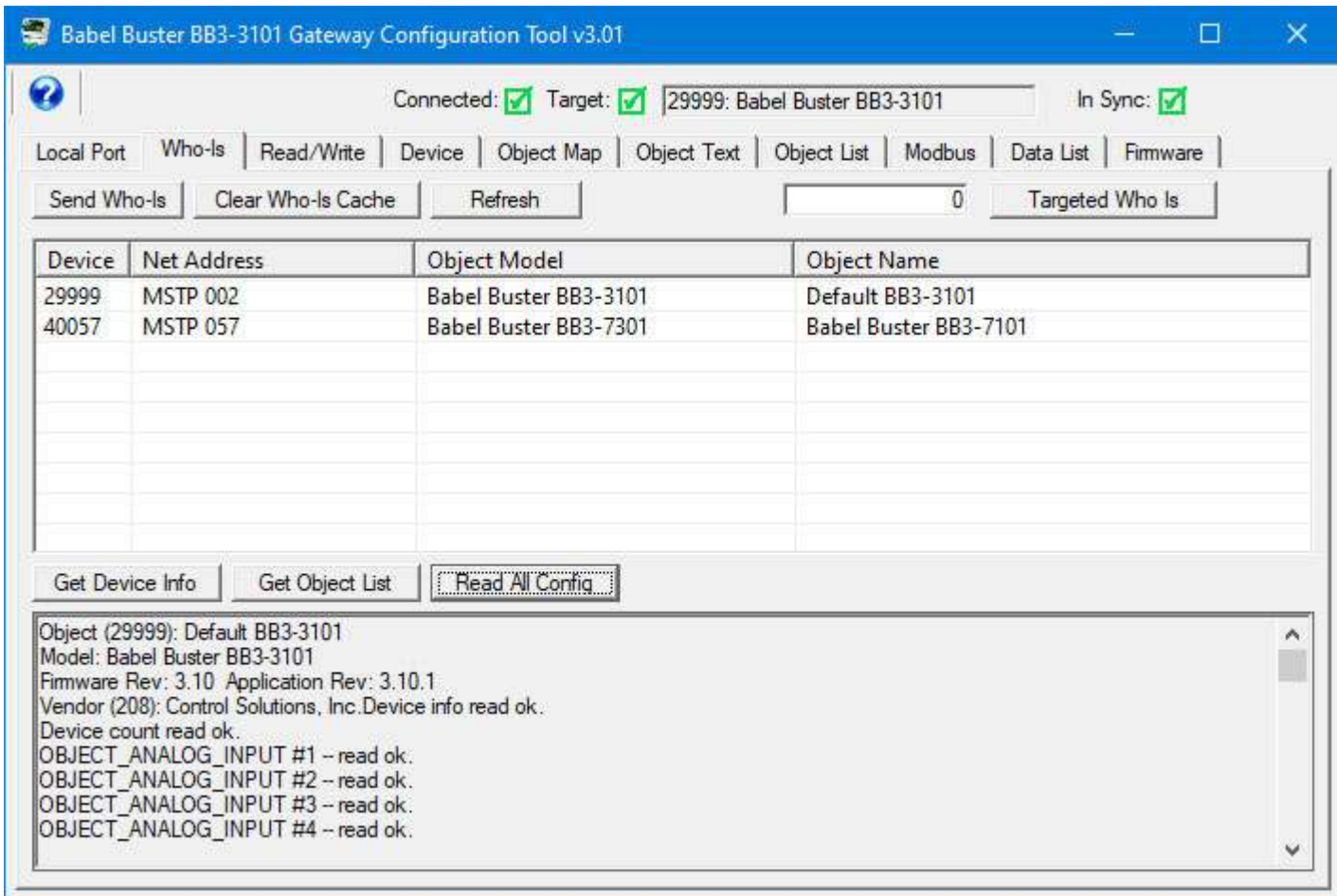
5.4 Clear Who-Is Cache

If you want to start over on the device discovery process, click the Clear Who-Is Cache button, followed by Send Who-Is, and then Refresh after devices appear. **The only time this step becomes a requirement is if you change the device instance of the gateway you are in the process of configuring. If the device instance is changed, you must rediscover it under its new identity.**



5.5 Read All Configuration

Click the Read All Config button to retrieve all configuration information about the connected gateway. This button only applies to Babel Buster BB3-3101. Using this button on any other type of device will fail because it will not contain gateway configuration properties in its objects. While configuration properties are being retrieved from the gateway, progress will be indicated in the status window at the bottom.

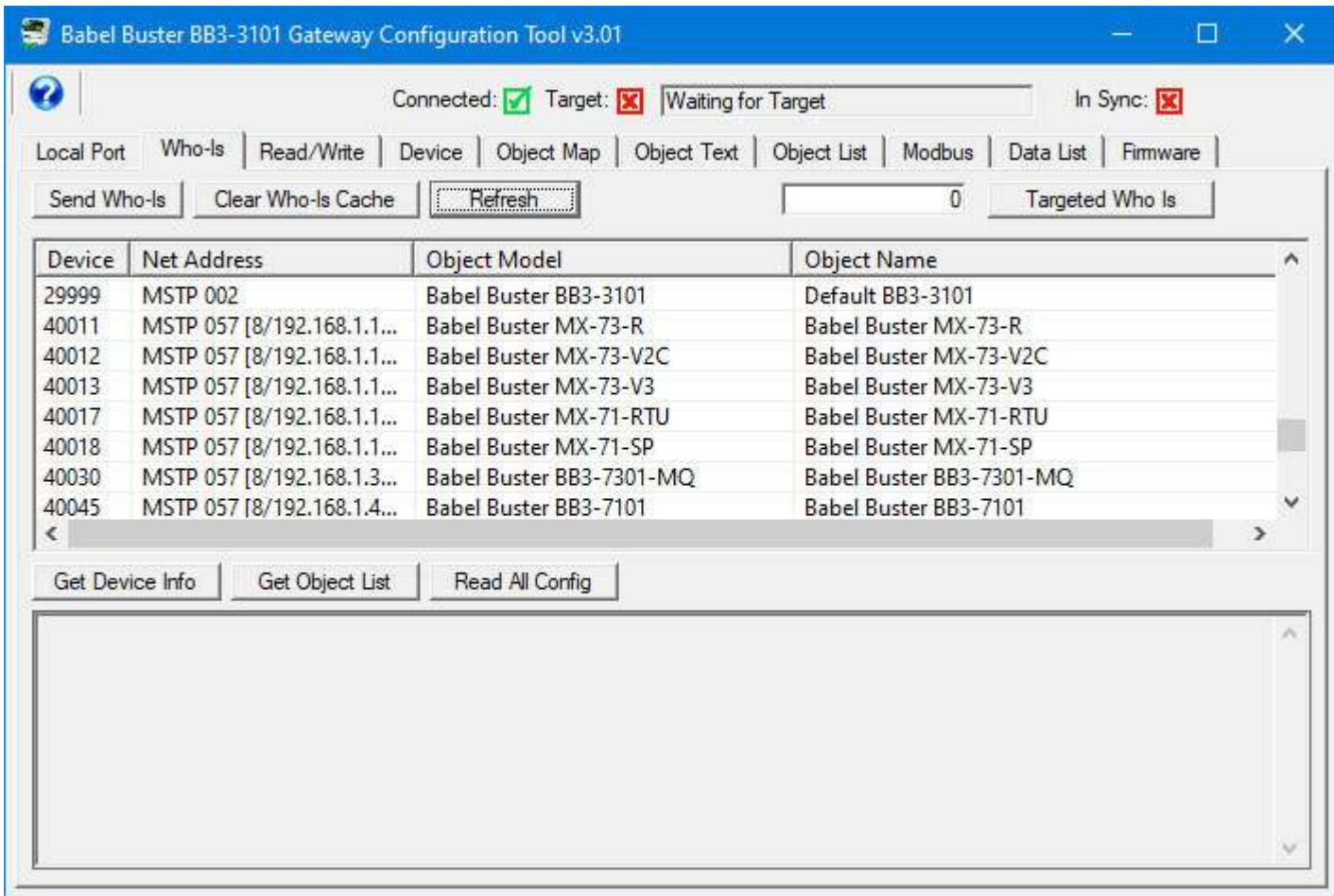


Once the "read all config" process is complete, the In Sync icon in the upper right corner will turn green. The "In Sync" icon that appears in the upper right corner of every page is the global "everything is in sync". If red, it means there is at least one red icon on some page somewhere in the configuration tool. A red icon (red X or red dot) means the configuration displayed on the screen in the configuration tool is not necessarily what is actually in the device. If the icon is green (green check mark or green dot), then the device is in sync with what is displayed.

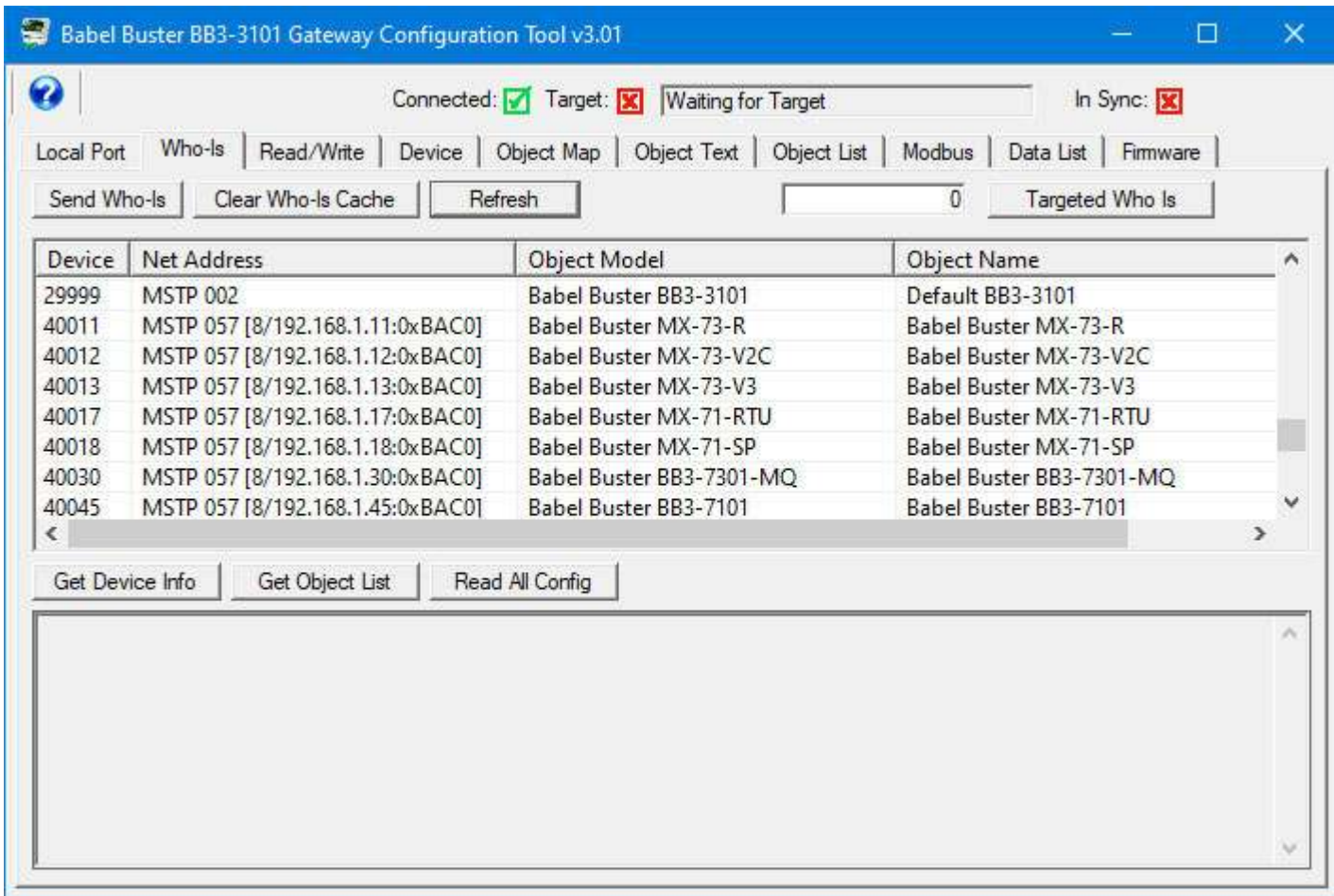
5.6 Targeted Who-Is

It is possible on a large network, especially when connected via IP, that there will be so many I-Am replies coming back in response to the Who-Is request that they do not all get through. It has been verified using WireShark that it is not a matter of the tool missing them, it is a matter of the I-Am simply not arriving at all because the network is flooded. The test scenario proving this involved multiple routers, multiple networks, and many devices. You can re-click the Send Who-Is button to try to capture the missing devices, but the same flood of I-Am replies is likely to happen.

In the example illustrated below, our BB3-3101 is at MS/TP MAC address 2. The remaining devices are showing up via an IP to MS/TP router. The BACnet router is on the MS/TP link at MAC address 57.



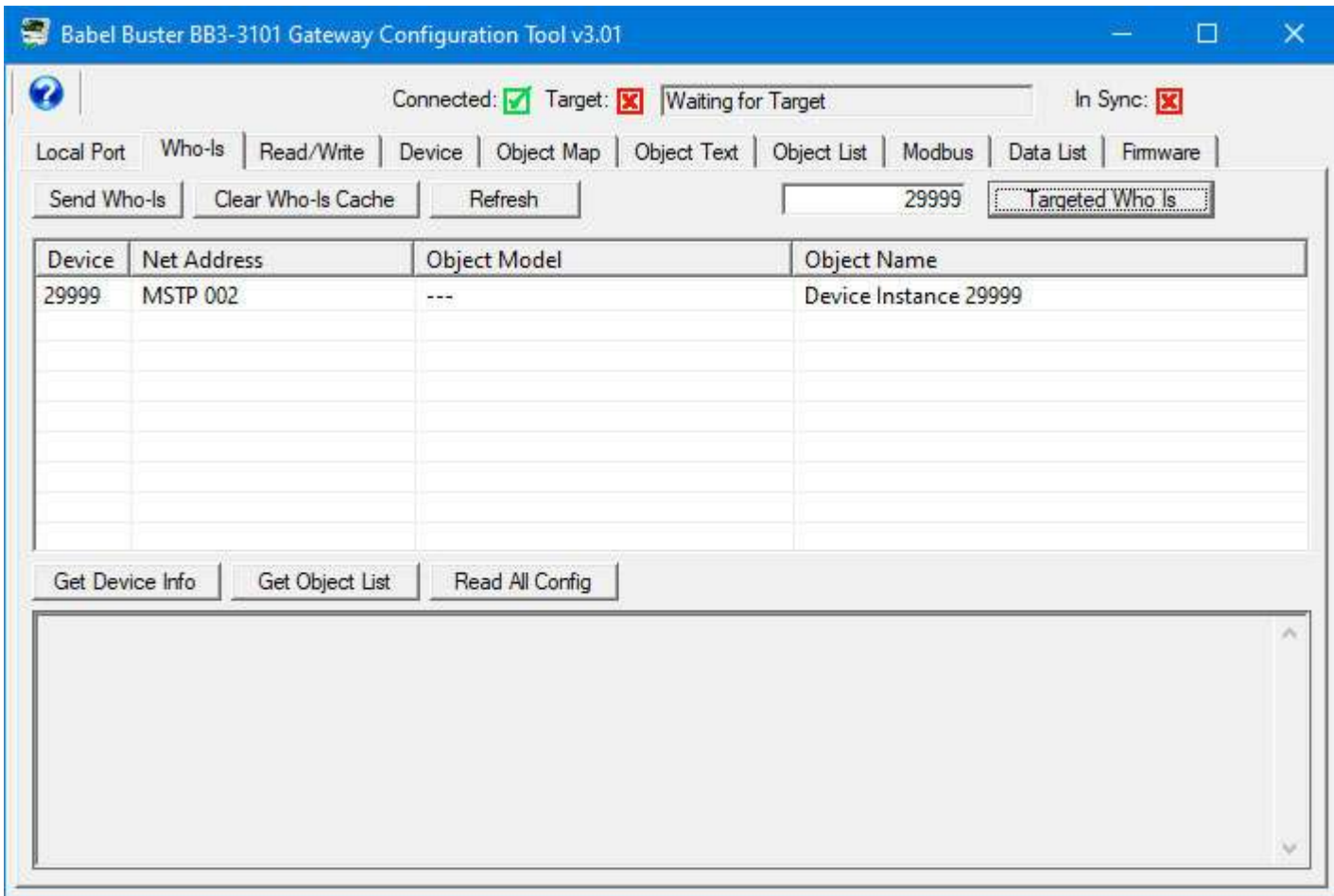
To see the full network address in cases like this, you need to expand the address column. Here we can see several BACnet IP devices accessible via the BACnet router.



The above example has nothing to do with configuring your BB3-3101. It is simply illustrating what you could see on a busy network.

You can avoid the flood of responses from devices if you know the device instance of the one device you are looking for. In the example below, the BB3-3101 is device instance 29999.

Enter the desired device instance in the window next to the Targeted Who Is button, and click that button. Only that device will now respond with its I-Am reply.



If the Refresh button is clicked at this point, only that one device's object information will be queried. You may now double click this one and only device to select it for configuring.

Babel Buster BB3-3101 Gateway Configuration Tool v3.01

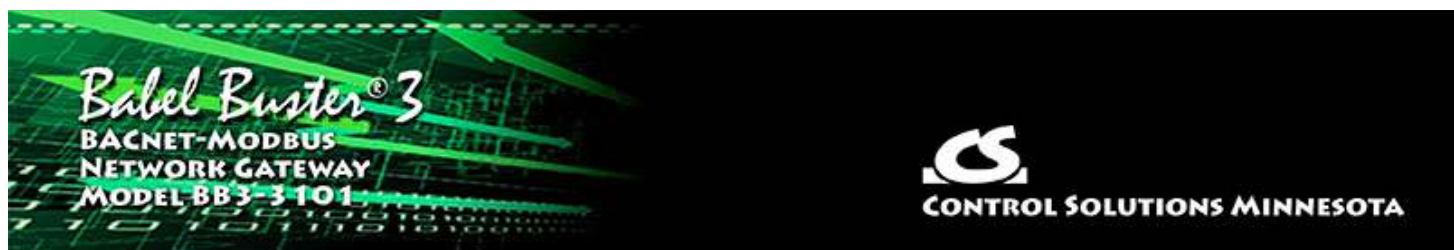
Connected: ☒ Target: ☒ Waiting for Target In Sync: ☒

Local Port Who-Is Read/Write Device Object Map Object Text Object List Modbus Data List Firmware

Send Who-Is Clear Who-Is Cache Refresh 29999 Targeted Who Is

Device	Net Address	Object Model	Object Name
29999	MSTP 002	Babel Buster BB3-3101	Default BB3-3101

Get Device Info Get Object List Read All Config



6 Tool 'Read/Write' Page

This page may be used as soon as a device is selected on the Who-Is page. The device does not have to be a BB3-3101 in order to use the generic object property read/write on this page. This can be a useful diagnostic for any BACnet device.

6.1 Read Property

Select object type, instance, property, and array index if any, and then click 'Read Property'. If the request is successful, the data will be displayed in the log window at the bottom.

A collection of most often used property types are included in the drop list. If the desired property is not shown, select 'Other-->' and enter the property type code in the window next to the list. The property type codes are those defined by the BACnet standard. For example, Present Value can also be obtained using 'Other --> 85'.

The screenshot shows the 'Babel Buster BB3-3101 Gateway Configuration Tool v3.01' window. The 'Read/Write' tab is selected. At the top, status indicators show 'Connected: [checked]', 'Target: [checked]', and 'In Sync: [X]'. The target address is '29999: Babel Buster BB3-3101'. Below the status bar is a tabbed menu with 'Local Port', 'Who-Is', 'Read/Write' (selected), 'Device', 'Object Map', 'Object Text', 'Object List', 'Modbus', 'Data List', and 'Firmware'. The main area contains fields for 'Object Type' (set to 'Analog Input'), 'Object Instance' (set to '1'), 'Property' (set to 'Present Value'), and 'Array Index' (set to '0'). There are 'Read Property' and 'Write Property' buttons. Below these are fields for 'Data Type' (set to 'Null'), 'Priority' (set to 'None'), and a 'Relinquish' checkbox. A 'Write Data' text box is also present. At the bottom, a log window displays the value '15.000000'.

6.2 Write Property

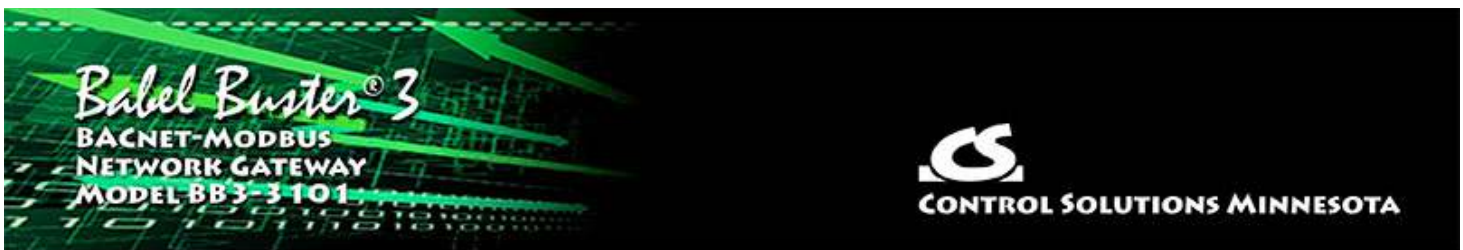
Select all of the same settings as you would for Read Property, and in addition, specify data type, priority (if commandable, use 'none' if not commandable), and data to write. Then click 'Write Property'.

Format for most data types is simply a numeric string. The data type 'character string' will consume everything found in the data window as ASCII text copied verbatim. Octet strings should consist of 8-bit values in hexadecimal notation (1 to 3 hex digits), with each octet separated by a comma. Bit strings should consist of a series of T and F characters, optionally separated by a comma. The first T or F is bit zero.

The screenshot shows the 'Babel Buster BB3-3101 Gateway Configuration Tool v3.01' window. The 'Read/Write' tab is selected. The 'Connected' status is checked, and the 'Target' is '29999: Babel Buster BB3-3101'. The 'In Sync' status is not checked. The 'Object Type' is 'Analog Output', 'Object Instance' is '1', and 'Property' is 'Present Value'. The 'Array Index' is blank, with a note '(leave blank for no index)'. The 'Data Type' is 'Real', 'Priority' is '3', and the 'Relinquish' checkbox is unchecked. The 'Write Data' field contains '19.5'. The 'Read Property' button is highlighted, and the 'Write Property' button is also visible. A data window at the bottom shows the value '19.500000'.

Local Port	Who-Is	Read/Write	Device	Object Map	Object Text	Object List	Modbus	Data List	Firmware

Object Type: Analog Output
Object Instance: 1
Property: Present Value
Array Index: (leave blank for no index)
Data Type: Real
Priority: 3
Relinquish: ☐
Write Data: 19.5
Read Property
Write Property
19.500000



7 Tool 'Device' Page

The "Device" page is where you create the gateway's identity as a BACnet device.

7.1 Retrieve Device Object Information from Device

Clicking the 'Get Info' button will read all of the information shown on the left half on this page, except that no password is ever read out (not permitted by the device).

Babel Buster BB3-3101 Gateway Configuration Tool v3.01

Connected: ☒ Target: ☒ 29999: Babel Buster BB3-3101 In Sync: ☒

Local Port | Who-Is | Read/Write | **Device** | Object Map | Object Text | Object List | Modbus | Data List | Firmware

Object ID (device instance) 29999 Reassign

Device Name
Default BB3-3101

Device Description
Modbus RTU to BACnet MS/TP Gateway

Device Location
Unassigned

☒ Auto-Reset Errors

Max Master 127 **Get Info**

MS/TP Baud Rate 38400 Change

MS/TP Station/MAC Addr 2

New Password

Old Password

Reinit Device Reconfigure Get Counts Check Fit

Device info read ok.

Object Allocation	Present	Pending
Analog Input Count	0	0
Analog Output Count	0	0
Analog Value Count	0	0
Binary Input Count	0	0
Binary Output Count	0	0
Binary Value Count	0	0
Multistate Input Count	0	0
Multistate Output Count	0	0
Multistate Value Count	0	0
Multistate Max State	0	0

Object ID	Object ID is the device instance of this gateway. It must be unique on the network. When a BACnet client broadcasts the "Who-Is" request, this number will be sent back as this device's identity. If more than one device respond with the same device instance, then the client will be unable to communicate correctly with the devices.
Device Name	This Device Name must be a unique name on your BACnet network. When a BACnet client broadcasts a "Who-Has" request asking for a device object by a given name, you will create problems on the network

	if more than one device responds with the same device name. Therefore, be careful to see that the name is unique.
Device Description	The Device Description property of the Device object is there for your own use in whatever way you see fit, and does not need to be unique.
Device Location	The Device Location property of the Device object is there for your own use in whatever way you see fit, and does not need to be unique.
Auto-Reset Errors	<p>Reliability codes will "latch" by default and require that you read the Reliability property in order to reset it to zero, assuming the problem has gone away. Once the non-zero reliability code has been read (by reading the Reliability property), it will reset to zero the next time the object is updated, provided the problem has been resolved.</p> <p>Since many systems do not automatically read Reliability codes, but do automatically respond to the Fault Status Flag associated with the non-zero reliability code, an auto-reset option is available. When set, reliability codes will return to zero as soon as the problem has been resolved, regardless of whether the non-zero reliability code was ever acknowledged by reading it.</p> <p>Many systems will report an object as "offline" because its fault status bit is set. It is not actually offline, and is in fact communicating just fine trying to tell you that there is a problem. But many front end systems don't recognize this and blindly claim "offline" as a result of the fault bit in the status property. If you are having this issue as the result of communication errors on the Modbus side, try setting the Auto-Reset Errors option here.</p>
Max Master	The Max Master determines the highest device MAC address that the gateway (or any MS/TP device) will try to poll for. It will default to 127, and you should leave this set to 127 unless you know exactly why you are changing it.
MS/TP Baud Rate	MS/TP Baud Rate needs to be set to match whatever speed your MS/TP network is running at. The BB2-3010/3060 will support 9600, 19200, 38400, and 76800.
MS/TP Station or MAC Addr	This is the MAC address of the gateway. The MAC address must be unique on the network. If more than one device is given the same MAC address, the MS/TP network will fail to function. Not only will this device fail to communicate, but the entire network will be corrupted by duplicate MAC addresses.
New Password	Use the New Password field if you want to change the default password to something else. Leave this field blank if you do not want to change the password. Any password in this window will take effect upon clicking the Change button.
Old Password	The Old Password is "buster" by default as shipped from the factory. You only need to use it when reinitializaing the device.

7.2 Change Device Object Configuration

Changes to any of the device information on the left-hand half of this page may be made. Once changes are made, click the 'Change' button to register all changes except device instance (see Reassign).

To cause changes such as baud rate, MAC address, or Max Master setting to take effect, it is necessary to reinitialize the device. The new parameters are saved, but not put into use until you reinitialize the device (enter Od Password and click Reinit Device button) or power cycle the gateway. This reinitialization is required so that you don't immediately lose contact with the device when changing baud rate or MAC address. When you change port parameters like baud rate, you will also need to restart to configuration tool at the different baud rate.

Babel Buster BB3-3101 Gateway Configuration Tool v3.01

Connected: ☒ Target: ☒ 29999: Babel Buster BB3-3101 In Sync: ☒

Local Port | Who-Is | Read/Write | Device | Object Map | Object Text | Object List | Modbus | Data List | Firmware

Object ID (device instance) 29999 Reassign

Device Name
Default BB3-3101

Device Description
Modbus RTU to BACnet MS/TP Gateway

Device Location
Unassigned

☒ ☐ Auto-Reset Errors

Max Master: 127 Get Info

MS/TP Baud Rate: 38400 Change

MS/TP Station/MAC Addr: 2

New Password

Old Password

Reinit Device

Reconfigure

Get Counts

Check Fit

Device info written ok.

Object Allocation	Present	Pending
Analog Input Count	0	0
Analog Output Count	0	0
Analog Value Count	0	0
Binary Input Count	0	0
Binary Output Count	0	0
Binary Value Count	0	0
Multistate Input Count	0	0
Multistate Output Count	0	0
Multistate Value Count	0	0
Multistate Max State	0	0

7.3 Reassign Device Instance

To change the Object ID (device instance) property, enter the new object ID (device instance) and click the 'Reassign' button.

Babel Buster BB3-3101 Gateway Configuration Tool v3.01

Connected: ☒ Target: ☒ 29999: Babel Buster BB3-3101 In Sync: ☒

Local Port | Who-Is | Read/Write | Device | Object Map | Object Text | Object List | Modbus | Data List | Firmware

Object ID (device instance) 29900

Device Name
Default BB3-3101

Device Description
Modbus RTU to BACnet MS/TP Gateway

Device Location
Unassigned

☒ ☐ Auto-Reset Errors

Max Master 127

MS/TP Baud Rate 38400

MS/TP Station/MAC Addr 2

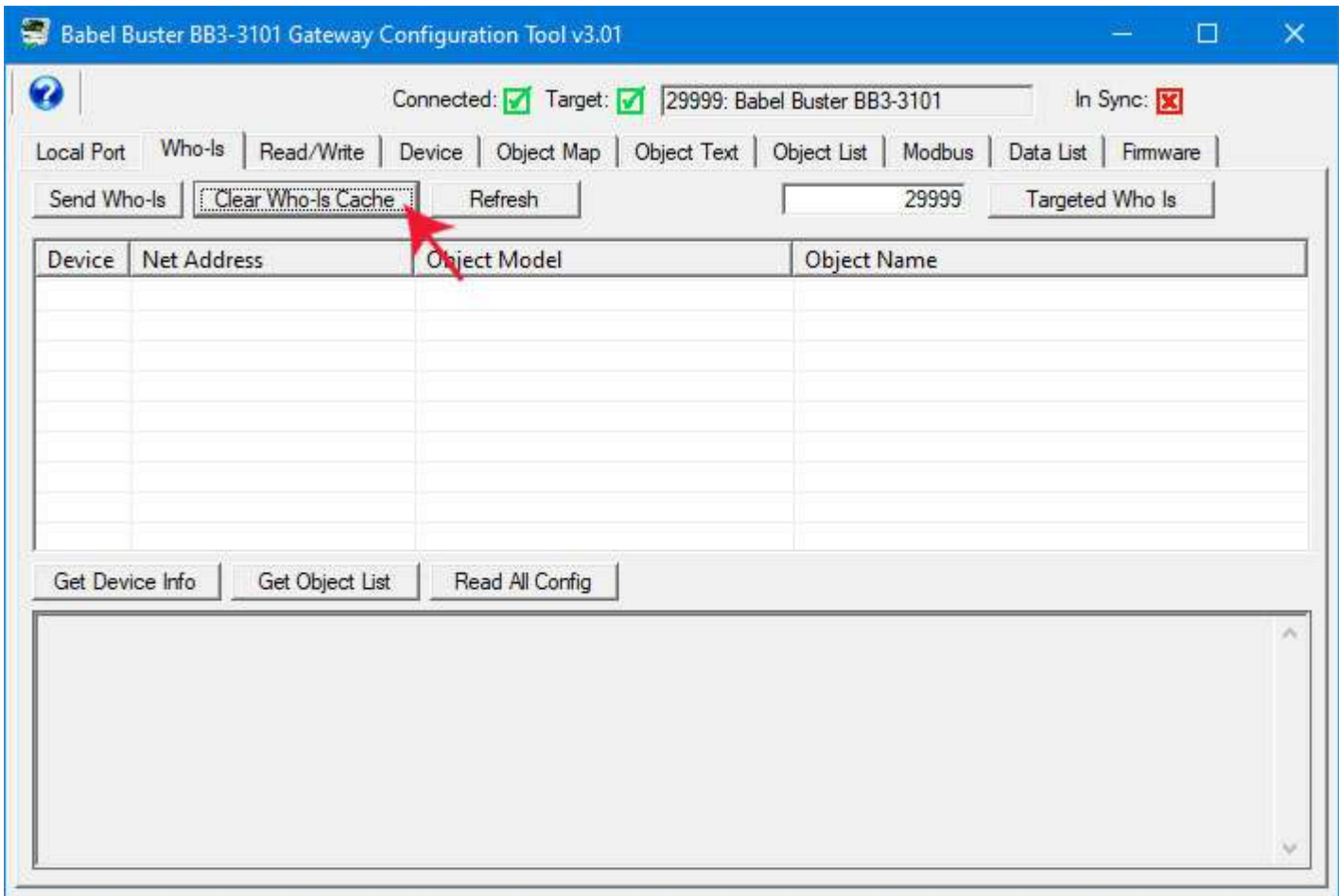
New Password

Old Password

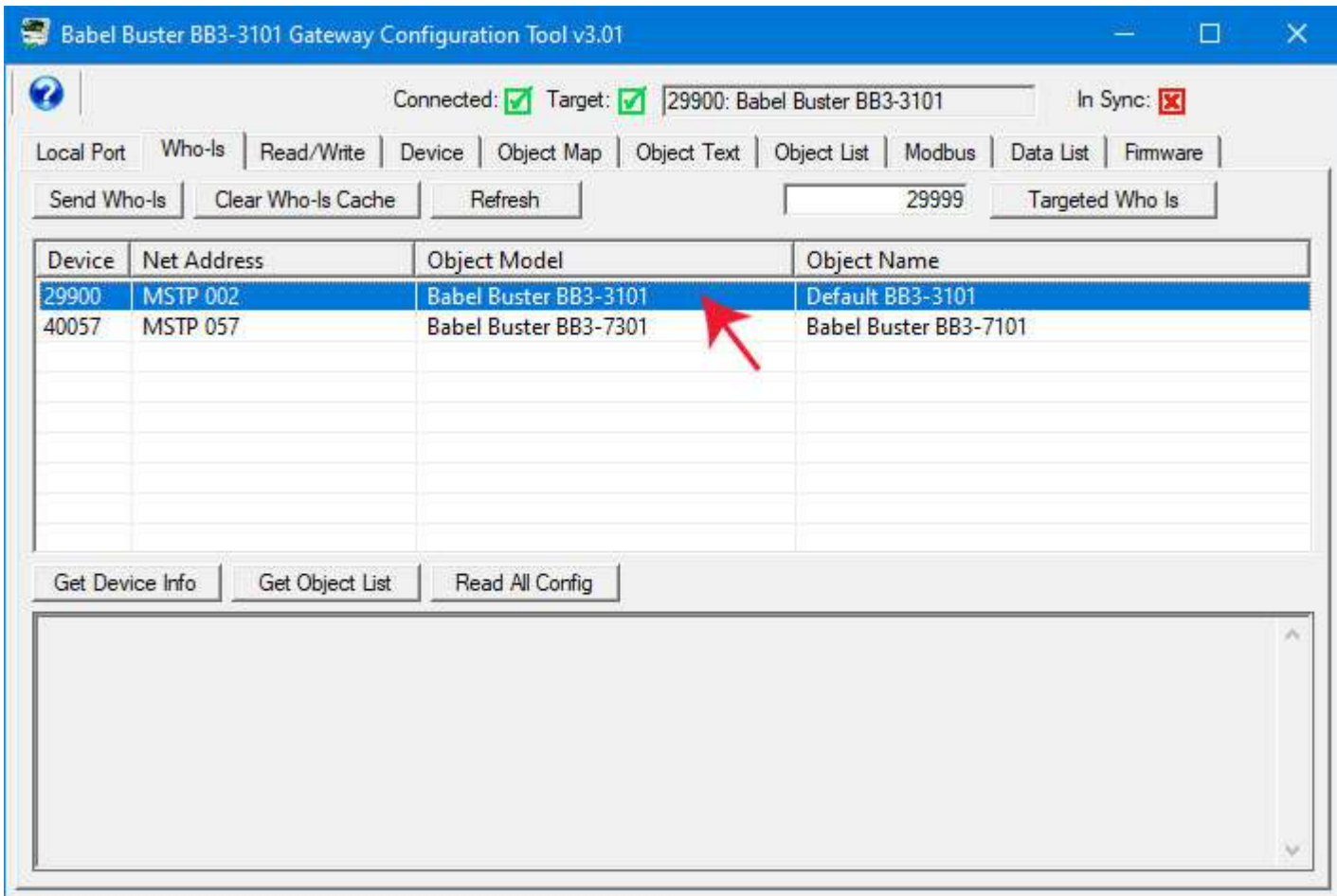
Device info written ok.

Object Allocation	Present	Pending
Analog Input Count	0	0
Analog Output Count	0	0
Analog Value Count	0	0
Binary Input Count	0	0
Binary Output Count	0	0
Binary Value Count	0	0
Multistate Input Count	0	0
Multistate Output Count	0	0
Multistate Value Count	0	0
Multistate Max State	0	0

Once you do this, you will need to return to the Who-Is page, clear the cache, repeat the Send Who-Is and Refresh, and reselect your target device. You will get an 'unknown device' error if you try to continue without reselecting the device since the device ID has now changed.



Repeat the device selection just as you did originally except now with the updated device instance.



You are now reconnected with your original device at its now device number. Click the Get Info button to re-read the device object properties just to verify that you are connected.

Babel Buster BB3-3101 Gateway Configuration Tool v3.01

Connected: ☒ Target: ☒ 29900: Babel Buster BB3-3101 In Sync: ☒

Local Port | Who-Is | Read/Write | Device | Object Map | Object Text | Object List | Modbus | Data List | Firmware

Object ID (device instance) 29900 Reassign

Device Name
Default BB3-3101

Device Description
Modbus RTU to BACnet MS/TP Gateway

Device Location
Unassigned

☒ ☐ Auto-Reset Errors

Max Master 127 Get Info

MS/TP Baud Rate 38400 Change

MS/TP Station/MAC Addr 2

New Password

Old Password

Reinit Device Reconfigure Get Counts Check Fit

Device info read ok.

Object Allocation	Present	Pending
Analog Input Count	0	0
Analog Output Count	0	0
Analog Value Count	0	0
Binary Input Count	0	0
Binary Output Count	0	0
Binary Value Count	0	0
Multistate Input Count	0	0
Multistate Output Count	0	0
Multistate Value Count	0	0
Multistate Max State	0	0

7.4 Get Object Counts from Device

Click 'Get Counts' to read the present object allocation configured in the device. The counts will show up in the 'Present' column. This set of counts tells you how many of each type of object are currently defined in the device. If a BACnet client performs an "auto discovery", this determines how many of each type of object will be discovered.

If you are working offline, i.e. creating a configuration ahead of time without any device connected, then entire the desired object counts in the Pending column, and click Get Counts.

If you will be using Multistate objects, the maximum state value is also set at the bottom of the object count list.

Babel Buster BB3-3101 Gateway Configuration Tool v3.01

Connected: ☒ Target: ☒ 29900: Babel Buster BB3-3101 In Sync: ☒

Local Port | Who-Is | Read/Write | Device | Object Map | Object Text | Object List | Modbus | Data List | Firmware

Object ID (device instance) 29900 Reassign

Device Name
Default BB3-3101

Device Description
Modbus RTU to BACnet MS/TP Gateway

Device Location
Unassigned

☒ ☐ Auto-Reset Errors

Max Master 127 Get Info

MS/TP Baud Rate 38400 Change

MS/TP Station/MAC Addr 2

New Password

Old Password

Reinit Device

Reconfigure

Get Counts

Check Fit

Device count read ok.

Object Allocation	Present	Pending
Analog Input Count	15	15
Analog Output Count	10	10
Analog Value Count	10	10
Binary Input Count	15	15
Binary Output Count	10	10
Binary Value Count	10	10
Multistate Input Count	5	5
Multistate Output Count	5	5
Multistate Value Count	5	5
Multistate Max State	10	10

7.5 Reconfigure Object Counts

Enter new object counts in the windows in the 'Pending' column.

Babel Buster BB3-3101 Gateway Configuration Tool v3.01

Connected: ☒ Target: ☒ 29999: Babel Buster BB3-3101 In Sync: ☒

Local Port | Who-Is | Read/Write | Device | Object Map | Object Text | Object List | Modbus | Data List | Firmware

Object ID (device instance) 29999 Reassign

Device Name
Default BB3-3101

Device Description
Modbus RTU to BACnet MS/TP Gateway

Device Location
Unassigned

☒ ☐ Auto-Reset Errors

Max Master 127 Get Info

MS/TP Baud Rate 38400 Change

MS/TP Station/MAC Addr 2

New Password

Old Password

Reinit Device Reconfigure Get Counts Check Fit

Object Allocation	Present	Pending
Analog Input Count	15	15
Analog Output Count	10	10
Analog Value Count	10	10
Binary Input Count	15	15
Binary Output Count	10	10
Binary Value Count	10	10
Multistate Input Count	5	5
Multistate Output Count	5	5
Multistate Value Count	5	5
Multistate Max State	10	10

Device count read ok.

Click 'Check Fit' to check to see if the counts are valid. Refer to Section 3.6, Changing Object Counts, for additional examples.

Babel Buster BB3-3101 Gateway Configuration Tool v3.01

Connected: ☒ Target: ☒ 29999: Babel Buster BB3-3101 In Sync: ☒

Local Port | Who-Is | Read/Write | Device | Object Map | Object Text | Object List | Modbus | Data List | Firmware

Object ID (device instance) 29999 Reassign

Device Name
Default BB3-3101

Device Description
Modbus RTU to BACnet MS/TP Gateway

Device Location
Unassigned

☒ ☐ Auto-Reset Errors

Max Master 127 Get Info

MS/TP Baud Rate 38400 Change

MS/TP Station/MAC Addr 2

New Password

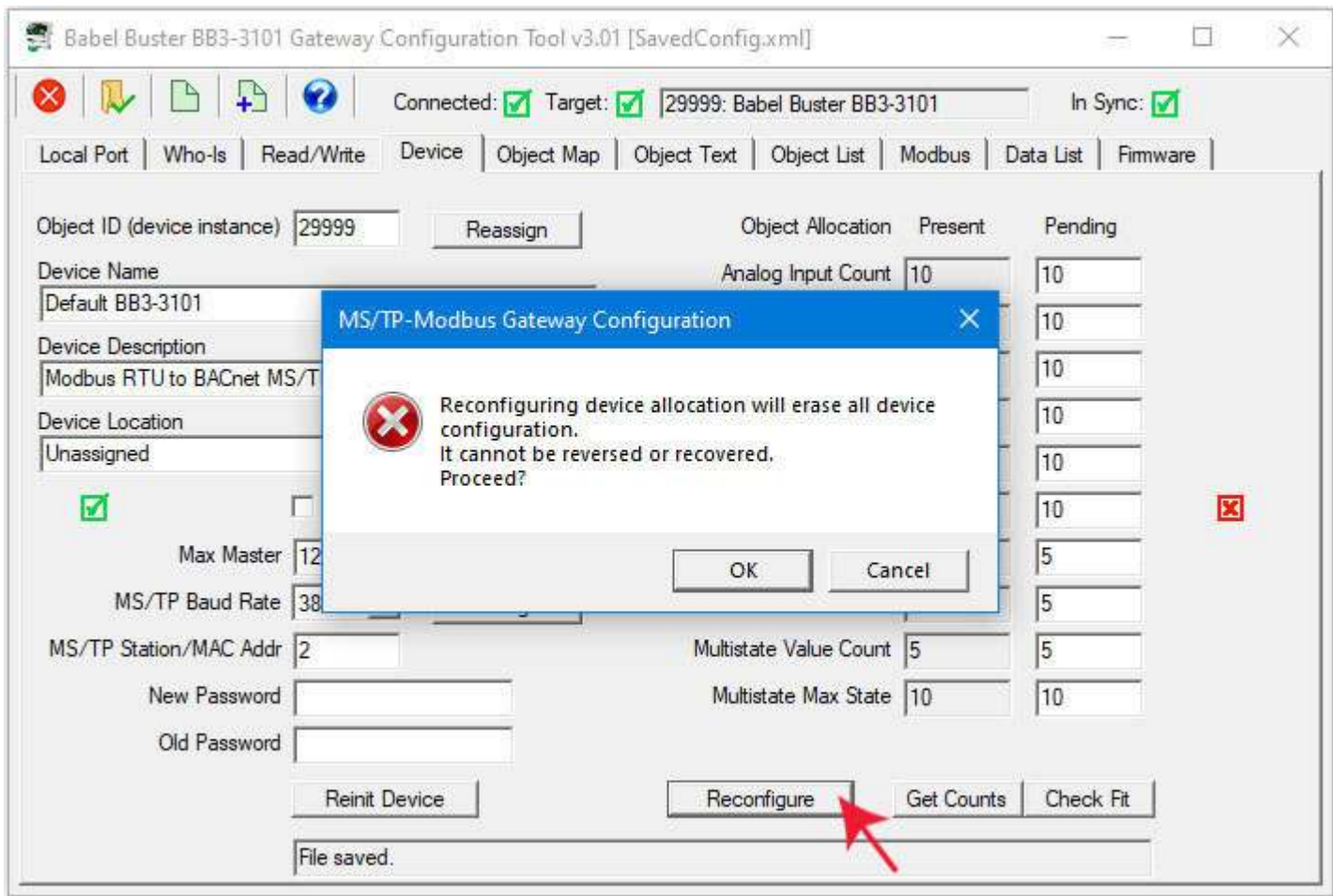
Old Password

Reinit Device Reconfigure Get Counts Check Fit

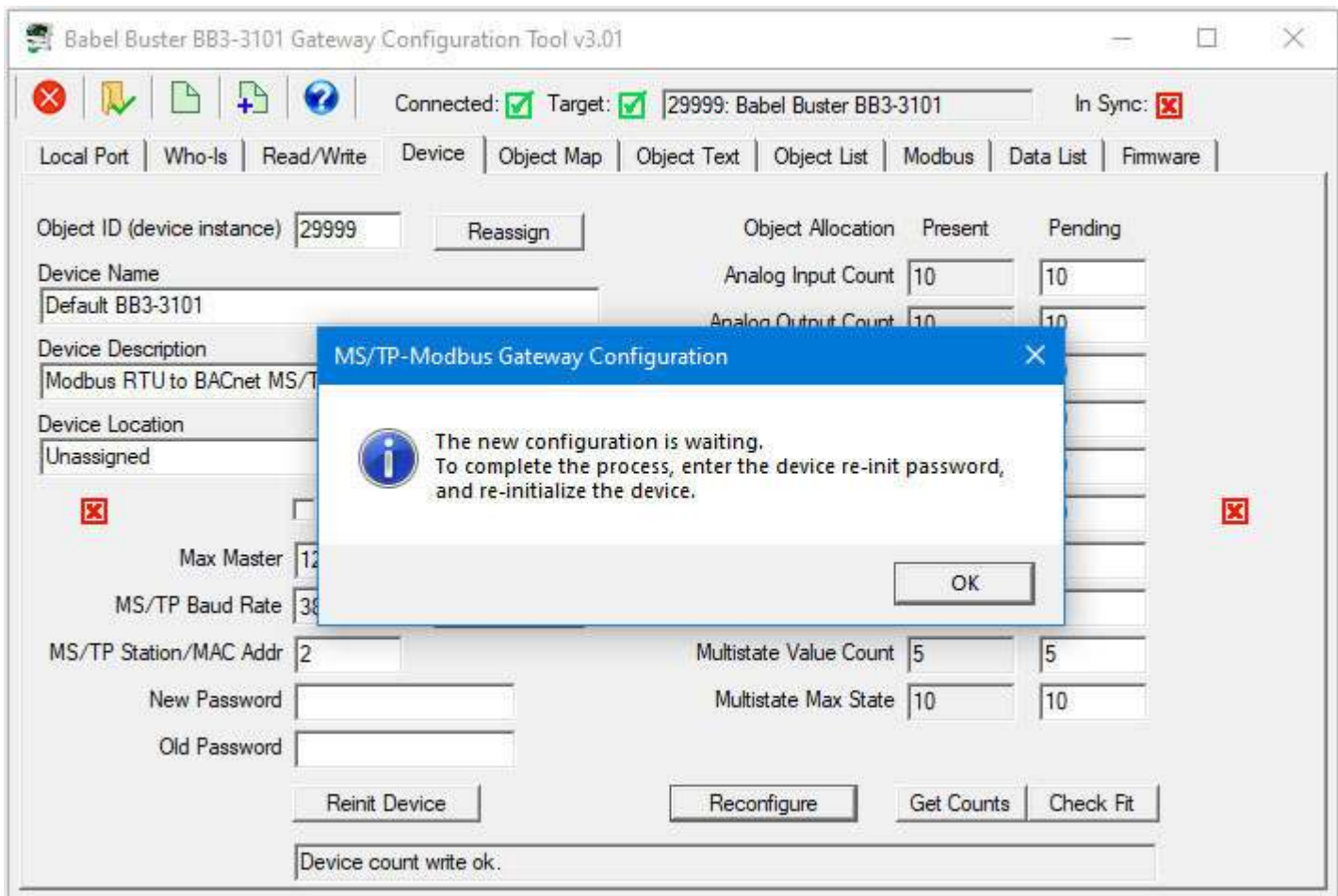
Device count read ok.

Object Allocation	Present	Pending
Analog Input Count	15	10
Analog Output Count	10	10
Analog Value Count	10	10
Binary Input Count	15	10
Binary Output Count	10	10
Binary Value Count	10	10
Multistate Input Count	5	5
Multistate Output Count	5	5
Multistate Value Count	5	5
Multistate Max State	10	10

When you are ready to reconfigure the device with the new object allocation, click the 'Reconfigure' button. *NOTE: This action will completely erase all prior object map configuration.*



The configuration is now ready to be applied.



At this point, the desired new object counts have been sent to the gateway device, but the device has not yet implemented the change. You need to reinitialize the device to finalize the process. Enter the password (default is "buster") and click the Reinit Device button.

Babel Buster BB3-3101 Gateway Configuration Tool v3.01 [SavedConfig.xml]

Connected: ☒ Target: ☒ 29999: Babel Buster BB3-3101 In Sync: ☒

Local Port | Who-Is | Read/Write | Device | Object Map | Object Text | Object List | Modbus | Data List | Firmware

Object ID (device instance) 29999 Reassign

Device Name Default BB3-3101

Device Description Modbus RTU to BACnet MS/TP Gateway

Device Location Unassigned

☒ Auto-Reset Errors

Max Master 127 Get Info

MS/TP Baud Rate 38400 Change

MS/TP Station/MAC Addr 2

New Password

Old Password buster

Reinit Device

Reconfigure Get Counts Check Fit

Device re-init sent.

Object Allocation	Present	Pending
Analog Input Count	10	10
Analog Output Count	10	10
Analog Value Count	10	10
Binary Input Count	10	10
Binary Output Count	10	10
Binary Value Count	10	10
Multistate Input Count	5	5
Multistate Output Count	5	5
Multistate Value Count	5	5
Multistate Max State	10	10

You can now click the Get Counts button again to verify the counts in the device. The numbers on the screen will be replaced with whatever is actually in the device, and if all went well, they will be what you thought you had asked for. But if you had failed to enter the correct password, your request was not executed (and there would have been an error message in the status window at the bottom of the screen).

Babel Buster BB3-3101 Gateway Configuration Tool v3.01

Connected: ☒ Target: ☒ 29999: Babel Buster BB3-3101 In Sync: ☒

Local Port | Who-Is | Read/Write | Device | Object Map | Object Text | Object List | Modbus | Data List | Firmware

Object ID (device instance) 29999

Device Name
Default BB3-3101

Device Description
Modbus RTU to BACnet MS/TP Gateway

Device Location
Unassigned

☒ ☐ Auto-Reset Errors

Max Master 127

MS/TP Baud Rate 38400

MS/TP Station/MAC Addr 2

New Password

Old Password

Object Allocation	Present	Pending
Analog Input Count	10	10
Analog Output Count	10	10
Analog Value Count	10	10
Binary Input Count	10	10
Binary Output Count	10	10
Binary Value Count	10	10
Multistate Input Count	5	5
Multistate Output Count	5	5
Multistate Value Count	5	5
Multistate Max State	10	10

Device count read ok.

7.6 Fixing "Invalid Configuration"

Depending on what the previous object counts had been and what you are trying to change them to, you may get the error indicated in the log window at the bottom of the screen illustrated below. This message appears after clicking OK on the dialog box that appears when you click the Reconfigure button.

Babel Buster BB3-3101 Gateway Configuration Tool v3.01

Connected: ☒ Target: ☒ 29999: Babel Buster BB3-3101 In Sync: ☒

Local Port | Who-Is | Read/Write | Device | Object Map | Object Text | Object List | Modbus | Data List | Firmware

Object ID (device instance) 29999 Reassign

Device Name
Default BB3-3101

Device Description
Modbus RTU to BACnet MS/TP Gateway

Device Location
Unassigned

☒ Auto-Reset Errors

Max Master: 127 Get Info

MS/TP Baud Rate: 38400 Change

MS/TP Station/MAC Addr: 2

New Password:

Old Password: buster

Reinit Device Reconfigure Get Counts Check Fit

Object Allocation	Present	Pending
Analog Input Count	800	980
Analog Output Count	120	5
Analog Value Count	20	5
Binary Input Count	20	5
Binary Output Count	20	5
Binary Value Count	10	5
Multistate Input Count	10	5
Multistate Output Count	10	5
Multistate Value Count	10	5
Multistate Max State	10	10

Error: Invalid configuration data or sequence

If you add up the object counts in the Pending column, the total does not exceed the maximum of 1024. So why is this error happening?

The writing of object counts from the configuration tool to the gateway is largely transparent - you just click the button. But what is really happening is that each count is a property value in the BACnet Device object in the gateway. To assure that any arbitrary object count can be written, it is necessary to re-process the total allocation as each new count value is written. When the first count of 980 in above example is written to request 980 Analog Input objects, the gateway is still retaining 120 Analog Output objects. That adds up to 1100, which is over the limit, and the result is the "invalid configuration" error reply.

In general, if counts higher on the list are trying to take away from counts lower on the list, you are likely to run into this problem. The solution is to temporarily assign zero objects all the way down the list. Repeat the object allocation with all zero counts as illustrated below.

Babel Buster BB3-3101 Gateway Configuration Tool v3.01

Connected: ☒ Target: ☒ 29999: Babel Buster BB3-3101 In Sync: ☒

Local Port | Who-Is | Read/Write | Device | Object Map | Object Text | Object List | Modbus | Data List | Firmware

Object ID (device instance) 29999 Reassign

Device Name
Default BB3-3101

Device Description
Modbus RTU to BACnet MS/TP Gateway

Device Location
Unassigned

☒ ☐ Auto-Reset Errors

Max Master 127 Get Info

MS/TP Baud Rate 38400 Change

MS/TP Station/MAC Addr 2

New Password

Old Password buster

Reinit Device

Reconfigure

Get Counts

Check Fit

Object Allocation	Present	Pending
Analog Input Count	0	0
Analog Output Count	0	0
Analog Value Count	0	0
Binary Input Count	0	0
Binary Output Count	0	0
Binary Value Count	0	0
Multistate Input Count	0	0
Multistate Output Count	0	0
Multistate Value Count	0	0
Multistate Max State	0	0

Device count read ok.

Now go back and retry the counts that were previously not working. Pictured below is the successful reconfiguration that had previously failed above.

Babel Buster BB3-3101 Gateway Configuration Tool v3.01

Connected: ☒ Target: ☒ 29999: Babel Buster BB3-3101 In Sync: ☒

Local Port | Who-Is | Read/Write | Device | Object Map | Object Text | Object List | Modbus | Data List | Firmware

Object ID (device instance) 29999 Reassign

Device Name
Default BB3-3101

Device Description
Modbus RTU to BACnet MS/TP Gateway

Device Location
Unassigned

☒ Auto-Reset Errors

Max Master 127 Get Info

MS/TP Baud Rate 38400 Change

MS/TP Station/MAC Addr 2

New Password

Old Password buster

Reinit Device Reconfigure Get Counts Check Fit

Device count read ok.

Object Allocation	Present	Pending
Analog Input Count	980	980
Analog Output Count	5	5
Analog Value Count	5	5
Binary Input Count	5	5
Binary Output Count	5	5
Binary Value Count	5	5
Multistate Input Count	5	5
Multistate Output Count	5	5
Multistate Value Count	5	5
Multistate Max State	10	10

7.7 Export and Import Configuration with XML File

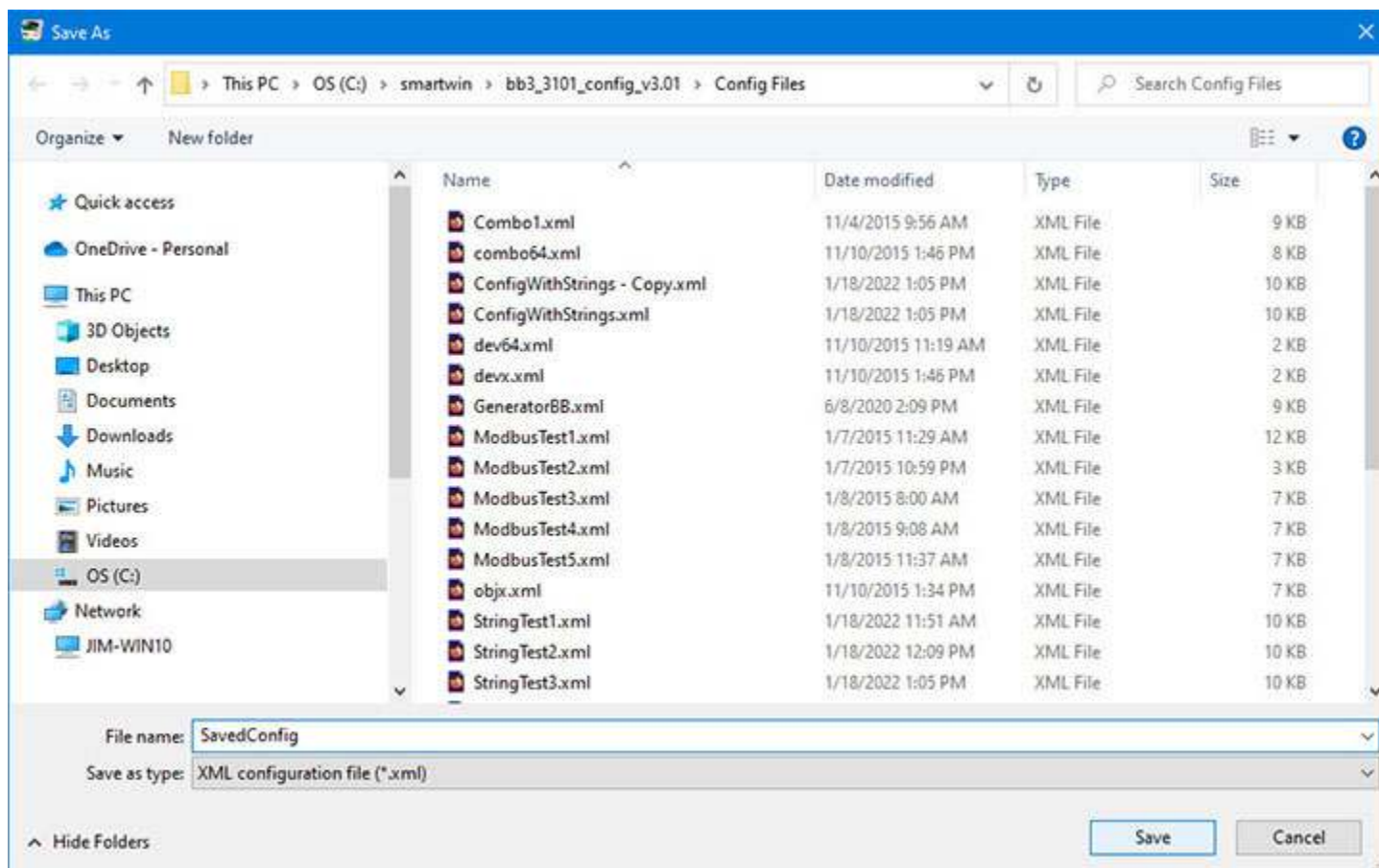
Click the New File icon to save your configuration to a new XML file. You have the option of saving only the device properties, or only the object maps, but the default (and most often preferred) option is to save both.

If you want to re-save a previously opened configuration file, following making some changes to configuration, click the first File icon rather than the second icon. The first green icon is "Save" and the second icon is "Save As", where "save as" means create a new file.

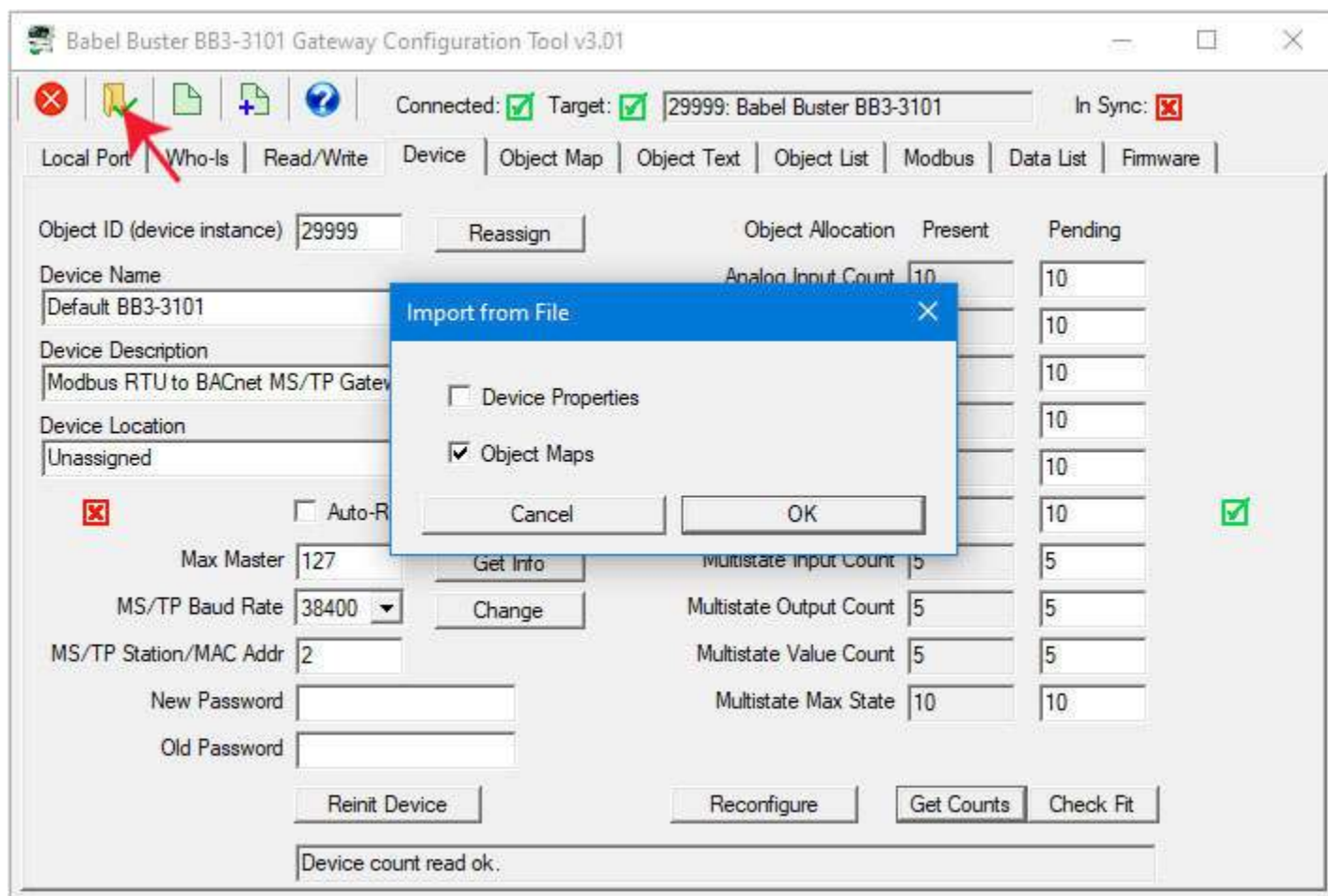
You have the option of saving just device properties, just object maps, or both. To copy object mapping to another gateway, you will usually want to save just the object maps. To create a full backup of the existing configuration for this gateway, save both.



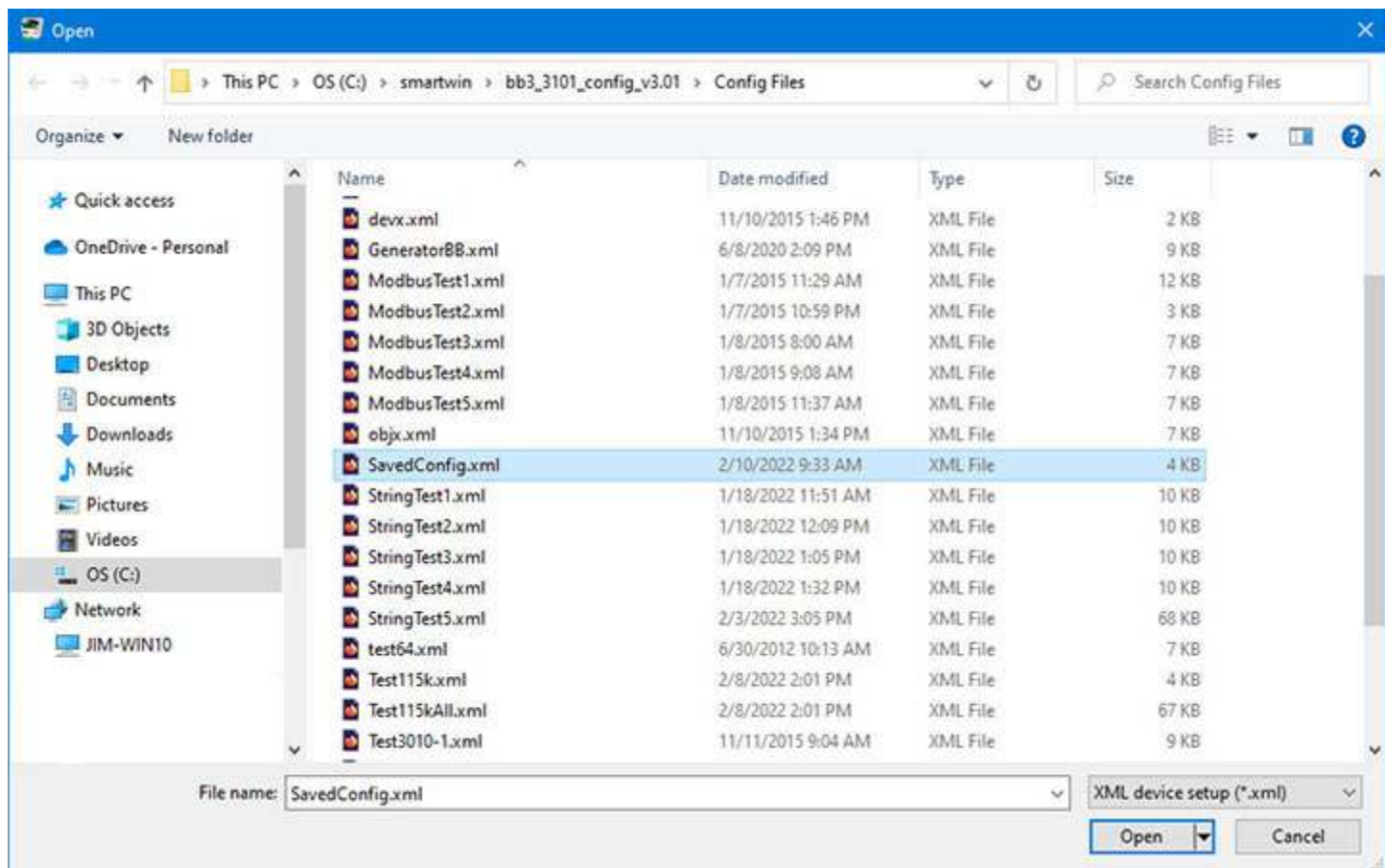
When you click OK in the "Export to File" dialog, you will next be directed to the Windows file dialog where you may enter a name for your new file.



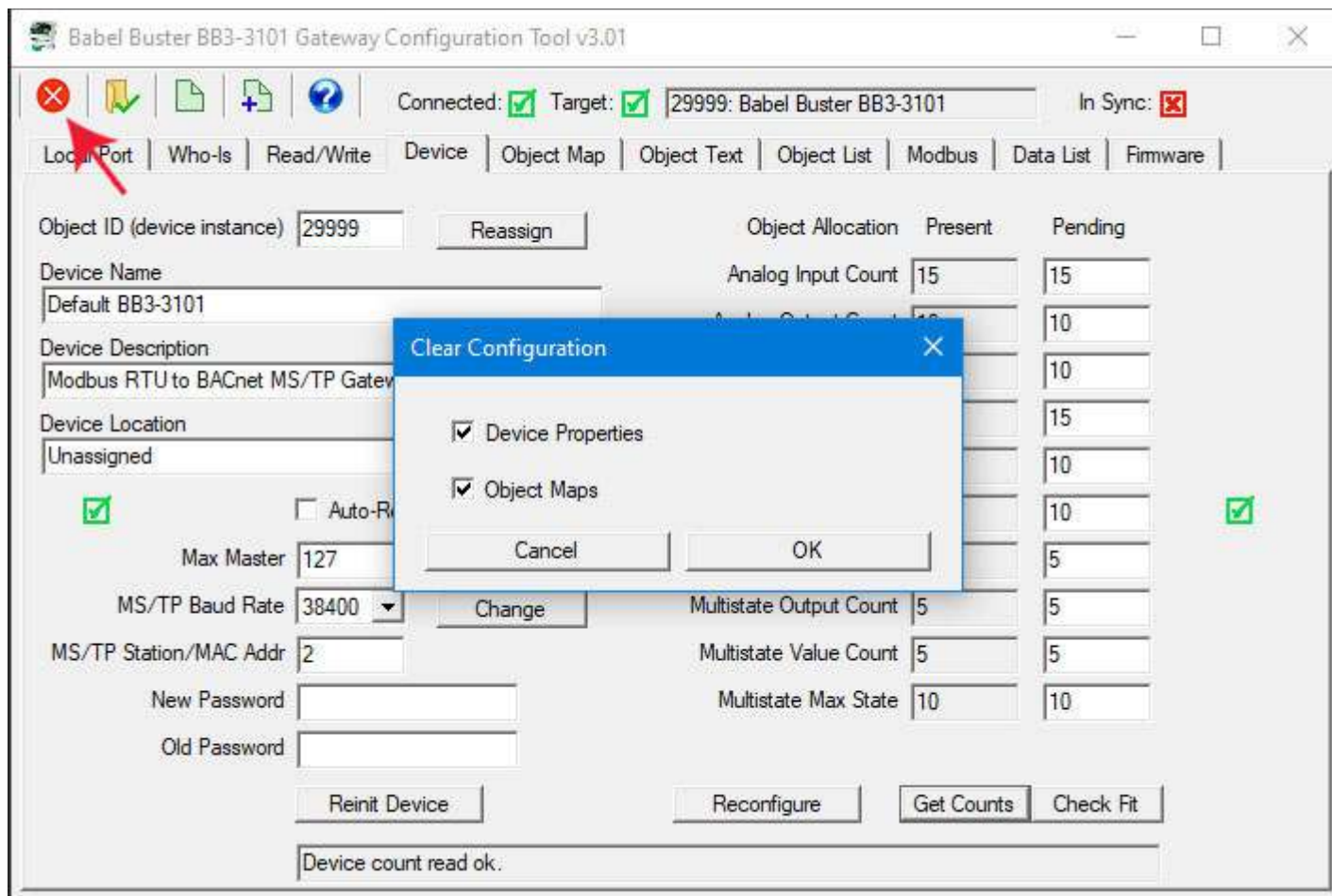
To load a previously saved configuration into the tool, click the File Open icon. You will have the option of importing device properties, object maps, or the default both.



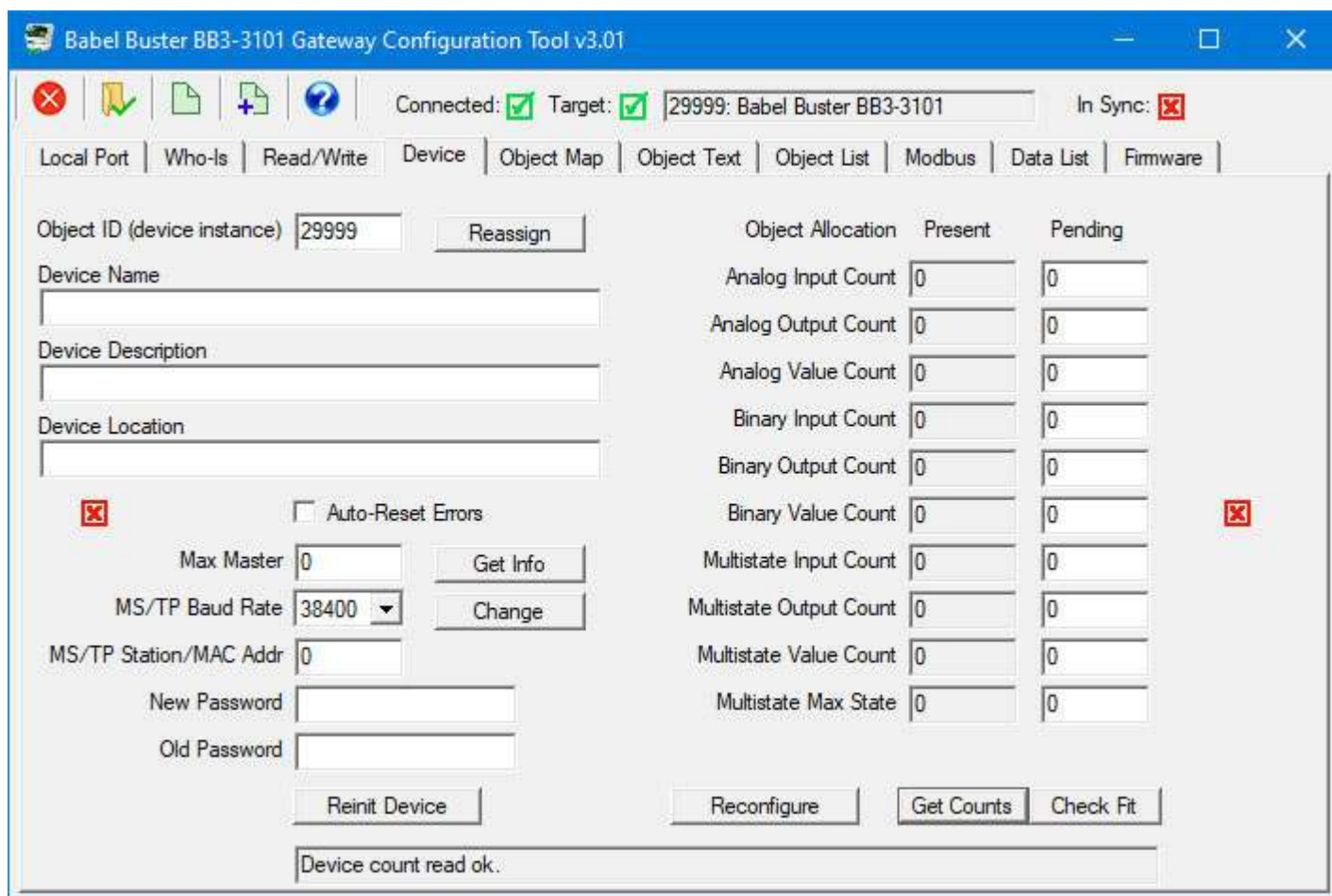
Upon clicking OK in the "Import from File" dialog, you will be directed to the familiar Windows file dialog where you can select the file to be opened.



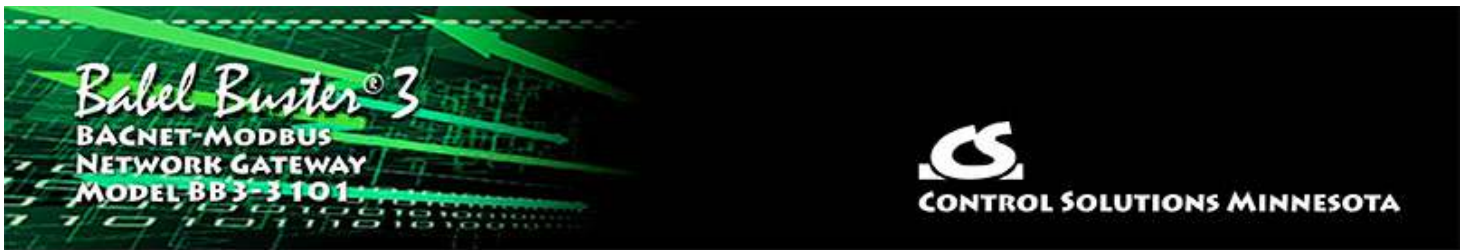
To wipe the slate completely clean in the configuration tool, click the Clear icon, which is the first icon on the toolbar. You will be given the option of clearing device properties, object maps, or both. Or just click Cancel if you clicked the Clear icon by mistake.



Clear everything will clear all but the device instance.



Note: If you have an XML configuration file saved from a BB2-3010, you can directly import that into your BB3-3101 by simply opening that file on this page.



8 Tool 'Object Map' Page

You gave the BB3-3101 gateway its identity as a BACnet device on the Device page in the previous section. The Object Map page is where you tell the gateway what to do.

8.1 Internal BACnet Object Configuration Parameters

The properties that define each BACnet object in the gateway are displayed on the upper section of the Object Map page.

The screenshot shows the 'Babel Buster BB3-3101 Gateway Configuration Tool v3.01' window. The 'Object Map' tab is selected. The interface includes a toolbar with icons for Local Port, Who-Is, Read/Write, Device, Object Map, Object Text, Object List, Modbus, Data List, and Firmware. The 'Object Map' tab displays configuration parameters for an 'Analog Input' object (Instance 1). The 'Object Type/Instance' dropdown is set to 'Analog Input' and '1'. The 'Object Name' is 'Name of Analog Input 1' and the 'Units' are 'no_units'. The 'Description' is 'Description of Analog Input 1'. There are checkboxes for 'Read Periodic', 'Write Periodic', 'Set Default on Power-Up', 'Enable Max Quiet Time', 'Write on Delta', and 'Set Default on Comm Fail'. The 'Poll Rate (Sec)' is 0, 'Slope/Scale Factor' is 0, 'Timeout (Sec)' is 0, 'Initial COV Increment' is 0, 'Intercept/Offset' is 0, 'Max. Quiet Time (Sec)' is 0, 'Initial COV Period' is 0, 'Delta for Send' is 0, 'Default Value' is 0, and 'Initial Relinquish Default' is empty. There are also checkboxes for 'Map Modbus Object' and 'Map BACnet Object'. At the bottom, there are buttons for 'Insert', 'Add', 'Delete', 'Read Device', and 'Write Device'. The status bar shows 'OBJECT_ANALOG_INPUT #1 -- read ok.'.

Object Type/Inst

Select BACnet object type from the list, and enter or scroll to the desired object instance. The object number will stop scrolling at the limit set by object counts on the Device page.

Object Name	Enter an alphanumeric name for the object. The name must be unique, and will be checked for uniqueness. If it is a duplicate name, you will see an error message and the dialog will remain open.
Description	Enter any arbitrary description for this object.
Units	Select a BACnet units type from the list.
Read Periodic	Check this box if the BACnet object will be periodically reading from the slave Modbus device or slave BACnet device. 'Read' is valid for Input and Value type objects.
Write Periodic	Check this box if the BACnet object will be periodically writing to the slave Modbus device or slave BACnet device. 'Write' is valid for Output and Value type objects.
Write on Delta	Check this box if the BACnet object will be writing to the slave device only when the object's value changes by a specified 'Delta'. It is valid to check both Write Periodic and Write on Delta at the same time, or check just one or the other. 'Write' is valid for Output and Value type objects.
Set Default on Power-Up	Check this box if the object should assume the default value every time the BB3-3101 powers up.
Set Default on Comm Fail	Check this box if the object should assume the default value when communication with the slave device has failed some number of times.
Enable Max Quiet Time	Check this box to enable maximum quiet time. This is applicable to an object which is set to Write on Delta. The result is that if there has been no change within the max quite time, the slave device will be re-written anyway.
Poll Rate (Sec)	This rate applies to periodic reading or writing, and simply specifies how often the slave device is read from or written to.
Initial COV Increment	The initial COV increment specified here will apply when another BACnet client subscribes to this object's COV with a simple Subscribe COV. This initial increment will remain in effect until some other BACnet client writes a new COV increment to this object. If the BACnet client does a Subscribe COV Property, then the COV increment specified in the subscription is used instead of this Initial COV Increment. COV Increment applies only to Analog objects. COV will be reported on any change in a Binary or Multi-state object.
Initial COV Period	The initial COV Period will take effect when the Babel Buster is first powered up. After that, any BACnet client can write a new COV Period to the object. The COV Period produces a periodic COV notification regardless of value change. COV Period is valid for all object types. Enter zero to disable periodic COV and provide notification only upon change specified by the increment.

Initial Relinquish Default	The initial Relinquish Default will take effect when the gateway is first powered up. After that, any BACnet client can write a new Relinquish Default value to the object. The Relinquish Default value applies to any Commandable object, and the Present Value of the object will assume this value when all 16 priorities have been relinquished.
Slope/Scale Factor	Scaling applies the formula $y=mx+b$. When reading from the slave, the raw data as read is multiplied by the scale factor, then the offset is added to produce the resulting Present Value. When writing to the slave, the offset is first subtracted from Present Value, and that result is divided by the scale factor to produce the raw data actually written to the slave device. NOTE: If no scale factor is given (zero is entered), no scaling will be done, as if slope=1 and intercept=0.
Intercept/Offset	The offset portion of the scaling as noted above.
Delta for Send	Enter the threshold for writing when configured to Write on Delta. If a value of 5.0 is entered, the value must change by more than 5.0 before the slave device will be written to. Delta applies only to Analog objects. Binary and Multi-state objects will send on any change when Write on Delta is enabled.
Timeout (Sec)	Timeout for Modbus queries is specified on the Modbus page. When the remote device is another BACnet device, the timeout given here specifies the amount of time the BB3-3101 will wait for a response before calling it an error.
Max. Quiet Time	This is applicable to an object which is set to Write on Delta. The result is that if there has been no change within this amount of time, the slave device will be re-written anyway. In addition to entering a nonzero value here, you must check the 'Enable Max Quiet Time' box.
Default Value	This is the value that should become the Present Value upon power-up or upon communications failure, if either of these options are selected by the appropriate check boxes above.
Read Fails before Fault	When 'Set Default on Comm Fail' is checked, this entry will allow you to disregard a small number of spurious errors. At least this many errors must occur consecutively before a fault will actually be flagged. Causing a fault as a result of a single instance of spurious noise on a communication line can be a nuisance. This setting allows quieting the nuisance fault notifications.

8.2 Modbus Master Configuration Parameters

You map the BACnet object to a Modbus register, assuming the gateway will be operating as a Modbus master, by clicking on "Map Modbus Object". When you do so, additional parameters specific to mapping of a Modbus register will appear.

Babel Buster BB3-3101 Gateway Configuration Tool v3.01

Connected: ☒ Target: ☒ 29999: Babel Buster BB3-3101 In Sync: ☒

Local Port | Who-Is | Read/Write | Device | Object Map | Object Text | Object List | Modbus | Data List | Firmware

Object Type/Instance: Analog Input 1 Units: no_units

Object Name: Name of Analog Input 1

Description: Description of Analog Input 1

☒ Read Periodic ☐ Write Periodic ☐ Set Default on Power-Up ☐ Enable Max Quite Time ☒

☐ Write on Delta ☐ Set Default on Comm Fail

Poll Rate (Sec): 2 Slope/Scale Factor: 0 Timeout (Sec): 0

Initial COV Increment: 0 Intercept/Offset: 0 Max. Quiet Time (Sec): 0

Initial COV Period: 0 Delta for Send: 0 Default Value: 0

Initial Relinquish Default: Read Fails before Fault: 0

☒ Map Modbus Object ☐ Map BACnet Object

Register Number 1..N: 1 Unit/Slave Addr: 1 ☒ Member of Packed Register

Register Type: Holding Register (fc 16) Mask (Hex): 0000F000 ☐ Pack Mixed Object Types

Register Format: Unsigned 32-bit (high reg first) Fill (Hex): 00000000

Insert Add Delete Read Device Write Device OBJECT_ANALOG_INPUT #1 - write ok.

Map Modbus Object	Check this box to map the displayed BACnet object to a register in a slave Modbus device. (If using the BB3-3101 as a Modbus slave, do not check this box, and refer instead to Appendix D for slave register mapping.)
Register Number 1..N	Enter the Modbus register number between 1 and 65535. The register number is raw address plus one. Therefore if your Modbus device's documentation indicates that the first register is at address zero, add one to every address to get the register 'number'. Note: Do not enter Modicon numbers like 40001 for the first holding register. If the Modbus device documentation indicates register 40001, this means you should select "holding register" as the register type, and just "1" for register number.
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.

Register Format	<p>Register format options include 16-bit, 32-bit, or 64-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. For 64-bit, you are reading/writing four 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 from the list of formats.</p> <p>There is also a Bit format. This ONLY applies to coil and discrete input. If you are attempting to read a single bit from a holding register, you must read the register as unsigned 16-bit (or 32-bit if applicable) and then apply a mask. The format 'Bit' refers to the format of data in the Modbus slave; it does not refer to your desired end result.</p>
Unit/Slave Addr	<p>This is the Modbus RTU 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 used to look up the IP address of the slave. This value is not used when the gateway itself is the slave.</p>
Mask (Hex)	<p>When extracting a single bit or set of bits from a packed Modbus register, this mask specifies where in the register the desired bits are found. This mask is entered as a 4-digit hexadecimal number representing 16 bits. After masking data read from Modbus by performing a bit-wise And between the data and this mask, the resulting data is right shifted so that the least significant bit of interest becomes the LSB of data. When writing, this process is reversed.</p> <p>NOTE: The mask is only applied if "Member of Packed Register" is selected.</p>
Fill (Hex)	<p>Applicable only when writing, this value is optionally used to always set specified bits. This value is entered as a 4-digit hexadecimal number representing 16 bits. The data to be written to the Modbus slave is bit-wise logical Or-ed with this value just before being written to the device.</p> <p>NOTE: The fill is only applied if "Member of Packed Register" is selected, and is only applicable when writing to Modbus.</p>
Member of Packed Register	<p>Check this box if the BB3-3101 should apply mask and fill, and should look at the next consecutive object map, or is included in the group started by a previous object map, to potentially combine this and at least one other object map to collect multiple BACnet objects into a single Modbus register for writing, or distribute from a single Modbus register to multiple BACnet objects when reading.</p>

Pack Mixed Object Types	Check this box if your packed Modbus register needs to be mapped to more than one type of object, e.g. both Binary and Analog objects are involved. Do not check this box unless truly needed because it does result in searching all object maps, looking for members of the group, every time this object is processed. Unchecking this box and making sure object maps are in consecutive order when only one object type is needed will result in more efficient processing.
-------------------------	--

The options available for Modbus register format are displayed in the drop-down list. The corresponding notation that should be used when creating a CSV file is shown in the table below. If no format column is included in your CSV file, unsigned 16-bit register will be assumed.

☒ Map Modbus Object
 ☐ Map BACnet Object

Register Number 1..N:
 Unit/Slave Addr:
☒ Member of Packed Register

Register Type: Holding Register (fc 16)
 Mask (Hex):
☐ Pack Mixed Object Types

Register Format: Unsigned 32-bit (high reg first)
 Fill (Hex):

Insert
 Add
 Delete

Undefined
 Signed 16-bit
 Unsigned 16-bit
 Signed 32-bit (high reg first)
Unsigned 32-bit (high reg first)
 Floating Point (high reg first)
 Bit (coil/discrete)
 Signed 32-bit (low reg first)
 Unsigned 32-bit (low reg first)
 Floating Point (low reg first)
 Signed 64-bit (high reg first)
 Signed 64-bit (low reg first)

OBJECT_ANALOG_INPUT #1 -- read ok.

Format designations as used in CSV or XML files:

CSV Notation	Description
SIGN	Signed 16-bit
UNSI	Unsigned 16-bit
SDBE	Signed 32-bit (high reg first)
UDBE	Unsigned 32-bit (high reg first)
FPBE	Floating Point (high reg first)
BBIT	Bit (coil/discrete)
SDLE	Signed 32-bit (low reg first)
UDLE	Unsigned 32-bit (low reg first)
FPLE	Floating Point (low reg first)
SQBE	Signed 64-bit (high reg first)
SQLE	Signed 64-bit (low reg first)

8.3 BACnet Client Configuration Parameters

You can map the BACnet object in the gateway to read or write a BACnet object in some other BACnet device on the same network. This is most often used if the gateway will be

treated as a Modbus slave, and some other Modbus master wants to read and write data in a BACnet device on your MS/TP network. When you click "Map BACnet Object, the parameters specific to mapping this object to another BACnet device will appear.

The screenshot shows the 'Object Map' tab of the Babel Buster BB3-3101 Gateway Configuration Tool v3.01. The interface includes a toolbar with icons for Local Port, Who-Is, Read/Write, Device, Object Map (selected), Object Text, Object List, Modbus, Data List, and Firmware. The main configuration area is divided into several sections:

- Object Type/Instance:** Analog Input, 4. Units: no_units.
- Object Name:** Name of Analog Input 4.
- Description:** Description of Analog Input 4.
- Read/Write Settings:**
 - ☒ Read Periodic
 - ☐ Write Periodic
 - ☐ Set Default on Power-Up
 - ☐ Enable Max Quiet Time ☒
 - ☐ Write on Delta
 - ☐ Set Default on Comm Fail
- Timing and Scaling:**
 - Poll Rate (Sec): 5
 - Slope/Scale Factor: 0
 - Timeout (Sec): 1
 - Initial COV Increment: 0
 - Intercept/Offset: 0
 - Max. Quiet Time (Sec): 0
 - Initial COV Period: 0
 - Delta for Send: 0
 - Default Value: 0
 - Initial Relinquish Default: (empty)
 - Read Fails before Fault: 0
- Mapping Options:**
 - ☐ Map Modbus Object
 - ☒ Map BACnet Object
- Remote Device Instance:** 40057
- Property:** Present Value, 85
- Bit Num:** 0
- Object Type:** Analog Input
- Instance:** 1
- Data Type:** Null
- Mac address (no who-is):** 0
- Array Index:** -1
- Priority:** None

At the bottom, there are buttons for Insert, Add, Delete, Read Device, and Write Device. The status bar shows 'OBJECT_ANALOG_INPUT #4 -- write ok.'

Map BACnet Object	Check this box when the BB3-3101 should look for another BACnet device as the slave device (instead of a Modbus slave). This box can be checked any time, but is of particular interest when using the BB3-3101 as a Modbus slave and BACnet master for connecting BACnet devices to a Modbus network.
Remote Device Instance	Enter the device instance of the remote BACnet server (slave). The remote device must be able to support Who-Is/I-Am for locating device by device instance.
Object Type	Select the object type from the list, such as Analog Input, etc.
Mac address (if no who-is)	Sets the MS/TP MAC address (station ID) of the remote device if it is a slave-only device or does not support dynamic binding (Who-Is/I-Am). Normally you would leave this set to zero so that the remote device will be discovered automatically when the gateway broadcasts a "Who-Is" message. This setting applies to MS/TP only. BACnet IP devices (on the other side of a router) are required to support dynamic binding (Who-Is/I-Am).
Property	Select the property to be read/written. A collection of most often used property types are included in the drop list. If the desired property is not shown, select 'Other-->' and enter the property type code in the window next to the list. The property

	type codes are those defined by the BACnet standard. For example, Present Value can also be obtained using 'Other --> 85'.
Instance	Select the object instance. There are usually multiple instances of objects in a device. This identifies which one you want.
Array Index	Some properties are an array of values. If applicable, enter the array index here. Enter -1 to indicate no array index is to be used. Zero is a valid index.
Bit Num	When the property type is a bit string, this entry may be used to specify which bit you wish to single out, especially when applying the result to a Binary object. If no bit number is provided, the entire bit string will be assembled as an integer value representing a mask containing all bits.
Data Type	Select the data type expected from this list. Note that some types are not supported here. Character string, for example, is one type of data that cannot be forced into Analog, Binary, or Multi-state objects by the BB3-3101.
Priority	You must use a priority level when writing to a Commandable object. If the object is not commandable, you must select 'None' here to avoid having the request rejected.

Babel Buster BB3-3101 Gateway Configuration Tool v3.01

Connected: ☒ Target: ☒ 29999: Babel Buster BB3-3101 In Sync: ☒

Local Port | Who-Is | Read/Write | Device | Object Map | Object Text | Object List | Modbus | Data List | Firmware

Object Type/Instance: Analog Output 1 Units: no_units

Object Name: Name of Analog Output 1

Description: Description of Analog Output 1

☐ Read Periodic ☐ Write Periodic ☐ Set Default on Power-Up ☐ Enable Max Quiet Time ☒

☒ Write on Delta ☐ Set Default on Comm Fail

Poll Rate (Sec): 0 Slope/Scale Factor: 0 Timeout (Sec): 1

Initial COV Increment: 0 Intercept/Offset: 0 Max. Quiet Time (Sec): 0

Initial COV Period: 0 Delta for Send: 0 Default Value: 0

Initial Relinquish Default: 0 Read Fails before Fault: 0

☐ Map Modbus Object ☒ Map BACnet Object

Remote Device Instance: 40057 Property: Present Value 85 Bit Num: 0

Object Type: Analog Output Instance: 1 Data Type: Real

Mac address (no who-is): 0 Array Index: -1 Priority: 7

Insert Add Delete Read Device Write Device OBJECT_ANALOG_OUTPUT #1 -- write ok.

Be sure to set the Priority when writing to a Commandable object. Failure to do so will result in the request being rejected. Conversely, be sure to select "None" when writing to

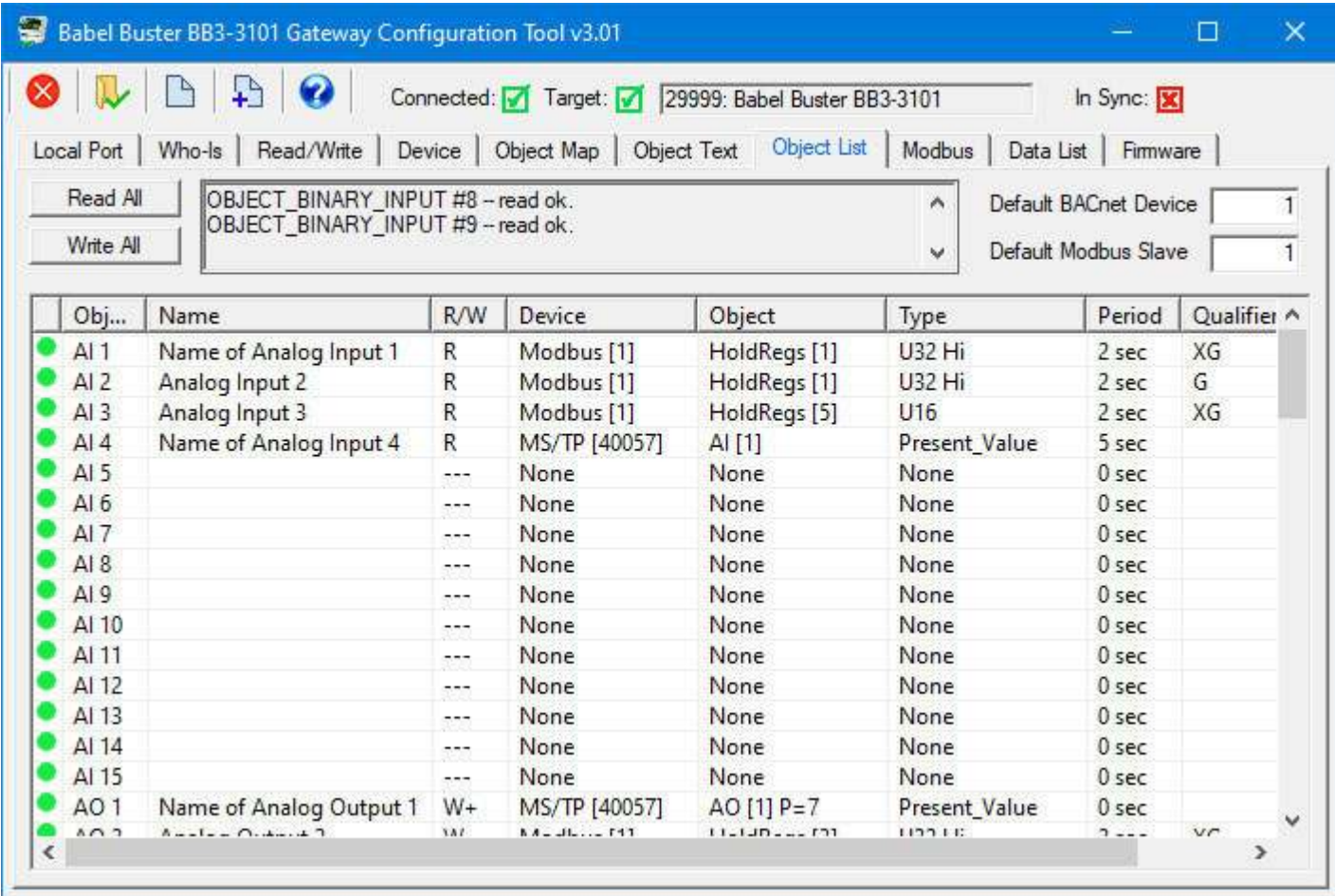
a non-commandable object. Including priority for an object that does not expect it will also result in the request being rejected.

NOTE: There is specific behavior that applies when a commandable object (e.g. Analog Output) is writing to another commandable object as illustrated above. In this instance, priority 7 will be used to set the Present Value of the remote object as long as the local object is not in the relinquished state. If no priorities are set in the local object, then NULL will be written to priority 7 in the remote object effectively relinquishing that priority in the remote object.

8.4 Insert, Add, Delete Objects

Clicking 'Insert' will insert a new map before the currently displayed object map. Clicking 'Add' will insert a new map after the currently displayed object map. Clicking 'Delete' will remove the currently displayed map and slide the remainder of the list up by one slot on the list.

The only time position matters much in the list is when configuring a packed register that maps a single Modbus register to multiple BACnet objects. The members of the packed register must appear in consecutive maps unless the 'Pack Mixed Object Types' is checked. Do not check 'mixed object types' unless you really need to mix analog and binary objects (for example) in the same packed Modbus register. Packing mixed object types is very inefficient since the gateway must scan the entire list of object maps for every instance of packed register. When packing the same object types, placing them in consecutive maps results in very efficient execution.



Our currently defined object list is as illustrated above on the Object List page. Go to

the Object Map page, select the insert point (object number) and click Insert.

Babel Buster BB3-3101 Gateway Configuration Tool v3.01

Connected: ☒ Target: ☒ 29999: Babel Buster BB3-3101 In Sync: ☒

Local Port | Who-Is | Read/Write | Device | Object Map | Object Text | Object List | Modbus | Data List | Firmware

Object Type/Instance: Analog Input 1 Units: no_units

Object Name: Name of Analog Input 1

Description: Description of Analog Input 1

☒ Read Periodic ☐ Write Periodic ☐ Set Default on Power-Up ☐ Enable Max Quiet Time ☒

☐ Write on Delta ☐ Set Default on Comm Fail

Poll Rate (Sec): 2 Slope/Scale Factor: 0 Timeout (Sec): 0

Initial COV Increment: 0 Intercept/Offset: 0 Max. Quiet Time (Sec): 0

Initial COV Period: 0 Delta for Send: 0 Default Value: 0

Initial Relinquish Default: Read Fails before Fault: 0

☒ Map Modbus Object ☐ Map BACnet Object

Register Number 1..N: 1 Unit/Slave Addr: 1 ☒ Member of Packed Register

Register Type: Holding Register (fc 16) Mask (Hex): 0000F000 ☐ Pack Mixed Object Types

Register Format: Unsigned 32-bit (high reg first) Fill (Hex): 00000000

Insert Add Delete Read Device Write Device OBJECT_ANALOG_INPUT #1 -- write ok.

After clicking Insert, the new map appears before the indicated object number, and existing object maps are moved down the list.

Babel Buster BB3-3101 Gateway Configuration Tool v3.01

Connected: ☒ Target: ☒ 29999: Babel Buster BB3-3101 In Sync: ☒

Local Port | Who-Is | Read/Write | Device | Object Map | Object Text | Object List | Modbus | Data List | Firmware

Read All Write All

Default BACnet Device 1
Default Modbus Slave 1

Obj...	Name	R/W	Device	Object	Type	Period	Qualifier
AI 1		---	None	None	None	0 sec	
AI 2	Name of Analog Input 1	R	Modbus [1]	HoldRegs [1]	U32 Hi	2 sec	XG
AI 3	Analog Input 2	R	Modbus [1]	HoldRegs [1]	U32 Hi	2 sec	G
AI 4	Analog Input 3	R	Modbus [1]	HoldRegs [5]	U16	2 sec	XG
AI 5	Name of Analog Input 4	R	MS/TP [40057]	AI [1]	Present_Value	5 sec	
AI 6		---	None	None	None	0 sec	
AI 7		---	None	None	None	0 sec	
AI 8		---	None	None	None	0 sec	
AI 9		---	None	None	None	0 sec	
AI 10		---	None	None	None	0 sec	
AI 11		---	None	None	None	0 sec	
AI 12		---	None	None	None	0 sec	
AI 13		---	None	None	None	0 sec	
AI 14		---	None	None	None	0 sec	
AI 15		---	None	None	None	0 sec	
AO 1	Name of Analog Output 1	W+	MS/TP [40057]	AO [1] P=7	Present_Value	0 sec	
AO 2		W+	Modbus [1]	U16	U16	2 sec	XG

Clicking Delete will remote the current object and move maps up the list.

Babel Buster BB3-3101 Gateway Configuration Tool v3.01

Connected: ☒ Target: ☒ 29999: Babel Buster BB3-3101 In Sync: ☒

Local Port | Who-Is | Read/Write | Device | Object Map | Object Text | Object List | Modbus | Data List | Firmware

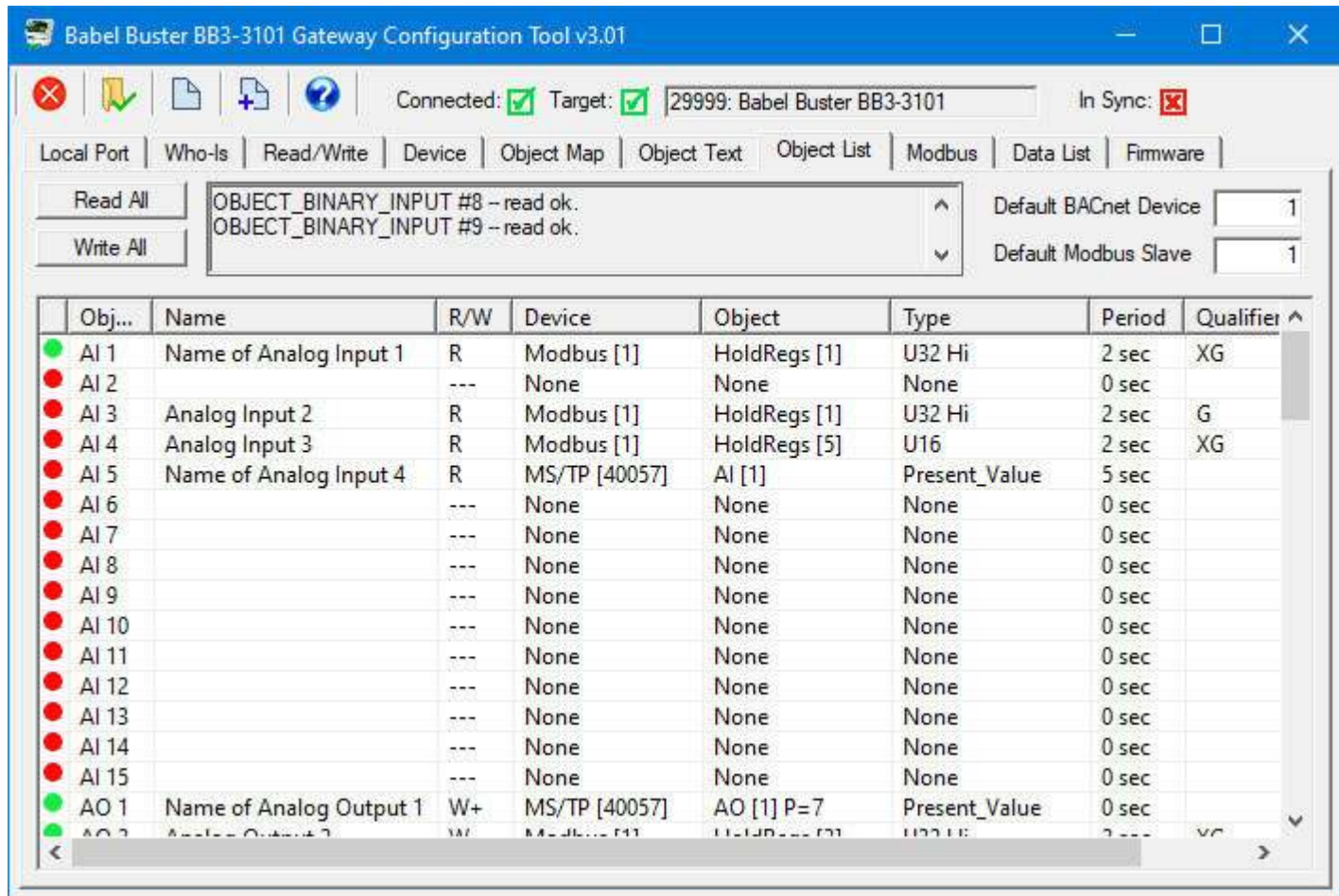
Read All Write All

OBJECT_MULTI_STATE_VALUE #4 -- read ok.
OBJECT_MULTI_STATE_VALUE #5 -- read ok.

Default BACnet Device 1
Default Modbus Slave 1

Obj...	Name	R/W	Device	Object	Type	Period	Qualifier
AI 1	Name of Analog Input 1	R	Modbus [1]	HoldRegs [1]	U32 Hi	2 sec	XG
AI 2	Analog Input 2	R	Modbus [1]	HoldRegs [1]	U32 Hi	2 sec	G
AI 3	Analog Input 3	R	Modbus [1]	HoldRegs [5]	U16	2 sec	XG
AI 4	Name of Analog Input 4	R	MS/TP [40057]	AI [1]	Present_Value	5 sec	
AI 5		---	None	None	None	0 sec	
AI 6		---	None	None	None	0 sec	
AI 7		---	None	None	None	0 sec	
AI 8		---	None	None	None	0 sec	
AI 9		---	None	None	None	0 sec	
AI 10		---	None	None	None	0 sec	
AI 11		---	None	None	None	0 sec	
AI 12		---	None	None	None	0 sec	
AI 13		---	None	None	None	0 sec	
AI 14		---	None	None	None	0 sec	
AI 15		---	None	None	None	0 sec	
AO 1	Name of Analog Output 1	W+	MS/TP [40057]	AO [1] P=7	Present_Value	0 sec	
AO 2		W+	Modbus [1]	U16	U16	2 sec	XG

Clicking Add will insert the new object after the current object, and move the remaining object mappings down by one.



8.5 Read/Write Object Map in Device

When changes are made, the in-sync icon changes to a red X on the right hand side of this page. When properties are either read or written, the device is now in sync, and the red X changes to a green check mark. This icon will change on an object by object basis indicating those objects that are in sync.

When all aspects of device configuration are in sync, between device and tool, the 'In Sync' icon in the top right corner of the screen will change to a green check box. This is the master In-Sync icon, and will be green only when all other in-sync icons are also green. This icon is effectively a summary icon.

Babel Buster BB3-3101 Gateway Configuration Tool v3.01

Connected: ☒ Target: ☒ 29999: Babel Buster BB3-3101 In Sync: ☒

Local Port | Who-Is | Read/Write | Device | Object Map | Object Text | Object List | Modbus | Data List | Firmware

Object Type/Instance: Analog Input 1 Units: no_units

Object Name: Name of Analog Input 1

Description: Description of Analog Input 1

☒ Read Periodic ☐ Write Periodic ☐ Set Default on Power-Up ☐ Enable Max Quiet Time ☒

☐ Write on Delta ☐ Set Default on Comm Fail

Poll Rate (Sec): 2 Slope/Scale Factor: 0 Timeout (Sec): 0

Initial COV Increment: 0 Intercept/Offset: 0 Max. Quiet Time (Sec): 0

Initial COV Period: 0 Delta for Send: 0 Default Value: 0

Initial Relinquish Default: Read Fails before Fault: 0

☒ Map Modbus Object ☐ Map BACnet Object

Register Number 1..N: 1 Unit/Slave Addr: 1 ☒ Member of Packed Register

Register Type: Holding Register (fc 16) Mask (Hex): 0000F000 ☐ Pack Mixed Object Types

Register Format: Unsigned 32-bit (high reg first) Fill (Hex): 00000000

Insert Add Delete **Read Device** Write Device OBJECT_ANALOG_INPUT #1 -- read ok.

8.6 Export and Import Configuration as XML File

The device properties and object maps are stored in the same XML file. Refer to section 7.6 of this user guide to review file import and export features. They work the same whether accessed from the Device page or Object Map page. The file type exported or imported from either of these pages is an XML file. Note that the file type is not the same on the Object List page, where it expects a CSV file instead.

Babel Buster BB3-3101 Gateway Configuration Tool v3.01

Connected: ☒ Target: ☒ 29999: Babel Buster BB3-3101 In Sync: ☒

Local Port | Who-Is | Read/Write | Device | Object Map | Object Text | Object List | Modbus | Data List | Firmware

Object ID (device instance): 29999 Reassign

Device Name: Default BB3-3101

Device Description: Modbus RTU to BACnet MS/TP Gateway

Device Location: Unassigned

☒ Auto-R

Max Master: 127

Object Allocation Present Pending

Analog Input Count: 10 10 10 10 10 10 10 10 10 10

Multistate Input Count: 5 5

Import from File

☐ Device Properties

☒ Object Maps

Cancel OK

Note: If you have an XML configuration file saved from a BB2-3010, you can directly

import that into your BB3-3101 by simply opening that file on this page.

8.7 Using Default on Comm Fail

One way of detecting a communication failure is to look at the reliability code in the gateway's local object. A reliability code of zero means everything is OK. A non-zero reliability code means there is a problem. For example, if the gateway's object is polling a remote Modbus device, then a reliability code of 64 means timeout, and that in turn most often means no connection. If the local object is mapped for polling an remote BACnet device, then a reliability code of 82 means timeout. Several other reliability codes are possible and these are listed in section 12.2.

The reliability code can be read as a property by another BACnet client. The reliability codes can be read as Modbus register values if the gateway is functioning as a Modbus slave.

There is another alternative for flagging data as invalid due to a communication error. You can configure a default value which the local object should assume if it fails to retrieve a value from the remote device. This only applies to local BACnet objects that are configured to read data from a remote device. Trying to write a default value upon communication failure doesn't make any sense since inability to access the remote device was the problem in the first place.

There are three values you need to configure to implement the default on fail. The configuration for an object mapped to read from a remote BACnet device is illustrated here.

The configuration for an object mapped to read from a remote Modbus device is illustrated here.

Babel Buster BB3-3101 Gateway Configuration Tool v3.02

Connected: ☒ Target: ☒ 29998: Babel Buster BB3-3101 In Sync: ☒

Local Port | Who-Is | Read/Write | Device | Object Map | Object Text | Object List | Modbus | Data List | Firmware

Object Type/Instance: Analog Input 4 Units: square_meters

Object Name: AI-4 Description:

☒ Read Periodic ☐ Write Periodic ☐ Set Default on Power-Up ☐ Enable Max Quiet Time ☒

☐ Write on Demand ☒ Set Default on Comm Fail

Poll Rate (Sec): 2 Slope/Scale Factor: 0 Timeout (Sec): 1

Initial COV Increment: 0 Intercept/Offset: 0 Max. Quiet Time (Sec): 0

Initial COV Period: 0 Delta for Send: 0 Default Value: -99

Initial Relinquish Default: Read Fails before Fault: 3

☒ Map Modbus Object ☐ Map BACnet Object

Register Number 1..N: 1 Unit/Slave Addr: 1 ☐ Member of Packed Register

Register Type: Holding Register (fc 16) Mask (Hex): 0000 ☐ Pack Mixed Object Types

Register Format: Unsigned 16-bit Fill (Hex): 0000

Insert Add Delete Read Device Write Device OBJECT_ANALOG_INPUT #3 - write ok.

The "Read Fails" count of greater than one prevents spurious communication problems from triggering the default value. The problem must persist before the default value will be applied.

The Data List page with these two objects in fail mode is illustrated here.

Babel Buster BB3-3101 Gateway Configuration Tool v3.02

Connected: ☒ Target: ☒ 29998: Babel Buster BB3-3101 In Sync: ☒

Local Port | Who-Is | Read/Write | Device | Object Map | Object Text | Object List | Modbus | Data List | Firmware

Read All

OBJECT_MULTI_STATE_VALUE #1 -- read ok.
OBJECT_MULTI_STATE_VALUE #2 -- read ok.

Obj...	R/W	Device	Object	Type	Present Value	Reliabil...	Status Flags
AI 1	R	MS/TP [20841]	AI [1]	Present_Value	0.000000	0	F,F,F,F { }
AI 2	R	MS/TP [20841]	AO [2] X=8 P=0	Prop 87, Real	0.000000	0	F,F,F,F { }
AI 3	R	MS/TP [20841]	AI [300]	Present_Value	-99.000000	83	F,T,F,F { Fault }
AI 4	R	Modbus [1]	HoldRegs [1]	U16	-99.000000	64	F,T,F,F { Fault }
AO 1	R	MS/TP [20841]	AO [1] P=0	Present_Value	0.000000	0	F,F,F,F { }
AO 2	W+	MS/TP [20841]	AO [2] P=8	Present_Value	55.000000	0	F,F,F,F { }
AV 1	R	MS/TP [20841]	AV [1]	Present_Value	32.000000	0	F,F,F,F { }
AV 2	W+	MS/TP [20841]	AV [2]	Present_Value	0.000000	0	F,F,F,F { }
AV 3	W+	MS/TP [20841]	AV [3]	Present_Value	0.000000	0	F,F,F,F { }
AV 4	W+	MS/TP [20841]	AV [4]	Present_Value	19.500000	0	F,F,F,F { }
BI 1	R	MS/TP [20841]	BI [1]	Present_Value	0	0	F,F,F,F { }
BO 1	R	MS/TP [20841]	BO [1] P=0	Present_Value	0	0	F,F,F,F { }
BO 2	W+	MS/TP [20841]	BO [2] P=8	Present_Value	0	0	F,F,F,F { }
BV 1	R	MS/TP [20841]	BV [1]	Present_Value	0	0	F,F,F,F { }
BV 2	W+	MS/TP [20841]	BV [2]	Present_Value	0	0	F,F,F,F { }
MI 1	R	MS/TP [20841]	MI [1]	Present_Value	1	0	F,F,F,F { }

8.8 Data Conversion Examples

The scale and offset, or slope and intercept, settings allow data conversion "on the fly". As noted earlier, scaling applies the formula $y=mx+b$. When reading from the slave, the raw data as read is multiplied by the scale factor, then the offset is added to produce the resulting Present Value. When writing to the slave, the offset is first subtracted from Present Value, and that result is divided by the scale factor to produce the raw data actually written to the slave device. NOTE: If no scale factor is given (zero is entered), no scaling will be done, as if slope=1 and intercept=0.

One of the more commonly used conversions is degrees Celsius (C) to Fahrenheit (F). The formula is

$$F = 1.8 * C + 32$$

Reading degrees C from remote device and converting on the fly to degrees F in the gateway's local object would be configured as illustrated below. The data read from the remote device is multiplied by 1.8, then the offset of 32 is added.

Babel Buster BB3-3101 Gateway Configuration Tool v3.02 [BACnet-Client2B.xml]

Connected: ☒ Target: ☒ 29998: Babel Buster BB3-3101 In Sync: ☒

Local Port | Who-Is | Read/Write | Device | Object Map | Object Text | Object List | Modbus | Data List | Firmware

Object Type/Instance: Analog Value 1 Units: square_meters

Object Name: AV-1

Description:

☒ Read Periodic ☐ Write Periodic ☐ Set Default on Power-Up ☐ Enable Max Quiet Time ☒

☐ Write on Delta ☐ Set Default on Comm Fail

Poll Rate (Sec): 5 Slope/Scale Factor: 1.8 Timeout (Sec): 0

Initial COV Increment: 0 Intercept/Offset: 32 Max. Quiet Time (Sec): 0

Initial COV Period: 0 Delta for Send: 0 Default Value: 0

Initial Relinquish Default: Read Fails before Fault: 0

☐ Map Modbus Object ☒ Map BACnet Object

Remote Device Instance: 20841 Property: Present Value 85 Bit Num: 0

Object Type: Analog Value Instance: 1 Data Type: Null

Mac address (no who-is): 0 Array Index: -1 Priority: None

Insert Add Delete Read Device Write Device File loaded.

Writing degrees F from gateway's local object to degrees C in the remote device is configured the same way as illustrated below. When writing, the math is reversed, so in this case, the offset of 32 is first subtracted, then the value is divided by 1.8.

Babel Buster BB3-3101 Gateway Configuration Tool v3.02 [BACnet-Client2B.xml]

Connected: ☒ Target: ☒ 29998: Babel Buster BB3-3101 In Sync: ☒

Local Port | Who-Is | Read/Write | Device | Object Map | Object Text | Object List | Modbus | Data List | Firmware

Object Type/Instance: Analog Value 3 Units: parts_per_million

Object Name: Analog Value 3 Description:

☐ Read Periodic ☐ Write Periodic ☐ Set Default on Power-Up ☒ Enable Max Quite Time ☒

☒ Write on Delta ☐ Set Default on Comm Fail

Poll Rate (Sec): 0 Slope/Scale Factor: 1.8 Timeout (Sec): 0

Initial COV Increment: 0 Intercept/Offset: 32 Max. Quiet Time (Sec): 30

Initial COV Period: 0 Delta for Send: 5 Default Value: 0

Initial Relinquish Default: Read Fails before Fault: 0

☐ Map Modbus Object ☒ Map BACnet Object

Remote Device Instance: 20841 Property: Present Value 85 Bit Num: 0

Object Type: Analog Value Instance: 3 Data Type: Real

Mac address (no who-is): 0 Array Index: -1 Priority: None

Insert Add Delete Read Device Write Device OBJECT_ANALOG_VALUE #3 -- write ok.

Writing degrees C from gateway's local object to degrees F in remote device is illustrated below:

Babel Buster BB3-3101 Gateway Configuration Tool v3.02 [BACnet-Client2B.xml]

Connected: ☒ Target: ☒ 29998: Babel Buster BB3-3101 In Sync: ☒

Local Port | Who-Is | Read/Write | Device | Object Map | Object Text | Object List | Modbus | Data List | Firmware

Object Type/Instance: Analog Value 2 Units: parts_per_million

Object Name: Analog Value 2 Description:

☐ Read Periodic ☐ Write Periodic ☐ Set Default on Power-Up ☐ Enable Max Quite Time ☒
☒ Write on Delta ☐ Set Default on Comm Fail

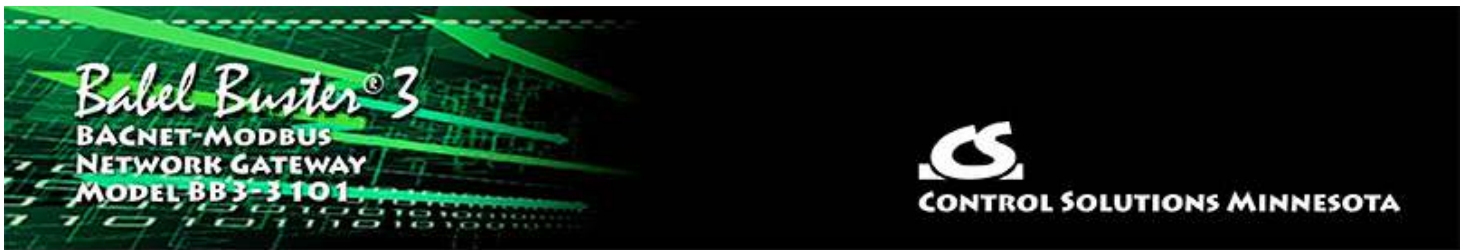
Poll Rate (Sec): 0 Slope/Scale Factor: 0.55555 Timeout (Sec): 0
Initial COV Increment: 0 Intercept/Offset: -17.778 Max. Quiet Time (Sec): 0
Initial COV Period: 0 Delta for Send: 0 Default Value: 0
Initial Relinquish Default: Read Fails before Fault: 0

☐ Map Modbus Object ☒ Map BACnet Object

Remote Device Instance: 20841 Property: Present Value 85 Bit Num: 0
Object Type: Analog Value Instance: 2 Data Type: Real
Mac address (no who-is): 0 Array Index: -1 Priority: None

Insert Add Delete Read Device Write Device File loaded:

If the numbers in this last example don't seem to make sense, revisit the previous example, and get out your calculator and try it out.



9 Tool 'Object Text' Page

Binary objects have an active and inactive state string associated with them. Multistate objects have state strings with one string per state. This page is where you edit those strings. Using state strings is not required, but if you do use them, this page is where you may edit them.

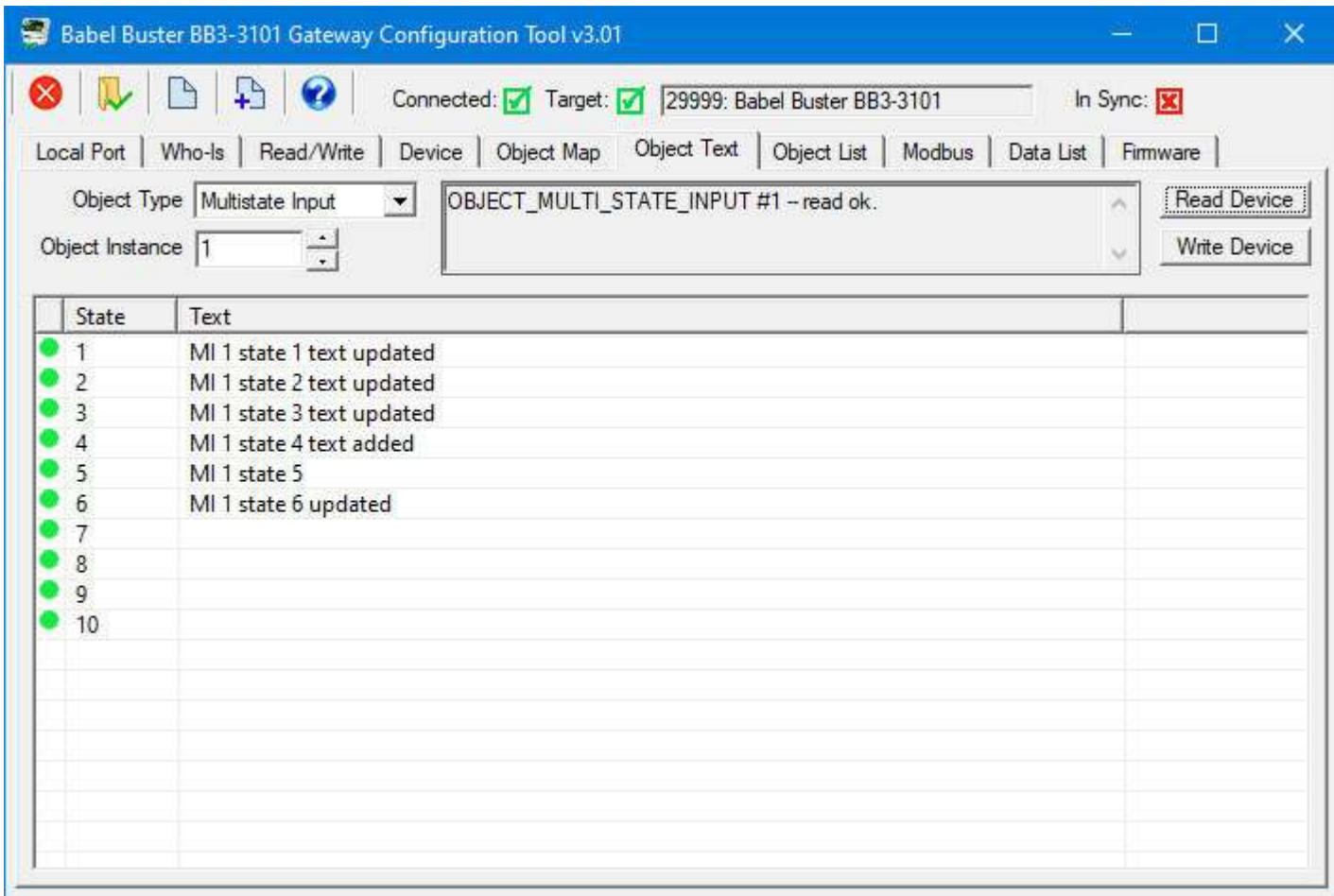
9.1 Read/Write Object Text

Select object type and instance, then click Read Device to retrieve the strings from the gateway. For Binary objects, there will be only two states labeled Inactive and Active.

The screenshot shows the 'Babel Buster BB3-3101 Gateway Configuration Tool v3.01' window. The 'Object Text' tab is selected. The 'Object Type' is set to 'Binary Input' and 'Object Instance' is '1'. The 'Read Device' button is highlighted. Below, a table displays the state strings for the selected object.

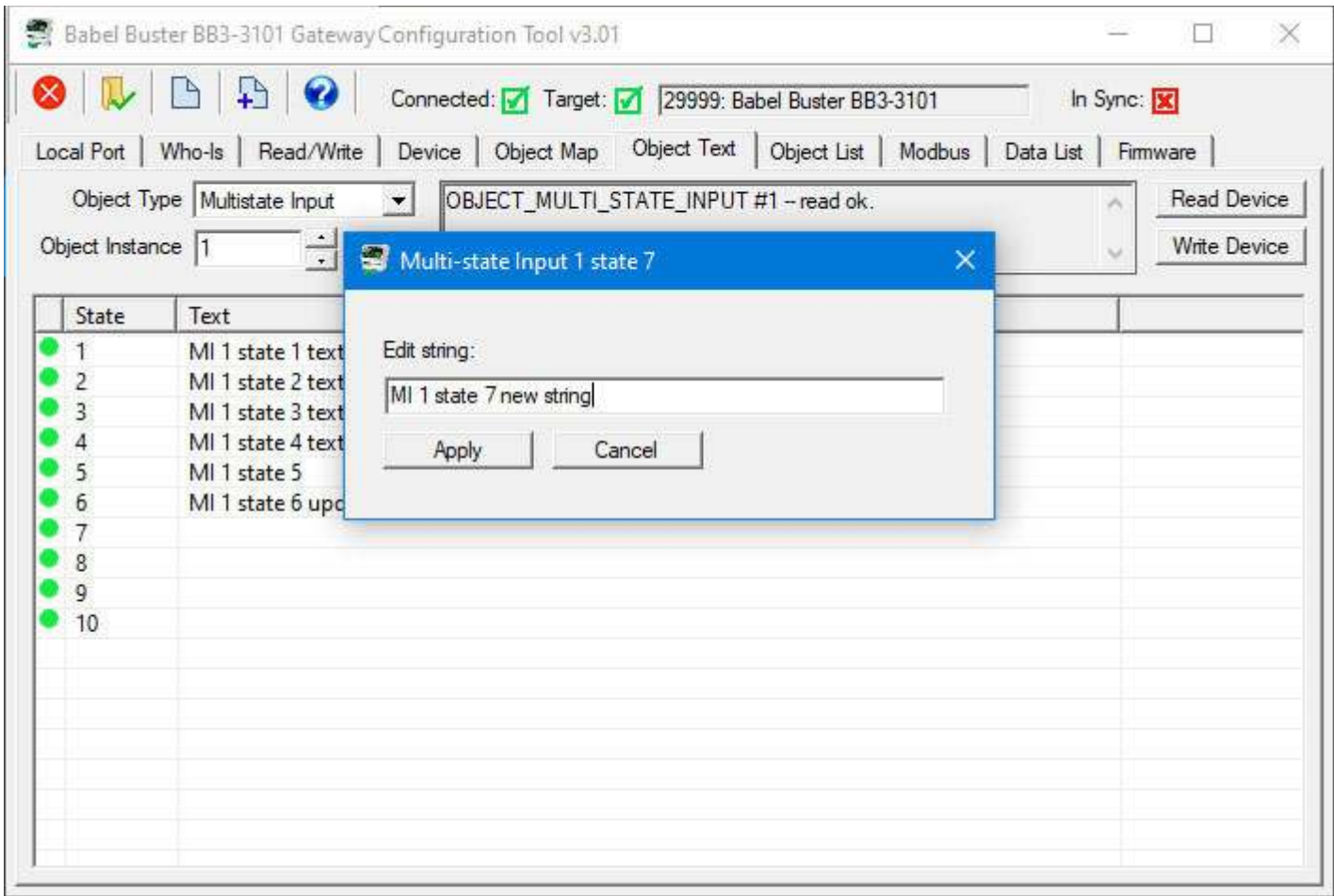
State	Text
Inactive	BI 1 Inctive Text Updated
Active	BI 1 Active Text Updated

When retrieving state strings for a Multistate object, there will be one string per state up to the maximum state set on the Device page at the bottom of the object count list.

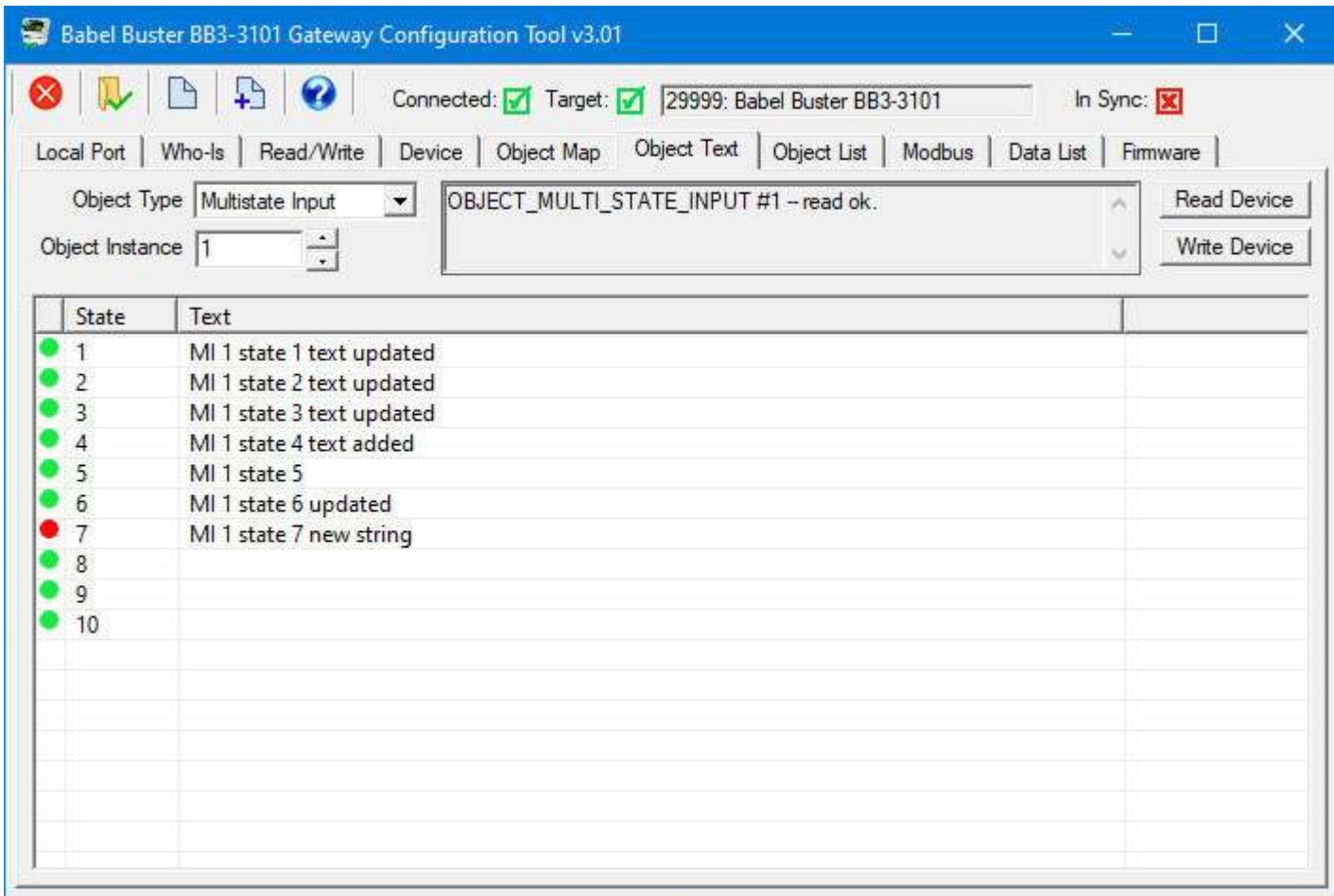


9.2 Edit Object Text

Double click on a line of text to open the edit dialog. You may edit or enter a new text string. Click Apply to apply the new or modified string.

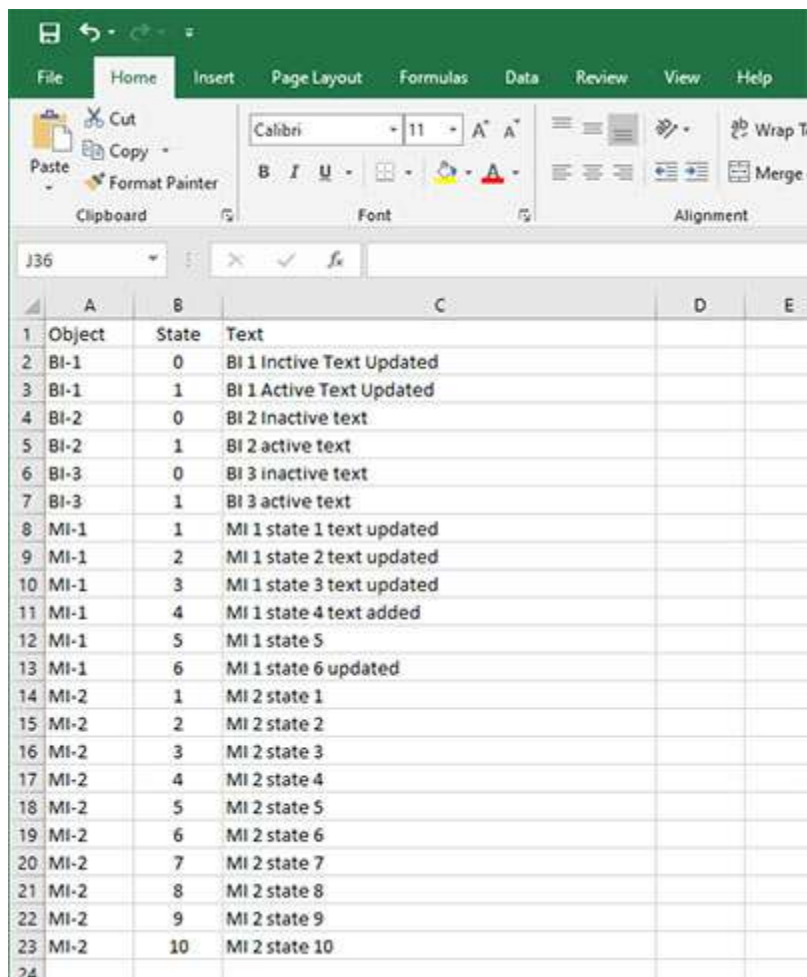


After making changes to text strings, the icon will change from green to red reminding you that you still need to send the text to the gateway by clicking Write Device.



9.3 Export and Import Text as CSV file

Text strings may be imported from a simple CSV file having only 3 columns: Object, state, and text. For Binary objects, the state should be 0=inactive, 1=active. For Multistate objects, states should begin with 1 and count up to the maximum value configured on the Device page.



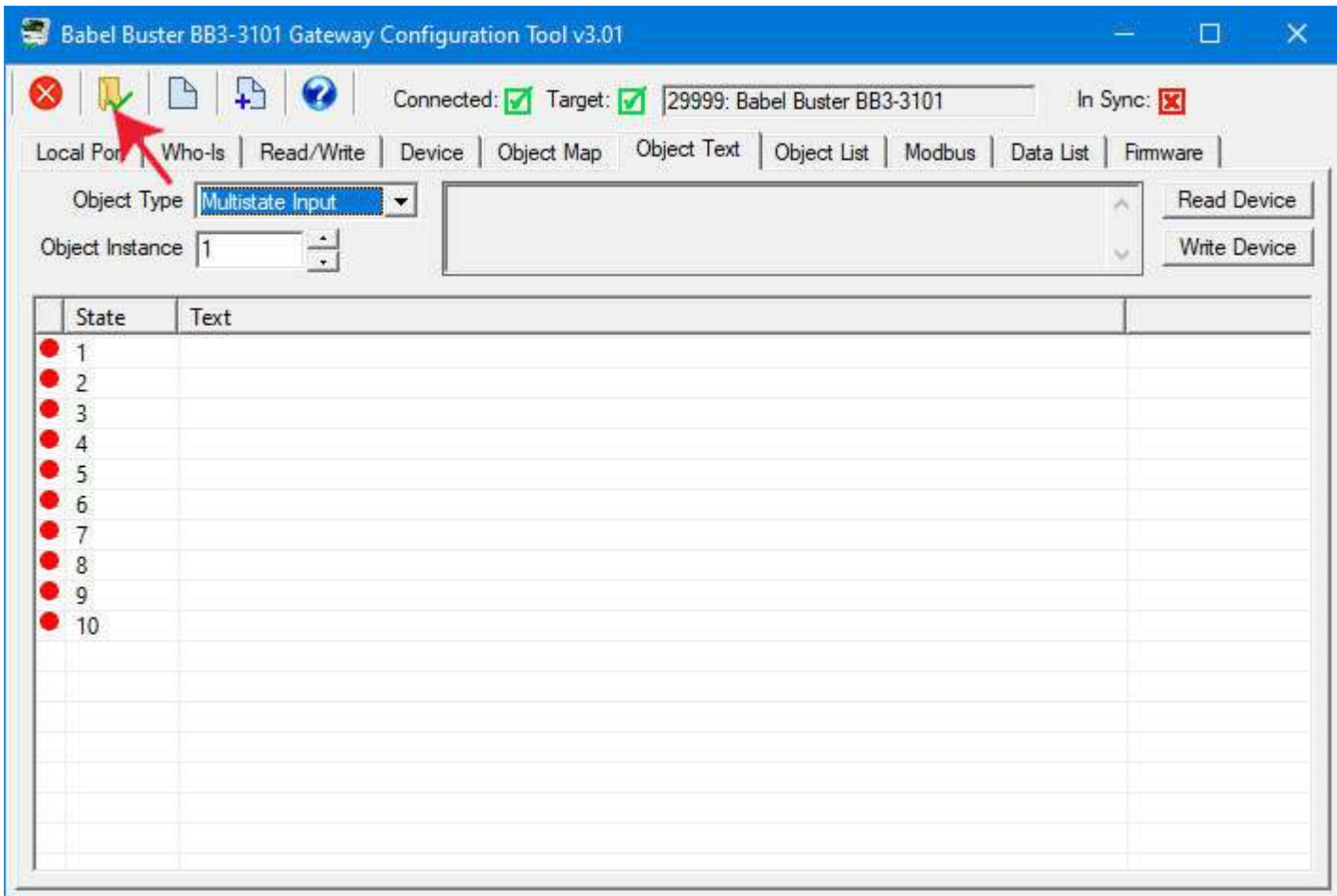
	A	B	C	D	E
1	Object	State	Text		
2	BI-1	0	BI 1 Inctive Text Updated		
3	BI-1	1	BI 1 Active Text Updated		
4	BI-2	0	BI 2 Inactive text		
5	BI-2	1	BI 2 active text		
6	BI-3	0	BI 3 inactive text		
7	BI-3	1	BI 3 active text		
8	MI-1	1	MI 1 state 1 text updated		
9	MI-1	2	MI 1 state 2 text updated		
10	MI-1	3	MI 1 state 3 text updated		
11	MI-1	4	MI 1 state 4 text added		
12	MI-1	5	MI 1 state 5		
13	MI-1	6	MI 1 state 6 updated		
14	MI-2	1	MI 2 state 1		
15	MI-2	2	MI 2 state 2		
16	MI-2	3	MI 2 state 3		
17	MI-2	4	MI 2 state 4		
18	MI-2	5	MI 2 state 5		
19	MI-2	6	MI 2 state 6		
20	MI-2	7	MI 2 state 7		
21	MI-2	8	MI 2 state 8		
22	MI-2	9	MI 2 state 9		
23	MI-2	10	MI 2 state 10		
24					

The text view of the same CSV file would appear as follows.

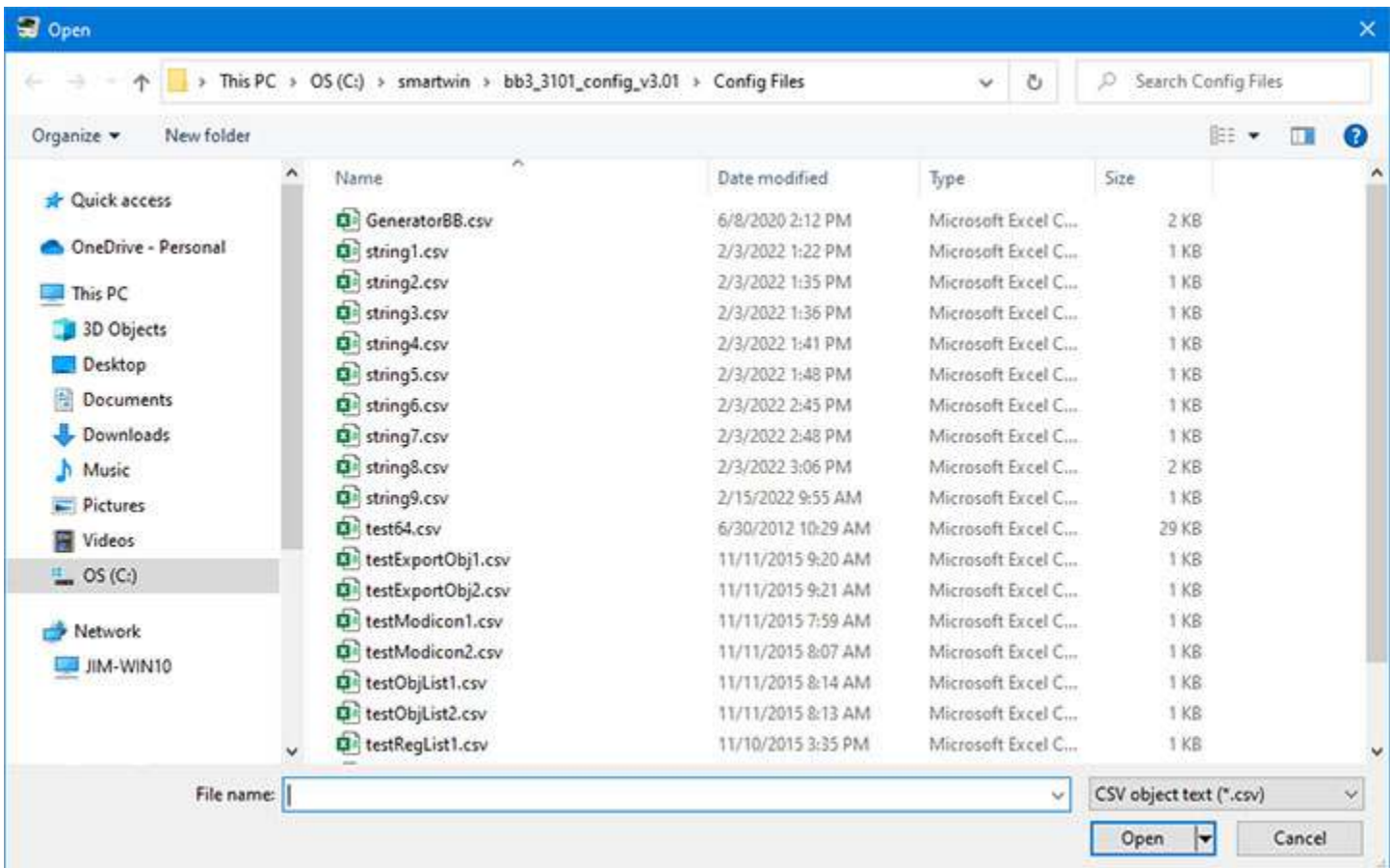


```
string9.csv - Notepad
File Edit Format View Help
Object,State,Text
BI-1,0,BI 1 Inctive Text Updated
BI-1,1,BI 1 Active Text Updated
BI-2,0,BI 2 Inactive text
BI-2,1,BI 2 active text
BI-3,0,BI 3 inactive text
BI-3,1,BI 3 active text
MI-1,1,MI 1 state 1 text updated
MI-1,2,MI 1 state 2 text updated
MI-1,3,MI 1 state 3 text updated
MI-1,4,MI 1 state 4 text added
MI-1,5,MI 1 state 5
MI-1,6,MI 1 state 6 updated
MI-2,1,MI 2 state 1
MI-2,2,MI 2 state 2
MI-2,3,MI 2 state 3
MI-2,4,MI 2 state 4
MI-2,5,MI 2 state 5
MI-2,6,MI 2 state 6
MI-2,7,MI 2 state 7
MI-2,8,MI 2 state 8
MI-2,9,MI 2 state 9
MI-2,10,MI 2 state 10
```

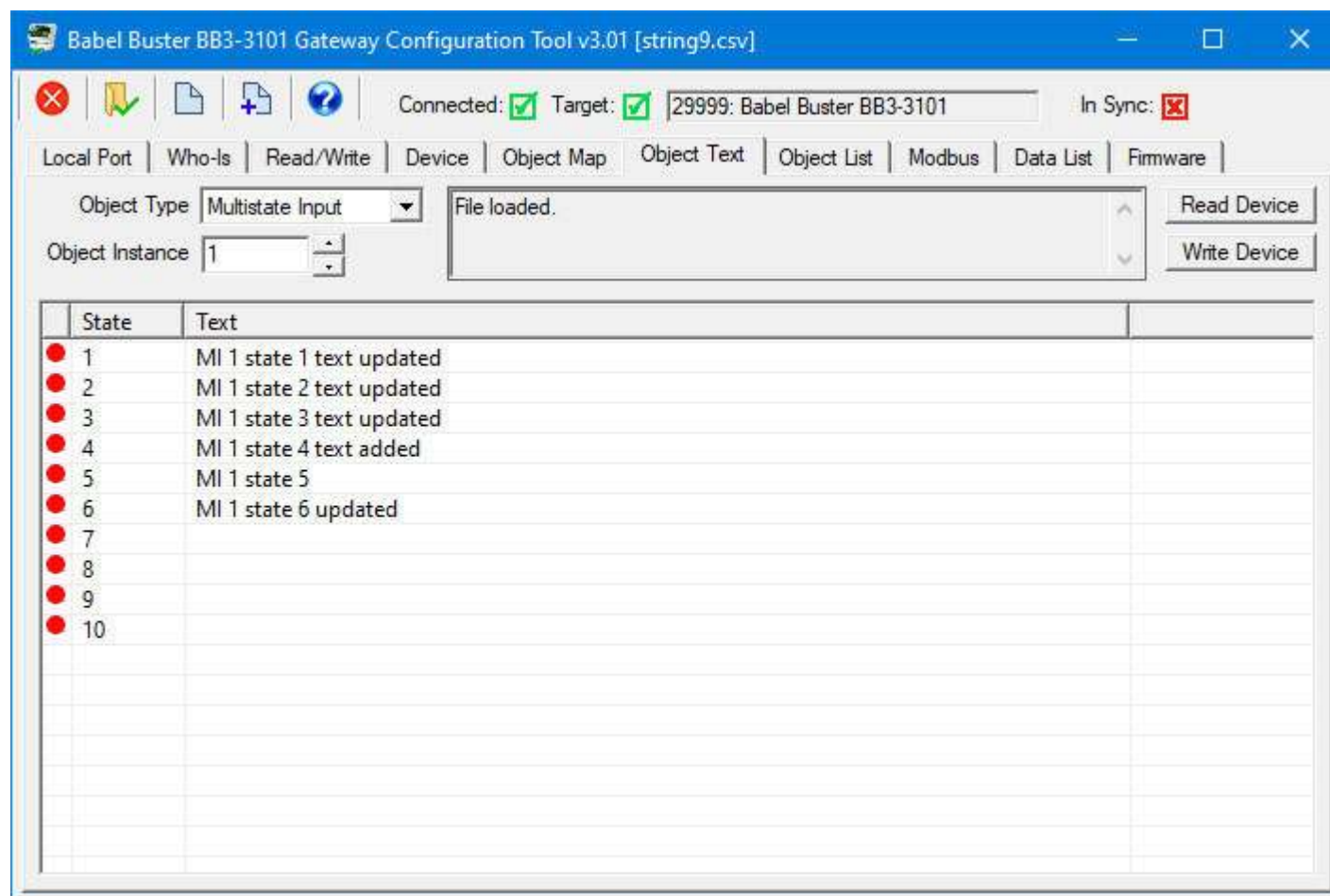
To import this file, click on the File Open icon.



The Windows file dialog will appear, allowing you to select the file to be loaded.



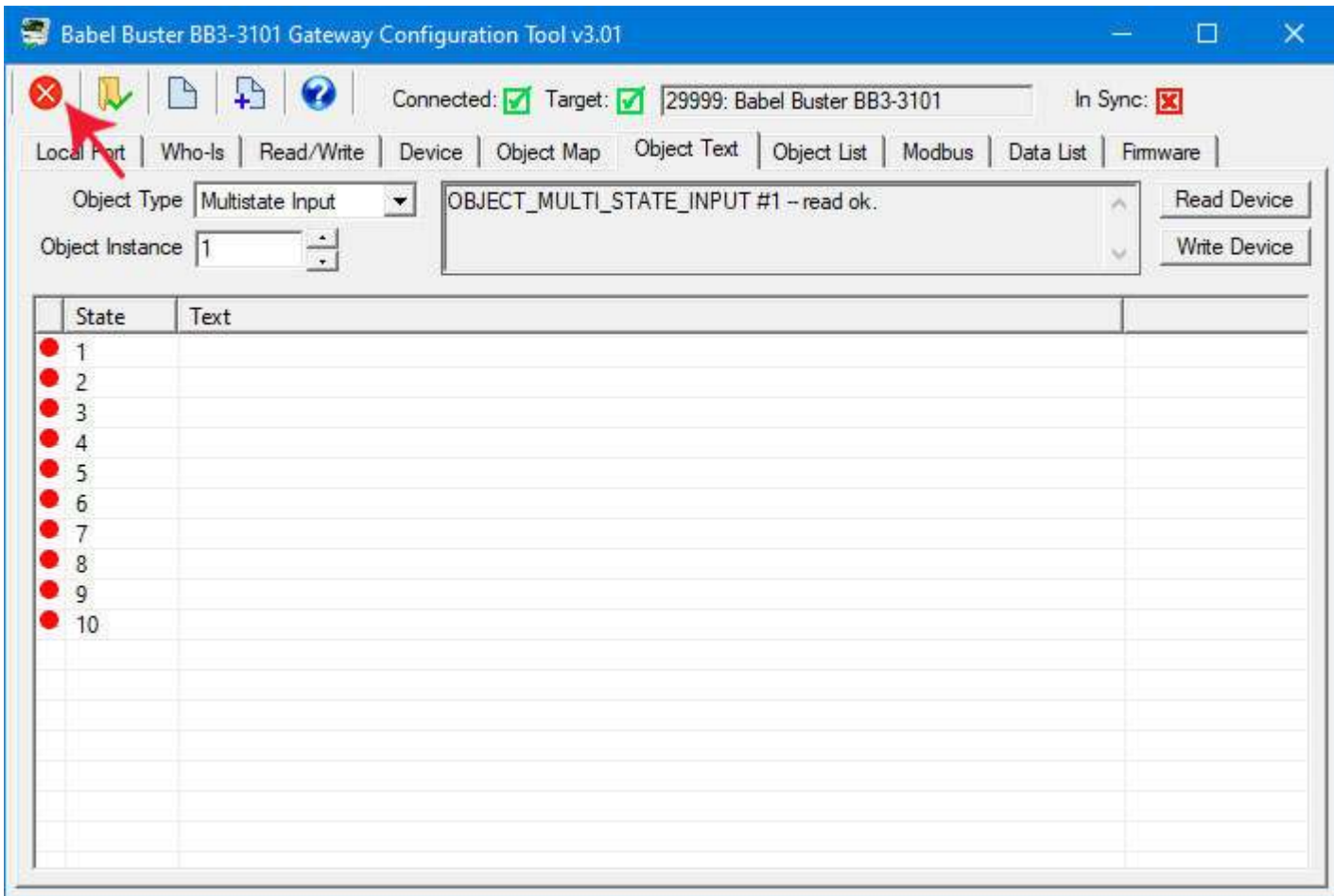
Upon loading the CSV file, the text strings will appear on the Object Text apge. You may click Write Device to write text strings one object at a time, or go to the Object List page and click Write All.

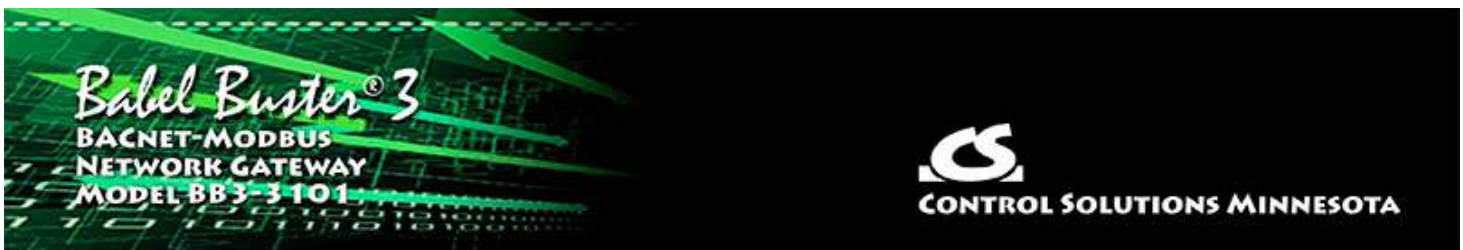


9.4 Clearing Object Text

Clicking the Clear icon (first icon on the left in the top toolbar) on the Object Text page only clears the object text. It does not affect object properties, object counts or device properties. If you want to clear any of those, go to the Object Map or Device pages and click the Clear icon there to clear. The Clear icon on the Object Text page clears text for all objects having text properties.

The buttons in the configuration tool only clear information stored by the tool software in your PC's memory. To completely clear configuration in the physical gateway device, you need to follow the "Reconfigure Object Counts" procedure discussed in Section 7 of this user guide. To clear just object text, use the Write Device button on the Object Text page to clear just one object, or use the Write All button on the Object List page.





10 Tool 'Object List' Page

The Object List page is where you can import and export object mappings in CSV file format. This is more convenient for making mass changes to the configuration. Refer to Section 3 of this user guide for more guidelines on practical use of the CSV file.

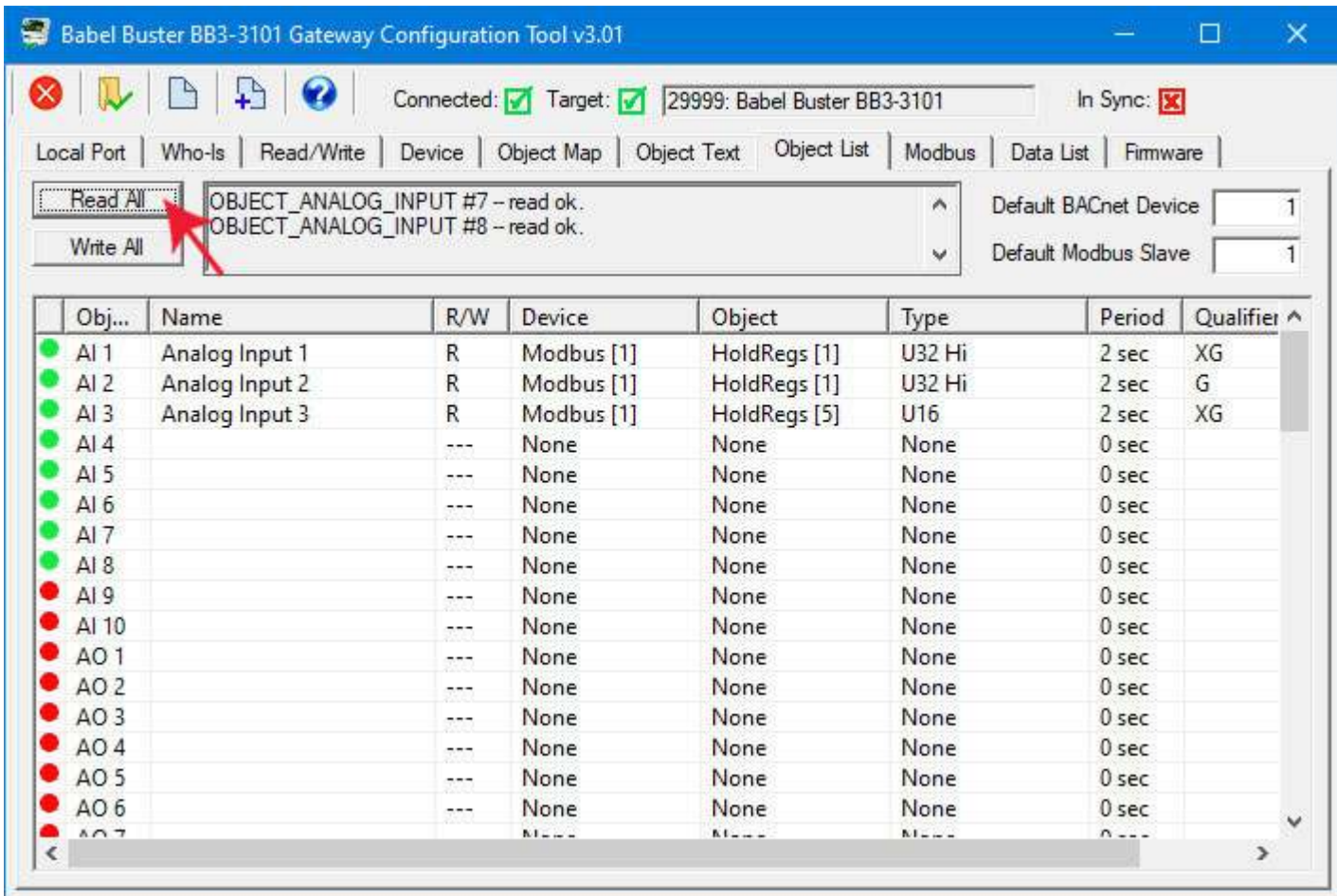
10.1 Read/Write All Object Maps in Device

You first need to confirm that you have some objects allocated on the Device page. Click Get Counts to retrieve the currently configured count of objects in the device. Refer to Section 7 of this user guide for more about object counts.

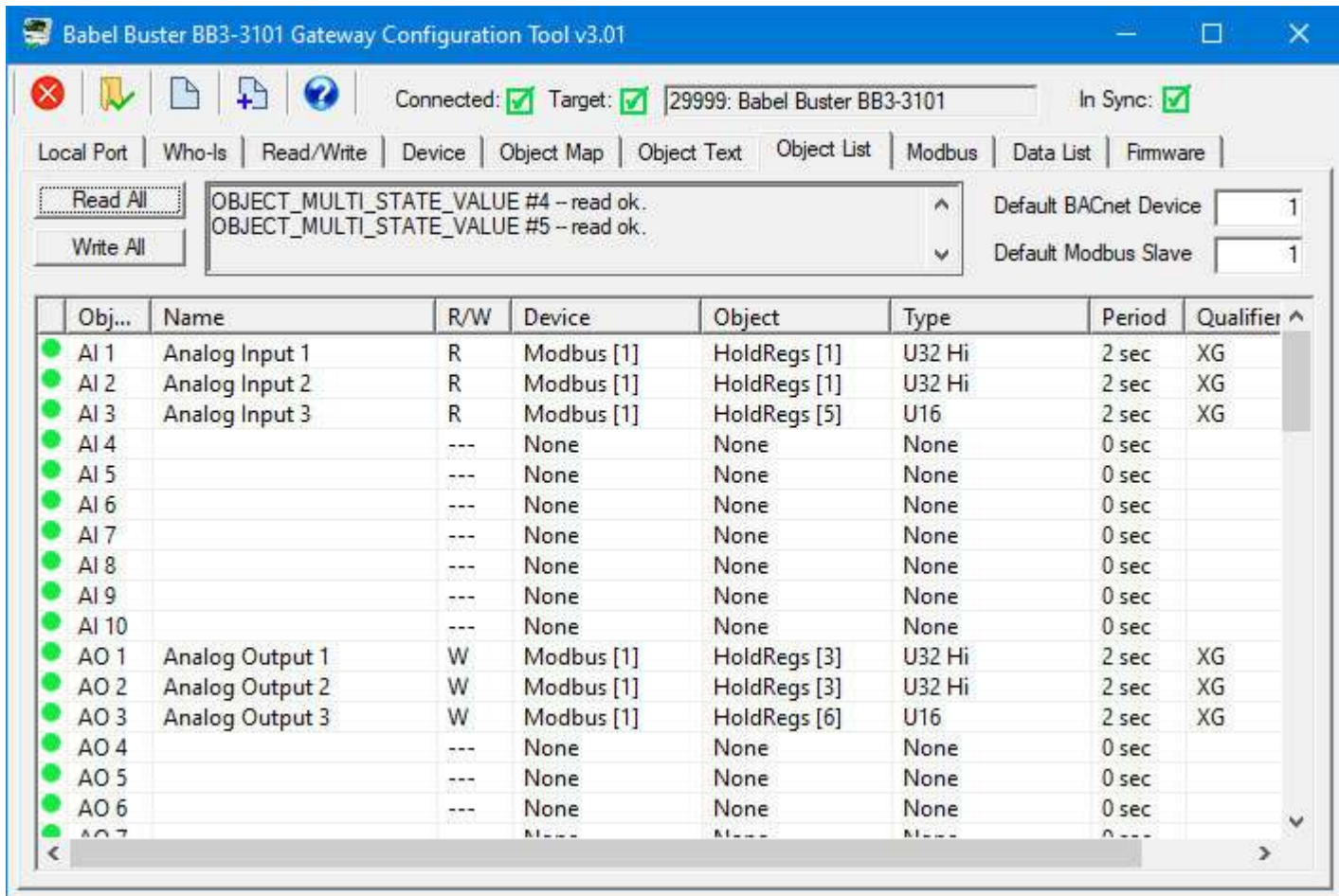
The screenshot shows the 'Babel Buster BB3-3101 Gateway Configuration Tool v3.01' window. The 'Object List' tab is selected. The interface includes a top status bar with 'Connected: [checked]', 'Target: [checked]', '29900: Babel Buster BB3-3101', and 'In Sync: [red X]'. Below this is a tabbed menu with 'Local Port', 'Who-Is', 'Read/Write', 'Device', 'Object Map', 'Object Text', 'Object List', 'Modbus', 'Data List', and 'Firmware'. The main area is divided into two sections. The left section contains fields for 'Object ID (device instance)' (29900), 'Device Name' (Default BB3-3101), 'Device Description' (Modbus RTU to BACnet MS/TP Gateway), 'Device Location' (Unassigned), and checkboxes for 'Auto-Reset Errors' (checked) and 'Reinit Device'. The right section is a table titled 'Object Allocation' with columns 'Present' and 'Pending'. The table lists various object types and their counts. A red arrow points to the 'Get Counts' button at the bottom right of the window. A status bar at the bottom indicates 'Device count read ok.'

Object Allocation	Present	Pending
Analog Input Count	15	15
Analog Output Count	10	10
Analog Value Count	10	10
Binary Input Count	15	15
Binary Output Count	10	10
Binary Value Count	10	10
Multistate Input Count	5	5
Multistate Output Count	5	5
Multistate Value Count	5	5
Multistate Max State	10	10

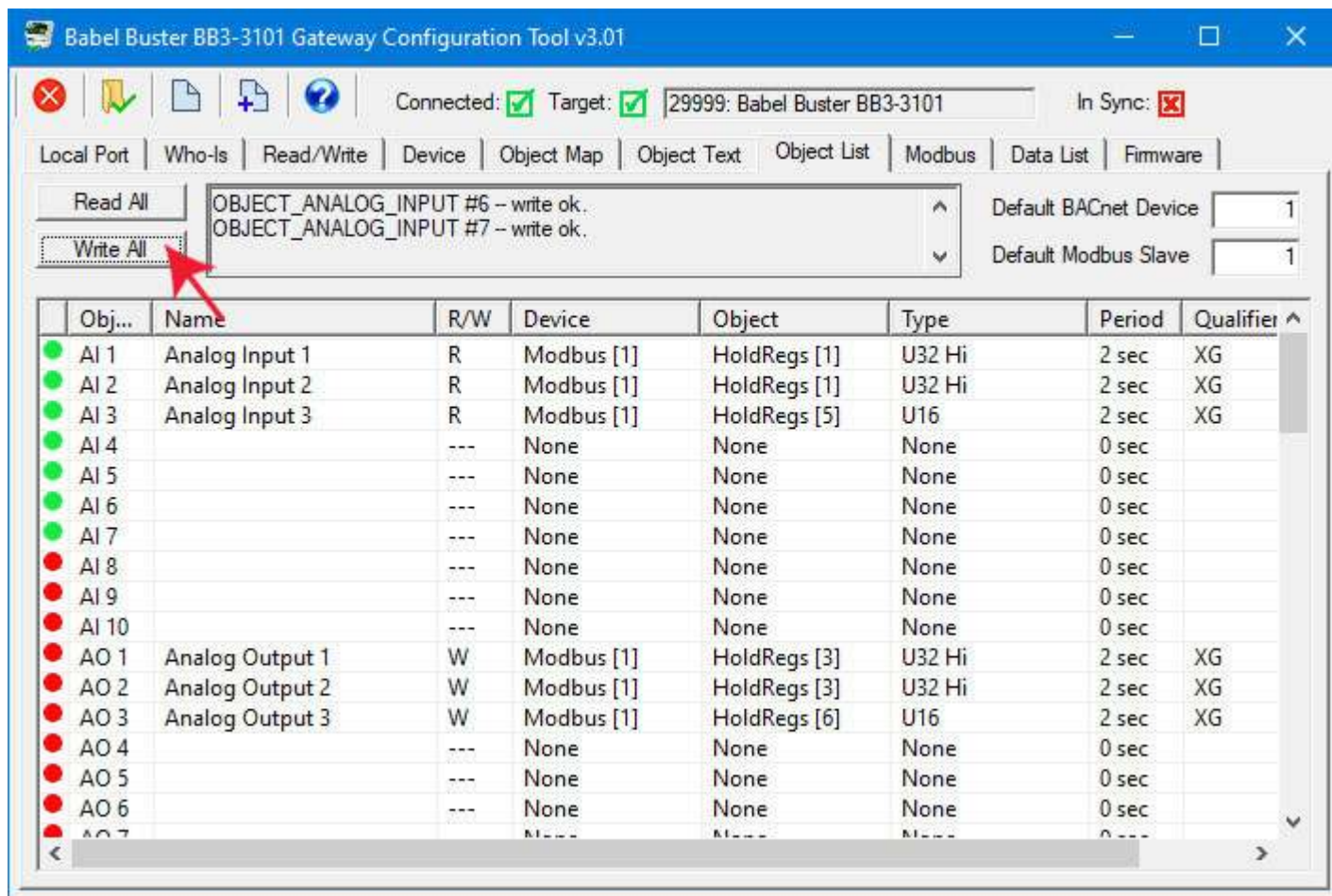
Click Read All to begin the process of reading all object maps from the gateway device. As objects are read, they will be populated in the object list, and the in-sync icon will change from red to green to indicate that this is the information actually stored in the device. Until the icon turns green, any configuration information displayed is a copy that only exists in the configuration tool software on your PC.



As the process continues, the "read ok" status lines will continue to scroll up in the status window. When this activity stops, it is generally safe to assume everything has been read. But the icons in the first column will confirm that. If green, then that line contains information actually read from the device.



When you make changes to object maps, the green in-sync icon will change to red. To write the updated configuration to the device, click the Write All button. As the process continues, the "write ok" will be written to the status window and the in-sync icons will turn from red to green.



10.2 Select an Object Map for Editing

You cannot make changes to object maps from the Object List page. To make changes, double click the line you wish to change. You will be automatically taken to the Object Map page where that object will be displayed. You may make changes there.

Babel Buster BB3-3101 Gateway Configuration Tool v3.01

Connected: ☒ Target: ☒ 29999: Babel Buster BB3-3101 In Sync: ☒

Local Port | Who-Is | Read/Write | Device | Object Map | Object Text | Object List | Modbus | Data List | Firmware

Read All Write All

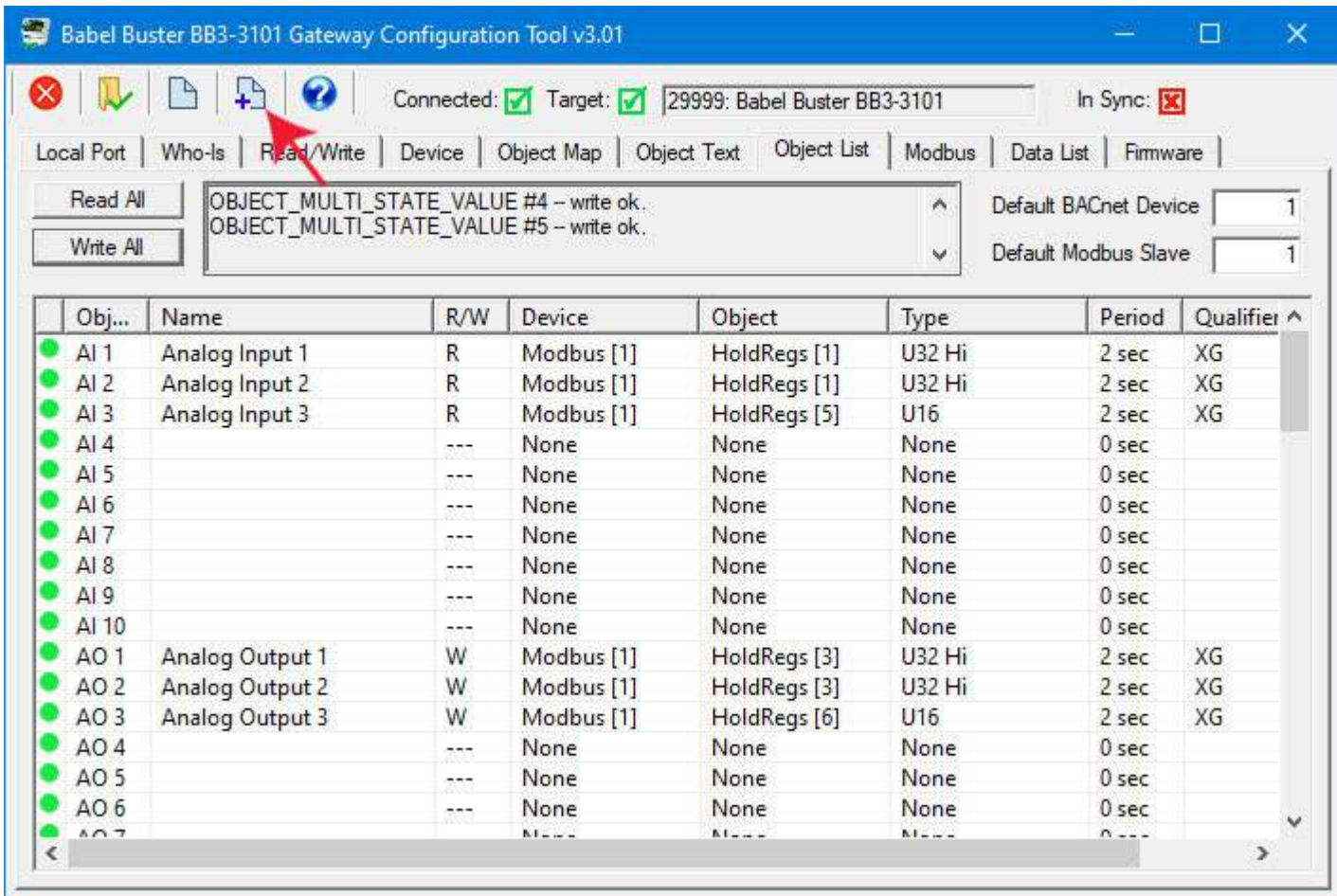
OBJECT_MULTI_STATE_VALUE #4 -- read ok.
OBJECT_MULTI_STATE_VALUE #5 -- read ok.

Default BACnet Device: 1
Default Modbus Slave: 1

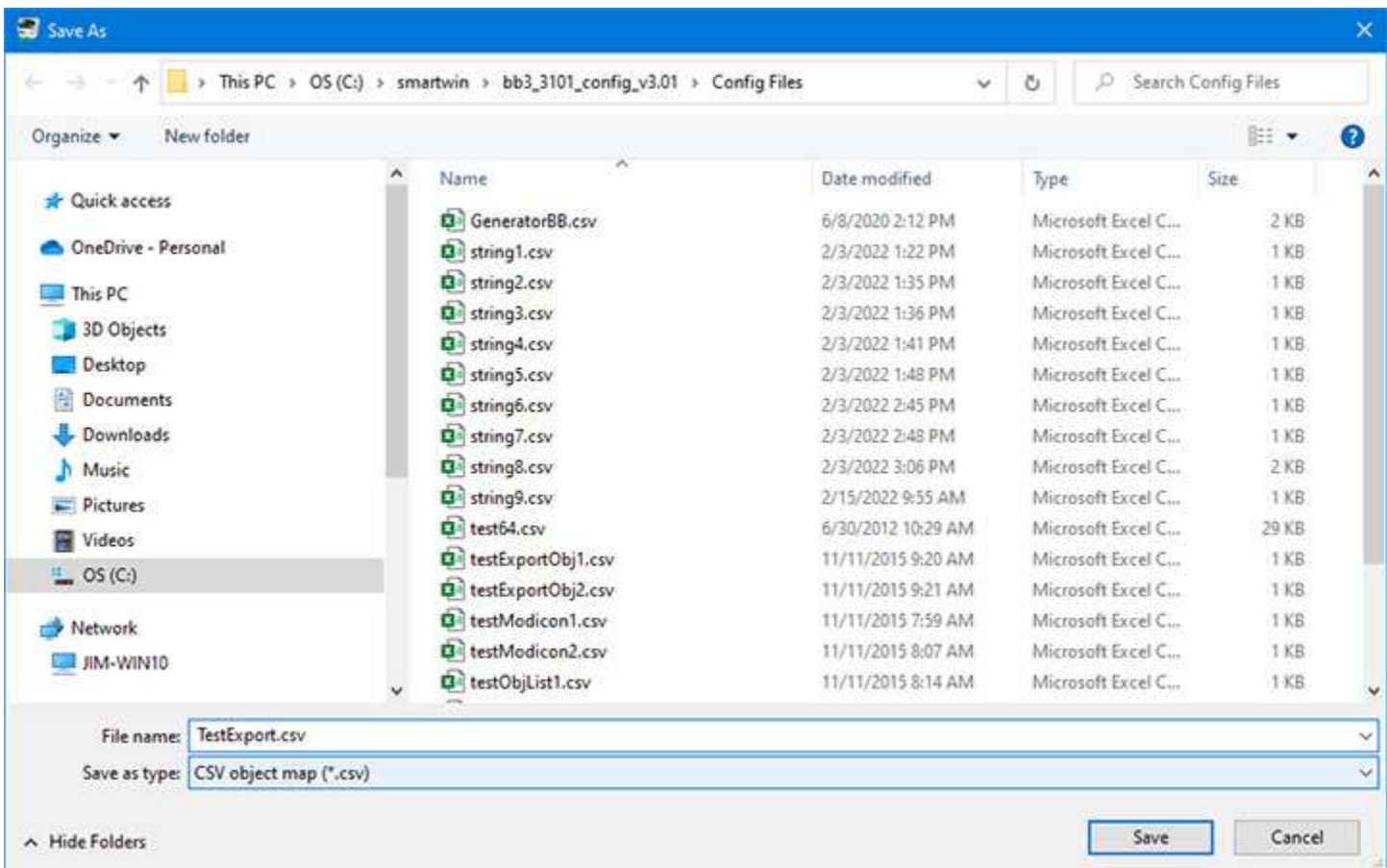
Obj...	Name	R/W	Device	Object	Type	Period	Qualifier
AI 1	Analog Input 1	R	Modbus [1]	HoldRegs [1]	U32 Hi	2 sec	XG
AI 2	Analog Input 2	R	Modbus [1]	HoldRegs [1]	U32 Hi	2 sec	G
AI 3	Analog Input 3	R	Modbus [1]	HoldRegs [5]	U16	2 sec	XG
AI 4		---	None	None	None	0 sec	
AI 5		---	None	None	None	0 sec	
AI 6		---	None	None	None	0 sec	
AI 7		---	None	None	None	0 sec	
AI 8		---	None	None	None	0 sec	
AI 9		---	None	None	None	0 sec	
AI 10		---	None	None	None	0 sec	
AO 1	Analog Output 1	W	Modbus [1]	HoldRegs [3]	U32 Hi	2 sec	XG
AO 2	Analog Output 2	W	Modbus [1]	HoldRegs [3]	U32 Hi	2 sec	XG
AO 3	Analog Output 3	W	Modbus [1]	HoldRegs [6]	U16	2 sec	XG
AO 4		---	None	None	None	0 sec	
AO 5		---	None	None	None	0 sec	
AO 6		---	None	None	None	0 sec	
AO 7		---	None	None	None	0 sec	

10.3 Export and Import Configuration as CSV file

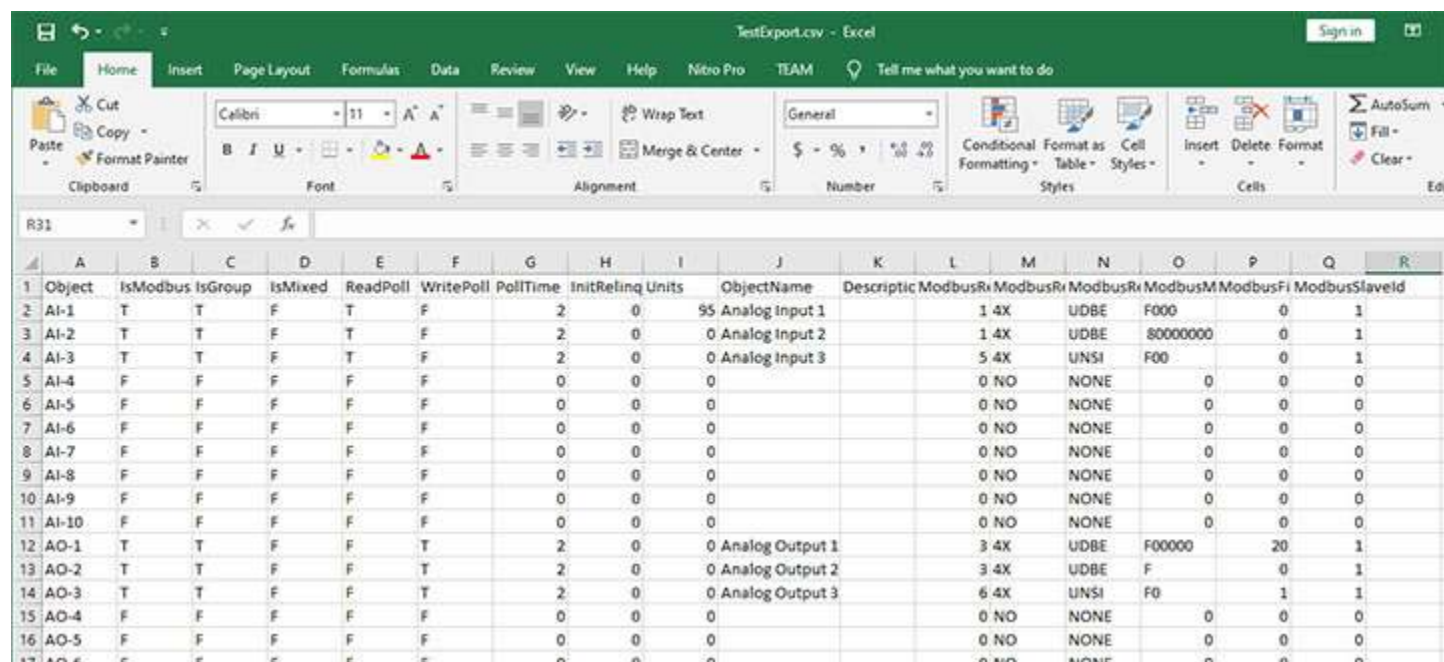
To save your object maps as a CSV file, click the New File icon.



The Windows file dialog will appear, and you may then enter the name under which the file should be saved.

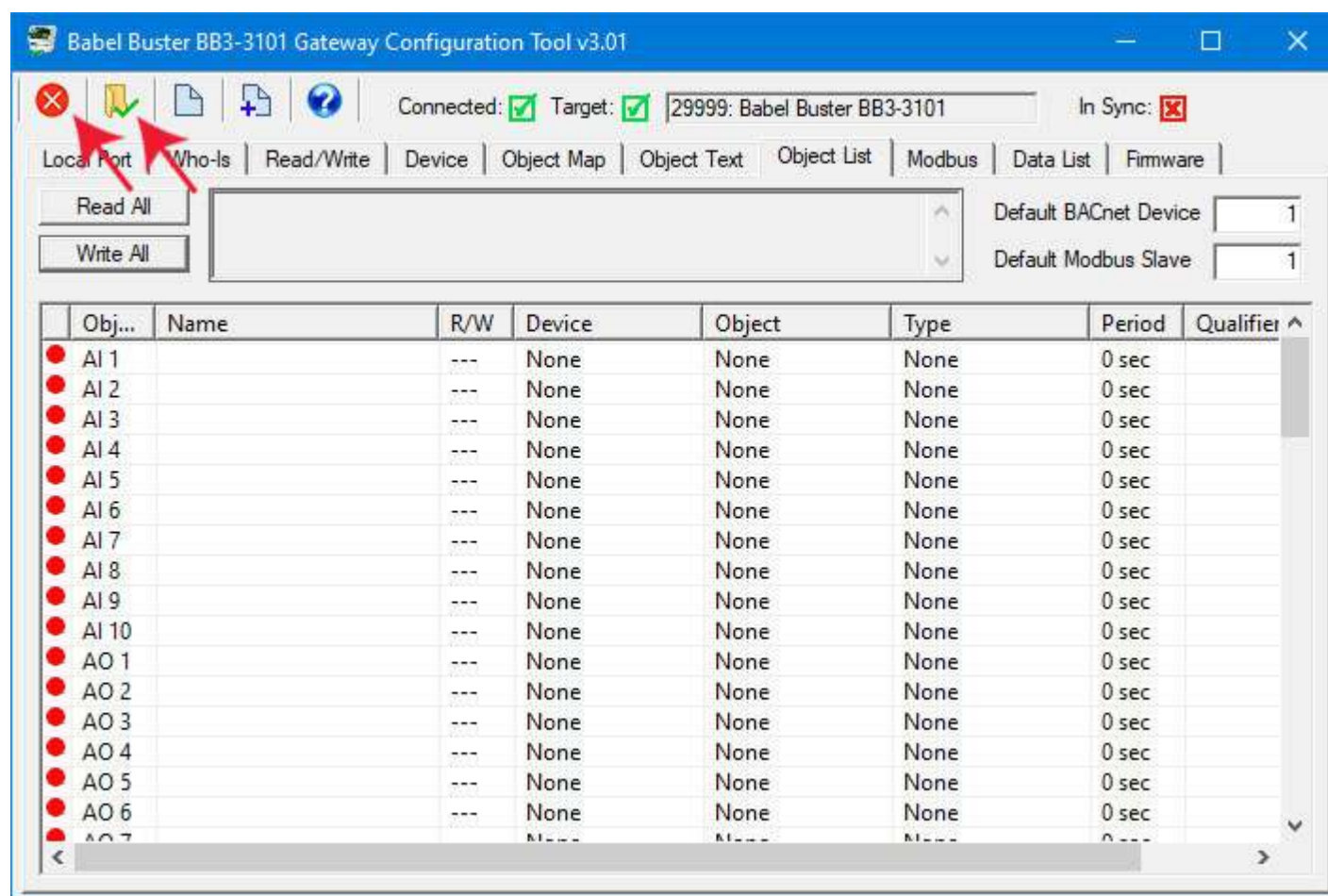


If you subsequently open the saved CSV file with a spread sheet program, it will appear something like this. Refer to Appendix A for a complete listing of all of the possible columns in the spread sheet. When saving a CSV file, the configuration tool will skip any columns that are not used at least once in your configuration.

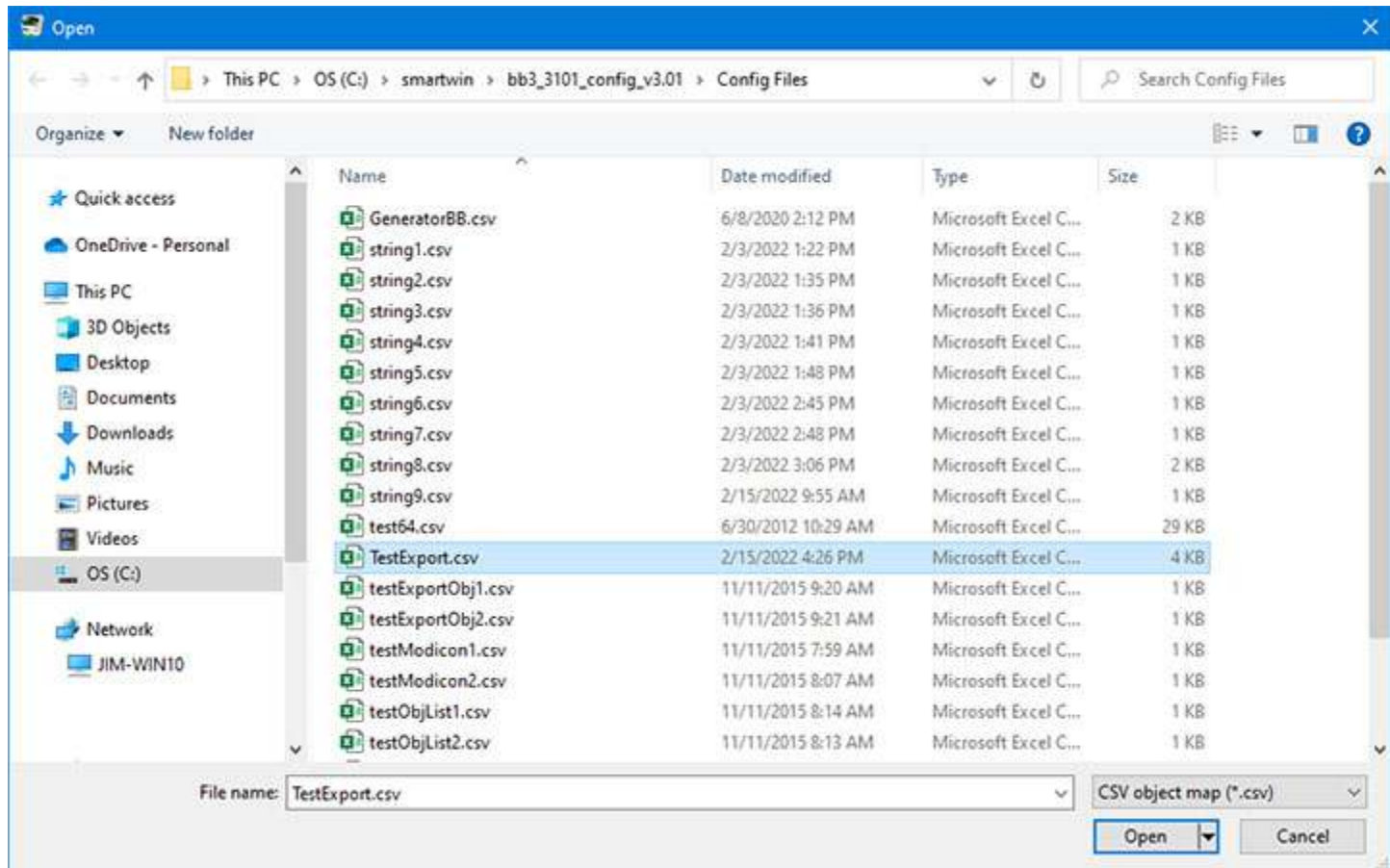


Object	IsModbus	IsGroup	IsMixed	ReadPoll	WritePoll	PollTime	InitReling	Units	ObjectName	Descriptive	ModbusR	ModbusRi	ModbusM	ModbusFi	ModbusSlaveId
AI-1	T	T	F	T	F		2	0	95 Analog Input 1		1 4X	UDBE	F000	0	1
AI-2	T	T	F	T	F		2	0	0 Analog Input 2		1 4X	UDBE	80000000	0	1
AI-3	T	T	F	T	F		2	0	0 Analog Input 3		5 4X	UNSI	F00	0	1
AI-4	F	F	F	F	F		0	0			0 NO	NONE		0	0
AI-5	F	F	F	F	F		0	0			0 NO	NONE		0	0
AI-6	F	F	F	F	F		0	0			0 NO	NONE		0	0
AI-7	F	F	F	F	F		0	0			0 NO	NONE		0	0
AI-8	F	F	F	F	F		0	0			0 NO	NONE		0	0
AI-9	F	F	F	F	F		0	0			0 NO	NONE		0	0
AI-10	F	F	F	F	F		0	0			0 NO	NONE		0	0
AO-1	T	T	F	F	T		2	0	0 Analog Output 1		3 4X	UDBE	F00000	20	1
AO-2	T	T	F	F	T		2	0	0 Analog Output 2		3 4X	UDBE	F	0	1
AO-3	T	T	F	F	T		2	0	0 Analog Output 3		6 4X	UNSI	F0	1	1
AO-4	F	F	F	F	F		0	0			0 NO	NONE		0	0
AO-5	F	F	F	F	F		0	0			0 NO	NONE		0	0

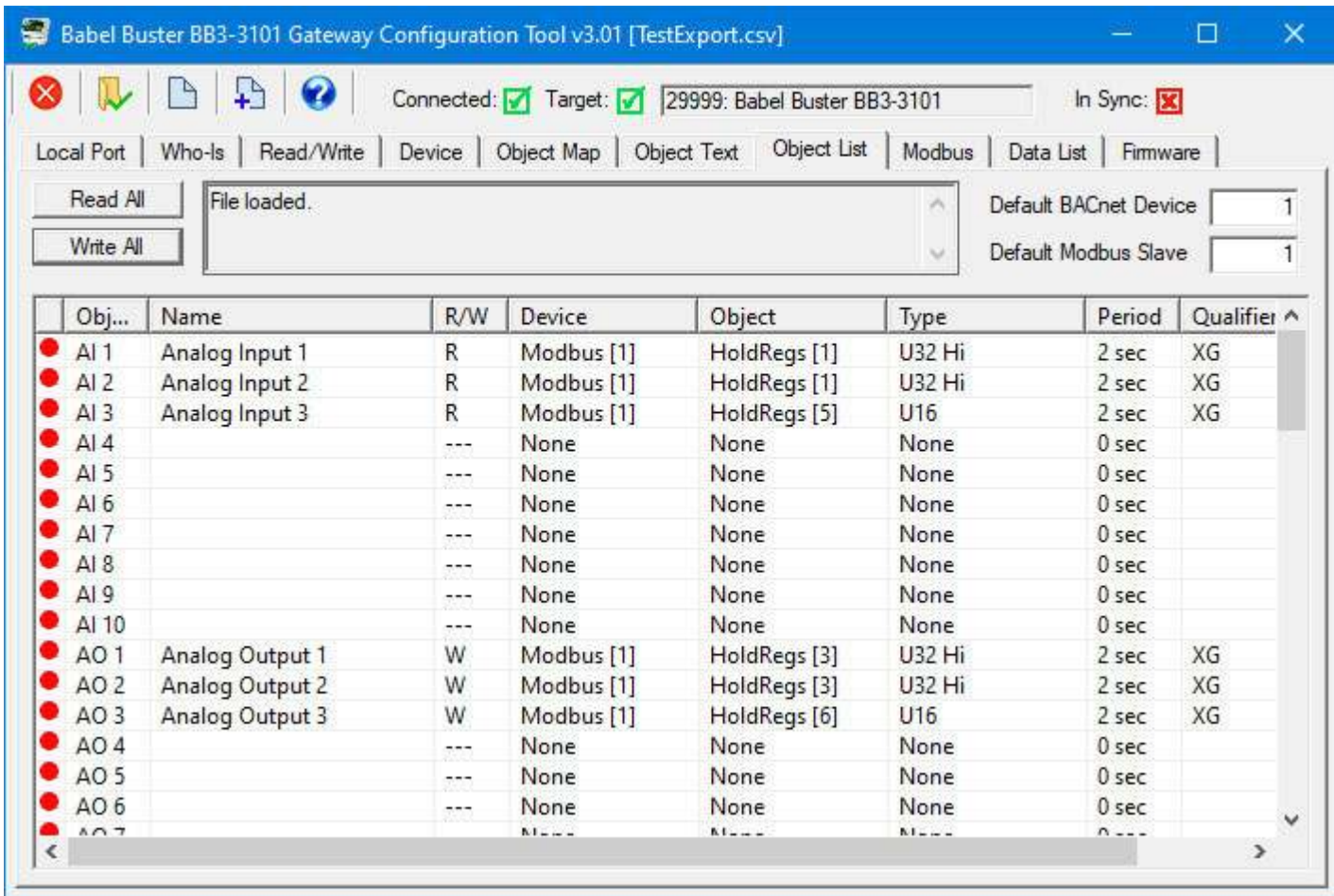
To clear the object maps and reload a previously saved CSV file, start by clicking the Clear icon (first icon on the left). Then click the File Open icon.



The Windows file dialog will appear, allowing you to select the file to be loaded.



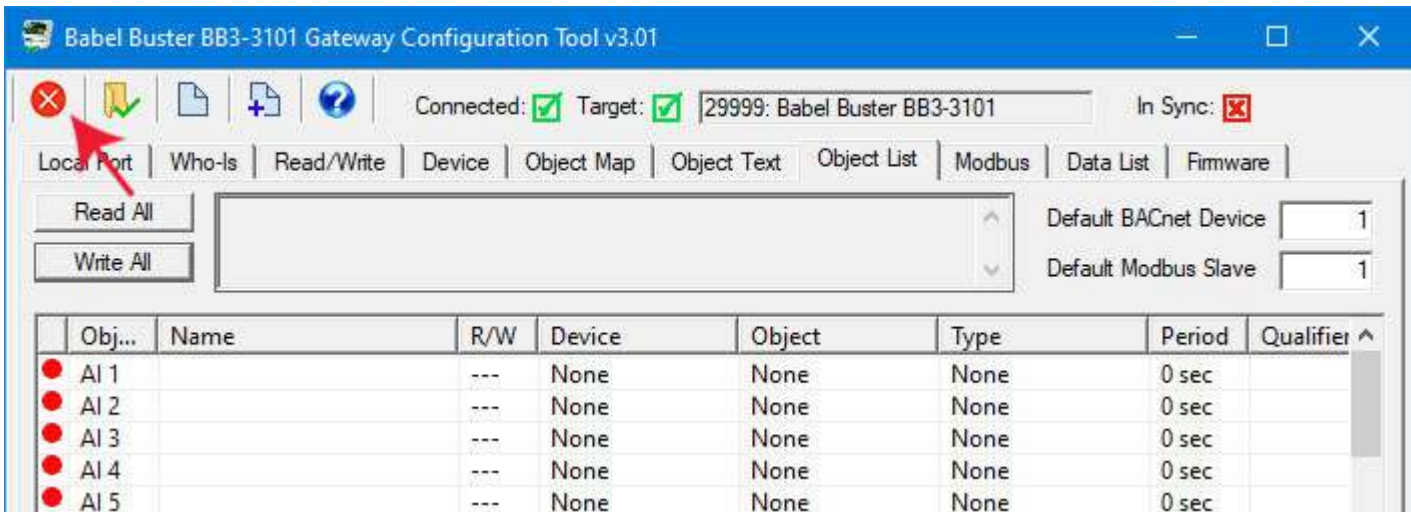
Upon loading the CSV file, the screen looks like it did when we first saved the file above, except the icons in the first column are red instead of green. Green means the object map is in sync with the device. Red means the object map contained within the configuration tool is not in sync with object maps actually contained within the gateway device itself. You need to click Write All to write the maps to the device, and then they will become in sync.

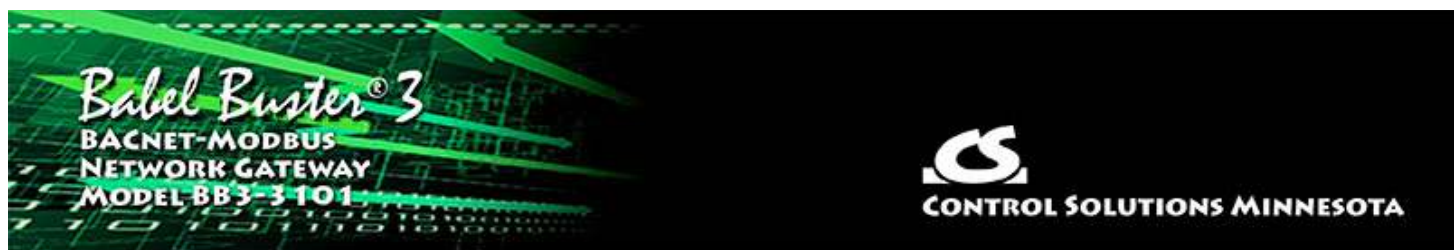


10.4 Clearing Object Maps, Counts, and Device

Clicking the Clear icon (first icon on the left in the top toolbar) on the Object List page only clears the object maps. It does not affect object counts or device properties. If you want to change any of those, go to the Device page and click the Clear icon there to clear.

The buttons in the configuration tool only clear information stored by the tool software in your PC's memory. To clear configuration in the physical gateway device, you need to follow the "Reconfigure Object Counts" procedure discussed in Section 7 of this user guide.





11 Tool 'Modbus' Page

11.1 Set Modbus Port Parameters

The Modbus page is where you configure the Modbus RTU port and check Modbus errors. Only the right half of the page is used for configuring Modbus. The left half is for diagnostics.

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.)
Alt Map	When the BB3-3101 gateway is being used as a Modbus slave, the Modbus register addresses are normally calculated as shown in Appendix D. If you prefer to use the alternate mapping scheme

	also talked about in Appendix D, then check this box. This selection only applies if the gateway is a Modbus slave.
Swap Registers if Slave	<p>Registers containing data longer than 16 bits are actually multiple registers treated as a single data value. In the case of 32-bit data values (32-bit integer or floating point), two consecutive registers are used. However, the order in which the registers appear in the register map is not standardized. Selection of the register ordering is done in the object maps (see Section 8) when the gateway is operating as Modbus master. When the gateway is a Modbus slave, then the register ordering is set here and it applies to all slave registers in the gateway.</p> <p>The BB3-3101 defaults to having the most significant data in the first register (lowest numbered register). If your Modbus master expects to read the least significant data first, then check "Swap Registers if Slave".</p> <p>If the data your Modbus master is receiving does not make any sense, try changing this "Swap" setting and see if you get better results. Remember that this only applies to 32-bit integer or floating point values. If your data is wrong for a 16-bit integer, your problem lies elsewhere.</p>
Baud Rate	Standard baud rates from 1200 to 115,200 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	<p>Timeout applies only when the gateway is operating as Modbus master. This is the amount of time (in seconds, fractions to tenths supported) that the gateway will wait for the slave device to respond. If the gateway has not received a response within this time, it is counted as a timeout and the no-response error status is indicated.</p> <p>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.</p>
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	<p>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.</p> <p>If your Modbus slave responds to some queries, but gets seemingly random timeouts ("no response" errors), it may be missing some requests due to the gateway sending requests too fast. Increase the pre-delay when a slave is getting some successful replies, but</p>

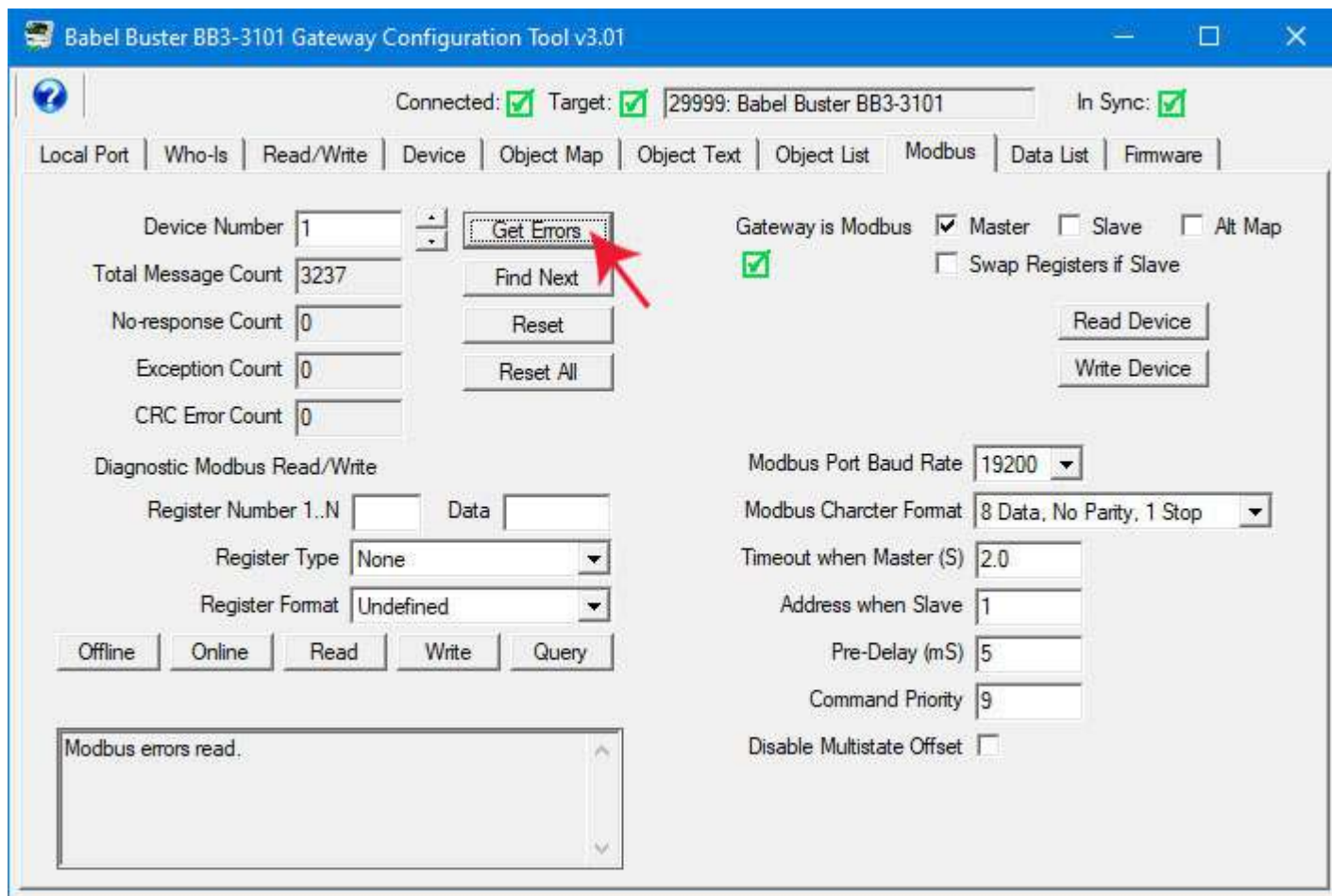
	intermittent timeouts with no other types of errors reported.
Command Priority	This priority is applied to commandable objects when written to by an external Modbus master when the BB3-3101 is functioning as a Modbus slave. The default is priority 10. Priority must be from 1 to 16, except 6 is reserved.
Disable Multistate Offset	BACnet protocol dictates that a Multistate object may not have a value of zero. Modbus registers often have a value of zero. By default, the BB3-3101 will translate from Modbus to BACnet Multistate with an offset of one so that Modbus zero is indicated as state 1 in the Multistate object. To disable this offset, check the Disable Multistate Offset box. See section 12.3 for additional discussion.

11.2 Check/Reset Errors

Enter or scroll to a desired Modbus slave address and click 'Get Errors' to see the message and error counts recorded by BB3-3101 for that device. Click 'Find Next' to have BB3-3101 automatically scan through all Modbus devices and report the next device that has non-zero error counts.

Click 'Reset' to clear the error counts and message count for the Modbus device whose address is currently displayed. Click 'Reset All' to clear all counts for all Modbus devices the BB3-3101 is polling.

Note: Total message count is a 32-bit count. No-response count is a 16-bit count and will stall at 65535. CRC and exception errors are 8-bit counts, and will therefore stall at 255. Normally you expect few if any errors once the system has been successfully commissioned, therefore space is not wasted on maintaining large counts. If you see a count of 255 exception errors, you know there were at least that many and likely more, but this will generally be observed only when first configuring the system and correcting configuration errors.



Babel Buster BB3-3101 Gateway Configuration Tool v3.01

Connected: ☒ Target: ☒ 29999: Babel Buster BB3-3101 In Sync: ☒

Local Port | Who-Is | Read/Write | Device | Object Map | Object Text | Object List | Modbus | Data List | Firmware

Device Number: 1 Gateway is Modbus: ☒ Master ☐ Slave ☐ Alt Map ☐ Swap Registers if Slave

Total Message Count: 3237

No-response Count: 0

Exception Count: 0

CRC Error Count: 0

Diagnostic Modbus Read/Write

Register Number 1..N: Data:

Register Type:

Register Format:

Modbus Port Baud Rate: 19200

Modbus Charcter Format: 8 Data, No Parity, 1 Stop

Timeout when Master (S): 2.0

Address when Slave: 1

Pre-Delay (mS): 5

Command Priority: 9

Disable Multistate Offset: ☐

Modbus errors read.

11.3 Diagnostic Modbus Read/Write

A very simple Modbus scanner is provided for diagnostic use, to see if your Modbus device is responding. The scanner uses the direct Modbus command pass-through, but does it in a fashion that is more user friendly. The scanner does not really 'scan' – we just call it that because the term scanner is widely used and understood by anyone with Modbus experience. In our instance, our 'scanner' sends a single request when you click Read or Write, and retrieves the corresponding query when you click Query.

Although the scanner will function at the same time gateway maps are generating Modbus traffic, it is easier to see error indications if you place the gateway in "offline" mode. The device LEDs (Request, Reply) will then only flash when you send a diagnostic command. The definition of "offline" is that read/write maps are not processed, but the device is functional in every other way. (Outside of this tool, the offline mode is achieved by writing NON_OPERATIONAL to the System Status property of the Device Object.)

To use the scanner, enter the register number you wish to read on the remote Modbus device (slave address given as Device Number above). Select a register type and format. Then click Read, or enter data in the data window and click Write. About a second or two later click Query to retrieve the result of the read or write. Commands and responses are shown in the log window at the bottom.

You will get an error in the reply if the Modbus device does not respond or any of several other problems exist. The following illustrates an exception reply because register 400 does not exist in this device:

Babel Buster BB3-3101 Gateway Configuration Tool v3.01

Connected: ☒ Target: ☒ 29999: Babel Buster BB3-3101 In Sync: ☒

Local Port | Who-Is | Read/Write | Device | Object Map | Object Text | Object List | Modbus | Data List | Firmware

Device Number: 1 Gateway is Modbus: ☒ Master ☐ Slave ☐ Alt Map

Total Message Count: 3237 ☐ Swap Registers if Slave

No-response Count: 0

Exception Count: 0

CRC Error Count: 0

Diagnostic Modbus Read/Write

Register Number 1..N: 400 Data:

Register Type: Holding Register (fc 16)

Register Format: Signed 16-bit

Modbus Port Baud Rate: 19200

Modbus Charcter Format: 8 Data, No Parity, 1 Stop

Timeout when Master (S): 2.0

Address when Slave: 1

Pre-Delay (mS): 5

Command Priority: 9

Disable Multistate Offset: ☐

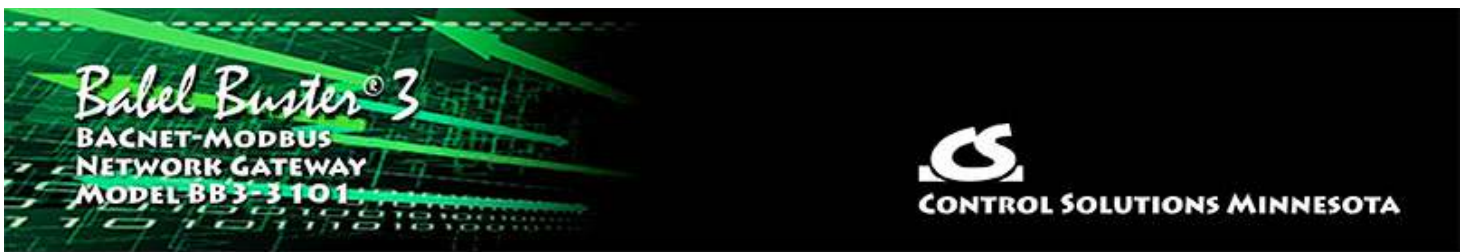
Request sent.
[01,03,01,8F,00,01,B4,1D]
Checking reply...
[01,83,02]
Error in reply.

The first byte in the reply is slave address or unit number. The second byte is function code. If the most significant bit is set, it means there is an exception error. The MSB set shows up in hexadecimal format is the first digit being 8 (hex 83 returned for function code 3). The third byte, when the reply is an exception error, will be one of the error codes shown in the table below.

If you see "[empty]" immediately after "Checking reply..." it means the gateway is still waiting. We tend to assume still waiting for a reply, but it can mean waiting for our turn to send the request in the first place. If the gateway is configured with a significant number of points, click Offline to tell the gateway to stop polling other Modbus registers. You should then get either a reply, or the response that indicates an actual valid timeout occurred.

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	<p>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.</p> <p><i>This is the most common error users observe with the BB3-3101. It typically means configuration is incorrect - the gateway is attempting to request a register that does not exist in the Modbus slave. The most frequent incorrect register problem is entering 40001 for register number when you should enter just 1 and select holding register for type.</i></p>

3	Illegal Data Value	The value contained in the query's data field is not acceptable to the slave.
4	Slave Device Failure	An unrecoverable error occurred.
6	Slave Device Busy	The slave is engaged in processing a long-duration command. The master should try again later.
0A (hex)	Gateway Path Unavailable	Gateway could not establish communication with target device.
0B (hex)	Gateway Target Device Failed to Respond	Specialized use in conjunction with gateways, indicates no response was received from the target device.
11 (hex)	Gateway Target Device Failed to Respond	No response from slave, request timed out.

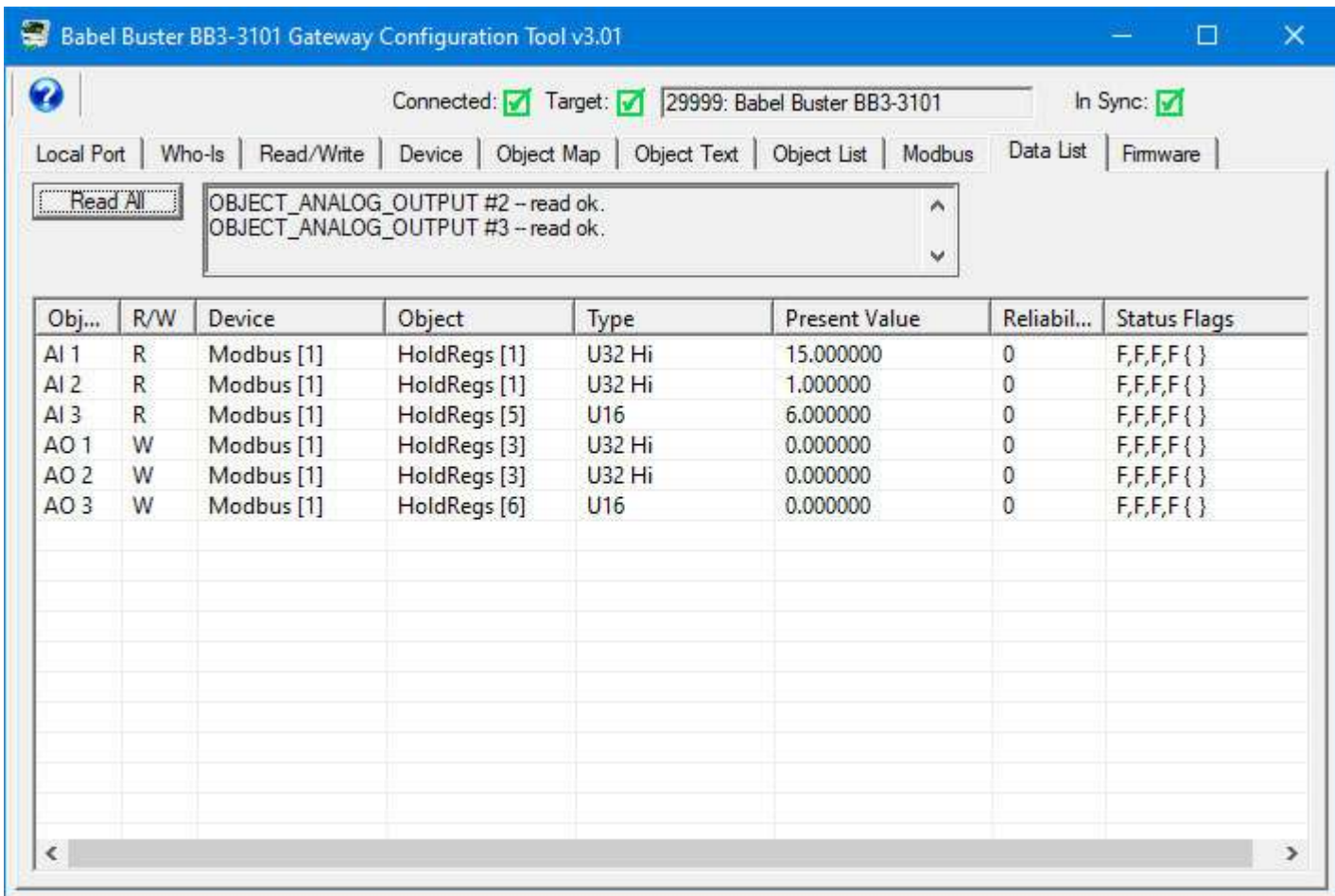


12 Tool 'Data List' Page

12.1 Read All Object Data

Click 'Read All' to cause the tool to query all BACnet objects in the BB3-3101 device. A brief summary of the object's mapping is displayed along with its Present Value, Reliability code, and Status Flags. Progress of the 'Read All' process will scroll through the log window at the top.

To manually change the values in any BACnet object in the gateway device, use the Read/Write page to access object properties.

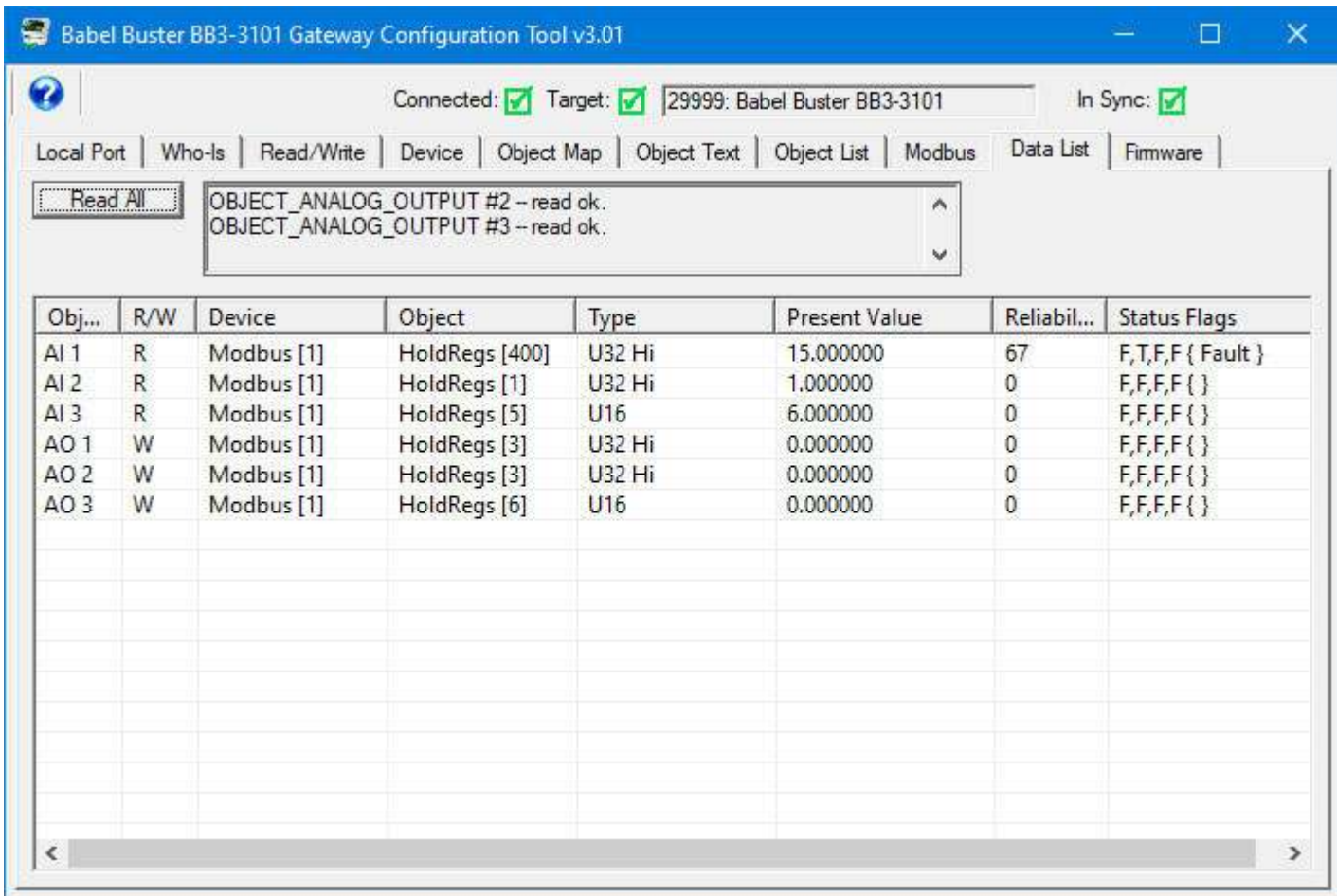


Obj... (Object)	The first "Obj..." column is the local object identification.
R/W	The R/W column indicates the following: R means periodic read, W means periodic write, W+ means write on delta. RW+ means a combination of those.

Device	The Device column will show what remote slave device this object is mapped to. 'Modbus[N]' means Modbus RTU device whose slave address is N. 'MS/TP[N]' means BACnet slave whose device instance is N.
Object	The second Object column shows what object in the slave device is mapped to this local object. If the object refers to a Modbus register, the register number is shown, e.g. 'HoldRegs[3]' means holding register 3. If the object is a BACnet object, the object type and instance are shown, e.g. 'AI[1]' means Analog Input 1. In the case of a commandable object, you will see something like "AO[2] P=3" - this means Analog Output 2, and write priority is set to 3.
Type	Type shows the remote object data format. If the remote device is Modbus, the data format will be a notation applicable to Modbus, such as Unsigned for 16-bit integer register. If the remote device is BACnet, the type will indicate property type, such as Present Value.
Present Value	Present Value shows a representation of the present value of the BACnet object in the BB2-3010/3060.
Reliability	Reliability code for the BACnet object in the BB2-3010/3060 is shown. These are listed below.
Status Flags	Status Flags shows the status flags for the object. The standard flags are: IN_ALARM, FAULT, OVERRIDDEN, and OUT_OF_SERVICE respectively (i.e. last T/F is out of service flag). The flag is F to show false/0 and T to show true/1.

12.2 Error Codes

If the slave device being queried by the gateway returns an error code, or there is simply a problem in communicating with the slave, this will be indicated by the gateway object's reliability code. When reliability code is non-zero, the fault flag is also set. Therefore, the Status Flags indication will typically be "F,T,F,F" any time the reliability code is something other than zero (zero means no errors to report).



Error codes returned by a Modbus slave device or BACnet slave device are encoded into reliability codes presented by the respective object. Reliability codes (read via the Reliability property) are as follows:

- no response (64)
- crc error (65)
- exception, illegal function code (66)
- exception, illegal data address (67)
- exception, illegal data value (68)
- exception, TCP gateway path unavailable (check IP address) (75)
- exception, TCP gateway target device failed to respond (connection problem) (76)
- exception, other code returned by device, code+65, (69..79)
- configuration property fault (80)
- exception, code not recognized (81)
- BACnet client read/write timeout (82)
- BACnet client received error response from slave (83)

If the object's reliability indicates code 83, meaning an error code was returned by a remote BACnet device, the actual code returned can be learned by reading the local object's properties 830 (class) and 831 (code). If an error exists and a reliability code other than 83 is indicated, reading properties 830-831 can also return class 64 (proprietary) and the reliability code+200 (i.e. Modbus illegal function code exception would be class 64, code 266).

12.3 Multistate Value Offset

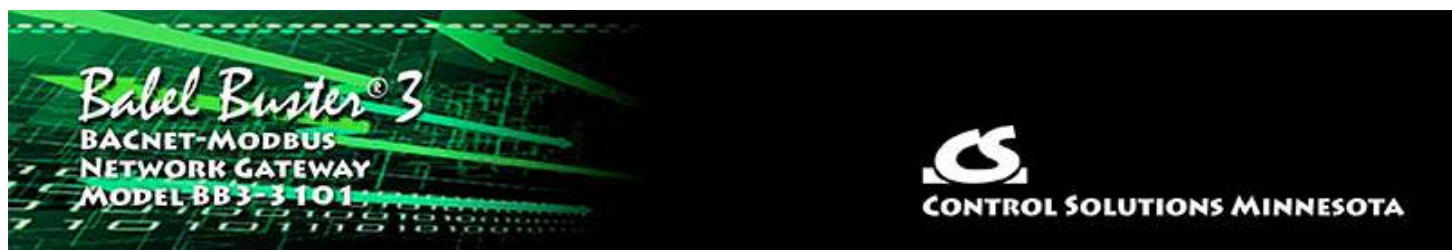
BACnet protocol dictates that a Multistate object may not have a value of zero. Modbus registers often have a value of zero. By default, the BB3-3101 will translate from Modbus to BACnet Multistate with an offset of one so that Modbus zero is indicated as state 1 in the Multistate object.

You have the option of disabling this offset by checking the Disable Multistate Offset box on the Modbus page. However, any values from Modbus will be bounded to legal Multistate values. Therefore, a Modbus value of zero will be forced to a value of one. At the same time, a value of one will remain a value of one. Therefore, without the offset, you cannot tell the difference between the Modbus values of zero and one.

The minimum value that a Multistate object can ever contain is one. The maximum value will be that maximum state value configured on the Device page. Values coming from Modbus that fall outside this range will be forced into bounds.

The bounding to legal values mentioned here only applies to Multistate objects.

The maximum state value is configurable on the Device page. The maximum value you will ever be able to set is 4096 since that is the maximum number of state strings available (based on memory capacity). The practical limit will be lower than that. If you attempt to set a limit that is out of range of available resources, you will get an "Invalid configuration" error message when attempting to reconfigure the gateway on the Device page. The pool of 4096 state strings is shared among all Binary and Multistate objects.



13 Tool 'Firmware' Page

Firmware update support is included as a page in the BB3-3101 configuration tool. The firmware update is done via MS/TP and can be done without taking the BB3-3101 offline except temporarily to restart when the new firmware is ready to run.

13.1 Firmware Organization

There are two copies of the application firmware - a primary and a backup image. The primary image is the one that is always running when the BB3-3101 is booted up. In the unlikely event that the primary image becomes corrupted, the backup image will replace it.

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

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

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

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

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

13.2 Verifying Primary Application Image

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

Babel Buster BB3-3101 Gateway Configuration Tool v3.02

Connected: ☒ Target: ☒ 29999: Babel Buster BB3-3101 In Sync: ☒

Local Port | Who-Is | Read/Write | Device | Object Map | Object Text | Object List | Modbus | Data List | Firmware

Code Size: 115856
Code Checksum: CEB4CC81
Model Number: 3101
Model Variant: 1
Revision: 3.10.1

☒ Primary ☐ Update ☐ File
☒ Stored ☐ Verified

Get Boot Info

Upload Firmware

Boot info: Valid
Query result: No error

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

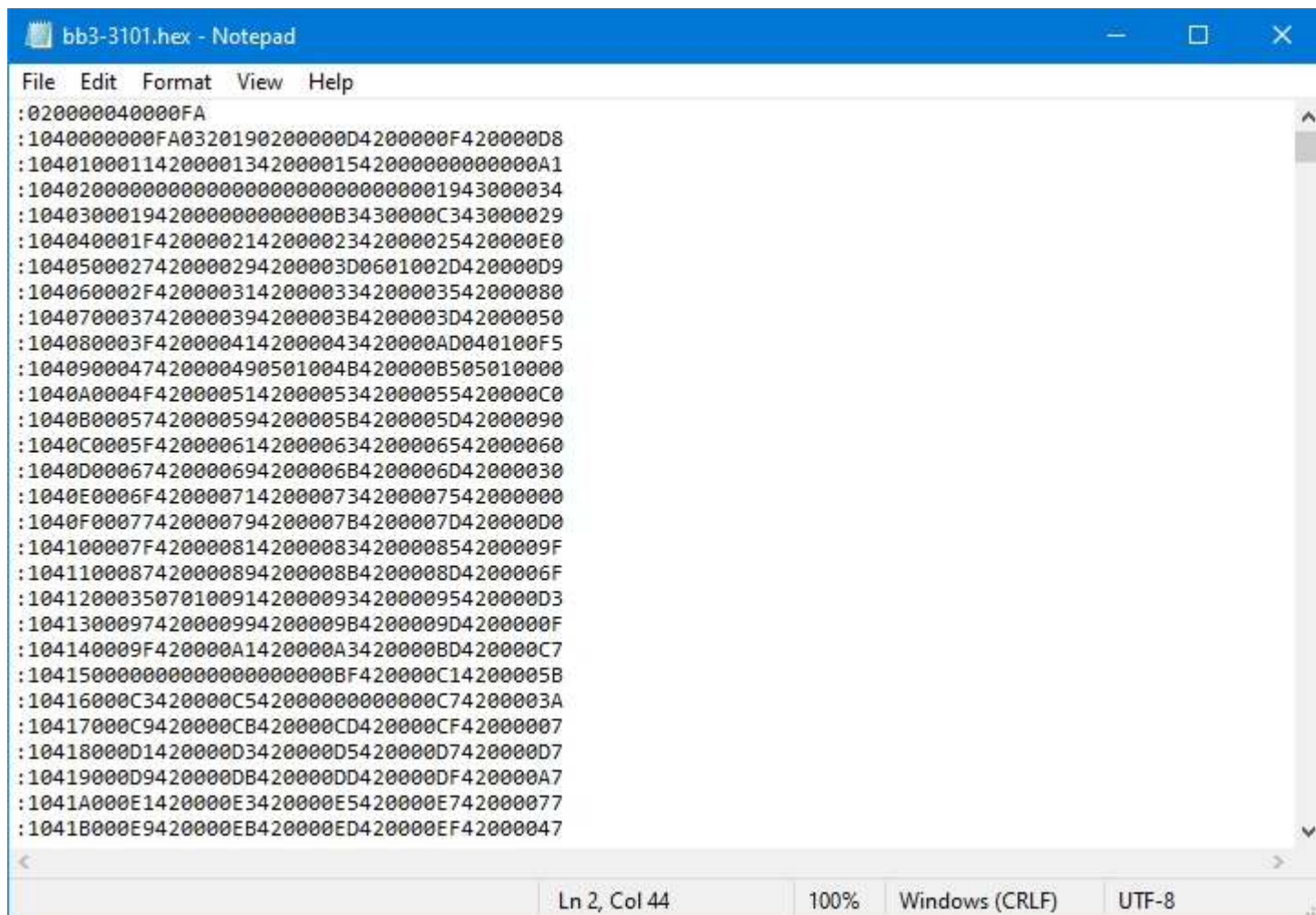
The screenshot shows the 'Firmware' tab of the 'Babel Buster BB3-3101 Gateway Configuration Tool v3.02'. The interface includes a status bar at the top with 'Connected: [checked]', 'Target: [checked]', and 'In Sync: [X]'. Below this is a tabbed menu with 'Local Port', 'Who-Is', 'Read/Write', 'Device', 'Object Map', 'Object Text', 'Object List', 'Modbus', 'Data List', and 'Firmware' (selected). The main area contains several input fields and checkboxes. On the left, there are fields for 'Code Size' (115856), 'Code Checksum' (CEB4CC81), 'Model Number' (3101), 'Model Variant' (1), and 'Revision' (3.10.1). On the right, there are checkboxes for 'Primary' (checked), 'Update' (unchecked), 'File' (unchecked), 'Stored' (unchecked), and 'Verified' (checked). Below these are two buttons: 'Get Boot Info' and 'Upload Firmware'. At the bottom, there is a text box displaying 'Boot info: Valid' and 'Query result: No error'.

Code Size	115856	<input checked="" type="checkbox"/> Primary	<input type="checkbox"/> Update	<input type="checkbox"/> File
Code Checksum	CEB4CC81	<input type="checkbox"/> Stored	<input checked="" type="checkbox"/> Verified	
Model Number	3101	<button>Get Boot Info</button>		
Model Variant	1			
Revision	3.10.1	<button>Upload Firmware</button>		

Boot info: Valid
Query result: No error

13.3 Firmware Update Process

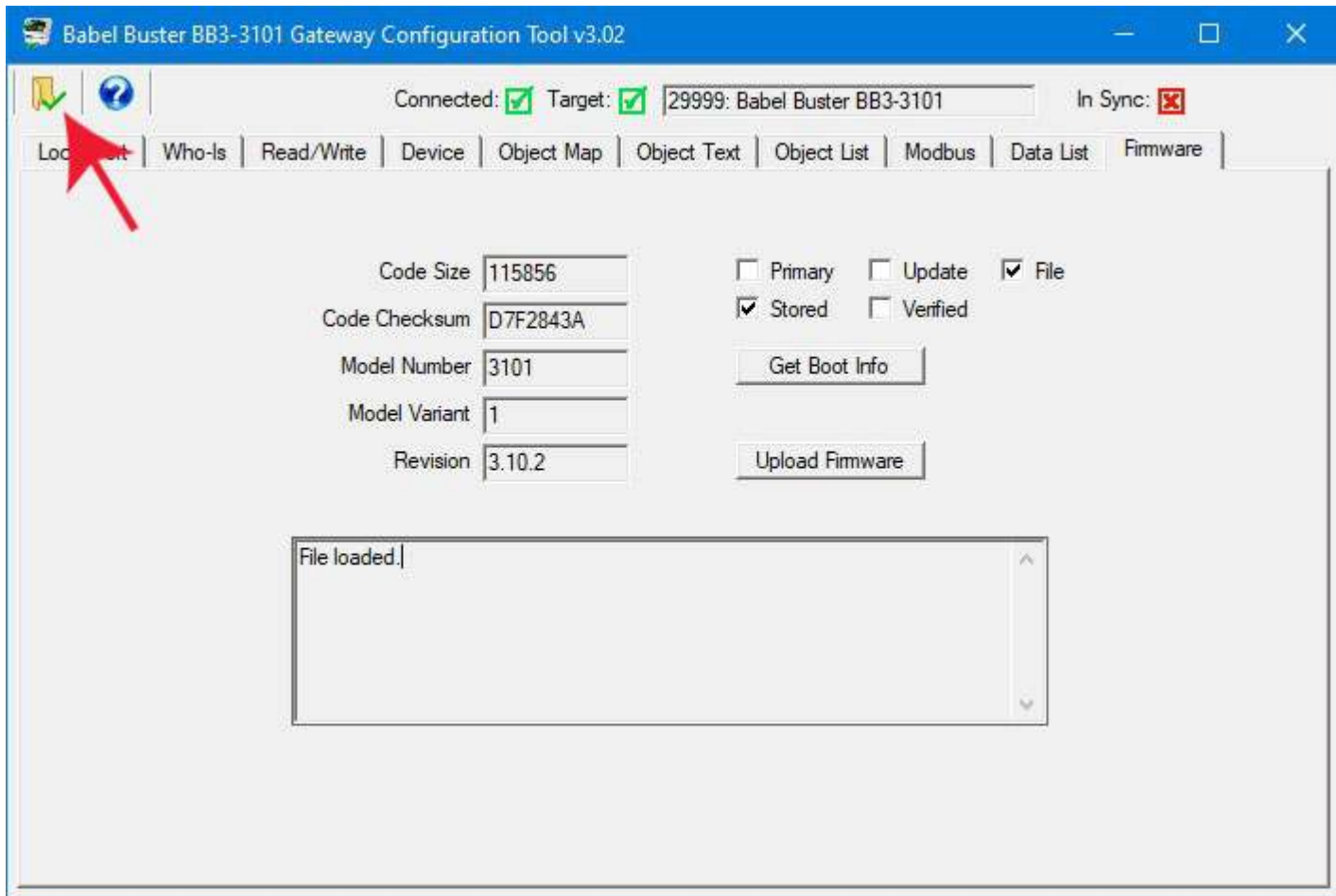
If there is newer firmware and you have good reason to believe a firmware update would resolve some issue you're having, then start by obtaining a copy of the firmware update from Control Solutions support. The update will be in the form of a "hex" file, typically named something like "bb3-3101.hex" and its contents, which will always be printable, will look something like this (abbreviated view):



```
bb3-3101.hex - Notepad
File Edit Format View Help
:020000040000FA
:1040000000FA0320190200000D4200000F420000D8
:1040100011420000134200001542000000000000A1
:10402000000000000000000000000000001943000034
:10403000194200000000000000B3430000C343000029
:104040001F420000214200002342000025420000E0
:1040500027420000294200003D0601002D420000D9
:104060002F42000031420000334200003542000080
:1040700037420000394200003B4200003D42000050
:104080003F4200004142000043420000AD040100F5
:1040900047420000490501004B420000B505010000
:1040A0004F420000514200005342000055420000C0
:1040B00057420000594200005B4200005D42000090
:1040C0005F42000061420000634200006542000060
:1040D00067420000694200006B4200006D42000030
:1040E0006F42000071420000734200007542000000
:1040F00077420000794200007B4200007D420000D0
:104100007F4200008142000083420000854200009F
:1041100087420000894200008B4200008D4200006F
:1041200035070100914200009342000095420000D3
:1041300097420000994200009B4200009D4200000F
:104140009F420000A1420000A3420000BD420000C7
:10415000000000000000000000BF420000C14200005B
:10416000C3420000C5420000C7420000000000C74200003A
:10417000C9420000CB420000CD420000CF42000007
:10418000D1420000D3420000D5420000D7420000D7
:10419000D9420000DB420000DD420000DF420000A7
:1041A000E1420000E3420000E5420000E742000077
:1041B000E9420000EB420000ED420000EF42000047
Ln 2, Col 44 100% Windows (CRLF) UTF-8
```

The BB3-3101 configuration tool can be connected locally to the MS/TP network via a Control Solutions MTX002 USB to MS/TP adapter, or via a BACnet IP to MS/TP router such as the Control Solutions BB3-7301. The ability to connect via an IP router means you can do a firmware update remotely. All firmware update support is accomplished via standard BACnet protocol using the Private Transfer service.

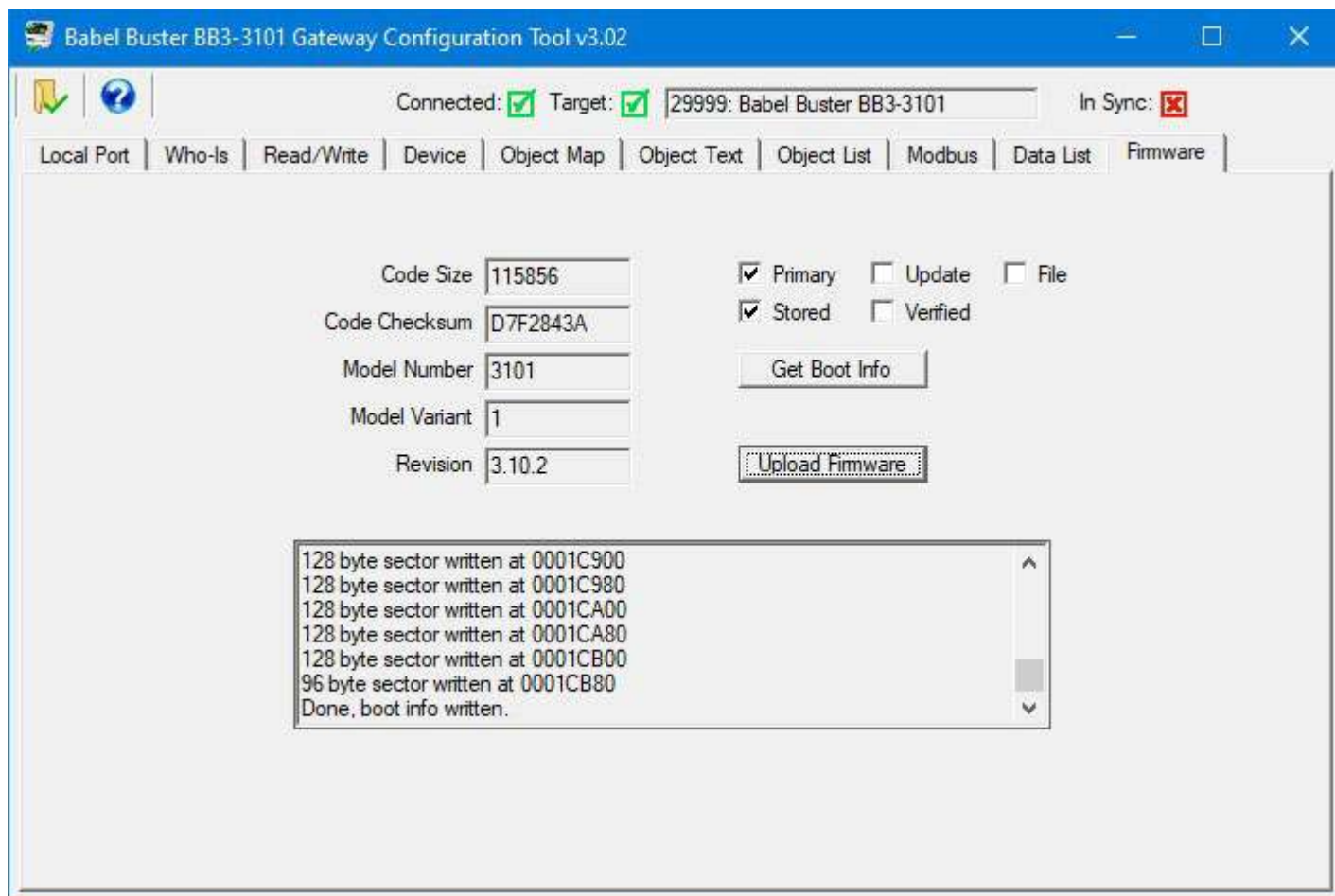
To upload a firmware update, start by clicking on the file open icon. Use the standard Windows file dialog that opens automatically to locate your copy of the .hex file and open it. After loading the file, the information displayed now is taken from the file just loaded. The File checkbox indicates that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



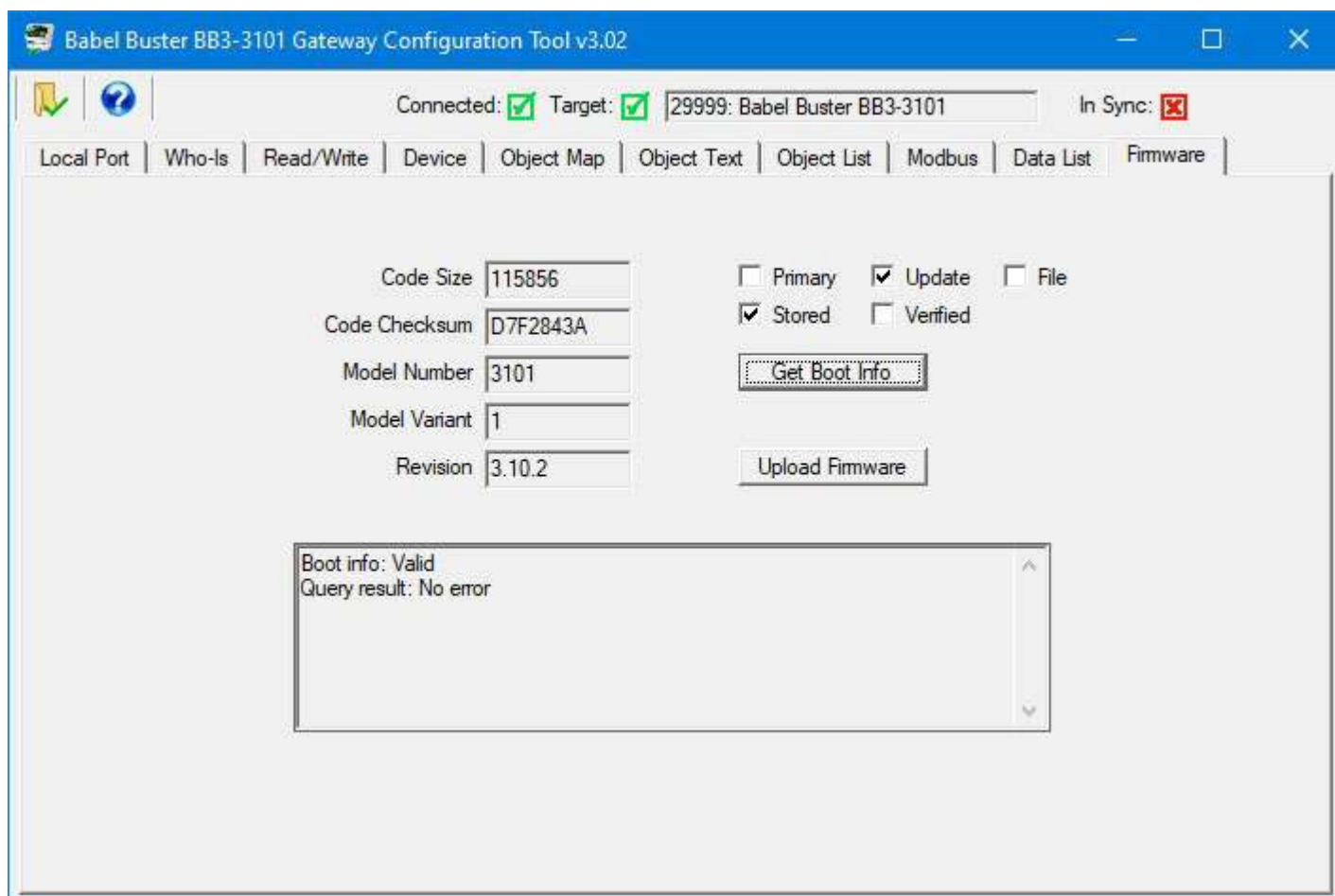
The upload process will take a couple of minutes. During this time, progress will continue to get posted to the status window. When the upload is complete, the boot info record for the update image is written.

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

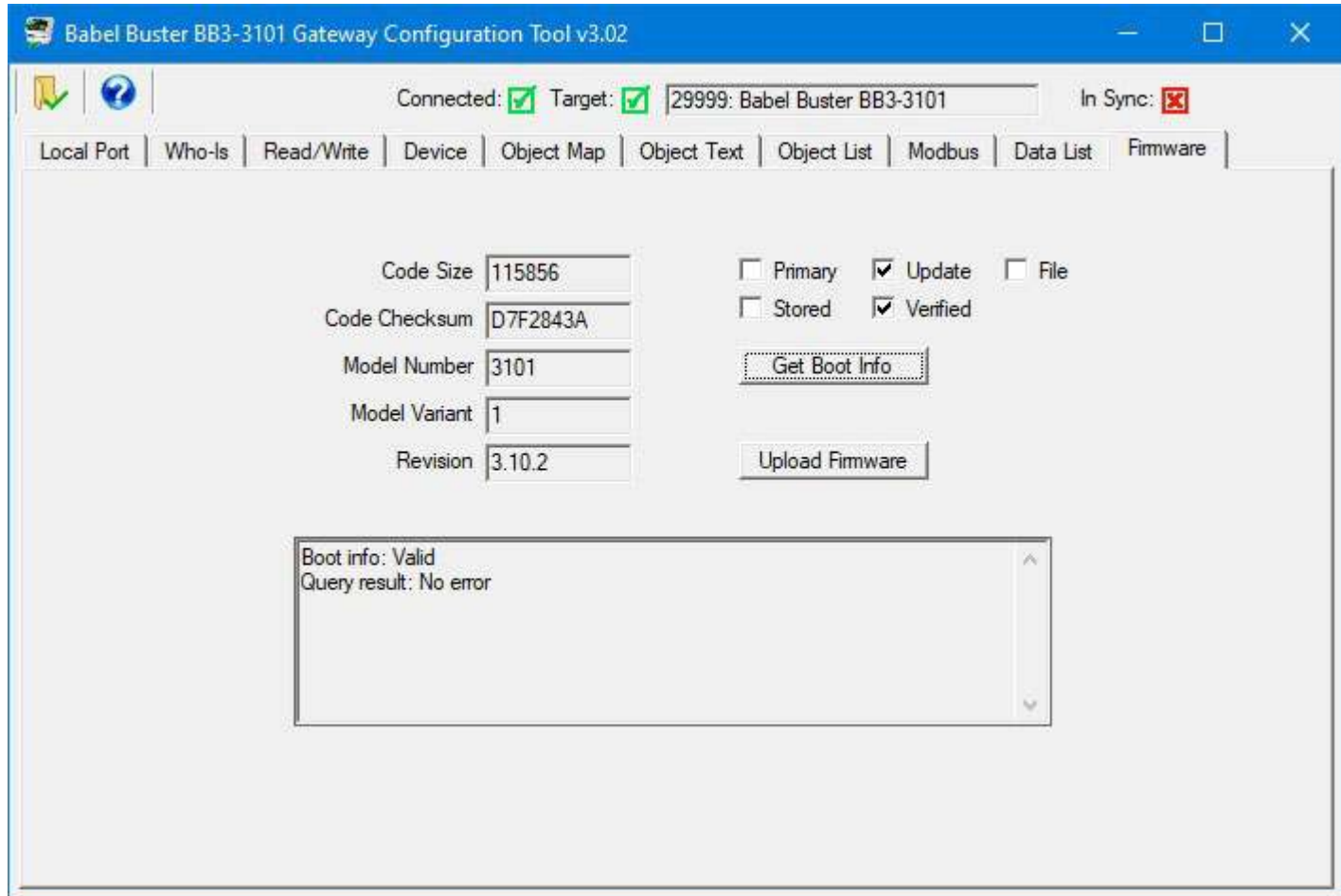
Because the firmware update is programmed into the backup image area, you may upload the firmware update while the gateway continues to function normally on the BACnet network. There is no need to take the gateway offline while uploading new firmware. The gateway will be offline only briefly when you restart the gateway after completing the upload.



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



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



Next, you will need to restart the gateway. Do this by going to the Device page, entering the reinitialize password (default is "buster"), and click the Reinit Device button. The gateway will now restart, and upon bootup, and upon seeing that a newer image is available, the backup image will be copied into the primary application flash area of memory.

Babel Buster BB3-3101 Gateway Configuration Tool v3.02

Connected: ☒ Target: ☒ 29999: Babel Buster BB3-3101 In Sync: ☒

Local Port | Who-Is | Read/Write | Device | Object Map | Object Text | Object List | Modbus | Data List | Firmware

Object ID (device instance) 29999 Reassign

Device Name
Default BB3-3101

Device Description
Modbus RTU to BACnet MS/TP Gateway

Device Location
Unassigned

☒ ☐ Auto-Reset Errors

Max Master 127 Get Info

MS/TP Baud Rate 38400 Change

MS/TP Station/MAC Addr 2

New Password

Old Password buster

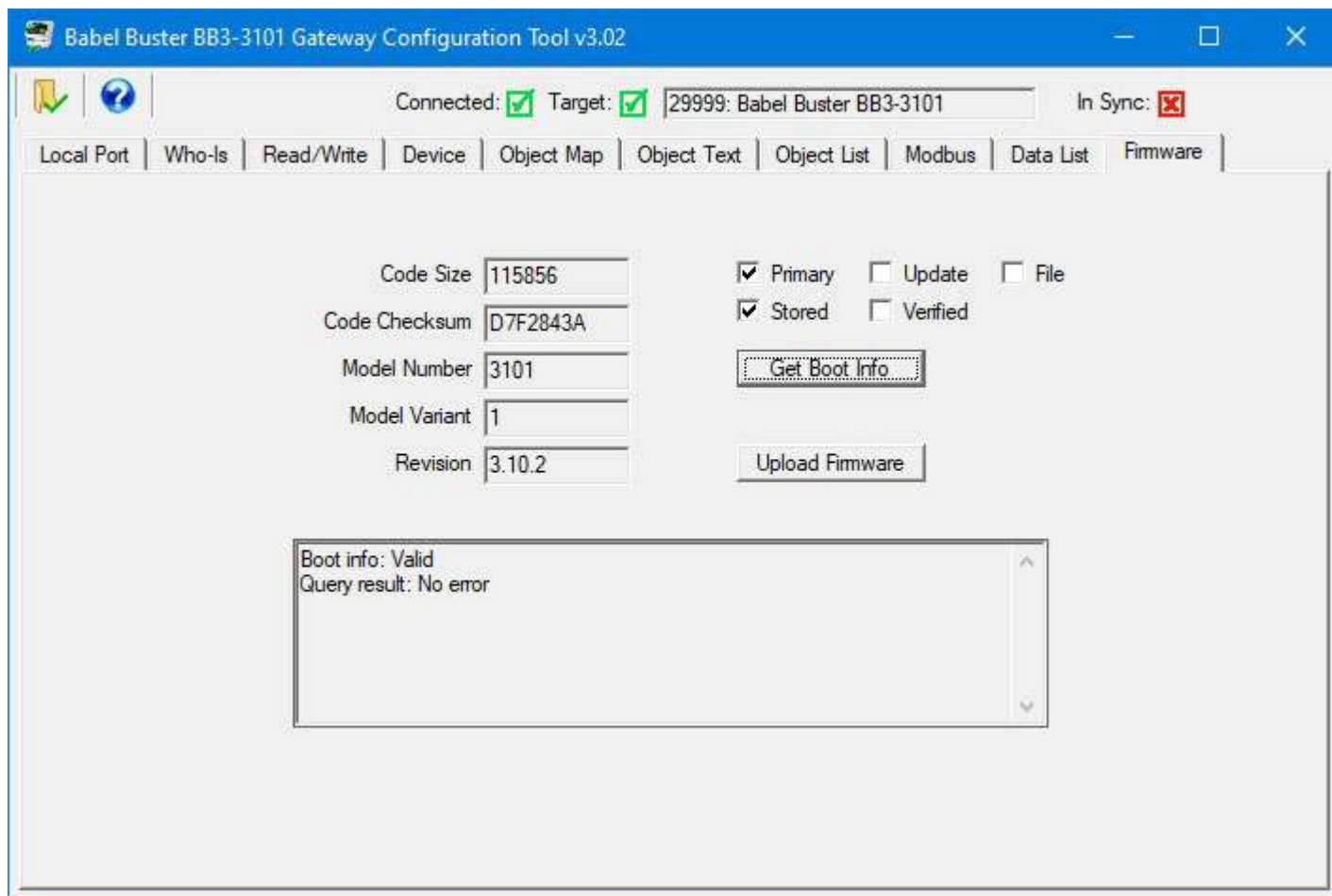
Reinit Device

Reconfigure Get Counts Check Fit

Device re-init sent.

Object Allocation	Present	Pending
Analog Input Count	10	10
Analog Output Count	10	10
Analog Value Count	10	10
Binary Input Count	10	10
Binary Output Count	10	10
Binary Value Count	10	10
Multistate Input Count	5	5
Multistate Output Count	5	5
Multistate Value Count	5	5
Multistate Max State	10	10

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



You can check Primary and Verified (and click Get Boot Info) to verify that the primary application did get correctly programmed into the primary flash area.

This verification actually happens automatically every time the gateway starts up. If the result was anything other than "Valid, No error", the backup restore will be attempted automatically if there is a valid backup image (which is also used as the update image), and there will be an error indication (red LED) alerting you to the fact this automatic verification failed. This verification via the configuration tool is redundant, but potentially useful if you are connected remotely and cannot see the LED.

The screenshot shows the 'Firmware' tab of the 'Babel Buster BB3-3101 Gateway Configuration Tool v3.02'. The interface includes a status bar at the top with 'Connected: [checked]', 'Target: [checked]', and 'In Sync: [X]'. Below this is a tabbed menu with 'Local Port', 'Who-Is', 'Read/Write', 'Device', 'Object Map', 'Object Text', 'Object List', 'Modbus', 'Data List', and 'Firmware' (selected). The main area contains fields for 'Code Size' (115856), 'Code Checksum' (D7F2843A), 'Model Number' (3101), 'Model Variant' (1), and 'Revision' (3.10.2). To the right are checkboxes for 'Primary' (checked), 'Update' (unchecked), 'File' (unchecked), 'Stored' (unchecked), and 'Verified' (checked). There are two buttons: 'Get Boot Info' and 'Upload Firmware'. At the bottom, a text box displays 'Boot info: Valid' and 'Query result: No error'.

Code Size	115856	<input checked="" type="checkbox"/> Primary	<input type="checkbox"/> Update	<input type="checkbox"/> File
Code Checksum	D7F2843A	<input type="checkbox"/> Stored	<input checked="" type="checkbox"/> Verified	
Model Number	3101	<button>Get Boot Info</button>		
Model Variant	1			
Revision	3.10.2	<button>Upload Firmware</button>		

Boot info: Valid
Query result: No error

You can now go about whatever business you had in configuring the gateway. You can go back to the Who-Is page and click Get Device Info to verify that the updated firmware version is reported to BACnet.

Babel Buster BB3-3101 Gateway Configuration Tool v3.02

Connected: ☒ Target: ☒ 29999: Babel Buster BB3-3101 In Sync: ☒

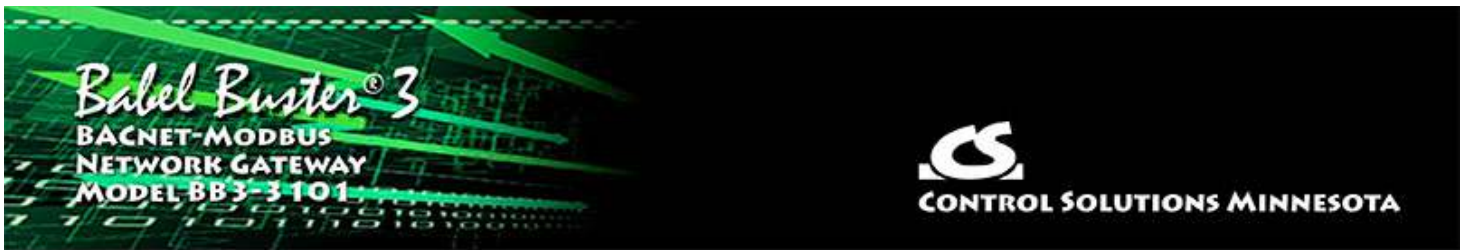
Local Port | Who-Is | Read/Write | Device | Object Map | Object Text | Object List | Modbus | Data List | Firmware

Send Who-Is | Clear Who-Is Cache | Refresh | 29999 | Targeted Who Is

Device	Net Address	Object Model	Object Name
29999	IP 192.168.1.57:0xBAC0 [...]	Babel Buster BB3-3101	Default BB3-3101

Get Device Info | Get Object List | Read All Config

Object (29999): Default BB3-3101
Model: Babel Buster BB3-3101
Firmware Rev: 3.10 Application Rev: 3.10.2
Vendor (208): Control Solutions, Inc.



Appendix A - CSV File Format

Refer to Section 3 of this user guide for examples and discussion of how to use CSV files.

A.1 Object Maps

The columns noted in the following table are used to identify and define BACnet objects in the gateway device.

Object	<p>Object - ASCII string identifying the local object type and instance, "TT-N" where N is instance 1..MAX_OBJECT, and TT is one of the following:</p> <ul style="list-style-type: none"> AI = Analog Input AO = Analog Output AV = Analog Value BI = Binary Input BO = Binary Output BV = Binary Value MI = Multi-state Input MO = Multi-state Output MV = Multi-state Value <p>Example: "AI-1" is Analog Input 1, "AO-14" is Analog Output 14, etc. (You do not need to include quote characters for this column in the CSV file.)</p>
IsModbus	T if mapped to Modbus, or F
IsGroup	T if member of packed Modbus register, or F
IsMixed	T if member of mixed type packed Modbus register, or F
IsBACnet	T if mapped to BACnet objects in another BACnet device, or F
ReadPoll	T if periodic Read, or F
WritePoll	T if periodic Write, or F
WriteDelta	T if Write on Delta, or F
DefPOR	T if default on POR enabled, or F
DefNOK	T if default on comm. fail enabled, or F
DefOOS	T if default to Out Of Service on power-up, or F
PollTime	Integer, Periodic poll time in seconds

Timeout	Integer, BACnet slave timeout in seconds
MaxQuiet	T if max. quiet enabled, or F
Scale	Real, scale factor
Offset	Real, offset for scaling
DefaultValue	Real or Integer as applicable to object type, default value
SendDelta	Real, delta threshold for Write on Delta
InitCOVincrement	Real, initial COV increment
InitCOVperiod	Integer, COV period in seconds
InitRelinquish	Real or Integer as applicable to object type, initial relinquish default
FailCount	Integer, count of comm. fails before Fault indicated
Units	Integer, Units code as defined by BACnet standard
ObjectName	Character String (must enclose in quotes if string includes comma character)
Description	Character String (must enclose in quotes if string includes comma character)

The columns noted in the following table are used to identify Modbus registers that will be polled by the gateway device, with content of the polled registers placed into (or taken from) BACnet objects in the gateway device.

ModbusRegNum	Integer, Modbus register number
ModbusRegType	RemoteRegType - ASCII string, 2-character code, representing Modbus register type: 'NO' - none '0X' - coil(s) - uses FC15 to write '1X' - discrete input '3X' - input register '4X' - holding register(s) - uses FC16 to write '0S' - coil, use FC5 to write single '4S' - holding register, use FC6 to write single
ModiconReg	Integer, Modbus register number in Modicon format (Do not include ModbusRegNum or ModbusRegType if using ModiconReg. Do not include ModiconReg if using ModbusRegNum and ModbusRegType.)
ModbusRegFormat	RemoteRegFormat - ASCII string, 4-character code, representing Modbus register format: 'NONE' - none 'SIGN' - signed integer (16-bit) 'UNSI' - unsigned integer (16-bit) 'SDBE' - signed double integer, big endian (32-bit, register pair) 'UDBE' - unsigned double integer, big endian (32-bit, register pair) 'FPBE' - floating point, big endian (register pair)

	'BBIT' - bit 'SDLE' - signed double integer, little endian (32-bit, register pair) 'UDLE' - unsigned double integer, little endian (32-bit, register pair) 'FPLE' - floating point, little endian (register pair)
ModbusMask	Integer, hexadecimal representation
ModbusFill	Integer, hexadecimal representation
ModbusSlaveId	Integer, Modbus slave address or unit number

The columns noted in the following table are used to identify BACnet objects in other BACnet devices that will be polled by the gateway device, with content of the polled objects placed into (or taken from) BACnet objects in the gateway device.

ObjDevice	Integer, BACnet device instance
ObjType	ASCII string (2 characters) identifying the local object type, where string is one of the following: AI = Analog Input AO = Analog Output AV = Analog Value BI = Binary Input BO = Binary Output BV = Binary Value MI = Multi-state Input MO = Multi-state Output MV = Multi-state Value
ObjInstance	Integer, BACnet object instance
ObjProperty	Integer, BACnet property code as defined by BACnet standard
ObjIndex	Integer, array index, -1 if no index
ObjPriority	Integer, priority 1-16 or 0 if none
ObjEncoding	Integer, object encoding (0) Null (1) Boolean (2) Unsigned Integer (3) Signed Integer (4) Real (8) Bit String (9) Enumerated
BitNumber	Integer, bit number
MacAddr	Integer (0..254), MAC address of MS/TP device if slave-only device or does not support dynamic binding (Who-Is/I-Am). Normally should be zero so that Who-Is is allowed to take care of device discovery.

Examples:

The header line for a CSV file containing only BACnet client mappings will be exported as

(exclude blanks, line breaks):

Object, IsBACnet, ReadPoll, WritePoll, WriteDelta, DefPOR, DefNOK, PollTime, Timeout, MaxQuiet, Scale, Offset, DefaultValue, SendDelta, InitRelinquish, FailCount, Units, ObjectName, Description, ObjDevice, ObjType, ObjInstance, ObjProperty, ObjIndex, ObjPriority, ObjEncoding, BitNumber, MacAddr

The header line for a CSV file containing only Modbus client mappings will be exported as (exclude blanks, line breaks):

Object, IsModbus, ReadPoll, WritePoll, PollTime, InitRelinquish, Units, ObjectName, Description, ModbusRegNum, ModbusRegType, ModbusRegFormat, ModbusSlaveId

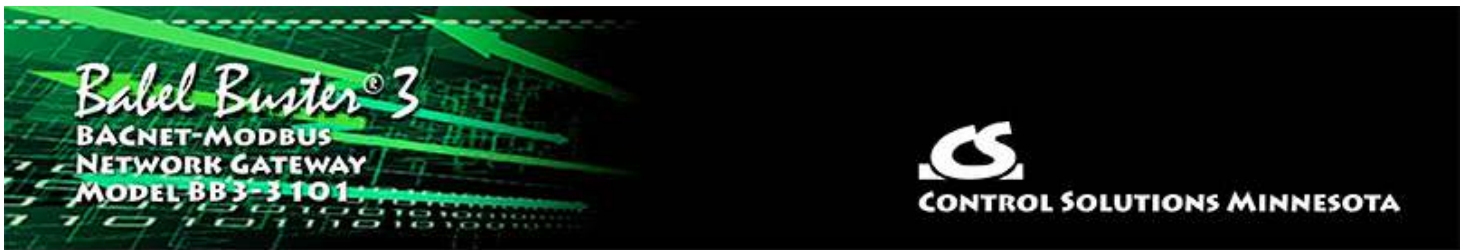
The header line for a CSV file containing both Modbus and BACnet client mappings will be exported as (exclude blanks, line breaks):

Object, IsModbus, IsBACnet, ReadPoll, WritePoll, WriteDelta, DefPOR, DefNOK, PollTime, Timeout, MaxQuiet, Scale, Offset, DefaultValue, SendDelta, InitRelinquish, FailCount, Units, ObjectName, Description, ModbusRegNum, ModbusRegType, ModbusRegFormat, ModbusSlaveId, ObjDevice, ObjType, ObjInstance, ObjProperty, ObjIndex, ObjPriority, ObjEncoding, BitNumber, MacAddr

A.2 Object Text

The columns noted in the following table are used to identify and define text strings for Binary and Multi-state objects in the gateway device.

Object	Object - ASCII string identifying the local object type and instance, "TT-N" where N is instance 1..MAX_OBJECT, and TT is one of the following: BI = Binary Input BO = Binary Output BV = Binary Value MI = Multi-state Input MO = Multi-state Output MV = Multi-state Value Example: "BI-1" is Binary Input 1, "BO-14" is Binary Output 14, etc. (You do not need to include quote characters for this column in the CSV file.)
State	For Binary objects: 0=inactive text, 1=active text For Multi-state objects: 1..N where N is maximum state configured for Multi-state objects
Text	Character String (must enclose in quotes if string includes comma character)



Appendix B - Modbus Trouble Shooting

B.1 Observing Modbus Errors, LED Indications

Refer to Section 11 of this user guide to see how error counts may be queried for specific Modbus slave devices.

The reliability code displayed for each object on the Data List page (see section 12) will indicate Modbus errors on a point by point basis. The reliability code generally indicates the nature of the problem.

Refer to Appendix J, Hardware Details, for a detailed description of what the LEDs are indicating.

B.2 Error Codes, Reliability Codes

If the Modbus slave device being queried by the gateway returns an error code, or there is simply a problem in communicating with the slave, this will be indicated by the gateway object's reliability code. When reliability code is non-zero, the fault flag is also set. Therefore, the Status Flags indication will typically be "F,T,F,F" any time the reliability code is something other than zero (zero means no errors to report).

Error codes returned by a Modbus slave device are encoded into reliability codes presented by the respective object. Reliability codes pertaining to polling of Modbus slaves (read via the Reliability property) are as follows:

- no response (64)
- crc error (65)
- exception, illegal function code (66)
- exception, illegal data address (67)
- exception, illegal data value (68)
- exception, other code returned by device, code+65, (69..79)
- configuration property fault (80)
- exception, code not recognized (81)

B.3 Auto-Reset Errors

Reliability codes will "latch" by default and require that you read the Reliability property in order to reset it to zero, assuming the problem has gone away. Once the non-zero reliability code has been read (by reading the Reliability property), it will reset to zero the next time the object is updated, provided the problem has been resolved.

Since many systems do not automatically read Reliability codes, but do automatically

respond to the Fault Status Flag associated with the non-zero reliability code, an auto-reset option is available (on the Device page). When set, reliability codes will return to zero as soon as the problem has been resolved, regardless of whether the non-zero reliability code was ever acknowledged by reading it.

Many systems will report an object as "offline" because its fault status bit is set. It is not actually offline, and is in fact communicating just fine trying to tell you that there is a problem. But many front end systems don't recognize this and blindly claim "offline" as a result of the fault bit in the status property. If you are having this issue as the result of communication errors on the Modbus side, try setting the Auto-Reset Errors option here. This only applies if you have "Map Modbus Object" checked on one or more Object Map pages, and the gateway is configured to be Modbus master.

B.4 Trouble Shooting Tips

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

No-response errors:

- Check to see that communication parameters are correct (baud rate, etc).
- Check to see that the slave address matches.
- Check to see that Pre-Delay is at least 50 mS.
- Check wiring and power.
- Check for reversed polarity on RS485 lines. If uncertain, just try swapping them.
- Check to see that slave device is enabled for Modbus communication (many devices default to disabled)

CRC errors:

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

Exception errors:

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

B.5 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.

6	Slave Device Busy	The slave is engaged in processing a long-duration command. The master should try again later.
10 (hex 0A)	Gateway Path Unavailable	Gateway could not establish communication with target device.
11 (hex 0B)	Gateway Target Device Failed to Respond	Specialized use in conjunction with gateways, indicates no response was received from the target device.
17 (hex 11)	Gateway Target Device Failed to Respond	No response from slave, request timed out.

Modicon convention notation for Modbus registers

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

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

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

Valid address ranges as originally defined for Modbus were 0 to 9999 for each of the above register types. Valid ranges allowed in the current specification are 0 to 65,535. The address range applies to each type of register, and one needs to look at the function code in the Modbus message packet to determine what register type is being referenced. The Modicon convention uses the first digit of a register reference to identify the register type.

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

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

Translating references to addresses, reference 40001 selects the holding register at address 0000, most often referred to as holding register number 1. The reference 40001 will appear in documentation using Modicon notation, but Babel Buster gateways require specifying "holding register" and entering that register number as just "1".

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

0x = Coil = 000001-065535

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

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

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

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

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

What does notation like 40001:7 mean?

This is a commonly used notation for referencing individual bits in a register. This particular example, 40001:7, references (Modicon) register 40001, bit 7. Bits are generally numbered starting at bit 0, which is the least significant or right most bit in the field of 16 bits found in a Modbus register.

How do I read individual bits in a register?

Documentation tends to be slightly different for every Modbus device. But if your device packs multiple bits into a single holding register, the documentation will note up to 16 different items found at the same register number or address. The bits may be identified with "Bn" or "Dn" or just "bit n". Most of the time, the least significant bit will be called bit 0 and the most significant will be bit 15. It is possible you could find reference to bit 1 through bit 16, in which case just subtract one from the number to reference the table below.

You cannot read just one bit from a holding register. There is no way to do that - Modbus protocol simply does not provide that function. You must read all 16 bits, and then test the individual bit you are interested in for true or false (1 or 0). Babel Buster gateways provide an automatic way of doing that by including a "mask" in each register map or rule. Each time the register is read, the mask will be logically AND-ed with the data from the register, and the result will be right justified to yield a 1 or 0 based on whether the selected bit was 1 or 0. Babel Buster gateways provide optimization when successive read maps or rules are selecting different bits from the same register. The Modbus register will be read from the slave once, and the 16-bit data will be shared with successive maps or rules, with each map or rule selecting its bit of interest.

The bit mask shown in the expanded form of the Babel Buster RTU read map is a 4 digit

hexadecimal (16 bit) value used to mask out one or more bits in a register. The selected bits will be right justified, so a single bit regardless of where positioned in the source register will be stored locally as 0 or 1. The hex bit mask values would be as follows:

B0/D0/bit 0 mask = 0001
B1/D1/bit 1 mask = 0002
B2/D2/bit 2 mask = 0004
B3/D3/bit 3 mask = 0008
B4/D4/bit 4 mask = 0010
B5/D5/bit 5 mask = 0020
B6/D6/bit 6 mask = 0040
B7/D7/bit 7 mask = 0080
B8/D8/bit 8 mask = 0100
B9/D9/bit 9 mask = 0200
B10/D10/bit 10 mask = 0400
B11/D11/bit 11 mask = 0800
B12/D12/bit 12 mask = 1000
B13/D13/bit 13 mask = 2000
B14/D14/bit 14 mask = 4000
B15/D15/bit 15 mask = 8000

Some Modbus devices also back two 8-bit values into a single 16-bit register. The two values will typically be documented as "high byte" and "low byte" or simply have "H" and "L" indicated. If you run into this scenario, the masking for bytes is as follows:

High byte mask = FF00
Low byte mask = 00FF

When the mask value in a Babel Buster gateway is more than just one bit, the mask is still logically AND-ed with the data from the Modbus slave, and the entire resulting value is right justified to produce an integer value of less than the original bit width of the original register.

There have been a few instances of documenting packed bits in a 32-bit register. Although Modbus protocol is strictly 16-bit registers, some implementations force you to read pairs of registers. If your device documents 32 packed bits, then you would insert 0000 in front of each mask above, and the remainder of the list would be as follows:

B16/D16/bit 16 mask = 00010000
B17/D17/bit 17 mask = 00020000
B18/D18/bit 18 mask = 00040000
B19/D19/bit 19 mask = 00080000
B20/D20/bit 20 mask = 00100000
B21/D21/bit 21 mask = 00200000
B22/D22/bit 22 mask = 00400000
B23/D23/bit 23 mask = 00800000
B24/D24/bit 24 mask = 01000000
B25/D25/bit 25 mask = 02000000
B26/D26/bit 26 mask = 04000000
B27/D27/bit 27 mask = 08000000
B28/D28/bit 28 mask = 10000000
B29/D29/bit 29 mask = 20000000
B30/D30/bit 30 mask = 40000000

B31/D31/bit 31 mask = 80000000

Deciphering Modbus Documentation

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

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

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

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

Can I put 2 gateways on the same Modbus network?

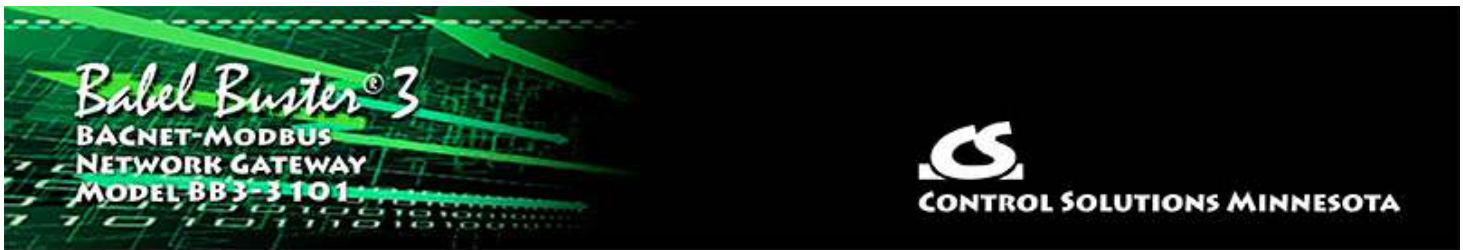
You can not have more than one Master on a Modbus RTU (RS-485) network. Therefore, if the gateway is to be configured as the Master, you can only have 1 gateway. You cannot use multiple gateways to read more points from the same Modbus slave device.

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 C - BACnet Trouble Shooting

C.1 Observing BACnet Errors, LED Indicators

The reliability code displayed for each object on the Data List page (see section 12) will indicate BACnet client errors on a point by point basis. The reliability code generally indicates the nature of the problem.

Refer to Appendix J, Hardware Details, for a detailed description of what the LEDs are indicating.

C.2 Error Codes, Reliability Codes

If the BACnet slave device being queried by the gateway returns an error code, or there is simply a problem in communicating with the slave, this will be indicated by the gateway object's reliability code. When reliability code is non-zero, the fault flag is also set. Therefore, the Status Flags indication will typically be "F,T,F,F" any time the reliability code is something other than zero (zero means no errors to report).

Error codes returned by a BACnet slave device are encoded into reliability codes presented by the respective object. Reliability codes pertaining to polling of BACnet slaves (read via the Reliability property) are as follows:

- BACnet client read/write timeout (82)

- BACnet client received error response from slave (83)

If the object's reliability indicates code 83, meaning an error code was returned by a remote BACnet device, the actual code returned can be learned by reading the local object's properties 830 (class) and 831 (code). If an error exists and a reliability code other than 83 is indicated, reading properties 830-831 can also return class 64 (proprietary) and the reliability code+200 (i.e. Modbus illegal function code exception would be class 64, code 266).

C.3 Auto-Reset Errors

Reliability codes will "latch" by default and require that you read the Reliability property in order to reset it to zero, assuming the problem has gone away. Once the non-zero reliability code has been read (by reading the Reliability property), it will reset to zero the next time the object is updated, provided the problem has been resolved.

Since many systems do not automatically read Reliability codes, but do automatically respond to the Fault Status Flag associated with the non-zero reliability code, an auto-reset option is available (on the Device page). When set, reliability codes will return to

zero as soon as the problem has been resolved, regardless of whether the non-zero reliability code was ever acknowledged by reading it.

Many systems will report an object as "offline" because its fault status bit is set. It is not actually offline, and is in fact communicating just fine trying to tell you that there is a problem. But many front end systems don't recognize this and blindly claim "offline" as a result of the fault bit in the status property. If you are having this issue as the result of communication errors related to polling other BACnet devices, try setting the Auto-Reset Errors option here. This only applies if you have "Map BACnet Object" checked on one or more Object Map pages.

C.4 Trouble Shooting Tips, USB Adapter

(a) Is the yellow LED next to the USB cable lit? If not, the PC does not recognize the adapter. Verify installation. Verify that MTX002 shows up as a Port in the Device Manager on your PC. The yellow LED next to the USB cable must be lit before continuing.

(b) Open the configuration tool. Select the port number indicated in the Device Manager. Select a known baud rate, max master setting, and unused Mac address. Click 'Connect'. Does the log window show "USB is responding"? If not, close the configuration tool, then disconnect and reconnect the USB cable to force a hard reset of the adapter. Retry the 'Connect'. If you get an error message about the COM port, see that the supercom.dll is found in the same directory as the configuration tool, and recheck the port number on the PC's Device Manager. You must get "USB is responding" followed by "Ready" or "Auto connected" before continuing.

(c) Is the Token LED on the USB adapter flickering? If not, either there is nothing on the MS/TP network or the network has not recognized the adapter's reply to poll for master. If the network was online before the adapter was connected, the adapter will wait for the token indefinitely. If no other device on the network grants the token, the adapter will never begin communicating. Failure to get in sync, assuming all devices appear otherwise functional, is most often caused by either a mismatch in baud rate, mismatch in max master setting, or unusable choice of mac address.

(d) Is the PFM (poll for master) LED on the USB adapter nearly solid on? This means the adapter is busy sending out polls for master but not seeing any response. This can be caused by nothing being on the network, a mismatch in baud rate, or other low level communication problems. The normal LED activity when the adapter is connected and in sync with the network will be continuous flickering of the Token LED, usually but not always some periodic flickering of the PFM LED, and occasional flashing of the Data LED. The error LED could also flicker once in a while, but hopefully not often.

NOTE: The two "not really connected" indications you are most likely to see are (i) None of the LEDs next to the terminal block on the adapter ever flash; (ii) The PFM (yellow) LED is always flickering (nearly constant on). It is a somewhat random coincidence whether other devices on the network or the adapter gets ahead in polling. Devices on the MS/TP network always wait for the right opportunity to respond. If they never get that opportunity due to being out of sync as a result of incorrect baud rate, etc, they will appear to be "offline". Meanwhile, a device polling to look for other devices will constantly just poll because it is not getting any response. Because the device doing the constant polling will generate enough traffic on the line to make the line look busy to other devices, the other devices will never come online.

TROUBLE SHOOTING TIP: If LEDs are not flashing on the MTX002 and/or you cannot discover any device on the Who-Is page, try disconnecting the MTX002 and restarting the configuration tool. Then, instead of selecting a baud rate and clicking Connect, just click Auto Connect. This will search for MS/TP on all possible baud rates, automatically determine the network's Max Master setting, and automatically assign an unused MAC address to the MTX002. When the MTX002 has found MS/TP (after a minute or two), it will say "Auto connected" where it normally just says "Ready". Discovering that the gateway is not set for the baud rate somebody thought it was is a common technical support finding.

C.5 Trouble Shooting Tips, Gateway Behavior

The above discussion is focused on getting the USB to MS/TP adapter connected. However, the situation involving one device being out of sync with the network can happen with any device on the network. This is the situation that creates the appearance of a single device refusing to go online, or a single device taking down the network. If a gateway appears to work normally when connected to only the USB adapter, but either refuses to go online or appears to take down the network when connected to a running network, its port settings are out of sync with the rest of the network. It will take a few seconds for the network to re-sync any time a new device is added, but if the network does not return to normal token passing within a reasonable time, the newly added device needs attention.

Trouble shooting the BB3-3101 MS/TP Connection

(a) Does the device have power? A blue LED inside the case, visible through the air vent slots, will be on if there is power present.

(b) Are any LEDs flickering on the front of the BB3-3101? If not, look through the vent holes to see if a red LED near the blue LED is on or blinking. If so, see System Fault Indications below. There should be a green LED near the blue LED that is mostly on, flashing off briefly about once every two seconds.

(c) The Token yellow/green LED on the front of the BB3-3101 should be flickering some color rather constantly. Green indicates token passing, yellow indicates poll for master. Constant yellow (never green) means no other devices are responding. Constant off (both colors) means other devices are generating traffic on the network, but it is not being recognized by the BB3-3101 as anything it can respond to. No token passing activity means either there are no other devices on the network, or there is a low level communications problem, namely one or more communication parameters are mismatched. There could also be a wiring problem. Check for reversed polarity on the data lines. If uncertain, just try swapping them – no physical harm will be done.

C.6 System Fault Indications

The red LED visible inside the BB3-3101 case, viewed through the vent slots, near the blue power LED, is the system fault indicator. It will be on during initial power-up boot mode operation, but should otherwise be off.

Following a device reinitialize upon command from BACnet, the system fault LED will flicker at a very rapid rate for approximately 10 seconds, then turn off (or indicate system fault). The rapid flicker is verification that the restart has occurred.

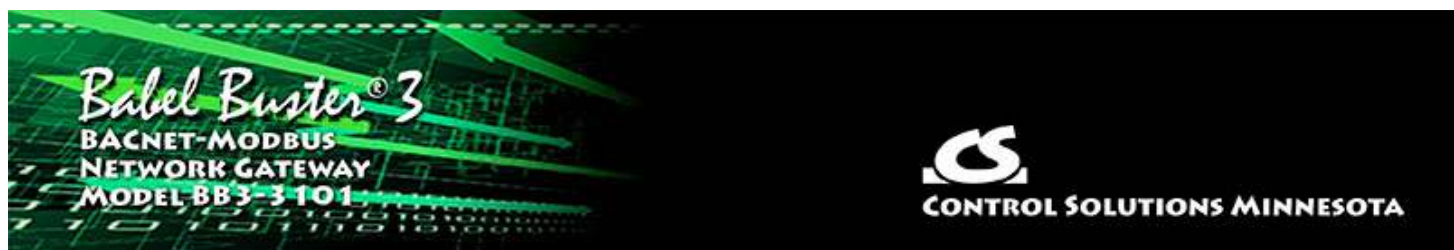
During normal operation, a watchdog timer is always running to force a soft restart the system in the event of a software hang. Should the system restart as a result of watchdog timeout, the system fault LED will flicker at a very rapid rate for approximately 10 seconds. An intentional software hang followed by watchdog timeout is used to force a restart upon receiving a restart command from BACnet. Therefore the indications are the same. However, if the restart indication is observed without sending the restart command from BACnet, please notify technical support.

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

Fault codes are as follows:

- (1) Processor hard fault
- (2) Unhandled IRQ
- (3) Stack overflow
- (4) Other OS fault
- (5) Resource allocation fail
- (6) System configuration checksum error
- (7) EEPROM read/write fail
- (8) Flash read/write fail
- (9) Application checksum error
- (10) Invalid application image
- (11) Unknown fault

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



Appendix D - Modbus Slave Register Map

The BB3-3101 is put into slave mode by checking the Slave check box on the Modbus page of the configuration tool, and then clicking Write Device. In slave mode, the object maps are only used to define polling of other BACnet devices. You do not create any Modbus maps. The Modbus registers are pre-defined and fixed. You only need to map BACnet objects so that BACnet data is exchanged between the fixed Modbus registers and the remote BACnet devices. The fixed Modbus registers are really just a means of accessing the BACnet objects in the BB3-3101, and the means of access amounts to calculating a specific register number that points to one of the BACnet objects. The Modbus registers can only access the Present Value property of the BACnet objects that are maintained by the BB3-3101.

The BB3-3101 may have a total of 1024 BACnet objects; however, as a Modbus slave, you can only address a maximum of 500 of any one type of object. So, for example, if you wanted to monitor 1000 BACnet analog values, you might use 500 Analog Inputs and 500 Analog Value objects. This limitation is due to the way in which Modbus register numbers are calculated as derived from BACnet object type and instance.

D.1 Modbus Slave Map - Calculated Registers

Modbus register numbers for accessing data objects in the BB3-3101 are calculated. The register number for binary and multi-state objects is $R = T * 1000 + I$ where T is the BACnet Object Type, and I is the instance (R is the resulting register number). The register number for analog objects, because they must be read as a register pair, is $R = T * 1000 + I * 2 - 1$ (R is the first register number in the pair). Register numbers start at 1. To create a raw address, subtract 1 from the register number.

Analog objects should be read as input registers or holding registers, and can only be written as holding registers. Binary and multi-state objects can be read as any register type (coil, discrete, input register, holding register), and can be written as coil or holding register.

Analog objects are always floating point data read as a register pair with most significant register first unless the Swap box is checked on the Modbus page in the configuration tool. Attempting to read or write an Analog object as a single register will produce an error.

Object types that may be used in Modbus register number calculation (as "T") are:

- 0 - Analog Input
- 1 - Analog Output
- 2 - Analog Value
- 3 - Binary Input

- 4 - Binary Output
- 5 - Binary Value
- 13 - Multistate Input
- 14 - Multistate Output
- 19 - Multistate Value

The reliability code of an object may be read as an unsigned integer by adding 20,000 to the calculated register number.

The following function codes are used by BB3-3101 as Modbus master, and are also recognized by BB3-3101 when functioning as a slave.

- 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

You can read or write Binary or Multistate objects as Modbus Coils. A coil on/off will correspond to Binary states on/off. A coil state of "off" will be a Multistate value of 1, and a coil state of "on" will be a Multistate value of 2. Accessing Multistate objects as coils is not particularly useful but is permitted.

You can read Binary or Multistate objects as Modbus Discrete Inputs. The values will be as noted for Coils.

Due to the fact that Analog objects map to register pairs, you cannot access Analog objects as coils or discrete inputs.

You can read Analog objects as Modbus Input Registers. You can read or write Analog objects as Modbus Holding Registers. In both cases, you must read or write register pairs for each object, and the data will be IEEE 754 floating point.

You can read Binary or Multistate objects as Modbus Input Registers but must read these as 16-bit integer values.

You can read or write Binary or Multistate objects as Modbus Holding Registers and must read or write these objects as 16-bit integer values.

D.2 Modbus Slave Map - Alternate Map

Alternate Map applies only if you are using the BB3-3101 as a Modbus slave. Although Modbus protocol specification provides for up to 65,535 holding registers (and 65,535 of each type of register), some older Modbus masters can only access 9,999 registers and their holding register range is usually documented as 40001 through 49999 (also known as Modicon notation). If your Modbus master cannot access the full range of registers, you will need to use the alternate mapping discussed here.

Enable the alternate Modbus slave map by checking the Alt Map box as illustrated below. This box can only be checked if you have also checked Slave.

Babel Buster BB3-3101 Gateway Configuration Tool v3.01

Connected: ☒ Target: ☒ 29999: Babel Buster BB3-3101 In Sync: ☒

Local Port | Who-Is | Read/Write | Device | Object Map | Object Text | Object List | Modbus | Data List | Firmware

Device Number: 1 Gateway is Modbus: ☒ Master ☒ Slave ☒ Alt Map
Total Message Count: 4861 ☐ Swap Registers if Slave
No-response Count: 0
Exception Count: 0
CRC Error Count: 0

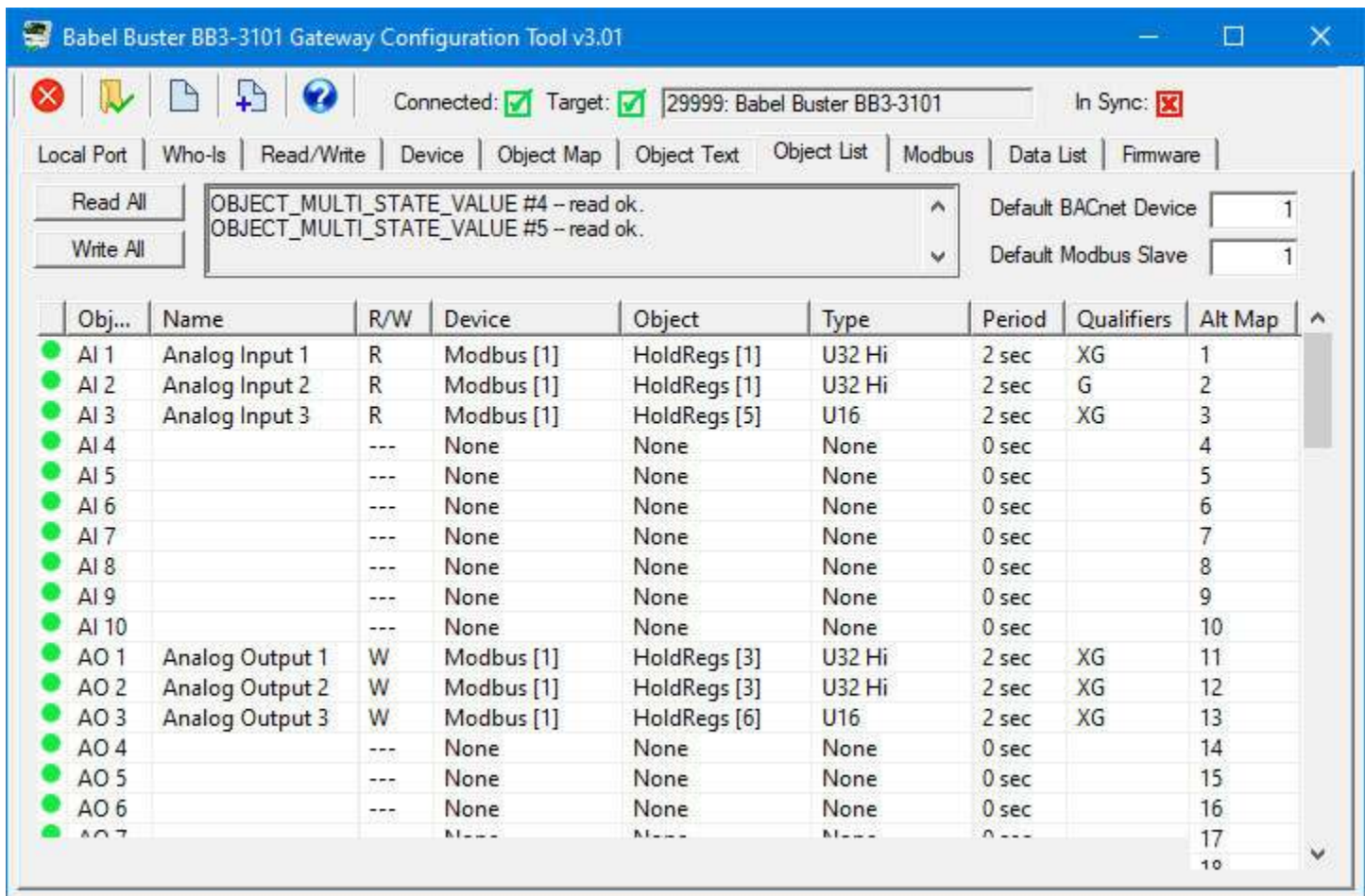
Diagnostic Modbus Read/Write

Register Number 1..N: 400 Data:
Register Type: Holding Register (fc 16)
Register Format: Signed 16-bit

Modbus Port Baud Rate: 19200
Modbus Character Format: 8 Data, No Parity, 1 Stop
Timeout when Master (S): 2.0
Address when Slave: 1
Command Priority: 9
Disable Multistate Offset: ☐

System status updated.

If you are using the BB3-3101 as a Modbus slave, then it would have to mean that all of your object maps are reading (or writing) MS/TP slave devices. Once you have allocated the desired number of objects on the Device tab, and configured your maps on the Object Map tab, go to the Object List view. Scroll the screen left, or expand the view by dragging the right border to stretch the screen. This will reveal the Alt Map column. This column is simply provided as a reference list and assigns a sequential number to each object map in the list without respect to type.



You will use the Alt Map reference number to calculate your Modbus register numbers. The Alt Map method only provides access to the first 500 objects on the list. In this mode, you are limited to accessing only 500 objects total. This limit will keep addressing within the confines of Modicon register numbering.

If your Modbus master does not make reference to the first holding register at 40001, but talks about address 0, or register 1, these all point to the same first holding register. The first holding register is address 0 on the wire, but is often documented as register 1, and in older documentation (or new documentation following old Modicon conventions), the first holding register will be listed as 40001.

Register number calculations are performed as noted in the following table. Since the restriction on number of accessible registers is most often associated with older equipment, we will use Modicon notation here since that same older equipment most likely uses this notation (i.e. 40001 is the first holding register).

You may read any of the BACnet objects mapped using any of the following data formats: Integer (16-bit), Unsigned integer (16-bit), Double signed integer (32-bit), Floating point (IEEE-754 32-bit). The different formats are read by using different address ranges to access the same object. Calculate addresses as illustrated in the following table, where M is the number in the Alt Map column above.

Alt Map	Signed Integer	Unsigned Integer	Double Integer	Floating Point	Reliability Code
Formula	$M + 40000$	$M + 41000$	$M * 2 + 42000$	$M * 2 + 43000$	$M + 44000$

Object 1 (M=1)	40001	41001	42002	43002	44001
Object 2 (M=2)	40002	41002	42004	43004	44002
Object 3 (M=3)	40003	41003	42006	43006	44003

The access rules are different when using ALT Mapping.

You can read or write any object type referenced as any Modbus register type when accessing signed or unsigned integer (16-bit). When accessing the object as double integer (32-bit) or floating point (also 32-bit), you can only access these as Modbus Input or Holding Registers, and must access these as register pairs.

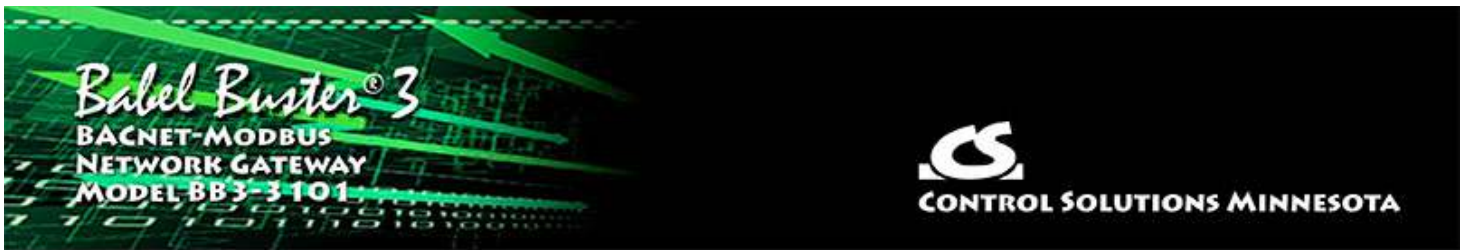
To illustrate the meaning of the above paragraph, you can read or write an Analog object as a Modbus Coil when using ALT mapping and referencing an integer register number. This is permitted, although not particularly useful as the only possible values you can write to the Analog object are zero and one when accessing it as a coil. The more practical use of ALT mapping for accessing an Analog object would be to access that Analog object as a 16-bit integer value (single Modbus Holding register), and this can be useful for a PLC that does not do well with floating point values.

D.3 Relinquishing Commandable Objects

The BB3-3101 can be configured as a Modbus slave and BACnet client, thus giving some other Modbus device like a PLC control over BACnet output objects via the BB3-3101 gateway. Because Modbus knows nothing about “relinquish” (or command priorities), there is no way to write to a Modbus register such that the BACnet object will relinquish a commandable priority level. When the BB3-3101 is acting as a Modbus slave, writing to holding register addresses that map to Output objects will result in the Modbus command priority being set. But there is no direct way to relinquish that command priority.

The remedy to this problem is that phantom registers have been added which are recognized only by the Modbus slave. The phantom registers are accessible by calculating the slave register number of the Output object as defined above, and then adding 40,000 to the holding register number. If using the “ALT” mapping, add 5,000 to the ALT map register number instead. Writing any value to this register number will relinquish the Modbus command priority for that object. An exception will be returned if you try to relinquish a non-commandable object.

In order to give Modbus full control over a commandable object in some other BACnet device, it is necessary to program your Modbus master to write to one holding register to set the output, and write to a different register (the phantom register) to relinquish the output. The addresses you use will be derived from the local Output object number you choose as the local object in the gateway. To cause that object to then be transmitted to the remote BACnet device, you need to set up a client write map as illustrated in section 8.3.



Appendix E - BACnet Object Properties

E.1 Data Object Properties (Analog, Binary, Multistate)

The following properties are found in the Analog, Binary, and Multi-state types of Input, Output, and Value objects. Some properties apply only to certain object types as noted where applicable.

Property	Encoding
Object_Identifier (75)	BACnetObjectIdentifier
Object_Name (77) (W)	CharacterString "Analog Input <i>n</i> "
Object_Description (28) (W)	Character String Same as Object_Name, is only alias for Object_Name
Object_Type (79)	BACnetObjectType ENUMERATED: analog-input (0) analog-output (1) analog-value (2) binary-input (3) binary-output (4) binary-value (5) device (8) multi-state-input (13) multi-state-output (14) multi-state-value (19)
Present_Value (85) (W)	REAL (analog objects) ENUMERATED (binary objects) Unsigned (multi-state objects) (no index) (priority required when writing commandable objects) (input objects writeable only when out of service)
Status_Flags (111)	BACnetStatusFlags BIT STRING: fault(1), out-of-service(3)
Event_State (36)	BACnetEventState ENUMERATED: normal(0), fault(1)

Reliability (103)	BACnetReliability ENUMERATED: normal(0) <i>Vendor specific:</i> no response (64) crc error (65) exception, illegal function code (66) exception, illegal data address (67) exception, illegal data value (68) exception, code+65, rarely used (69..79) configuration property fault (80) exception, code not recognized (81) BACnet client read/write timeout (82) BACnet client received error response from slave (83)
Out_Of_Service (81) (W)	BOOLEAN
COV_Increment (22) (W)	REAL (analog objects only)
COV_Period (180) (W)	Unsigned
Priority_Array (87)	BACnetPriorityArray (commandable objects only) SEQUENCE SIZE (16) OF BACnetPriorityValue REAL (each element, analog output objects) ENUMERATED (each element, binary output objects) Unsigned (each element, multi-state output objects)
Relinquish_Default (104) (W)	REAL (analog objects) ENUMERATED (binary objects) Unsigned (multi-state objects)
Polarity (84)	BACnetPolarity (binary objects only) ENUMERATED: normal(0)
Number_Of_States (74)	Unsigned (multi-state objects only)
Units (117)	BACnetEngineeringUnits (analog objects only)
Active_Text (4)	CharacterString (binary objects only)
Inactive_Text (46)	CharacterString (binary objects only)
State_Text (110)	CharacterString (multistate objects only)
<i>Vendor Specific Object Properties:</i>	

Object_Map_Usage (801) (W)	BIT STRING: (0) object is mapped to Modbus (1) object included in packed group (2) object included in mixed type packed group (3) reserved (4) set default on power-up (5) set default on comm. fail (6) read periodic (7) write periodic (8) write on delta (9) enable max quiet (10) object is mapped to BACnet
Poll_Period (802) (W)	Unsigned Modbus poll/update time in seconds
Note: Properties 803 through 808 can only be written when bit 0 of the Object_Map_Usage bit string is set. Properties 819 through 826 can only be written when bit 10 of the Object_Map_Usage bit string is set. Bits 0 and 10 of the Object_Map_Usage bit string cannot both be set at the same time (doing so will cause unpredictable behavior).	
Register_Number (803) (W)	Unsigned Modbus register number 1..65535
Register_Type (804) (W)	ENUMERATED: none(0), coils(1), discrete-input(2), input-register(3), holding-registers(4), coil (5), (write single, FC5) holding-register (6) (write single, FC6)
Register_Format (805) (W)	ENUMERATED: none(0), signed-integer(1), unsigned-integer(2), double-signed-integer(3), double-unsigned-integer(4), floating-point(5), bit(6), double-signed-integer-swapped(7), double-unsigned-integer-swapped(8), floating-point-swapped(9)
Modbus_Slave_ID (806) (W)	ENUMERATED: 1..MAX_RTU_DEVICE
Register_Bit_Mask (807) (W)	Unsigned
Register_Bit_Fill (808) (W)	Unsigned

Slope (809) (W)	REAL $\text{BACnet} = \text{Modbus} * \text{slope} + \text{intercept}$ $\text{Modbus} = (\text{BACnet} - \text{intercept}) / \text{slope}$
Intercept (810) (W)	REAL
Default_Value (811) (W)	REAL (analog objects) ENUMERATED (binary objects) Unsigned (multi-state objects)
Max_Quiet_Time (812) (W)	Unsigned
BACnet_Slave_Timeout (813) (W)	Unsigned
Max_Read_Fail_Count (814) (W)	Unsigned
Send_On_Delta (815) (W)	REAL (analog objects only)
Initial_COV_Increment (816) (W)	REAL (analog objects only)
Initial_COV_Period (817) (W)	Unsigned
Initial_Relinquish_Default (818) (W)	REAL (analog objects) ENUMERATED (binary objects) Unsigned (multi-state objects)
BACnet_Slave_Device (819) (W)	Unsigned
Slave_Object_Type (820) (W)	ENUMERATED
Slave_Object_Instance (821) (W)	Unsigned
Slave_Index (822) (W)	Unsigned (value is index+1, 0=no index)
Slave_Priority (823) (W)	ENUMERATED (1..16)
Slave_Property_Type (824) (W)	ENUMERATED (see BACnet protocol specification for type codes)
Slave_Data-Encoding (825) (W)	ENUMERATED (0) Null, (1) Boolean, (2) Unsigned integer, (3) Signed Integer, (4) Real (floating point), (8) Bit string, (9) Enumerated

Slave _Bit_Position (826) (W)	ENUMERATED
Error_Class (830)	Unsigned
Error_Code (831)	Unsigned
Config_Write_Command (832) (W)	ENUMERATED (1) write config to EEPROM

E.2 Device Object Properties

The following properties are found in the Device object of the BB3-3101. In addition to standard Device properties, configuration properties that apply at a system level to the whole device are included here.

Property	Encoding
Object_Identifier (75)	BACnetObjectIdentifier
Object_Name (77)	CharacterString
Object_Type (79)	BACnetObjectType ENUMERATED: device (8)
System_Status (112)	BACnetDeviceStatus
Vendor_Name (121)	CharacterString
Vendor_Identifier (120)	Unsigned16 (should always return 208)
Model_Name (70)	CharacterString
Firmware_Revision (44)	CharacterString
Application_Software_Version (12)	CharacterString
Protocol_Version (98)	Unsigned
Protocol_Revision (139)	Unsigned
Protocol_Services_Supported (97)	BACnetServicesSupported
Protocol_Object_Types_Supported (96)	BACnetObjectTypesSupported
Object_List (76)	BACnetARRAY[N] of BACnetObjectIdentifier
Max_APDU_Length_Accepted (62)	Unsigned
Segmentation_Supported (107)	BACnetSegmentation
APDU_Timeout (11)	Unsigned
Number_Of_APDU_Retries (73)	Unsigned

Device_Address_Binding (30)	List of BACnetAddressBinding
Database_Revision (155)	Unsigned
Description (28)	CharacterString
Location (58)	CharacterString
Max_Segments_Accepted (167)	Unsigned
APDU_Segment_Timeout (10)	Unsigned
Active_COV_Subscriptions (152)	BACnetLIST of BACnetCOVSubscription
Last_Restart_Reason (196)	BACnetRestartReason
Max_Master (64)	Unsigned (0..127)
Max_Info_Frames (63)	Unsigned
Restart_Notification_Recipients (202)	BACnetLIST of BACnetRecipient
Time_Of_Device_Restart (203)	BACnetTimeStamp
<i>Vendor Specific Object Properties:</i>	
RTU_Port_Baud_Rate (851) (W)	ENUMERATED: (1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200)
RTU_Character_Forma t(852) (W)	BITSTRING: (0) double registers are swapped (1) gateway is slave (2) enable parity (3) set odd parity (even otherwise if parity enabled) (4) enable two stop bits (no parity)
RTU_Slave_Timeout (853) (W)	Unsigned Index: 1..MAX_RTU_DEVICE
RTU_Slave_Address (854) (W)	ENUMERATED (1..247)
RTU_Pre_Delay (855) (W)	Unsigned
Max_State_Per_Multistate (856) (W)	Unsigned
RTU_Message_Count (861) (W*)	Unsigned Index: 1..MAX_RTU_DEVICE
RTU_Exception_Count (862) (W*)	Unsigned Index: 1..MAX_RTU_DEVICE
RTU_Bad_CRC_Count (863) (W*)	Unsigned Index: 1..MAX_RTU_DEVICE
RTU_No_Response_Count (864) (W*)	Unsigned Index: 1..MAX_RTU_DEVICE

W* -- Registers are "writeable" for purposes of resetting error counts. Writing any value will reset count to zero. Total message count will stop at 65535. Error counts will stop at 255. Counting will resume when reset to zero.

RTU_Error_Scan (865) (W)	Unsigned Index: 1..MAX_RTU_DEVICE (index is starting point to scan for next device with errors) (returns next index and error count, or zero) (write index to reset errors for that device, or zero to reset all)
RTU_Raw_Command (871) (W)	OCTET_STRING (write raw Modbus packet to send, read reply)
MSTP_Port_Baud_Rate (1201) (W)	ENUMERATED: (9600, 19200, 38400, 76800) NOTE: Changes to port settings will take effect only after a COLDSTART or WARMSTART command is issued.
MSTP_Station_ID (1202) (W)	Unsigned
Enable_Auto_Reset (1203) (W)	BOOLEAN (True if reliability codes should auto return to zero)
Reinit_Password (1205) (WO)	CharacterString Note: This property is write-only, and is only writeable with unlock sequence.
Password_Unlock (1206) (W)	Unsigned Note: Must write device's own device instance to this property to unlock password for writing, and next property written must be 1205. Any other sequence will reset and lock out password write.
Device_Allocation (1251) (W)	Unsigned Index: Use object type analog-input (0) analog-output (1) analog-value (2) binary-input (3) binary-output (4) binary-value (5) multi-state-input (13) multi-state-output (14) multi-state-value (19) max-state-per-multistate(128) (value is count allocated of objects of type indexed) (attempting to write a count that is not acceptable will return error class Services, code Invalid Configuration Data) NOTE: After writing all 10 object allocation counts, a WARMSTART must be issued. All object maps will be

cleared.



Appendix F - BACnet Codes

F.1 BACnet Object Property Codes

BACnet property type codes may be found in your copy of the BACnet protocol specification, ANSI/ASHRAE Standard 135. That document is copyrighted, but the C enumeration shown below for reference is taken from open source code available under GPL at <http://sourceforge.net>, and provides essentially the same information (copyrighted by Steve Karg, licensed under GPL as noted at <http://sourceforge.net>).

```
typedef enum {
    PROP_ACKED_TRANSITIONS = 0,
    PROP_ACK_REQUIRED = 1,
    PROP_ACTION = 2,
    PROP_ACTION_TEXT = 3,
    PROP_ACTIVE_TEXT = 4,
    PROP_ACTIVE_VT_SESSIONS = 5,
    PROP_ALARM_VALUE = 6,
    PROP_ALARM_VALUES = 7,
    PROP_ALL = 8,
    PROP_ALL_WRITES_SUCCESSFUL = 9,
    PROP_APDU_SEGMENT_TIMEOUT = 10,
    PROP_APDU_TIMEOUT = 11,
    PROP_APPLICATION_SOFTWARE_VERSION = 12,
    PROP_ARCHIVE = 13,
    PROP_BIAS = 14,
    PROP_CHANGE_OF_STATE_COUNT = 15,
    PROP_CHANGE_OF_STATE_TIME = 16,
    PROP_NOTIFICATION_CLASS = 17,
    PROP_BLANK_1 = 18,
    PROP_CONTROLLED_VARIABLE_REFERENCE = 19,
    PROP_CONTROLLED_VARIABLE_UNITS = 20,
    PROP_CONTROLLED_VARIABLE_VALUE = 21,
    PROP_COV_INCREMENT = 22,
    PROP_DATE_LIST = 23,
    PROP_DAYLIGHT_SAVINGS_STATUS = 24,
    PROP_DEADBAND = 25,
    PROP_DERIVATIVE_CONSTANT = 26,
    PROP_DERIVATIVE_CONSTANT_UNITS = 27,
    PROP_DESCRIPTION = 28,
    PROP_DESCRIPTION_OF_HALT = 29,
    PROP_DEVICE_ADDRESS_BINDING = 30,
    PROP_DEVICE_TYPE = 31,
    PROP_EFFECTIVE_PERIOD = 32,
    PROP_ELAPSED_ACTIVE_TIME = 33,
    PROP_ERROR_LIMIT = 34,
    PROP_EVENT_ENABLE = 35,
    PROP_EVENT_STATE = 36,
    PROP_EVENT_TYPE = 37,
    PROP_EXCEPTION_SCHEDULE = 38,
```

```
PROP_FAULT_VALUES = 39,  
PROP_FEEDBACK_VALUE = 40,  
PROP_FILE_ACCESS_METHOD = 41,  
PROP_FILE_SIZE = 42,  
PROP_FILE_TYPE = 43,  
PROP_FIRMWARE_REVISION = 44,  
PROP_HIGH_LIMIT = 45,  
PROP_INACTIVE_TEXT = 46,  
PROP_IN_PROCESS = 47,  
PROP_INSTANCE_OF = 48,  
PROP_INTEGRAL_CONSTANT = 49,  
PROP_INTEGRAL_CONSTANT_UNITS = 50,  
PROP_ISSUE_CONFIRMED_NOTIFICATIONS = 51,  
PROP_LIMIT_ENABLE = 52,  
PROP_LIST_OF_GROUP_MEMBERS = 53,  
PROP_LIST_OF_OBJECT_PROPERTY_REFERENCES = 54,  
PROP_LIST_OF_SESSION_KEYS = 55,  
PROP_LOCAL_DATE = 56,  
PROP_LOCAL_TIME = 57,  
PROP_LOCATION = 58,  
PROP_LOW_LIMIT = 59,  
PROP_MANIPULATED_VARIABLE_REFERENCE = 60,  
PROP_MAXIMUM_OUTPUT = 61,  
PROP_MAX_APDU_LENGTH_ACCEPTED = 62,  
PROP_MAX_INFO_FRAMES = 63,  
PROP_MAX_MASTER = 64,  
PROP_MAX_PRES_VALUE = 65,  
PROP_MINIMUM_OFF_TIME = 66,  
PROP_MINIMUM_ON_TIME = 67,  
PROP_MINIMUM_OUTPUT = 68,  
PROP_MIN_PRES_VALUE = 69,  
PROP_MODEL_NAME = 70,  
PROP_MODIFICATION_DATE = 71,  
PROP_NOTIFY_TYPE = 72,  
PROP_NUMBER_OF_APDU_RETRIES = 73,  
PROP_NUMBER_OF_STATES = 74,  
PROP_OBJECT_IDENTIFIER = 75,  
PROP_OBJECT_LIST = 76,  
PROP_OBJECT_NAME = 77,  
PROP_OBJECT_PROPERTY_REFERENCE = 78,  
PROP_OBJECT_TYPE = 79,  
PROP_OPTIONAL = 80,  
PROP_OUT_OF_SERVICE = 81,  
PROP_OUTPUT_UNITS = 82,  
PROP_EVENT_PARAMETERS = 83,  
PROP_POLARITY = 84,  
PROP_PRESENT_VALUE = 85,  
PROP_PRIORITY = 86,  
PROP_PRIORITY_ARRAY = 87,  
PROP_PRIORITY_FOR_WRITING = 88,  
PROP_PROCESS_IDENTIFIER = 89,  
PROP_PROGRAM_CHANGE = 90,  
PROP_PROGRAM_LOCATION = 91,  
PROP_PROGRAM_STATE = 92,  
PROP_PROPORTIONAL_CONSTANT = 93,  
PROP_PROPORTIONAL_CONSTANT_UNITS = 94,  
PROP_PROTOCOL_CONFORMANCE_CLASS = 95, /* deleted in version 1 revision 2  
*/  
PROP_PROTOCOL_OBJECT_TYPES_SUPPORTED = 96,  
PROP_PROTOCOL_SERVICES_SUPPORTED = 97,  
PROP_PROTOCOL_VERSION = 98,
```

```
PROP_READ_ONLY = 99,  
PROP_REASON_FOR_HALT = 100,  
PROP_RECIPIENT = 101,  
PROP_RECIPIENT_LIST = 102,  
PROP_RELIABILITY = 103,  
PROP_RELINQUISH_DEFAULT = 104,  
PROP_REQUIRED = 105,  
PROP_RESOLUTION = 106,  
PROP_SEGMENTATION_SUPPORTED = 107,  
PROP_SETPPOINT = 108,  
PROP_SETPPOINT_REFERENCE = 109,  
PROP_STATE_TEXT = 110,  
PROP_STATUS_FLAGS = 111,  
PROP_SYSTEM_STATUS = 112,  
PROP_TIME_DELAY = 113,  
PROP_TIME_OF_ACTIVE_TIME_RESET = 114,  
PROP_TIME_OF_STATE_COUNT_RESET = 115,  
PROP_TIME_SYNCHRONIZATION_RECIPIENTS = 116,  
PROP_UNITS = 117,  
PROP_UPDATE_INTERVAL = 118,  
PROP_UTC_OFFSET = 119,  
PROP_VENDOR_IDENTIFIER = 120,  
PROP_VENDOR_NAME = 121,  
PROP_VT_CLASSES_SUPPORTED = 122,  
PROP_WEEKLY_SCHEDULE = 123,  
PROP_ATTEMPTED_SAMPLES = 124,  
PROP_AVERAGE_VALUE = 125,  
PROP_BUFFER_SIZE = 126,  
PROP_CLIENT_COV_INCREMENT = 127,  
PROP_COV_RESUBSCRIPTION_INTERVAL = 128,  
PROP_CURRENT_NOTIFY_TIME = 129,  
PROP_EVENT_TIME_STAMPS = 130,  
PROP_LOG_BUFFER = 131,  
PROP_LOG_DEVICE_OBJECT = 132,  
/* The enable property is renamed from log-enable in  
   Addendum b to ANSI/ASHRAE 135-2004(135b-2) */  
PROP_ENABLE = 133,  
PROP_LOG_INTERVAL = 134,  
PROP_MAXIMUM_VALUE = 135,  
PROP_MINIMUM_VALUE = 136,  
PROP_NOTIFICATION_THRESHOLD = 137,  
PROP_PREVIOUS_NOTIFY_TIME = 138,  
PROP_PROTOCOL_REVISION = 139,  
PROP_RECORDS_SINCE_NOTIFICATION = 140,  
PROP_RECORD_COUNT = 141,  
PROP_START_TIME = 142,  
PROP_STOP_TIME = 143,  
PROP_STOP_WHEN_FULL = 144,  
PROP_TOTAL_RECORD_COUNT = 145,  
PROP_VALID_SAMPLES = 146,  
PROP_WINDOW_INTERVAL = 147,  
PROP_WINDOW_SAMPLES = 148,  
PROP_MAXIMUM_VALUE_TIMESTAMP = 149,  
PROP_MINIMUM_VALUE_TIMESTAMP = 150,  
PROP_VARIANCE_VALUE = 151,  
PROP_ACTIVE_COV_SUBSCRIPTIONS = 152,  
PROP_BACKUP_FAILURE_TIMEOUT = 153,  
PROP_CONFIGURATION_FILES = 154,  
PROP_DATABASE_REVISION = 155,  
PROP_DIRECT_READING = 156,  
PROP_LAST_RESTORE_TIME = 157,
```



```
PROP_MAINTENANCE_REQUIRED = 158,  
PROP_MEMBER_OF = 159,  
PROP_MODE = 160,  
PROP_OPERATION_EXPECTED = 161,  
PROP_SETTING = 162,  
PROP_SILENCED = 163,  
PROP_TRACKING_VALUE = 164,  
PROP_ZONE_MEMBERS = 165,  
PROP_LIFE_SAFETY_ALARM_VALUES = 166,  
PROP_MAX_SEGMENTS_ACCEPTED = 167,  
PROP_PROFILE_NAME = 168,  
PROP_AUTO_SLAVE_DISCOVERY = 169,  
PROP_MANUAL_SLAVE_ADDRESS_BINDING = 170,  
PROP_SLAVE_ADDRESS_BINDING = 171,  
PROP_SLAVE_PROXY_ENABLE = 172,  
PROP_LAST_NOTIFY_TIME = 173,  
PROP_SCHEDULE_DEFAULT = 174,  
PROP_ACCEPTED_MODES = 175,  
PROP_ADJUST_VALUE = 176,  
PROP_COUNT = 177,  
PROP_COUNT_BEFORE_CHANGE = 178,  
PROP_COUNT_CHANGE_TIME = 179,  
PROP_COV_PERIOD = 180,  
PROP_INPUT_REFERENCE = 181,  
PROP_LIMIT_MONITORING_INTERVAL = 182,  
PROP_LOGGING_DEVICE = 183,  
PROP_LOGGING_RECORD = 184,  
PROP_PRESCALE = 185,  
PROP_PULSE_RATE = 186,  
PROP_SCALE = 187,  
PROP_SCALE_FACTOR = 188,  
PROP_UPDATE_TIME = 189,  
PROP_VALUE_BEFORE_CHANGE = 190,  
PROP_VALUE_SET = 191,  
PROP_VALUE_CHANGE_TIME = 192,  
/* enumerations 193-206 are new */  
PROP_ALIGN_INTERVALS = 193,  
PROP_GROUP_MEMBER_NAMES = 194,  
PROP_INTERVAL_OFFSET = 195,  
PROP_LAST_RESTART_REASON = 196,  
PROP_LOGGING_TYPE = 197,  
PROP_MEMBER_STATUS_FLAGS = 198,  
PROP_NOTIFICATION_PERIOD = 199,  
PROP_PREVIOUS_NOTIFY_RECORD = 200,  
PROP_REQUESTED_UPDATE_INTERVAL = 201,  
PROP_RESTART_NOTIFICATION_RECIPIENTS = 202,  
PROP_TIME_OF_DEVICE_RESTART = 203,  
PROP_TIME_SYNCHRONIZATION_INTERVAL = 204,  
PROP_TRIGGER = 205,  
PROP_UTC_TIME_SYNCHRONIZATION_RECIPIENTS = 206,  
/* enumerations 207-211 are used in Addendum d to ANSI/ASHRAE 135-2004 */  
PROP_NODE_SUBTYPE = 207,  
PROP_NODE_TYPE = 208,  
PROP_STRUCTURED_OBJECT_LIST = 209,  
PROP_SUBORDINATE_ANNOTATIONS = 210,  
PROP_SUBORDINATE_LIST = 211,  
/* enumerations 212-225 are used in Addendum e to ANSI/ASHRAE 135-2004 */  
PROP_ACTUAL_SHED_LEVEL = 212,  
PROP_DUTY_WINDOW = 213,  
PROP_EXPECTED_SHED_LEVEL = 214,  
PROP_FULL_DUTY_BASELINE = 215,
```

```
/* enumerations 216-217 are used in Addendum i to ANSI/ASHRAE 135-2004 */
PROP_BLINK_PRIORITY_THRESHOLD = 216,
PROP_BLINK_TIME = 217,
/* enumerations 212-225 are used in Addendum e to ANSI/ASHRAE 135-2004 */
PROP_REQUESTED_SHED_LEVEL = 218,
PROP_SHED_DURATION = 219,
PROP_SHED_LEVEL_DESCRIPTIONS = 220,
PROP_SHED_LEVELS = 221,
PROP_STATE_DESCRIPTION = 222,
/* enumerations 223-225 are used in Addendum i to ANSI/ASHRAE 135-2004 */
PROP_FADE_TIME = 223,
PROP_LIGHTING_COMMAND = 224,
PROP_LIGHTING_COMMAND_PRIORITY = 225,
/* enumerations 226-235 are used in Addendum f to ANSI/ASHRAE 135-2004 */
PROP_DOOR_ALARM_STATE = 226,
PROP_DOOR_EXTENDED_PULSE_TIME = 227,
PROP_DOOR_MEMBERS = 228,
PROP_DOOR_OPEN_TOO_LONG_TIME = 229,
PROP_DOOR_PULSE_TIME = 230,
PROP_DOOR_STATUS = 231,
PROP_DOOR_UNLOCK_DELAY_TIME = 232,
PROP_LOCK_STATUS = 233,
PROP_MASKED_ALARM_VALUES = 234,
PROP_SECURED_STATUS = 235,
/* enumerations 236-243 are used in Addendum i to ANSI/ASHRAE 135-2004 */
PROP_OFF_DELAY = 236,
PROP_ON_DELAY = 237,
PROP_POWER = 238,
PROP_POWER_ON_VALUE = 239,
PROP_PROGRESS_VALUE = 240,
PROP_RAMP_RATE = 241,
PROP_STEP_INCREMENT = 242,
PROP_SYSTEM_FAILURE_VALUE = 243,
/* enumerations 244-311 are used in Addendum j to ANSI/ASHRAE 135-2004 */
PROP_ABSENTEE_LIMIT = 244,
PROP_ACCESS_ALARM_EVENTS = 245,
PROP_ACCESS_DOORS = 246,
PROP_ACCESS_EVENT = 247,
PROP_ACCESS_EVENT_AUTHENTICATION_FACTOR = 248,
PROP_ACCESS_EVENT_CREDENTIAL = 249,
PROP_ACCESS_EVENT_TIME = 250,
PROP_ACCESS_RULES = 251,
PROP_ACCESS_RULES_ENABLE = 252,
PROP_ACCESS_TRANSACTION_EVENTS = 253,
PROP_ACCOMPANIED = 254,
PROP_ACTIVATION_TIME = 255,
PROP_ACTIVE_AUTHENTICATION_POLICY = 256,
PROP_ASSIGNED_ACCESS_RIGHTS = 257,
PROP_AUTHENTICATION_FACTOR_INPUT_LIST = 258,
PROP_AUTHENTICATION_FACTORS = 259,
PROP_AUTHENTICATION_POLICY_LIST = 260,
PROP_AUTHENTICATION_POLICY_NAMES = 261,
PROP_AUTHORIZATION_MODE = 262,
PROP_BELONGS_TO = 263,
PROP_CREDENTIAL_DISABLE = 264,
PROP_CREDENTIAL_STATUS = 265,
PROP_CREDENTIALS = 266,
PROP_CREDENTIALS_IN_ZONE = 267,
PROP_DAYS_REMAINING = 268,
PROP_ENTRY_POINTS = 269,
PROP_EXIT_POINTS = 270,
```

```
PROP_EXPIRY_TIME = 271,  
PROP_EXTENDED_TIME_ENABLE = 272,  
PROP_FAILED_ATTEMPT_EVENTS = 273,  
PROP_FAILED_ATTEMPTS = 274,  
PROP_FAILED_ATTEMPTS_TIME = 275,  
PROP_FORMAT_CLASS_SUPPORTED = 276,  
PROP_FORMAT_TYPE = 277,  
PROP_LAST_ACCESS_EVENT = 278,  
PROP_LAST_ACCESS_POINT = 279,  
PROP_LAST_CREDENTIAL_ADDED = 280,  
PROP_LAST_CREDENTIAL_ADDED_TIME = 281,  
PROP_LAST_CREDENTIAL_REMOVED = 282,  
PROP_LAST_CREDENTIAL_REMOVED_TIME = 283,  
PROP_LAST_USE_TIME = 284,  
PROP_LOCKDOWN = 285,  
PROP_LOCKDOWN_RELINQUISH_TIME = 286,  
PROP_MASTER_EXEMPTION = 287,  
PROP_MAX_FAILED_ATTEMPTS = 288,  
PROP_MEMBERS = 289,  
PROP_MASTER_POINT = 290,  
PROP_NUMBER_OF_AUTHENTICATION_POLICIES = 291,  
PROP_OCCUPANCY_COUNT = 293,  
PROP_OCCUPANCY_COUNT_ENABLE = 294,  
PROP_OCCUPANCY_COUNT_EXEMPTION = 295,  
PROP_OCCUPANCY_LOWER_THRESHOLD = 296,  
PROP_OCCUPANCY_LOWER_THRESHOLD_ENFORCED = 297,  
PROP_OCCUPANCY_STATE = 298,  
PROP_OCCUPANCY_UPPER_LIMIT = 299,  
PROP_OCCUPANCY_UPPER_LIMIT_ENFORCED = 300,  
PROP_PASSBACK_EXEMPTION = 301,  
PROP_PASSBACK_MODE = 302,  
PROP_PASSBACK_TIMEOUT = 303,  
PROP_POSITIVE_ACCESS_RULES = 304,  
PROP_READ_STATUS = 305,  
PROP_REASON_FOR_DISABLE = 306,  
PROP_THREAT_AUTHORITY = 307,  
PROP_THREAT_LEVEL = 308,  
PROP_TRACE_FLAG = 309,  
PROP_TRANSACTION_NOTIFICATION_CLASS = 310,  
PROP_USER_EXTERNAL_IDENTIFIER = 311,  
/* enumerations 312-313 are used in Addendum k to ANSI/ASHRAE 135-2004 */  
PROP_CHARACTER_SET = 312,  
PROP_STRICT_CHARACTER_MODE = 313,  
/* enumerations 312-313 are used in Addendum k to ANSI/ASHRAE 135-2004 */  
PROP_BACKUP_AND_RESTORE_STATE = 314,  
PROP_BACKUP_PREPARATION_TIME = 315,  
PROP_RESTORE_PREPARATION_TIME = 316,  
/* enumerations 317-323 are used in Addendum j to ANSI/ASHRAE 135-2004 */  
PROP_USER_INFORMATION_REFERENCE = 317,  
PROP_USER_NAME = 318,  
PROP_USER_TYPE = 319,  
PROP_USES_REMAINING = 320,  
PROP_VENDOR_FORMAT_IDENTIFIER = 321,  
PROP_ZONE_FROM = 322,  
PROP_ZONE_TO = 323,  
/* enumerations 324-325 are used in Addendum i to ANSI/ASHRAE 135-2004 */  
PROP_BINARY_ACTIVE_VALUE = 324,  
PROP_BINARY_INACTIVE_VALUE = 325  
    /* The special property identifiers all, optional, and required */  
    /* are reserved for use in the ReadPropertyConditional and */  
    /* ReadPropertyMultiple services or services not defined in this standard.
```

```

*/
    /* Enumerated values 0-511 are reserved for definition by ASHRAE. */
    /* Enumerated values 512-4194303 may be used by others subject to the */
    /* procedures and constraints described in Clause 23. */
} BACNET_PROPERTY_ID;

```

F.2 BACnet Engineering Units Codes

BACnet engineering units codes may be found in your copy of the BACnet protocol specification, ANSI/ASHRAE Standard 135. That document is copyrighted, but the C enumeration shown below for reference is taken from open source code available under GPL at <http://sourceforge.net>, and provides essentially the same information (copyrighted by Steve Karg, licensed under GPL as noted at <http://sourceforge.net>).

```

typedef enum {
    /* Acceleration */
    UNITS_METERS_PER_SECOND_PER_SECOND = 166,
    /* Area */
    UNITS_SQUARE_METERS = 0,
    UNITS_SQUARE_CENTIMETERS = 116,
    UNITS_SQUARE_FEET = 1,
    UNITS_SQUARE_INCHES = 115,
    /* Currency */
    UNITS_CURRENCY1 = 105,
    UNITS_CURRENCY2 = 106,
    UNITS_CURRENCY3 = 107,
    UNITS_CURRENCY4 = 108,
    UNITS_CURRENCY5 = 109,
    UNITS_CURRENCY6 = 110,
    UNITS_CURRENCY7 = 111,
    UNITS_CURRENCY8 = 112,
    UNITS_CURRENCY9 = 113,
    UNITS_CURRENCY10 = 114,
    /* Electrical */
    UNITS_MILLIAMPERES = 2,
    UNITS_AMPERES = 3,
    UNITS_AMPERES_PER_METER = 167,
    UNITS_AMPERES_PER_SQUARE_METER = 168,
    UNITS_AMPERE_SQUARE_METERS = 169,
    UNITS_FARADS = 170,
    UNITS_HENRYS = 171,
    UNITS_OHMS = 4,
    UNITS_OHM_METERS = 172,
    UNITS_MILLIOHMS = 145,
    UNITS_KILOHMS = 122,
    UNITS_MEGOHMS = 123,
    UNITS_SIEMENS = 173, /* 1 mho equals 1 siemens */
    UNITS_SIEMENS_PER_METER = 174,
    UNITS_TESLAS = 175,
    UNITS_VOLTS = 5,
    UNITS_MILLIVOLTS = 124,
    UNITS_KILOVOLTS = 6,
    UNITS_MEGAVOLTS = 7,
    UNITS_VOLT_AMPERES = 8,
    UNITS_KILOVOLT_AMPERES = 9,
    UNITS_MEGAVOLT_AMPERES = 10,
    UNITS_VOLT_AMPERES_REACTIVE = 11,
    UNITS_KILOVOLT_AMPERES_REACTIVE = 12,
    UNITS_MEGAVOLT_AMPERES_REACTIVE = 13,
    UNITS_VOLTS_PER_DEGREE_KELVIN = 176,

```

```
UNITS_VOLTS_PER_METER = 177,
UNITS_DEGREES_PHASE = 14,
UNITS_POWER_FACTOR = 15,
UNITS_WEBERS = 178,
/* Energy */
UNITS_JOULES = 16,
UNITS_KILOJOULES = 17,
UNITS_KILOJOULES_PER_KILOGRAM = 125,
UNITS_MEGAJOULES = 126,
UNITS_WATT_HOURS = 18,
UNITS_KILOWATT_HOURS = 19,
UNITS_MEGAWATT_HOURS = 146,
UNITS_BTUS = 20,
UNITS_KILO_BTUS = 147,
UNITS_MEGA_BTUS = 148,
UNITS_THERMS = 21,
UNITS_TON_HOURS = 22,
/* Enthalpy */
UNITS_JOULES_PER_KILOGRAM_DRY_AIR = 23,
UNITS_KILOJOULES_PER_KILOGRAM_DRY_AIR = 149,
UNITS_MEGAJOULES_PER_KILOGRAM_DRY_AIR = 150,
UNITS_BTUS_PER_POUND_DRY_AIR = 24,
UNITS_BTUS_PER_POUND = 117,
/* Entropy */
UNITS_JOULES_PER_DEGREE_KELVIN = 127,
UNITS_KILOJOULES_PER_DEGREE_KELVIN = 151,
UNITS_MEGAJOULES_PER_DEGREE_KELVIN = 152,
UNITS_JOULES_PER_KILOGRAM_DEGREE_KELVIN = 128,
/* Force */
UNITS_NEWTON = 153,
/* Frequency */
UNITS_CYCLES_PER_HOUR = 25,
UNITS_CYCLES_PER_MINUTE = 26,
UNITS_HERTZ = 27,
UNITS_KILOHERTZ = 129,
UNITS_MEGAHERTZ = 130,
UNITS_PER_HOUR = 131,
/* Humidity */
UNITS_GRAMS_OF_WATER_PER_KILOGRAM_DRY_AIR = 28,
UNITS_PERCENT_RELATIVE_HUMIDITY = 29,
/* Length */
UNITS_MILLIMETERS = 30,
UNITS_CENTIMETERS = 118,
UNITS_METERS = 31,
UNITS_INCHES = 32,
UNITS_FEET = 33,
/* Light */
UNITS_CANDELAS = 179,
UNITS_CANDELAS_PER_SQUARE_METER = 180,
UNITS_WATTS_PER_SQUARE_FOOT = 34,
UNITS_WATTS_PER_SQUARE_METER = 35,
UNITS_LUMENS = 36,
UNITS_LUXES = 37,
UNITS_FOOT_CANDLES = 38,
/* Mass */
UNITS_KILOGRAMS = 39,
UNITS_POUNDS_MASS = 40,
UNITS_TONS = 41,
/* Mass Flow */
UNITS_GRAMS_PER_SECOND = 154,
UNITS_GRAMS_PER_MINUTE = 155,
```



```
UNITS_KILOGRAMS_PER_SECOND = 42,  
UNITS_KILOGRAMS_PER_MINUTE = 43,  
UNITS_KILOGRAMS_PER_HOUR = 44,  
UNITS_POUNDS_MASS_PER_SECOND = 119,  
UNITS_POUNDS_MASS_PER_MINUTE = 45,  
UNITS_POUNDS_MASS_PER_HOUR = 46,  
UNITS_TONS_PER_HOUR = 156,  
/* Power */  
UNITS_MILLIWATTS = 132,  
UNITS_WATTS = 47,  
UNITS_KILOWATTS = 48,  
UNITS_MEGAWATTS = 49,  
UNITS_BTUS_PER_HOUR = 50,  
UNITS_KILO_BTUS_PER_HOUR = 157,  
UNITS_HORSEPOWER = 51,  
UNITS_TONS_REFRIGERATION = 52,  
/* Pressure */  
UNITS_PASCALS = 53,  
UNITS_HECTOPASCALS = 133,  
UNITS_KILOPASCALS = 54,  
UNITS_MILLIBARS = 134,  
UNITS_BARS = 55,  
UNITS_POUNDS_FORCE_PER_SQUARE_INCH = 56,  
UNITS_CENTIMETERS_OF_WATER = 57,  
UNITS_INCHES_OF_WATER = 58,  
UNITS_MILLIMETERS_OF_MERCURY = 59,  
UNITS_CENTIMETERS_OF_MERCURY = 60,  
UNITS_INCHES_OF_MERCURY = 61,  
/* Temperature */  
UNITS_DEGREES_CELSIUS = 62,  
UNITS_DEGREES_KELVIN = 63,  
UNITS_DEGREES_KELVIN_PER_HOUR = 181,  
UNITS_DEGREES_KELVIN_PER_MINUTE = 182,  
UNITS_DEGREES_FAHRENHEIT = 64,  
UNITS_DEGREE_DAYS_CELSIUS = 65,  
UNITS_DEGREE_DAYS_FAHRENHEIT = 66,  
UNITS_DELTA_DEGREES_FAHRENHEIT = 120,  
UNITS_DELTA_DEGREES_KELVIN = 121,  
/* Time */  
UNITS_YEARS = 67,  
UNITS_MONTHS = 68,  
UNITS_WEEKS = 69,  
UNITS_DAYS = 70,  
UNITS_HOURS = 71,  
UNITS_MINUTES = 72,  
UNITS_SECONDS = 73,  
UNITS_HUNDREDTHS_SECONDS = 158,  
UNITS_MILLISECONDS = 159,  
/* Torque */  
UNITS_NEWTON_METERS = 160,  
/* Velocity */  
UNITS_MILLIMETERS_PER_SECOND = 161,  
UNITS_MILLIMETERS_PER_MINUTE = 162,  
UNITS_METERS_PER_SECOND = 74,  
UNITS_METERS_PER_MINUTE = 163,  
UNITS_METERS_PER_HOUR = 164,  
UNITS_KILOMETERS_PER_HOUR = 75,  
UNITS_FEET_PER_SECOND = 76,  
UNITS_FEET_PER_MINUTE = 77,  
UNITS_MILES_PER_HOUR = 78,  
/* Volume */
```

```

UNITS_CUBIC_FEET = 79,
UNITS_CUBIC_METERS = 80,
UNITS_IMPERIAL_GALLONS = 81,
UNITS_LITERS = 82,
UNITS_US_GALLONS = 83,
/* Volumetric Flow */
UNITS_CUBIC_FEET_PER_SECOND = 142,
UNITS_CUBIC_FEET_PER_MINUTE = 84,
UNITS_CUBIC_METERS_PER_SECOND = 85,
UNITS_CUBIC_METERS_PER_MINUTE = 165,
UNITS_CUBIC_METERS_PER_HOUR = 135,
UNITS_IMPERIAL_GALLONS_PER_MINUTE = 86,
UNITS_LITERS_PER_SECOND = 87,
UNITS_LITERS_PER_MINUTE = 88,
UNITS_LITERS_PER_HOUR = 136,
UNITS_US_GALLONS_PER_MINUTE = 89,
/* Other */
UNITS_DEGREES_ANGULAR = 90,
UNITS_DEGREES_CELSIUS_PER_HOUR = 91,
UNITS_DEGREES_CELSIUS_PER_MINUTE = 92,
UNITS_DEGREES_FAHRENHEIT_PER_HOUR = 93,
UNITS_DEGREES_FAHRENHEIT_PER_MINUTE = 94,
UNITS_JOULE_SECONDS = 183,
UNITS_KILOGRAMS_PER_CUBIC_METER = 186,
UNITS_KW_HOURS_PER_SQUARE_METER = 137,
UNITS_KW_HOURS_PER_SQUARE_FOOT = 138,
UNITS_MEGAJOULES_PER_SQUARE_METER = 139,
UNITS_MEGAJOULES_PER_SQUARE_FOOT = 140,
UNITS_NO_UNITS = 95,
UNITS_NEWTON_SECONDS = 187,
UNITS_NEWTONS_PER_METER = 188,
UNITS_PARTS_PER_MILLION = 96,
UNITS_PARTS_PER_BILLION = 97,
UNITS_PERCENT = 98,
UNITS_PERCENT_OBSCURATION_PER_FOOT = 143,
UNITS_PERCENT_OBSCURATION_PER_METER = 144,
UNITS_PERCENT_PER_SECOND = 99,
UNITS_PER_MINUTE = 100,
UNITS_PER_SECOND = 101,
UNITS_PSI_PER_DEGREE_FAHRENHEIT = 102,
UNITS_RADIANS = 103,
UNITS_RADIANS_PER_SECOND = 184,
UNITS_REVOLUTIONS_PER_MINUTE = 104,
UNITS_SQUARE_METERS_PER_NEWTON = 185,
UNITS_WATTS_PER_METER_PER_DEGREE_KELVIN = 189,
UNITS_WATTS_PER_SQUARE_METER_DEGREE_KELVIN = 141,
; /* Enumerated values 0-255 are reserved for definition by ASHRAE. */
/* Enumerated values 256-65535 may be used by others subject to */
/* the procedures and constraints described in Clause 23. */
/* The last enumeration used in this version is 189. */
MAX_UNITS = 190
} BACNET_ENGINEERING_UNITS;

```

F.3 BACnet Error Codes

BACnet error codes may be found in your copy of the BACnet protocol specification, ANSI/ASHRAE Standard 135. That document is copyrighted, but the C enumeration shown below for reference is taken from open source code available under GPL at <http://sourceforge.net>, and provides essentially the same information (copyrighted by Steve Karg, licensed under GPL as noted at <http://sourceforge.net>).

BACnet error codes actually consist of two codes: The error Class and the error Code. The classes are the shorter list shown first below, and the codes that may occur for each class follow.

```
typedef enum {
ERROR_CLASS_DEVICE = 0,
ERROR_CLASS_OBJECT = 1,
ERROR_CLASS_PROPERTY = 2,
ERROR_CLASS_RESOURCES = 3,
ERROR_CLASS_SECURITY = 4,
ERROR_CLASS_SERVICES = 5,
ERROR_CLASS_VT = 6,
ERROR_CLASS_COMMUNICATION = 7,
/* Enumerated values 0-63 are reserved for definition by ASHRAE. */
/* Enumerated values 64-65535 may be used by others subject to */
/* the procedures and constraints described in Clause 23. */
MAX_BACNET_ERROR_CLASS = 8
} BACNET_ERROR_CLASS;
```

```
/* These are sorted in the order given in
Clause 18. ERROR, REJECT AND ABORT CODES
The Class and Code pairings are required
to be used in accordance with Clause 18. */
```

```
typedef enum {
/* valid for all classes */
ERROR_CODE_OTHER = 0,
```

```
/* Error Class - Device */
ERROR_CODE_DEVICE_BUSY = 3,
ERROR_CODE_CONFIGURATION_IN_PROGRESS = 2,
ERROR_CODE_OPERATIONAL_PROBLEM = 25,
```

```
/* Error Class - Object */
ERROR_CODE_DYNAMIC_CREATION_NOT_SUPPORTED = 4,
ERROR_CODE_NO_OBJECTS_OF_SPECIFIED_TYPE = 17,
ERROR_CODE_OBJECT_DELETION_NOT_PERMITTED = 23,
ERROR_CODE_OBJECT_IDENTIFIER_ALREADY_EXISTS = 24,
ERROR_CODE_READ_ACCESS_DENIED = 27,
ERROR_CODE_UNKNOWN_OBJECT = 31,
ERROR_CODE_UNSUPPORTED_OBJECT_TYPE = 36,
```

```
/* Error Class - Property */
ERROR_CODE_CHARACTER_SET_NOT_SUPPORTED = 41,
ERROR_CODE_DATATYPE_NOT_SUPPORTED = 47,
ERROR_CODE_INCONSISTENT_SELECTION_CRITERION = 8,
ERROR_CODE_INVALID_ARRAY_INDEX = 42,
ERROR_CODE_INVALID_DATA_TYPE = 9,
ERROR_CODE_NOT_COV_PROPERTY = 44,
ERROR_CODE_OPTIONAL_FUNCTIONALITY_NOT_SUPPORTED = 45,
ERROR_CODE_PROPERTY_IS_NOT_AN_ARRAY = 50,
/* ERROR_CODE_READ_ACCESS_DENIED = 27, */
ERROR_CODE_UNKNOWN_PROPERTY = 32,
ERROR_CODE_VALUE_OUT_OF_RANGE = 37,
ERROR_CODE_WRITE_ACCESS_DENIED = 40,
```

```
/* Error Class - Resources */
ERROR_CODE_NO_SPACE_FOR_OBJECT = 18,
ERROR_CODE_NO_SPACE_TO_ADD_LIST_ELEMENT = 19,
ERROR_CODE_NO_SPACE_TO_WRITE_PROPERTY = 20,
```

```
/* Error Class - Security */
```

```
ERROR_CODE_AUTHENTICATION_FAILED = 1,
/* ERROR_CODE_CHARACTER_SET_NOT_SUPPORTED = 41, */
ERROR_CODE_INCOMPATIBLE_SECURITY_LEVELS = 6,
ERROR_CODE_INVALID_OPERATOR_NAME = 12,
ERROR_CODE_KEY_GENERATION_ERROR = 15,
ERROR_CODE_PASSWORD_FAILURE = 26,
ERROR_CODE_SECURITY_NOT_SUPPORTED = 28,
ERROR_CODE_TIMEOUT = 30,

/* Error Class - Services */
/* ERROR_CODE_CHARACTER_SET_NOT_SUPPORTED = 41, */
ERROR_CODE_COV_SUBSCRIPTION_FAILED = 43,
ERROR_CODE_DUPLICATE_NAME = 48,
ERROR_CODE_DUPLICATE_OBJECT_ID = 49,
ERROR_CODE_FILE_ACCESS_DENIED = 5,
ERROR_CODE_INCONSISTENT_PARAMETERS = 7,
ERROR_CODE_INVALID_CONFIGURATION_DATA = 46,
ERROR_CODE_INVALID_FILE_ACCESS_METHOD = 10,
ERROR_CODE_ERROR_CODE_INVALID_FILE_START_POSITION = 11,
ERROR_CODE_INVALID_PARAMETER_DATA_TYPE = 13,
ERROR_CODE_INVALID_TIME_STAMP = 14,
ERROR_CODE_MISSING_REQUIRED_PARAMETER = 16,
/* ERROR_CODE_OPTIONAL_FUNCTIONALITY_NOT_SUPPORTED = 45, */
ERROR_CODE_PROPERTY_IS_NOT_A_LIST = 22,
ERROR_CODE_SERVICE_REQUEST_DENIED = 29,

/* Error Class - VT */
ERROR_CODE_UNKNOWN_VT_CLASS = 34,
ERROR_CODE_UNKNOWN_VT_SESSION = 35,
ERROR_CODE_NO_VT_SESSIONS_AVAILABLE = 21,
ERROR_CODE_VT_SESSION_ALREADY_CLOSED = 38,
ERROR_CODE_VT_SESSION_TERMINATION_FAILURE = 39,

/* unused */
ERROR_CODE_RESERVED1 = 33,
/* new error codes from new addenda */
ERROR_CODE_ABORT_BUFFER_OVERFLOW = 51,
ERROR_CODE_ABORT_INVALID_APDU_IN_THIS_STATE = 52,
ERROR_CODE_ABORT_PREEMPTED_BY_HIGHER_PRIORITY_TASK = 53,
ERROR_CODE_ABORT_SEGMENTATION_NOT_SUPPORTED = 54,
ERROR_CODE_ABORT_PROPRIETARY = 55,
ERROR_CODE_ABORT_OTHER = 56,
ERROR_CODE_INVALID_TAG = 57,
ERROR_CODE_NETWORK_DOWN = 58,
ERROR_CODE_REJECT_BUFFER_OVERFLOW = 59,
ERROR_CODE_REJECT_INCONSISTENT_PARAMETERS = 60,
ERROR_CODE_REJECT_INVALID_PARAMETER_DATA_TYPE = 61,
ERROR_CODE_REJECT_INVALID_TAG = 62,
ERROR_CODE_REJECT_MISSING_REQUIRED_PARAMETER = 63,
ERROR_CODE_REJECT_PARAMETER_OUT_OF_RANGE = 64,
ERROR_CODE_REJECT_TOO_MANY_ARGUMENTS = 65,
ERROR_CODE_REJECT_UNDEFINED_ENUMERATION = 66,
ERROR_CODE_REJECT_UNRECOGNIZED_SERVICE = 67,
ERROR_CODE_REJECT_PROPRIETARY = 68,
ERROR_CODE_REJECT_OTHER = 69,
ERROR_CODE_UNKNOWN_DEVICE = 70,
ERROR_CODE_UNKNOWN_ROUTE = 71,
ERROR_CODE_VALUE_NOT_INITIALIZED = 72,
ERROR_CODE_INVALID_EVENT_STATE = 73,
ERROR_CODE_NO_ALARM_CONFIGURED = 74,
ERROR_CODE_LOG_BUFFER_FULL = 75,
```

```
ERROR_CODE_LOGGED_VALUE_PURGED = 76,  
ERROR_CODE_NO_PROPERTY_SPECIFIED = 77,  
ERROR_CODE_NOT_CONFIGURED_FOR_TRIGGERED_LOGGING = 78,  
ERROR_CODE_UNKNOWN_SUBSCRIPTION = 79,  
ERROR_CODE_PARAMETER_OUT_OF_RANGE = 80,  
ERROR_CODE_UNKNOWN_FILE_SIZE = 81,  
ERROR_CODE_BUSY = 82,  
ERROR_CODE_COMMUNICATION_DISABLED = 83,  
MAX_BACNET_ERROR_CODE = 84  
/* Enumerated values 0-255 are reserved for definition by ASHRAE. */  
/* Enumerated values 256-65535 may be used by others subject to */  
/* the procedures and constraints described in Clause 23. */  
/* The last enumeration used in this version is 50. */  
} BACNET_ERROR_CODE;
```



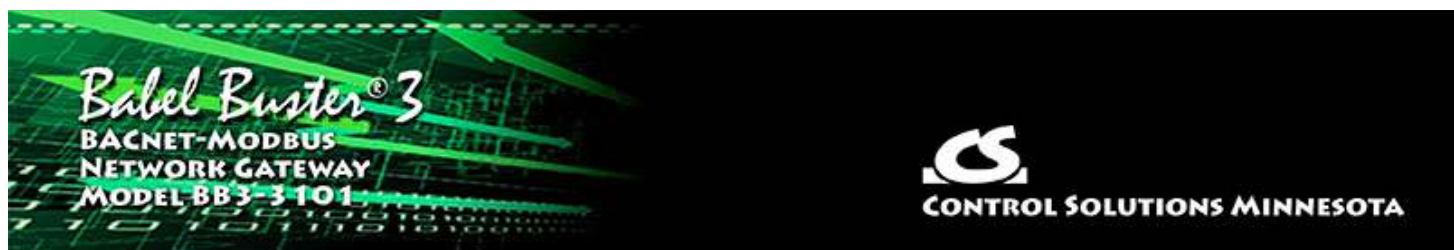

Appendix G - Configuration XML File Format

G.1 Configuration File Editing

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 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. Manual editing should be limited to creating and modifying the CSV file. In the process of importing the CSV file, you effectively auto-create much of the configuration.

The configuration software (user interface) includes a number of error checking steps that are applied as you enter information in the various pages of the software tool, 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 TRY TO FIX THE FILE FOR YOU.**



Appendix H USB Driver Installation

H.1 Driver Installation

The USB driver provided in Control Solutions' driver package will install in Windows XP through 10, and both 32-bit and 64-bit versions. It includes the necessary driver signing verified through Verisign/Symantec/DigiCert.

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



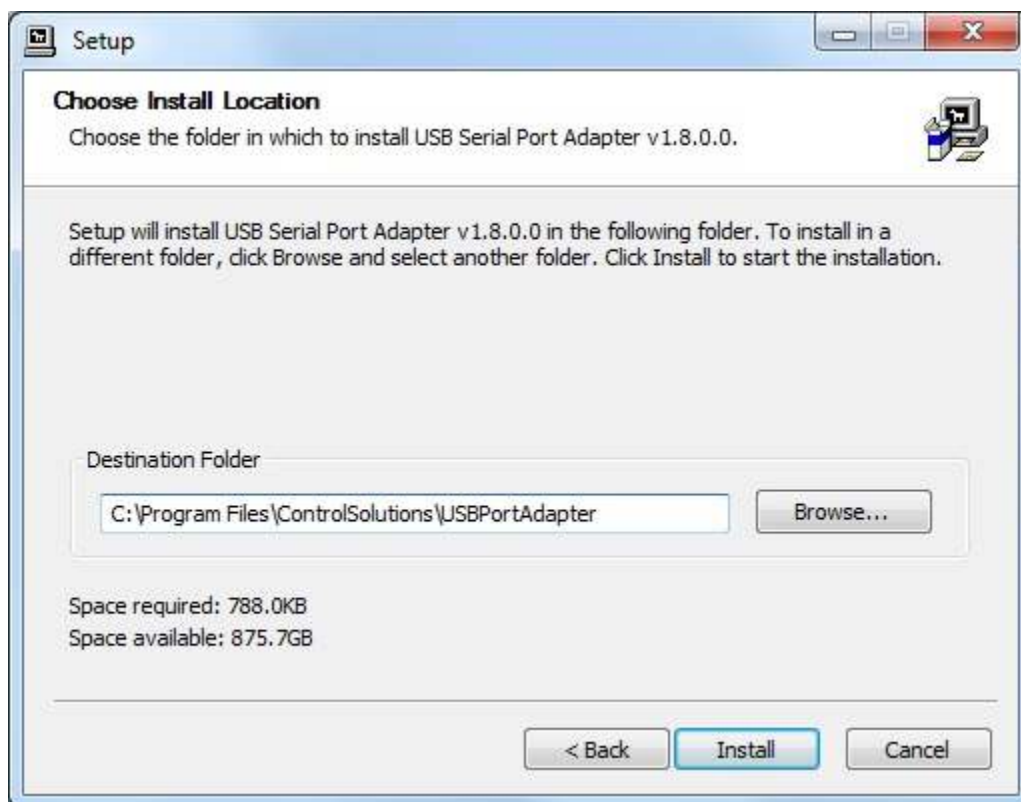
Unzip the contents of this file into a directory somewhere on your PC. The contents will look like this:

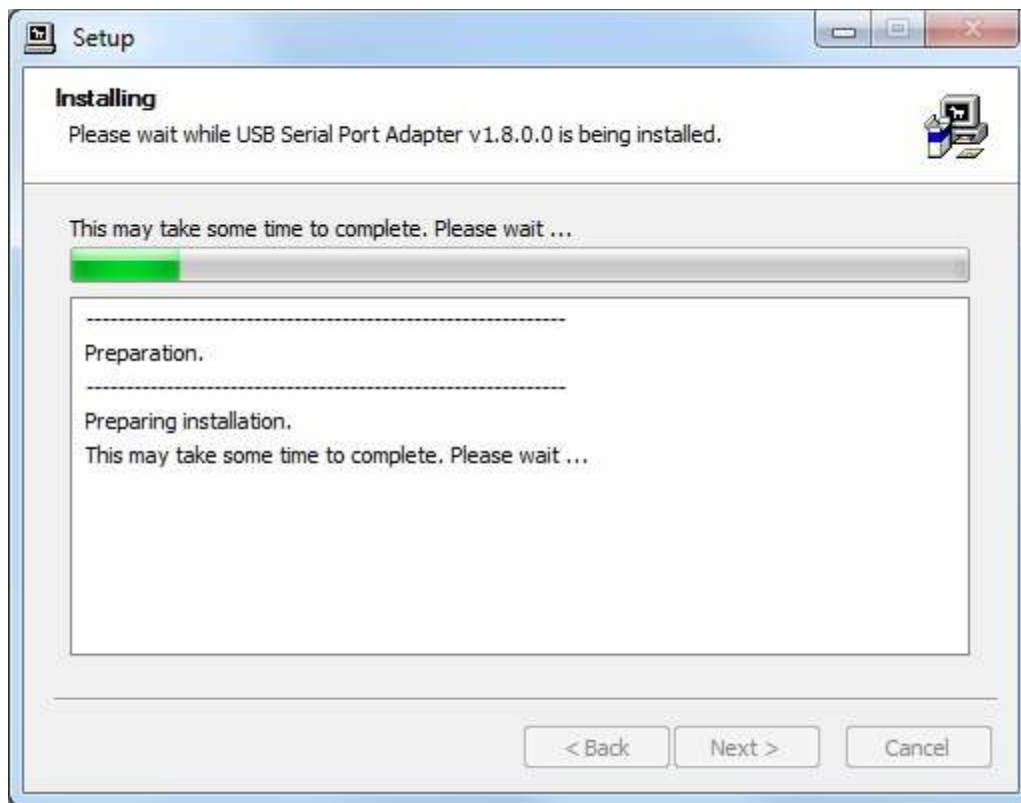


Double click "setup.exe". Say "yes" to any questions about whether to trust this software.

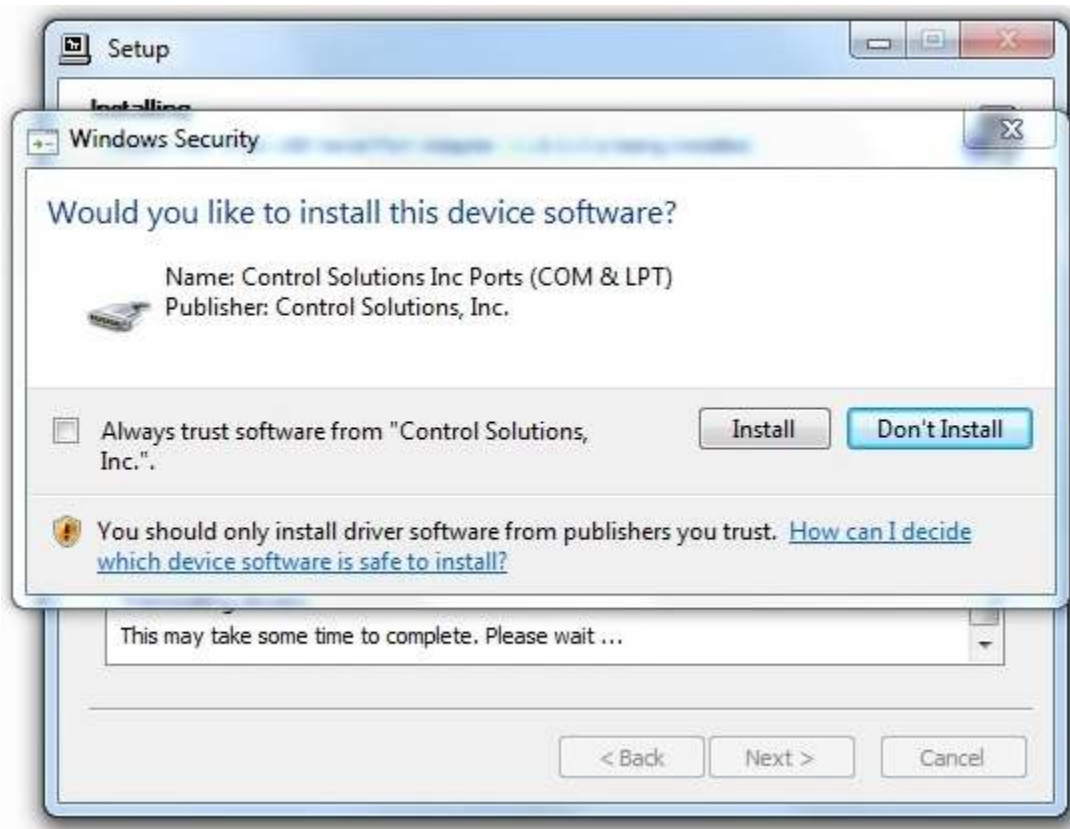
A sample of the series of screens you will see appears below. Basically all you need to do is follow the prompts, and click "yes", "next", "continue anyway", etc, as applicable.

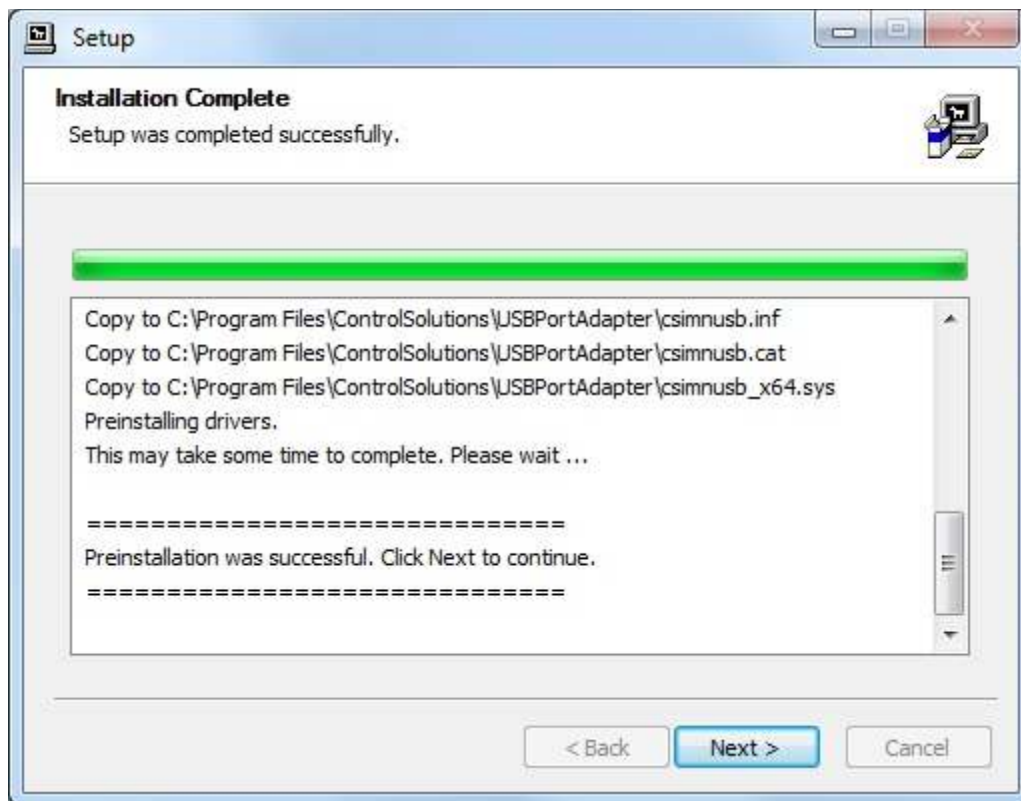
Technically, what you are doing in the process illustrated in the screen shots below is "driver pre-installation". After initial installation of this package, the device will automatically find the right driver when you plug it in, and driver installation will be finalized.





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





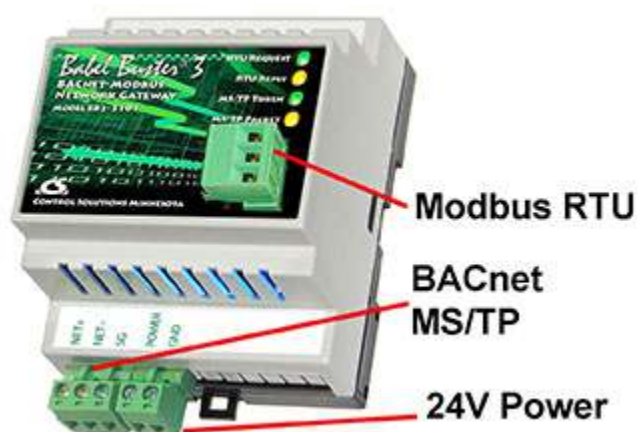
When you get to this screen, you're done. Now plug in your USB device (MTX002), allow the PC to finalize installation, and then go to the Device Manager via your PC's control panel to see which port the USB device got assigned to. Select this "COM" port in the Control Solutions configuration tool's "Connect"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



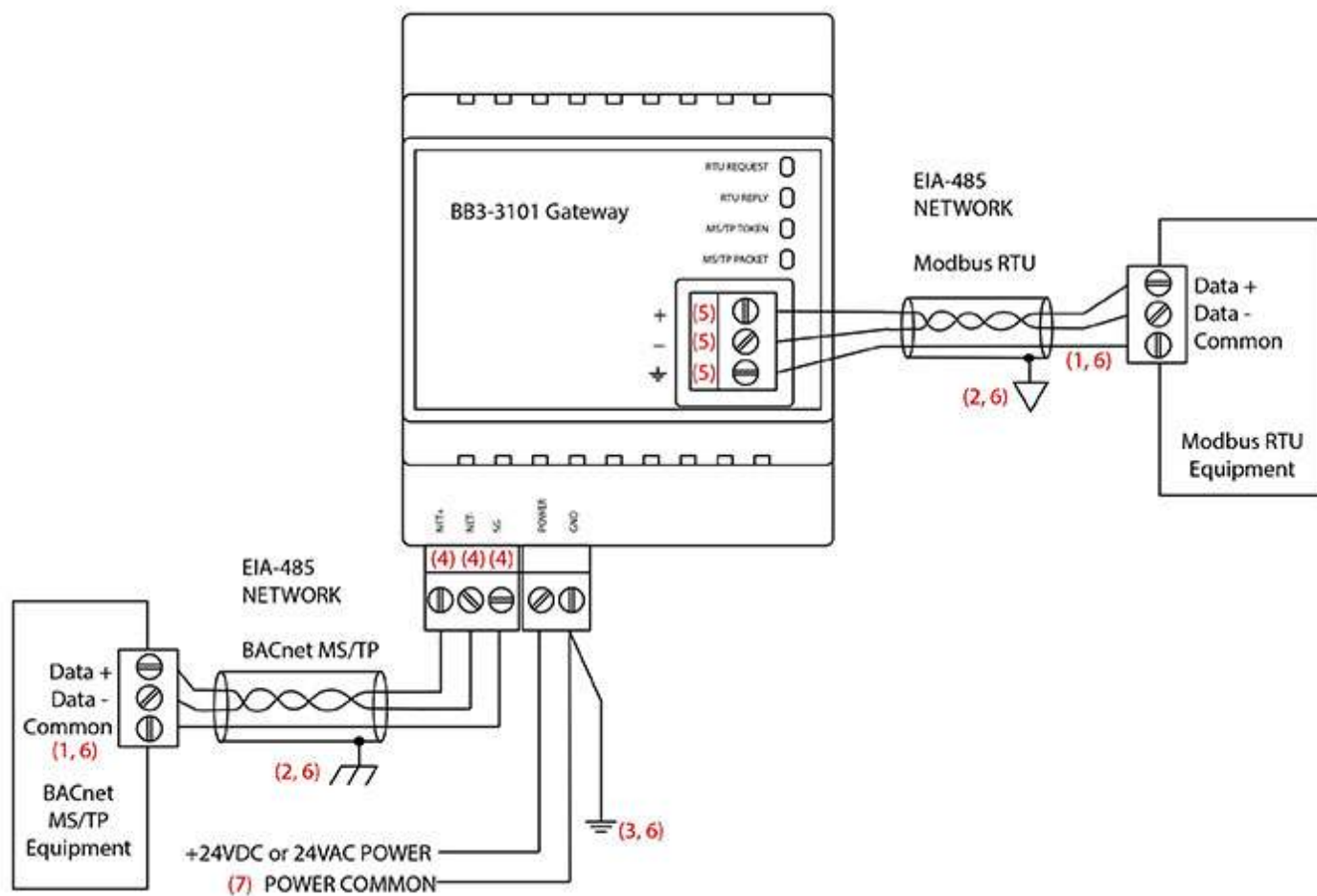


Appendix J Hardware Details

J.1 Wiring



Wiring connections for the BB3-3101 are as follows:



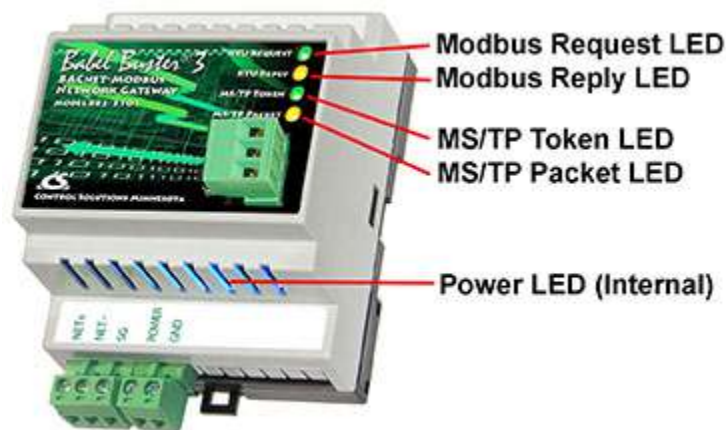
Notes:

- (1) Signal return required. Although EIA-485 (RS485) is thought of as a 2-wire network, you MUST include a third conductor connected to signal common at each device so that all devices are operating at close to the same signal reference.
- (2) Cable shield terminated per wiring standards, most often connected to common or ground at field end. Do not connect Modbus and BACnet shields to signal common at the gateway if isolation is to be maintained.
- (3) ESD protection depends on a ground suitable for dissipation of static discharge, necessary if not already provided by power common.
- (4) These terminals are electrically isolated from all other terminals
- (5) These terminals are electrically isolated from all other terminals IF the isolated Modbus port option is in use, otherwise referenced to Power Common.
- (6) The various Common connections must be isolated from each other if communication port isolation is to be maintained. The power and shield grounds should be connected as specified by your network engineer, but typically as noted in (2) above.
- (7) Gateway may share common power supply with either the BACnet equipment or

the Modbus equipment, but not both. Furthermore, if other equipment shares power with either BACnet or Modbus, then it is advisable to provide a separate isolated power supply for the gateway only.

J.2 Front Panel LED Indicators

Power-up LED behavior for BB2-3010 MS/TP gateway: 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.



The LEDs will indicate BACnet traffic as indicated in the table below.

BB3-3101 MS/TP LEDs		
Mode	Token LED	Packet LED
Gateway is Client or Server	Flash yellow each time the gateway sends a "poll for master".	Flash green any time a packet is sent on the network. This can be either a request or a response to a request
	Flash green each time the token is passed.	Flash red any time an error is detected. The error can be an error code reply, or timeout. The object reliability code will indicate which one it was.

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.

Two particular indications of the MS/TP token LED are worth noting.

If the token LED appears to be on nearly solid yellow (or amber), this means the gateway is doing nothing but poll for token, which implies it is not finding any other devices on the network. You will see this any time you apply power to the gateway without connecting any network. If the network is connected, there is a problem with connections or port settings.

If the token LED is completely off and never flashes on, this means the gateway is hearing other traffic on the network, but just not connecting. An MS/TP device is required to "be quiet until spoken to". Therefore, if there is chatter on the network, but the gateway cannot find anything that is addressed to the gateway, it will remain quiet. This problem is usually the result of problems with port settings.

A gateway (or any MS/TP device) can alternate between the above two conditions. The "be quiet until spoken to" rule is up for grabs when the entire network is powered up simultaneously. Whoever wakes up first will be generating all the chatter while everybody else on the network listens.

If the token LED is very intermittent, flashes rapidly a few times (including green on the token LED), then is off completely for a period of time, this usually means it is trying to talk, thought it was talking or actually was for a bit, but something got out of sync. If there are duplicate device addresses on the network or mismatching Max Master settings, this will result in the appearance of intermittent communications.

Check your wiring. Noise on the line resulting from not following RS-485 wiring guidelines will result in communication problems ranging from no connection at all to frequent errors. Also be sure taht you have a ground connection between all devices on the network. The RS-485 network is not a 2-wire network. You need two data lines plus signal ground. If you consider cable shield a connection, then you need 4 connections.

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

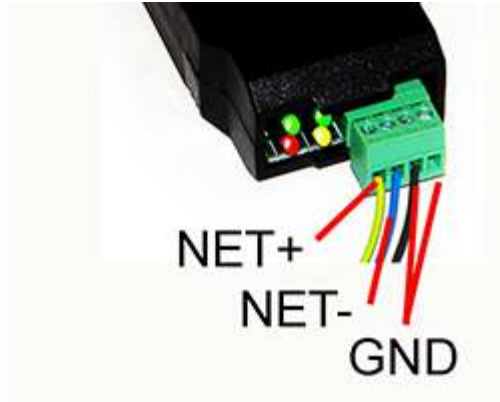
Mode	Request LED	Reply LED
Gateway is Master	Flash yellow each time master (gateway) sends a request to a remote slave.	Flash green when master receives a good response. Flash red when master receives exception message from slave, or if timed out with no response from slave (including CRC error in response which is discarded).
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.

J.3 MTX002 USB to MS/TP Adapter

The MTX002 is an intelligent USB to MS/TP adapter - it is not a generic USB to RS-485 adapter. The MTX002 is itself a BACnet MS/TP device. Its purpose is to offload the token passing from the Windows PC since Windows often does not do a sufficiently accurate job of timing the token passing. In addition, PC hardware is incapable of

operating at the 76800 baud rate common to MS/TP. The MTX002 does allow the PC to access MS/TP networks running at 76800 baud. Configuration software provided for the MS/TP gateways expects to use the MTX002 adapter, or access MS/TP via a BACnet IP to MS/TP router provided by the user.

Note that you must install the USB driver (see Appendix H) prior to attempting to use the MTX002.



Connect the NET+ and NET- terminals to the corresponding MS/TP terminals on the gateway (or to +/- on the MS/TP network). Connect GND to network common.

DO NOT connect power to the terminal block on the adapter. The adapter gets its power from the USB port of your PC. The adapter only draws a very small amount of power, well below the limit provided by standard USB ports.



The yellow LED next to the USB connector will light up any time the PC recognizes this device. There is a blue power LED inside the adapter visible by looking into the adapter through the slot next to the USB adapter. The power LED should always be on if connected to the PC. The yellow LED will only come on when recognized by PC software.

The LEDs next to the terminal block indicate network traffic on the MS/TP network.

The Token LED flashes green each time the adapter passes the token. The adapter functions as another MS/TP device on the network, and gets into the token passing loop just like any other MS/TP device.

The PFM LED flashes yellow each time the adapter sends a Poll For Master (PFM).

The Data LED flashes green each time a good packet other than token or PFM is

received, or a packet other than token or PFM is sent, by the adapter.

The Error LED flashes each time a valid packet is received by the adapter, but resulted in an error. The Error LED can also indicate a timeout waiting for the slave device to respond (typically the device being configured).

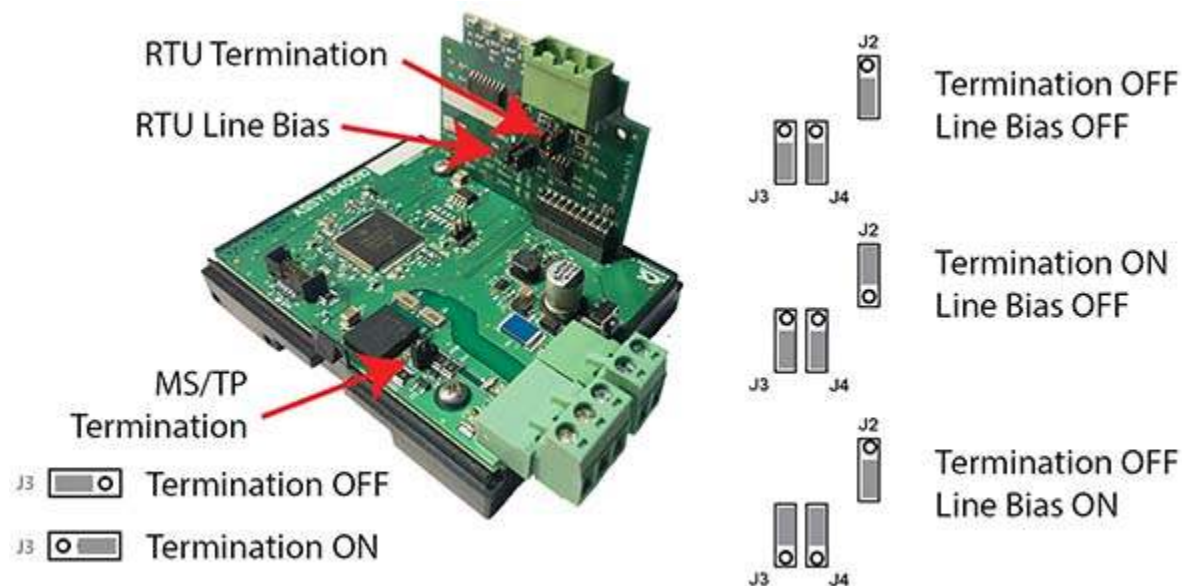
J.4 RS-485 Line Termination & Bias

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.

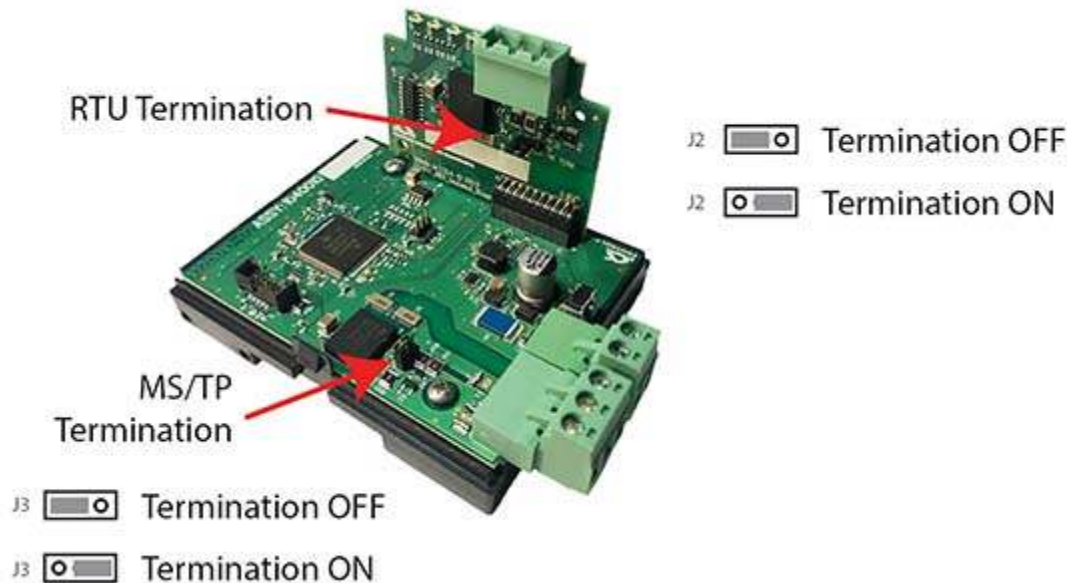
Enable line bias when needed. Line bias should only be enabled at one point on the network, and does not have to be the end point. Line bias holds the line in a known neutral state when no devices are transmitting. Without bias, the transition from offline to online by a transmitter can look like a false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

Termination header locations for BB3-3101 with standard non-isolated Modbus RTU port:

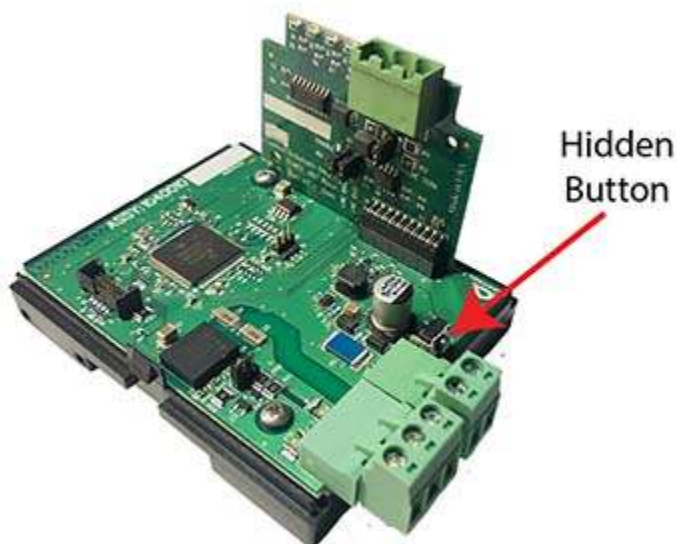


Termination header locations for BB3-3101 with optional isolated Modbus RTU port:



J.5 Restore Factory Defaults

There is a hidden button used to restore factory defaults. The button is shown with the cover removed in the picture below so you can see where it is, but it is not necessary to remove the cover to press the button. Simply press on the plastic tab next to the terminal block and you should feel a light click.



There are two "codes" for restoring factory default.

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

Code 2: Reset communication parameters and erase all configuration.

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

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

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

Code 1: Yellow/amber LEDs flash once and then pause (and then repeat).

Code 2: Red LEDs flash twice and then pause (and then repeat).

Confirmation: All LEDs flash green showing either code 1 or code 2 (one or two flashes followed by a pause, and then repeat).