



User Guide

Babel Buster 2 Model BB2-6040 Modbus ZigBee Gateway

Rev. 1.0 – October 2010

User Guide

Babel Buster 2 Model BB2-6040 Modbus ZigBee Gateway

Rev. 1.0 – October 2010

IMPORTANT SAFETY CONSIDERATIONS:

Proper system design is required for reliable and safe operation of distributed control systems incorporating Babel Buster series gateways and other such devices. 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 the Babel Buster series gateway or any other 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.

© 2010 Control Solutions, Inc.

BACnet® is a registered trademark of American Society of Heating, Refrigerating and Air-Conditioning Engineers (ASHRAE). Babel Buster® is a registered trademark of Control Solutions, Inc., Minnesota, USA. All other trademarks mentioned in this document are the property of their respective owners. Information in this document is subject to change without notice and does not represent a commitment on the part of Control Solutions, Inc. This document 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 manual or in the product(s) and/or the program(s) described in this manual at any time. This product could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes may be incorporated in new editions of the publication.

Contents

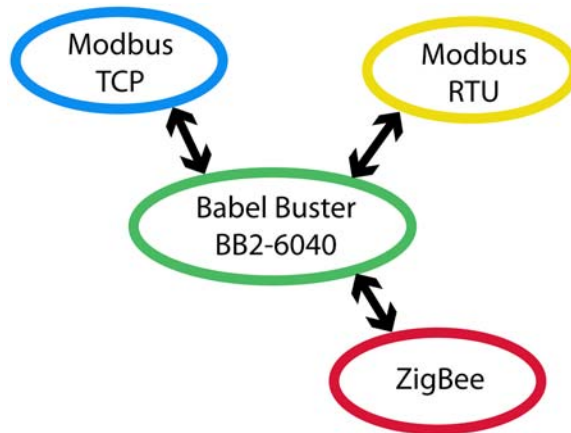
1	Overview	1
1.1	How to Use This Guide	1
1.2	The BB2-6040's Role on the ZigBee Network	1
1.3	Object Server Model for a Gateway	2
1.4	Overview of the BB2-6040	2
2	Out of the Box – Connecting for the First Time	4
3	Initial Connection of ZigBee Device	8
4	Creating a Data Parse Mask	12
5	Setting Up the ZigBee Device List	14
5.1	Network Identifier	14
5.2	Fixed 64-bit Address	14
5.3	Search String	15
6	Setting Up ZigBee Client Read and Write Maps	16
6.1	ZigBee Client Read Maps	16
6.2	ZigBee Client Write Maps	19
7	Setting Up Modbus RTU	22
7.1	Modbus RTU Device Configuration	22
7.2	Modbus RTU Slave Operation	23
7.3	Modbus RTU Master Read Maps	23
7.4	Modbus RTU Master Write Maps	25
7.5	Modbus RTU Master Data Displayed Per Slave	27
7.6	Modbus RTU Errors	28
8	Setting Up Modbus TCP	30
8.1	Modbus TCP Device Configuration	30
8.2	Modbus TCP Client Read Maps	31
8.3	Modbus TCP Client Write Maps	33
8.4	Modbus TCP Errors	35
8.5	Modbus TCP Server (Slave) Operation	36
8.6	Modbus TCP Server Virtual Device Configuration	36
9	System Setup	40
9.1	Configuration Files	40
9.2	Local Port Settings	41
9.3	User Accounts and Passwords	42
10	Data Objects	43
11	ZigBee Diagnostic Pages	44
12	Programming ZigBee I/O in Basic	49
12.1	Web Interface Pages for Programming	49
12.2	Script Basic Enhancements Specific to ZigBee	51
13	Trouble Shooting	54
13.1	Modbus RTU Trouble Shooting	54
13.2	Modbus TCP Trouble Shooting	56
13.3	ZigBee Trouble Shooting	58
Appendix A	Connecting Control Solutions Wireless Devices	59
A.1	Configuring the Temperature Sensor	59
A.2	Configuring the Router	59
A.3	Generic Configuration and the X-CTU Profile	59
Appendix B	Connecting Digi International Wireless Devices	61
B.1	Configuring the Digi XBee Sensor	61
B.2	Configuring the Digi XBee SmartPlug	61
B.3	Configuring the Digi Wall Router	61
Appendix C	Connecting Point Six Wireless Devices	62
C.1	Changes Required in the BB2-6040	62
C.2	Compatibility Between Point Six and Digi Sensors	62
C.3	Configuring BB2-6040 to Recognize Point Six Sensors	62
Appendix D	Gateway Setup Quick Start	66
	Revision History	67



1 Overview

1.1 How to Use This Guide

Section 1 gives an overview of the BB2-6040 gateway. Section 2 talks about connecting your PC to the BB2-6040 for the first time. Section 3 talks about connecting your first ZigBee device. The remaining sections provide additional detail about using the BB2-6040.



1.2 The BB2-6040's Role on the ZigBee Network

The Babel Buster BB2-6040 will function as the Coordinator on the ZigBee network. Other devices on the network should be Endpoints or Routers. If you will have 10 endpoints or less, you do not need routers except to solve range problems. If you will have more than 10 endpoints, you need routers to coordinate traffic back to the coordinator.

The most anticipated application scenario that the BB2-6040 is intended to support is a network of wireless sensors connected to a wired Modbus network. In this instance, it is important for the BB2-6040 to be the coordinator since it needs to be on the receiving end of all ZigBee traffic. The other possible application for the BB2-6040 is to make a wireless connection to a piece of Modbus equipment where the BB2-6040 and the Modbus equipment is intended to be a

slave to a larger system. In this instance, the BB2-6040 should be reprogrammed with the ZigBee module functioning as a router. Contact the factory about obtaining a router version of the BB2-6040.

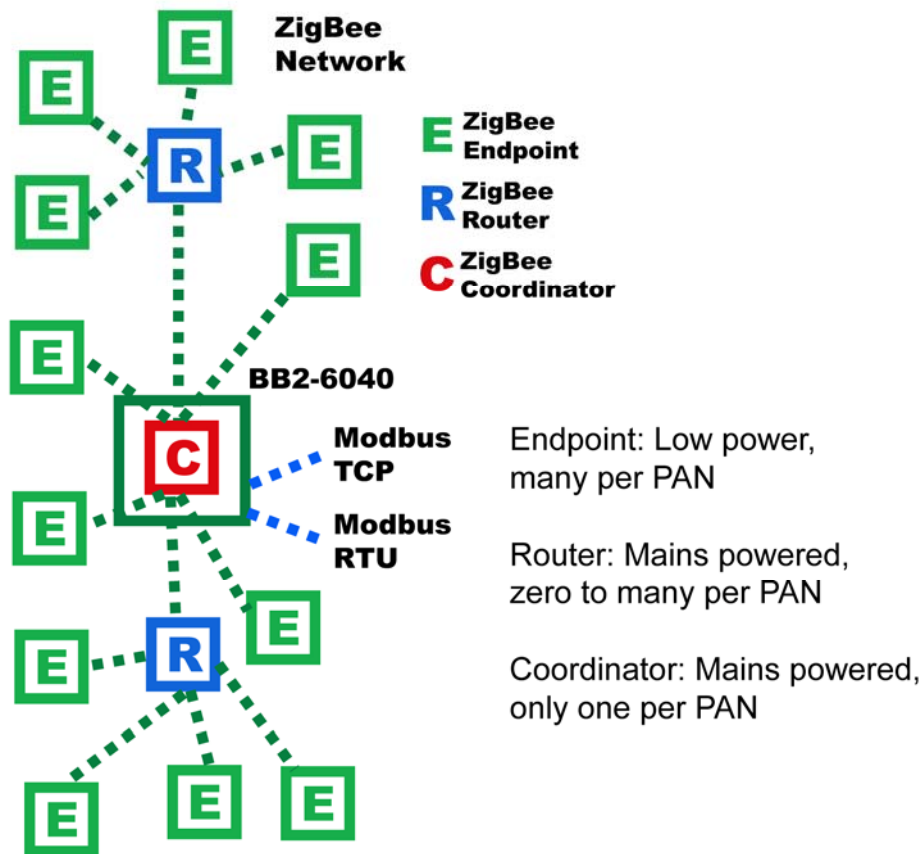
Using ZigBee as a wireless link between a Modbus device and the Modbus network, with two BB2-6040's – one on each end – is possible. However, only one of the BB2-6040's can be a coordinator, meaning one must be reprogrammed as a router. Exchange of a small set of simple data points can be supported in this manner, although standard WiFi would be a better choice for wireless Modbus TCP.

1.3 Object Server Model for a Gateway

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

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

Control Solutions gateways 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.



1.4 Overview of the BB2-6040

Control Solutions' Babel Buster® BB2-6040 is a Modbus client/server device that functions as a ZigBee Coordinator and gateway. After setting up the appropriate mappings in the BB2-6040, data from ZigBee sensors will appear as

Modbus registers on Modbus TCP and/or Modbus RTU. Data written to Modbus registers can also be propagated to ZigBee actuators by the BB2-6040.

The BB2-6040 expects most ZigBee endpoints to be low power devices that spend most of their time sleeping. Therefore the traditional gateway behavior of polling slave devices will not work. The BB2-6040 operates primarily in Listen mode, simply waiting for ZigBee devices to report in. When the ZigBee devices do report in, the BB2-6040 searches its known device list and data maps to see if the reported data should be saved as a local Modbus register's value.

The collection of Modbus registers found in the BB2-6040 will maintain the most recently received ZigBee. Other Modbus clients (masters) may then query the BB2-6040 server to obtain that most recent data using a simple holding register read.

Modbus clients may use a holding register write request to any of the BB2-6040's local registers to send data to a ZigBee device when mapped for writing. The BB2-6040 will transmit to the ZigBee device, and a properly configured ZigBee device will receive the transmission at the end of its sleep period.

The BB2-6040 may also function as a Modbus client, and will use holding register write requests to share data with other Modbus servers (slaves). The BB2-6040 therefore has the ability to transfer data between ZigBee devices and other Modbus devices that operate as a slave only.

The BB2-6040 initial release provides some degree of higher level support for ZigBee products transmitting raw data packets. The ZigBee Alliance has approved two functional profiles as of this writing. The HVAC profile is still in development. As specifications become available, along with devices that implement the profiles, the BB2-6040 will be updated. Firmware in the BB2-6040 is field upgradeable.

The 8/18/2009 version of the XBee OEM manual includes some discussion of ZigBee Device Objects (ZDO) and XBee support of ZDO. The BB2-6040 will not prevent use of any of the ZDO support; it simply does not automatically do ZDO support for you. It will at some later time as implementation of ZDO becomes more wide spread.

The BB2-6040 is based on the XBee OEM module from Digi International. The API command reference may be found in section 10 in the document at this link (as of March 2010):
http://ftp1.digi.com/support/documentation/90000976_D.pdf

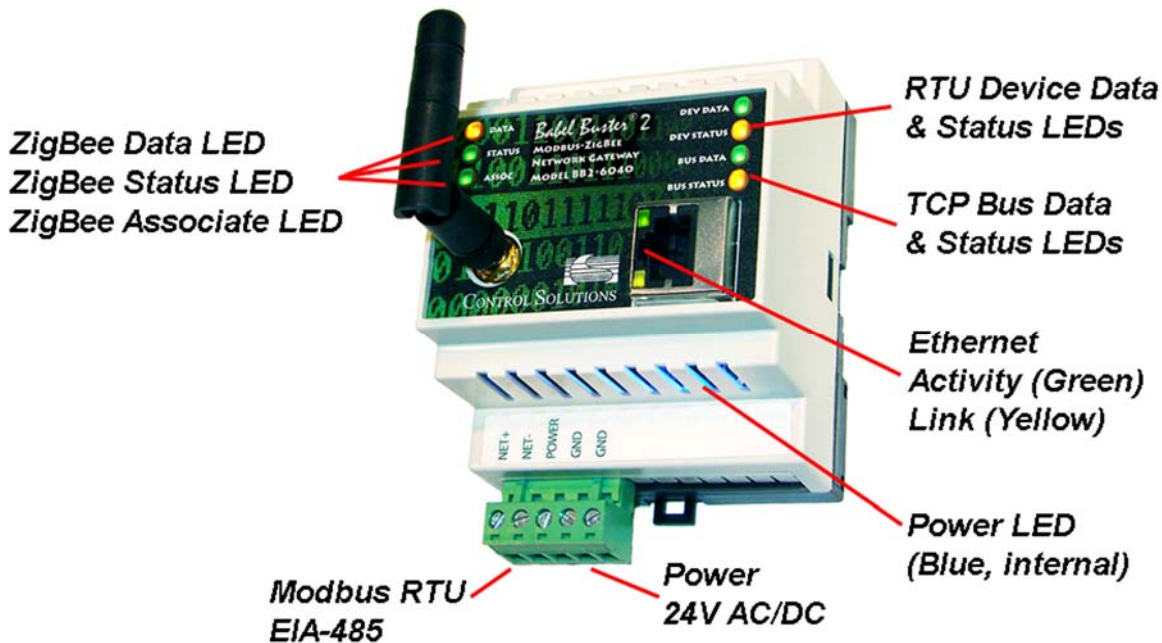
If this link is no longer valid, contact support@csimn.com for an updated link.

Available Models:

BB2-6040-US	Standard BB2-6040 for North America and Australia, ZigBee Coordinator
BB2-6040-EU	Standard BB2-6040 for Europe (lower powered RF), ZigBee Coordinator
BB2-6040-US-R	BB2-6040 North America/Australia, ZigBee Router (non-stock, build to order)
BB2-6040-EU-R	BB2-6040 Europe, ZigBee Router (non-stock, build to order)

2 Out of the Box – Connecting for the First Time

You only need to connect 24V power and Ethernet to begin working with the BB2-6040. Connect the Ethernet via a switch (or crossover cable) as you would any other Ethernet network device.



Upon power-up, the embedded server will take about a minute to boot up. A blue LED should be visible inside the device indicating power is present. Both the Ethernet Link and Activity LEDs should light up upon power-up assuming Ethernet is connected to a switch. After bootup, the Link LED indicates a network connection is present, and the Activity LED will flash any time network traffic is detected.

The ZigBee Status LED should come on solid green and remain on. The Associate LED should flash green at a rate of about once a second. When data is received by the ZigBee radio, the data light will flicker. If you do not have any ZigBee devices powered up, this LED will remain off.

You will need to start by establishing a connection between your PC and the BB2-6040. The BB2-6040 comes preconfigured to a static IP address of 10.0.0.101. This will be the local domain your PC is already on with certain types of network routers. If this is the case, you need to be sure nothing else is at 10.0.0.101 before putting it on the network, or simply do not connect it to your office network.

If your PC is on some other domain (192.168.0.0 is a common one), you will need to set up a temporary routing table entry on your PC. Start by finding out what IP address your PC is currently at. Do this by opening a command prompt, and typing the command 'ipconfig'. Note the address.

```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\Jim Hogenson>ipconfig

Windows IP Configuration

Ethernet adapter Local Area Connection:

    Connection-specific DNS Suffix  . : Home
    IP Address. . . . .               : 192.168.1.101
    Subnet Mask . . . . .             : 255.255.255.0
    Default Gateway . . . . .         : 192.168.1.1

C:\Documents and Settings\Jim Hogenson>
```

Once you have identified your PC's IP address, use the route command to add a routing table entry. The syntax, using the above IP address as the example, would be:

```
route add 10.0.0.0 mask 255.255.255.0 192.168.1.101
```

Note that the mask you use should also match whatever you found with ipconfig.

```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\Jim Hogenson>ipconfig

Windows IP Configuration

Ethernet adapter Local Area Connection:

    Connection-specific DNS Suffix  . : Home
    IP Address. . . . .               : 192.168.1.101
    Subnet Mask . . . . .             : 255.255.255.0
    Default Gateway . . . . .         : 192.168.1.1

C:\Documents and Settings\Jim Hogenson>route add 10.0.0.0 mask 255.255.255.0 192
.168.1.101

C:\Documents and Settings\Jim Hogenson>_
```

Once you have done the route command, you can simply do 'route print' to verify your routing table. The result of 'route print' should look somewhat like the example that follows, although this will vary greatly from one network and PC to another.


```

C:\WINDOWS\system32\cmd.exe
=====
Interface List
0x1 ..... MS TCP Loopback interface
0x2 ..00 1e 4f ea a1 36 ..... Broadcom NetXtreme 57xx Gigabit Controller - Pac
ket Scheduler Miniport
=====
Active Routes:
Network Destination        Netmask          Gateway          Interface        Metric
0.0.0.0                    0.0.0.0          192.168.1.1     192.168.1.101    20
10.0.0.0                   255.255.255.0    192.168.1.101  192.168.1.101    1
127.0.0.0                  255.0.0.0        127.0.0.1       127.0.0.1        1
169.254.0.0                255.255.0.0      192.168.1.101  192.168.1.101    20
192.168.1.0                255.255.255.0    192.168.1.101  192.168.1.101    20
192.168.1.101              255.255.255.255  127.0.0.1       127.0.0.1        20
192.168.1.255              255.255.255.255  192.168.1.101  192.168.1.101    20
224.0.0.0                  240.0.0.0        192.168.1.101  192.168.1.101    20
255.255.255.255           255.255.255.255  192.168.1.101  192.168.1.101    1
Default Gateway:          192.168.1.1
=====
Persistent Routes:
Network Address           Netmask          Gateway Address  Metric
10.0.0.0                  255.255.255.0    192.168.1.101    1
C:\Documents and Settings\Jim Hogenson>

```

Once your route entry is established, open your browser, and browse to “http://10.0.0.101”. You should see the “home” page of the BB2-6040 illustrated below.



[Hardware Guide](#)

[System Capacity Summary](#)

[Modbus Notes](#)

Model BB2-6040
sv3.02.2/5.1 cv3.1

Quick Help

You may now begin to browse your BB2-6040. You will need to log in with a password. There are two default user accounts pre-configured as shipped:

- User: system Password: admin
- User: root Password: buster (must log in as ‘root’ to change IP address)

Once you have established a connection with the gateway, you can proceed to change its IP address. Set the IP address and subnet mask to something more compatible with your network. If your BB2-6040 will be attempting to connect to other IP devices, you must also specify the gateway.

You must be logged in as “root” in order to change IP address. If not logged in as root, close your browser, and re-login. Then change the IP address and click the Change IP button. The new address will be saved to non-volatile memory which takes several seconds. Once the update is done, you must reboot (power cycle) to get the new IP address to take effect.



Babel Buster 2
MODBUS-ZIGBEE
NETWORK GATEWAY
MODEL BB2-6040

CONTROL SOLUTIONS, INC.
MINNESOTA

Data Objects | ZigBee | Modbus | **System Setup**

Setup | Programming

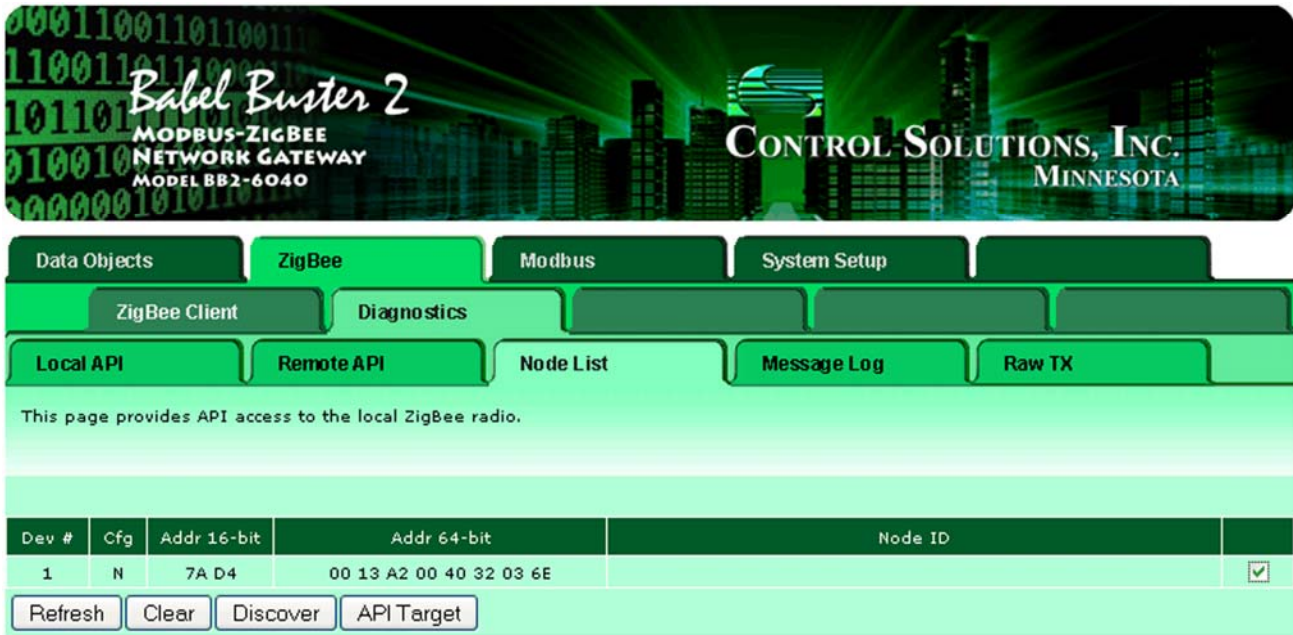
Config File | Local Host | User

This page allows you to change this device's IP address, and select whether double registers are swapped when returned to a remote client accessing this server.

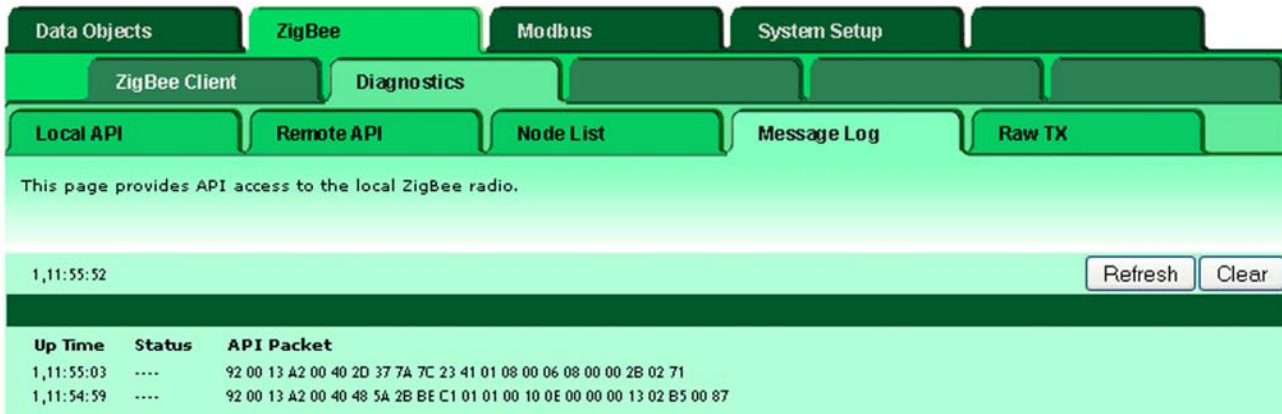
IP Address	<input type="text" value="192.168.1.65"/>	192.168.1.65	<input type="button" value="- Refresh -"/>
Subnet Mask	<input type="text" value="255.255.255.0"/>	255.255.255.0	<input type="button" value="Change IP"/>
Gateway	<input type="text" value="192.168.1.1"/>	192.168.1.1	
HTTP Port	<input type="text" value="80"/> (default 80)		<input type="button" value="Set Port"/>

3 Initial Connection of ZigBee Device

This section will show the general sequence of activity for connecting a ZigBee device for the first time. We will be using a generic XBee Series 2 ZigBee device with firmware programmed to be an endpoint using API mode. This will be typical of Control Solutions sensors, Digi International sensors, and various other ZigBee devices.

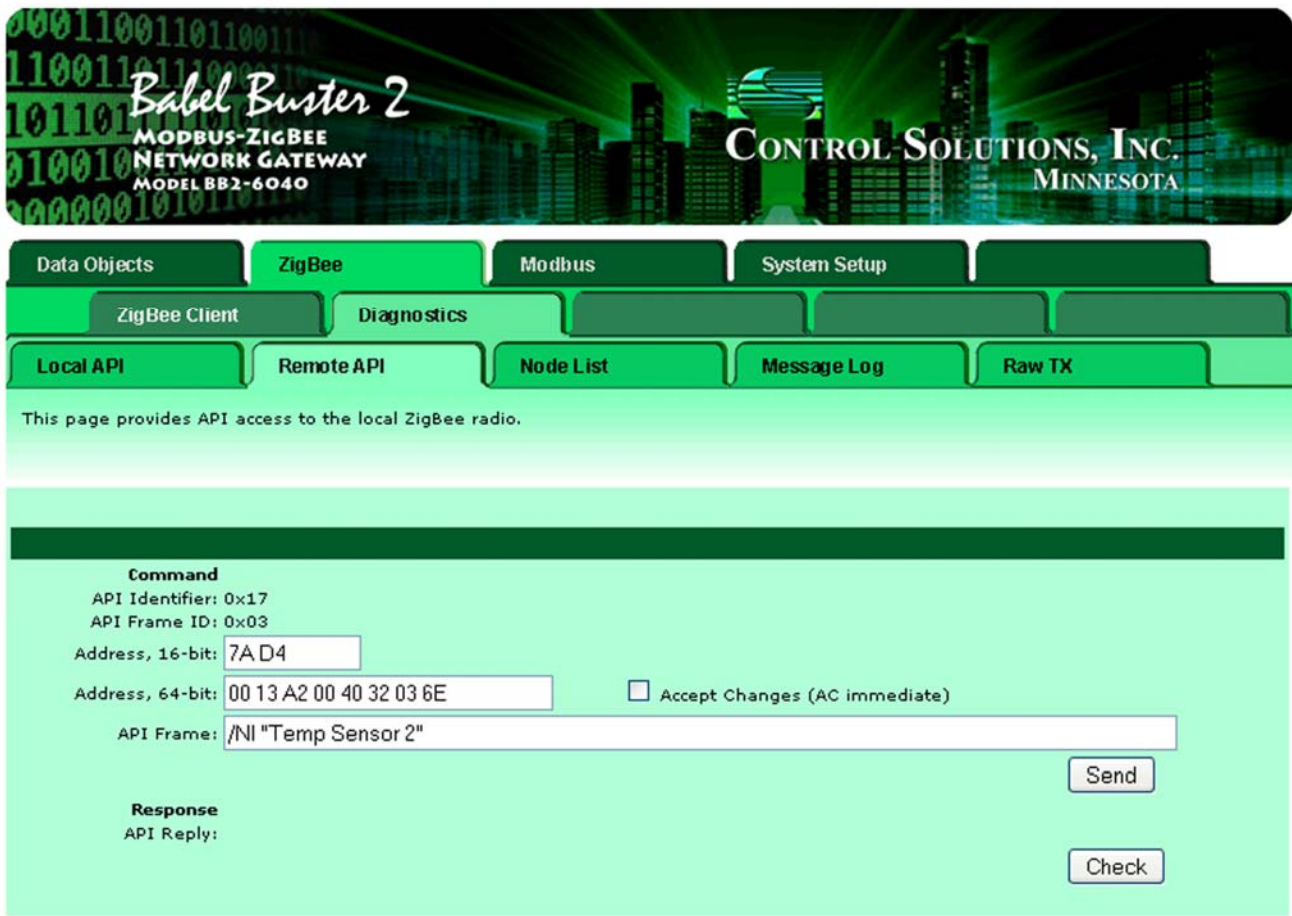


When a device is first connected, it will hopefully show up in the Node List. For this to happen, it must be configured with the same stack profile as the BB2-6040 (ZS parameter), and must be set for the same PAN ID, or set to “join any” PAN. Devices configured to use AT mode instead of API mode will also not show up here, but that does not mean you cannot communicate with them. Check the Message Log page to see if your device is sending anything the BB2-6040 can hear.



The first byte in the packet is always the packet type. The next 8 bytes are the device’s 64-bit address. This is always unique per device. It will generally be found on a label or documentation accompanying the device. It is essentially the MAC address of this ZigBee device.

Assuming your device shows up on the Node List, you can check the box at the end of the row and click API Target. This will automatically copy this device’s address to the address windows of the Remote API page. Next, go to the Remote API page.

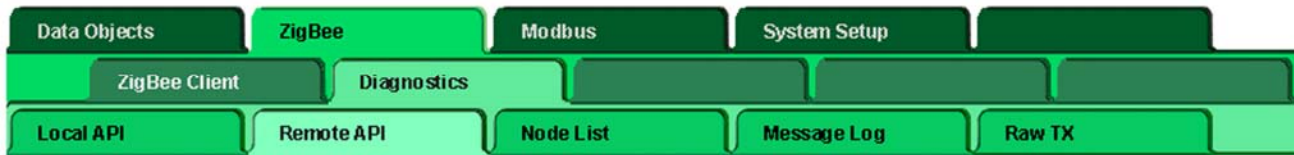


One of several commands you will send to a remote device, assuming it recognizes remote API commands, will be the NI or Network Identifier command. This assigns a name to this device, and after the WR command is sent, it becomes stored in the remote ZigBee device's non-volatile memory.

Commands sent to remote devices ultimately turn into a string of raw binary bytes most often represented as two hexadecimal digits. If you recognize 'FF' as meaning decimal 255, you've got this hex stuff figured out already. If not, go to Wikipedia.com and search for 'hexadecimal'.

The BB2-6040 provides some short cuts in its command syntax to make the commands more user friendly. Any AT command to be sent to the remote device using remote API commands can be typed as /AB where 'AB' is the 2-character AT command. Follow the command with hexadecimal bytes separated by spaces, character strings preceded by '/' (with no embedded spaces), or a string including spaces delimited by quotes.

The above example shows setting the Network Identifier to be "Temp Sensor 2". As soon as you click Send, the hexadecimal interpretation actually sent will be displayed instead, as illustrated below. Initially there will be no reply showing. It takes a little bit of time for a reply to come back most (but not all) of the time.



This page provides API access to the local ZigBee radio.

Command

API Identifier: 0x17
 API Frame ID: 0x07

Address, 16-bit:

Address, 64-bit: Accept Changes (AC immediate)

API Frame:

Response

API Reply:

Once Send has been clicked and you see the confirmation of the packet actually sent in hexadecimal form, click Check to see if you have received a reply yet. The reply will be displayed as illustrated below when received.

When a device is in sleep mode, it is important to note that your Send will not be received until the end of the device's sleep period. As first shipped, devices are often not configured to sleep. But once configured, you will have to wait a while for your response. While it may seem inconvenient, the longer you have to wait for each response, the longer your battery life is going to be since you now know the sleep period is long.

The reply received in response to the above command is illustrated below. The first several bytes are confirmation of the address of the device. The last three bytes are of interest. They should be the 2-character AT command followed by a status byte. A status byte of '00' means the command was executed ok. A nonzero value means something was wrong, such as invalid parameter, invalid command, etc.

This short example is simply an overview of seeing a ZigBee device come online for the first time. The /NI command is an example of only one of several commands you will need to send to a typical sensor before it is fully configured. More discussion about configuration is yet to come.



Navigation menu with buttons: Data Objects, ZigBee, Modbus, System Setup, ZigBee Client, Diagnostics, Local API, Remote API, Node List, Message Log, Raw TX.

This page provides API access to the local ZigBee radio.

Command
API Identifier: 0x17
API Frame ID: 0x07
Address, 16-bit:
Address, 64-bit: Accept Changes (AC immediate)
API Frame:

Response
API Reply: 97 07 00 13 A2 00 40 32 03 6E 7A D4 4E 49 00

Status codes for XBee endpoint devices using API mode are as follows:

- 00 No error
- 01 Error (unspecified)
- 02 Invalid Command
- 03 Invalid Parameter
- 04 Remote command transmission failed.

4 Creating a Data Parse Mask

The data parse mask may have been provided for you in a related document pertaining to a specific endpoint, or in an appendix of this document. If you need to create a new data parse mask for a new device, start by getting the device online and sending its data packets to the BB2-6040. Correlate the packets observed with the manufacturers documentation to begin to recognize its data.

The screen shot below shows several packets observed in the Message Log. The line marked “a” was received with no rule recognizing that packet. The packets marked “b” were received and recognized by rules with data parse masks. The packets at “c” show the gateway sending a command to a device and receiving its reply.

The screenshot shows the Babel Buster 2 Modbus-ZigBee Network Gateway interface. The top banner includes the product name and the logo for CONTROL SOLUTIONS, INC. MINNESOTA. Below the banner are navigation tabs for Data Objects, ZigBee, Modbus, and System Setup. Under the ZigBee tab, there are sub-tabs for ZigBee Client, Diagnostics, Local API, Remote API, Node List, Message Log, and Raw TX. The Message Log tab is active, displaying a table of API packets. The first packet is highlighted in red and labeled 'a'. The second and third packets are also highlighted in red and labeled 'b'. The fourth and fifth packets are highlighted in red and labeled 'c'.

Up Time	Status	API Packet
1,23:10:02	----	92 00 13 A2 00 40 48 5A 2B BE C1 01 01 00 10 0E 00 00 00 11 02 BB 00 87 a
1,23:09:45	->Gateway	92 00 13 A2 00 40 32 2F 9D E5 6C 01 01 08 00 0E 08 00 00 35 02 68 02 02 b
1,23:09:44	->Gateway	92 00 13 A2 00 40 32 2F 93 9D 8E 41 01 08 00 0E 08 00 00 3F 02 74 02 04 b
1,23:09:40	->Gateway	92 00 13 A2 00 40 2D A0 F5 45 34 01 01 08 00 0E 08 00 00 2E 02 7B 02 0E
1,23:09:38	->Gateway	97 95 00 13 A2 00 40 48 5A 2B BE C1 44 34 00
1,23:09:38	Gateway->	17 95 00 13 A2 00 40 48 5A 2B BE C1 02 44 34 04 c

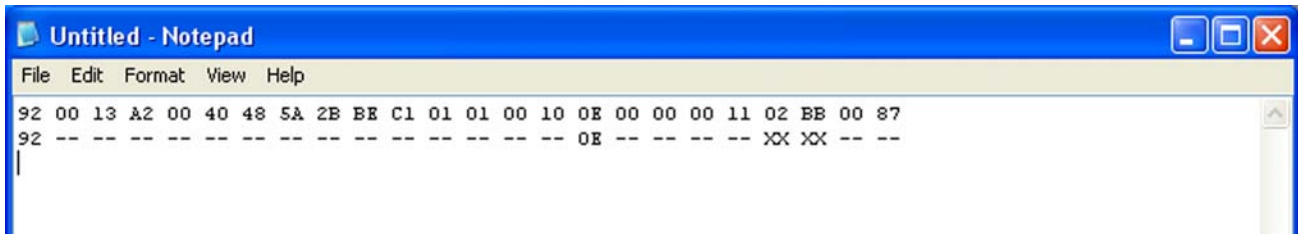
Let us copy the unrecognized packet and paste into a program like Notepad.

The screenshot shows the Babel Buster 2 interface with the message log. The first packet is highlighted in blue, and its hex value is copied to the clipboard.

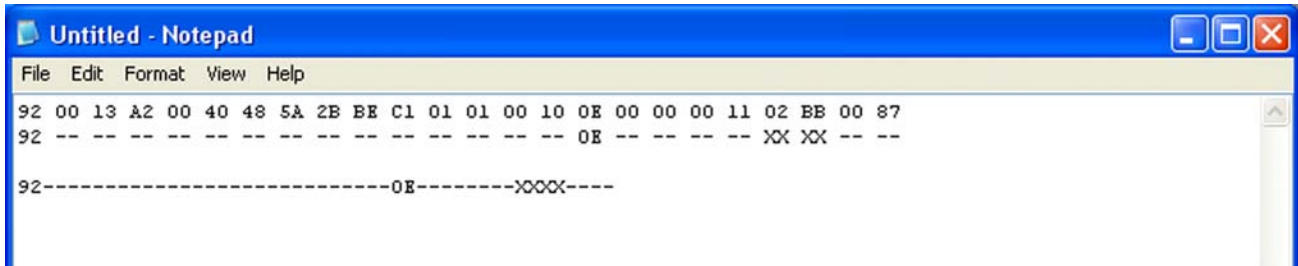
Up Time	Status	API Packet
1,23:10:02	----	92 00 13 A2 00 40 48 5A 2B BE C1 01 01 00 10 0E 00 00 00 11 02 BB 00 87
1,23:09:45	->Gateway	92 00 13 A2 00 40 32 2F 9D E5 6C 01 01 08 00 0E 08 00 00 35 02 68 02 02
1,23:09:44	->Gateway	92 00 13 A2 00 40 32 2F 93 9D 8E 41 01 08 00 0E 08 00 00 3F 02 74 02 04
1,23:09:40	->Gateway	92 00 13 A2 00 40 2D A0 F5 45 34 01 01 08 00 0E 08 00 00 2E 02 7B 02 0E

The screenshot shows a Notepad window titled "Untitled - Notepad". The text in the window is the hex value of the first packet from the message log: 92 00 13 A2 00 40 48 5A 2B BE C1 01 01 00 10 0E 00 00 00 11 02 BB 00 87.

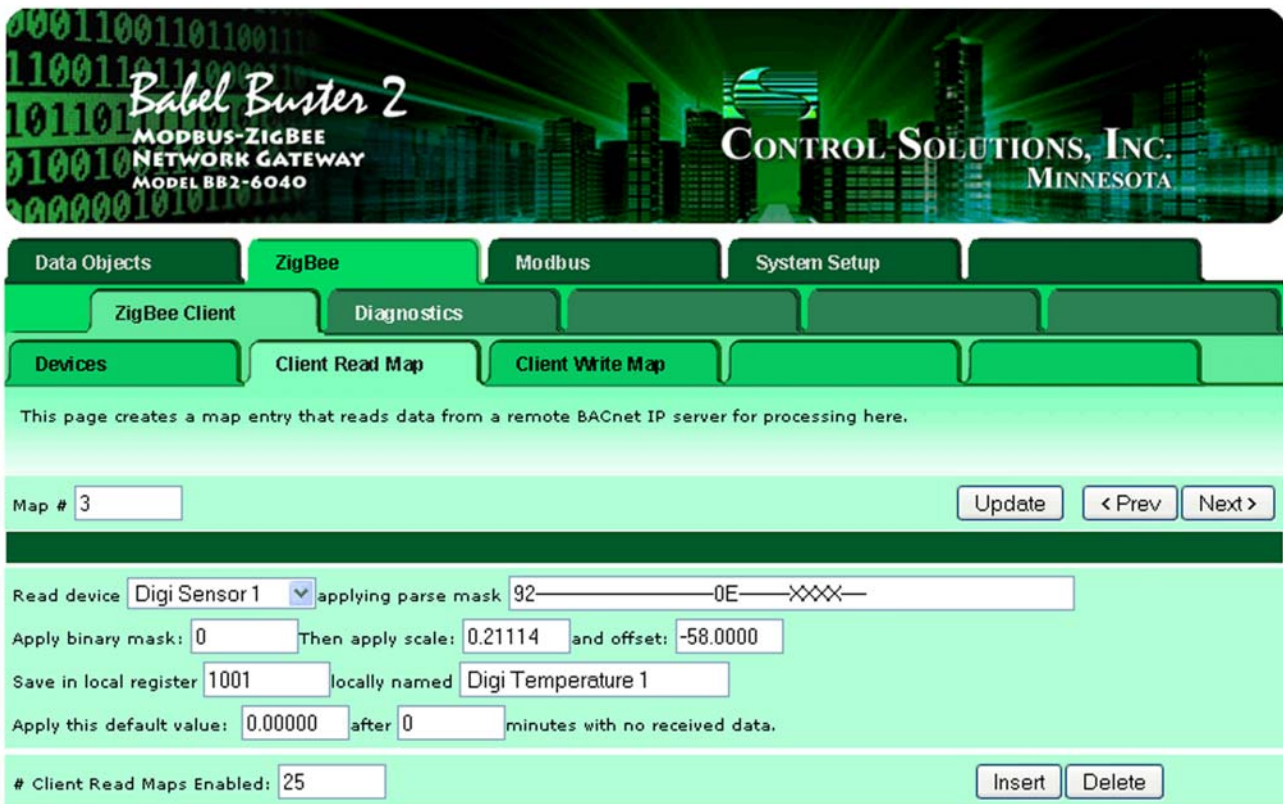
We now begin to construct the data parse mask based on documentation provided by the manufacturer. This example happens to be a Digi XBee sensor. We know that the first byte of the API packet should always be (hex) 92. We will not check the address here since the device handling should do that for us. We can therefore treat the address bytes as “don’t care” at this level, making device substitution easier at a later time. We know that the channel mask should always be (hex) 0E. We know that the next to last pair of bytes is a data point of interest, in this case raw temperature reading.



After typing mask characters out in a one to one correlation with the packet, make another copy of that line and then eliminate all embedded blanks.



Finally, copy the data parse mask into the Client Read Map, select the device from the device list, add scale and offset, select a local object as the destination of the data, and click Update. The BB2-6040 will now be listening for this data.



See Section 6 for further detail on constructing a Data Parse Mask.

5 Setting Up the ZigBee Device List

There are three ways to get the BB2-6040 to recognize a device. Devices must be recognized one way or another so that we know the right data is ending up in the right Modbus registers.

5.1 Network Identifier

The screenshot shows the web interface for the Babel Buster 2 Modbus-ZigBee Network Gateway. The header includes the product name and logo for Control Solutions, Inc. The navigation menu is set to 'ZigBee' > 'Client Read Map'. The main content area is titled 'Devices' and shows configuration for device # 2 of 100. The configuration fields are:

- Node ID (name): CSI Temp Sensor 1
- Discovered Address: 85 3E 00 13 A2 00 40 33 19 80
- Fixed Address: 00 00 00 00 00 00 00 00
- Device Search String: (empty)
- Offset: 0
- Rx Timeout: 10 Minutes
- 0.40 Minutes since last Rx

Buttons for 'Update', '< Prev', and 'Next >' are visible at the top right. A 'Reset' button is located next to the Discovered Address field.

The Network Identifier is a 20-character ASCII string naming the device. The name needs to be unique on the PAN. The combination of a valid ASCII string and non-zero Rx Timeout value signal your desire to have the BB2-6040 try to find this device by Network Identifier. The name will be broadcast, asking any ZigBee device matching that name to reply with its 64-bit address. From that point forward, the BB2-6040 will use the 64-bit address that was reported.

5.2 Fixed 64-bit Address

The screenshot shows the web interface for the Babel Buster 2 Modbus-ZigBee Network Gateway, similar to the previous one but with the Fixed Address field populated. The configuration fields are:

- Node ID (name): CSI Temp Sensor 1
- Discovered Address: 85 3E 00 13 A2 00 40 33 19 80
- Fixed Address: 00 13 A2 00 40 33 19 80
- Device Search String: (empty)
- Offset: 0
- Rx Timeout: 10 Minutes
- 0.24 Minutes since last Rx

Buttons for 'Update', '< Prev', and 'Next >' are visible at the top right. A 'Reset' button is located next to the Discovered Address field.

In the event setting the Network Identifier is not possible, but the 64-bit address is known, you can enter that fixed address along with non-zero Rx Timeout. The name becomes a local reference only. The 64-bit address will be automatically used by the BB2-6040 any time this device is referenced.

5.3 Search String

The screenshot shows a configuration window for a ZigBee device. At the top, it displays "Device # 1 of 100" and navigation buttons for "Update", "< Prev", and "Next >". Below this is a dark green header bar. The main configuration area has a light green background and contains the following fields and buttons:

- Node ID (name):** A text input field containing "Point Six Sensor".
- Discovered Address:** A text input field containing "FF FE 00 13 A2 00 40 3D DB 12". To its right is a "Reset" button.
- Fixed Address:** A text input field containing "00 00 00 00 00 00 00 00".
- Device Search String:** A text input field containing "A01170D0".
- Offset:** A text input field containing "48".
- Rx Timeout:** A text input field containing "10" followed by the text "Minutes".
- 0.34 Minutes since last Rx:** A label indicating the time since the last reception.

A method used by some ZigBee devices for identification is including an ASCII string, such as device serial number, in every data stream sent by that device. When this method is used, enter the string to search for and the byte offset indicating the location in the packet where the search should begin. It is expected that the search string will always begin in a fixed location within the packet. A non-zero Rx Timeout must also be entered to signal that this device entry is now valid.

The Rx Timeout listed here is the amount of time the BB2-6040 will wait for a response to the device search. If the device does not respond within this time, the BB2-6040 simply moves on to the next device and comes back for a retry in about 2 minutes. The "last Rx" time displayed may be either the response to device search, or data received from the device once found.

The Rx Timeout is also used as the reply timeout when a write map sends a remote API packet. The ZigBee radio should respond with either an 'ok' or an error code within a relatively short time after sending a remote API frame (0x17). Transmit frames other than remote API are sent without automatic checking for reply.

6 Setting Up ZigBee Client Read and Write Maps

6.1 ZigBee Client Read Maps

The screenshot shows the Babel Buster 2 Modbus-ZigBee Network Gateway web interface. The header includes the product name and logo for Control Solutions, Inc. The navigation menu is set to 'ZigBee' > 'Client Read Map'. A descriptive text block explains the page's function. Below this is a pagination control showing 'Showing 1 to 15 of 26' and buttons for 'Update', '< Prev', and 'Next >'. The main content is a table with 5 columns: Map #, Remote Device, Data Parse Mask for API Frame, Local Register, and Local Name. The table lists 8 entries for various sensors and their corresponding data parse masks and local registers.

Map #	Remote Device	Data Parse Mask for API Frame	Local Register	Local Name
1	Temp Sensor 2	92-----02XXXX	1001	CSI Temp Sensor
2	Temp Sensor 3	92-----0F-----XXXX--	1003	XBee Temp Sensor
3	Digi Sensor 1	92-----0E-----XXXX--	1005	Digi Temperature 1
4	Digi Sensor 1	92-----0E-----XXXX	1007	Digi Humidity 1
5	Digi Sensor 2	92-----0E-----XXXX--	1009	Digi Temperature 2
6	Digi Sensor 2	92-----0E-----XXXX	1011	Digi Humidity 2
7	Digi Sensor 3	92-----0E-----XXXX--	1013	Digi Temperature 3
8	Digi Sensor 3	92-----0E-----XXXX	1015	Digi Humidity 3

A note about "reading" ZigBee devices: ZigBee is optimized for low power battery operated sensors that sleep most of the time. This means the traditional "polling" of the remote device cannot be done with ZigBee as it can with Modbus or other remote hard wired devices. Most Babel Buster family members are set up to periodically poll remote devices. While the same terms such as "read map" are used in the BB2-6040, the definition of "read" really means "prepared to receive". The read maps found here never initiate any radio traffic, they only respond to receiving radio transmissions from remote ZigBee devices.

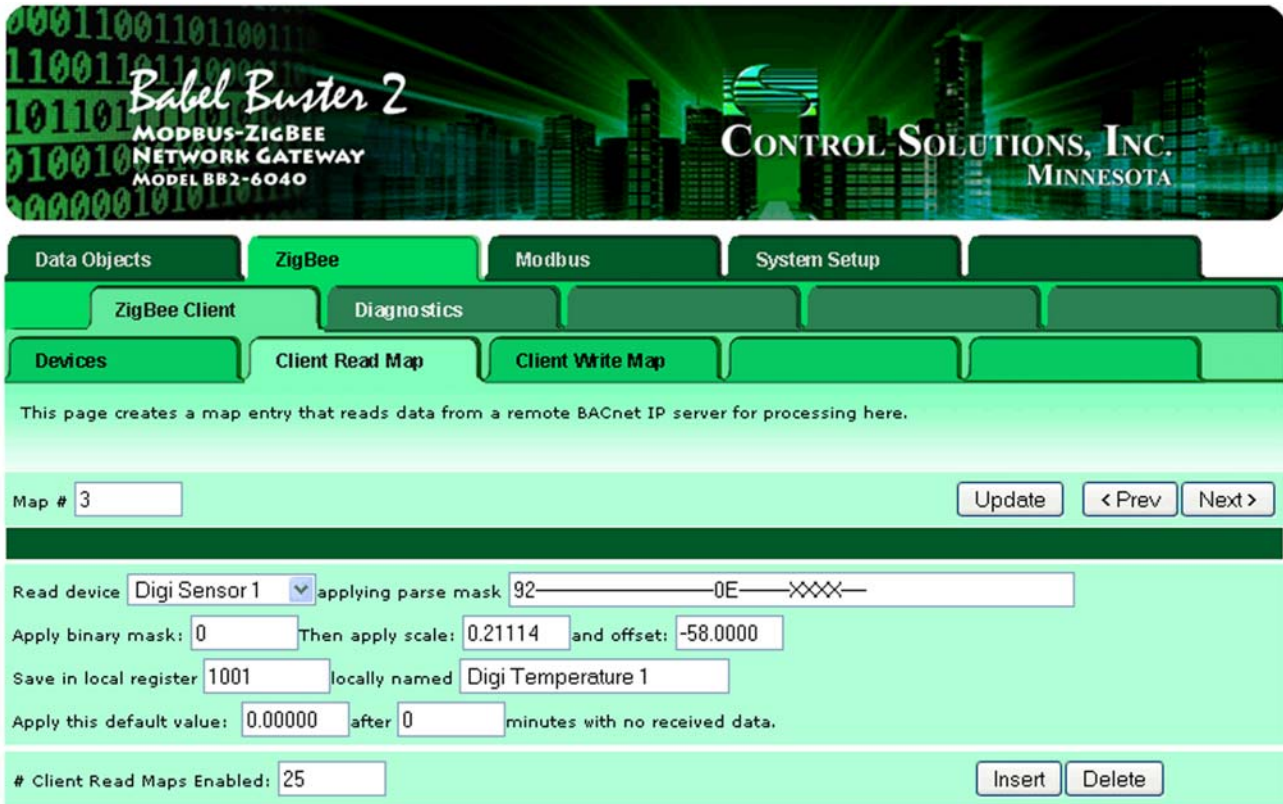
Map number simply tells you where you're at on the list of object maps. Click "next" and "prev" to scroll through the list. To advance directly to a specific map, enter the desired number in the "Showing" box, then click Update.

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

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

For each remote object to be read, select the device from the list, and provide a data parse map. Select a local register where the resulting data should be placed. Details about the data parse mask may be found on the extended map view page by clicking the map number in the left most column.

Local register numbers are 1-999 for integer values, and 1001-1999 accessed as register pairs for floating point. If you try to enter an even number above 1001, you will get an error message. All floating point register pairs start on odd boundaries. All local registers are accessed via Modbus as holding registers.



Clicking map number in the first column of the tabular map list gets you to this view. Rule number simply tells you where you're at on the list of object maps. Click "next" and "prev" to scroll through the list. To advance directly to a specific map, enter the desired number in the "Map #" box, then click Update.

For each remote object to be received, select an object name from the list and provide a data parse mask (see below). The names in the device list are defined in the Devices page.

When the remote data is received, data may be manipulated before being written to the local register. First, binary data is parsed from the received packet based on the data parse mask. Second, the raw data is masked by ANDing with the binary mask if non-zero. Next, the value will be converted to floating point, and the value will be multiplied by the scale factor, then the offset is added. The final result is written to the local register number given (with conversion back to binary or integer as applicable). The name is optional and used only for display purposes.

The periodic poll time determines how often the remote object will be read. This number, if nonzero, will override the default poll time given in the Devices page for the remote device being read.

The default value will be stored into the local register after the timeout period elapses with no data received. Setting the count to zero will disable the default, and the register will retain the most recent value obtained.

Delete will remove the rule number shown in the "Map #" box. Insert will insert a new map before the map number shown, and is used for placing maps between existing maps. It is not necessary to use Insert to add maps to the bottom of the list or to define any map presently having zero for a source object or "none" for remote type.

Selecting "none" for remote device effectively deletes the map even though it will still appear in the list until deleted. Unused maps at the end of the list will always show none as the type. If you wish to prevent these from being displayed, reduce the number of maps enabled.

The number of maps enabled simply limits the scope of map review so that you do not have to review a lot of unused maps. If the displayed maps are used up and you need more, increase the enabled number.

DATA PARSE MASK

As of v3.01, the BB2-6040 is prepared to parse raw binary (integer) data from received ZigBee packets. Most ZigBee devices have very limited resources (keeping power consumption low), and therefore will not support floating point. The BB2-6040 is prepared to accept binary/integer data and scale and convert as necessary to produce analog floating point data. In the event devices are found which do use IEEE754 floating point, support for that format will be added in a later revision (contact tech support).

Getting acquainted with a sensor for the first time will require some experimenting. You must start by configuring and commissioning the ZigBee sensor, causing it to transmit periodic sensor data samples. When data packets start to show up, between inspection of the packets and review of sensor documentation, you can begin to construct a data parsing mask that will result in getting the data of interest into Modbus registers. When a ZigBee transmission is received, its data packet will be parsed into as many registers as match the parsing mask. There may be multiple parsing masks for any given remote device.

An example of a packet received from a remote XBee sensor appears on the first line below:

```
92 00 13 A2 00 40 48 5A 2A 39 A3 01 01 00 10 0E 00 00 00 0F 02 AB 00 89
92 -- -- -- -- -- -- -- -- -- -- -- -- -- 01 -- -- 0E -- -- -- -- -- -- -- XX XX
```

The easiest way to construct a mask for this type of packet is copy it to a text editor, then begin typing mask characters on the line below it as illustrated above. Once a complete mask is assembled, remove the embedded blanks, and copy the mask that appears below into the mask window on this page (above).

```
92-----01----0E-----XXXX
```

The data mask fields are always 2-character pairs (with the exception of “[nn]”), and may be any of the following:

- "--" is a don't care byte. It must be two hyphens for 2 hex digits, and must be in pairs as only even bytes can be skipped.
- "NN" is an exact match value where N is a hex digit; for example if the byte must be 0x92, we put 92 in the mask.
- "XX" is a hex value field to be parsed; field may be XX to XXXX to XXXXXXXX for 8, 16, 32-bit values respectively
- "HH" is a hex digit that is ASCII encoded in the data stream. If ASCII is used, HHHH would be equivalent to XX.
- "[nn]" means skip nn (decimal) bytes; [04] is the same as 8 hyphens.

The first byte in the above example is the API frame type code. The next 10 bytes are the 64-bit and 16-bit addresses of the remote device the packet was received from. Do not force matching on these bytes. These are automatically checked against the addresses found in the device list. You only need to select a device by name.

The example above says we must see 0x92 in the first byte of the API frame, and values 0x01 and 0x0E where shown. We are going to extract data from the positions indicted by XXXX. In this case, we are extracting the current (amps) level from a Digi International Smart Plug. Visit the BB2-6040 product page at www.csimn.com for links to Digi documentation.

6.2 ZigBee Client Write Maps

This page creates a map entry that writes data to one or more remote ZigBee devices from data contained here. Click on map number to see more detail and insert/delete rules.

Showing 1 to 3 of 3 Update < Prev Next >

Map #	Remote Device	Data Format Mask for API Frame	Local Register	Local Name
1	Smart Plug 2	[17][F][A][N][02]D4[X]	1	Smart Plug 2 Relay
2	BB2-7040 ROUTER	[10][F][A][0FF][0FE][00][00][0F][X]	2	To Router
3	None		0	

A note about "writing" ZigBee devices: ZigBee is optimized for low power battery operated sensors that sleep most of the time. This means that any attempt to "write" really means queuing up a message for some time later. Each write map includes a timeout. The meaning of timeout is simply that the local register whose data is supposed to be transmitted will be queued up for a retry. The timeout should always be set longer than the sleep period of the remote ZigBee device intended to receive this data.

Map number simply tells you where you're at on the list of object maps. Click "next" and "prev" to scroll through the list. To advance directly to a specific map, enter the desired number in the "Showing" box, then click Update.

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

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

For each remote object to be written, select the device to write and provide a data formatting mask. Select the local register from which data will be written to the remote device. The names in the device list are defined in the Devices page. Details about the data format mask may be found on the extended map view page by clicking the map number in the left most column.

Selecting "none" for remote device effectively deletes the map even though it will still appear in the list until deleted. Unused maps at the end of the list will always show none as the type.

Babel Buster 2
 MODBUS-ZIGBEE
 NETWORK GATEWAY
 MODEL BB2-6040

CONTROL SOLUTIONS, INC.
 MINNESOTA

Data Objects | **ZigBee** | Modbus | System Setup

ZigBee Client | **Diagnostics** | Client Write Map

Devices | Client Read Map | **Client Write Map**

This page creates a map entry that writes data to one or more remote BACnet IP servers from data contained here.

Map #

Read local register named

Write remote data any time local object has changed by or when minutes have elapsed with no change.

Apply scale: offset: AND mask: OR mask: in format string.

Format string: Write to device

Client Write Maps Enabled:

Clicking the map number in the first column of the tabular map list gets you to this view. Rule number simply tells you where you're at on the list of object maps. Click "next" and "prev" to scroll through the list. To advance directly to a specific map, enter the desired number in the "Map #" box, then click Update.

The local register data may be written periodically, or when it changes, or both. To send upon change (send on delta), check the check box and enter the amount by which the local register must change before being written to the remote device. To guarantee that the remote object will be written at least occasionally even if the data does not change, enter a time value in minutes for periodic write.

Data from the local register may be manipulated before being written to the remote device. The local data is first multiplied by the scale factor. The offset is then added to it. The data is then converted to integer/binary in preparation for transmission to the ZigBee device. After conversion to integer, the data is ANDed with the AND mask, and then ORed with the OR mask. The resulting data is then inserted into the format string (see below).

Delete will remove the rule number shown in the "Map #" box. Insert will insert a new map before the map number shown, and is used for placing maps between existing maps. It is not necessary to use Insert to add maps to the bottom of the list or to define any map presently having zero for a source object or "none" for remote type.

Selecting "none" for remote device effectively deletes the map even though it will still appear in the list until deleted. Unused maps at the end of the list will always show none as the type. If you wish to prevent these from being displayed, reduce the number of maps enabled.

The number of maps enabled simply limits the scope of map review so that you do not have to review a lot of unused maps. If the displayed maps are used up and you need more, increase the enabled number.

DATA FORMAT MASK

As of v3.01, the BB2-6040 is prepared to format raw binary (integer) data transmitted in ZigBee packets. Most ZigBee devices have very limited resources (keeping power consumption low), and therefore will not support floating point. The BB2-6040 is prepared to send binary/integer data and scale and convert as necessary to convert from analog floating

point data. In the event devices are found which do use IEEE754 floating point, support for that format will be added in a later revision (contact tech support).

An example of a format mask is as follows:

```
[17][F][A][N][02]D4[X]
```

For this particular example, we select a device from the list which names a Digi International Smart Plug. Further, we use a value of 1 for the AND mask, and 4 for the OR mask. The result of this configuration is that we will send D4=4 or D4=5 to switch the Smart Plug's relay on and off. In our example, we are taking data from a local Binary Output object. The result is that we now have BACnet control of the Smart Plug's relay being operated via ZigBee.

The fields available for a format mask are as follows:

[nn] is interpreted as a hex number producing 1 byte in the transmit packet

[F] is replaced with the next auto-generated frame ID as 1 byte, hex 0x0F as a number must be [0F]

[A] is replaced with the 64-bit address (as 8 bytes) of the device named in the map

[N] is replaced with 16-bit address (as 2 bytes) of the device named in the map

[X] is replaced with object data producing one byte (8-bit) from the value

[XX] is replaced with object data producing 2 bytes (16-bit)

[XXXX] is replaced with object data producing 4 bytes (32-bit) - data is big endian as required by ZigBee

All other ASCII characters are copied verbatim with their ASCII code placed in the respective byte position in the packet.

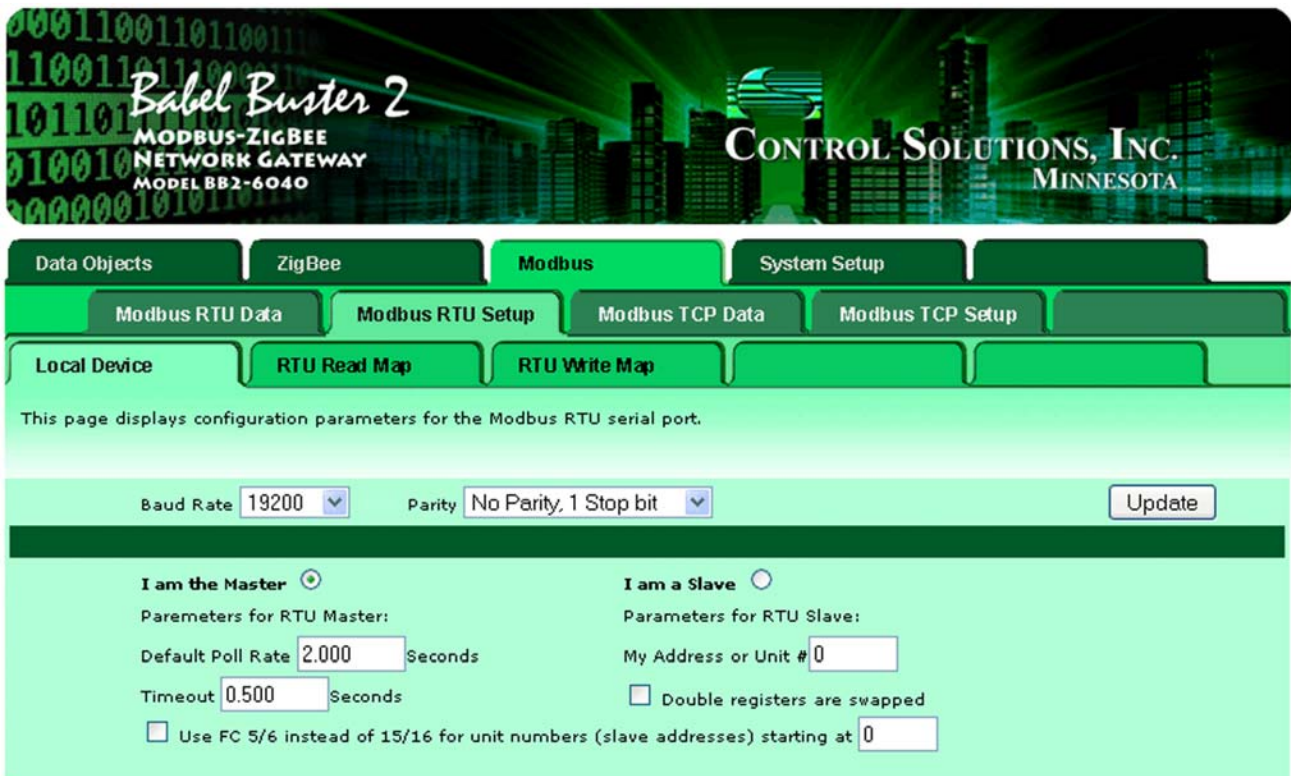
7 Setting Up Modbus RTU

The BB2-6040 can be a Modbus RTU master or slave. As a master you can read Modbus data from, or write Modbus data to, other Modbus slaves. The BB2-6040 will periodically poll the other Modbus devices according to register maps you set up. To read from a remote Modbus device, configure a Read Map. To write to a remote Modbus device, configure Write Map.

Data read from a remote device is stored in a local register when received. Data written to a remote device is taken from a local register when sent. The local registers are the same collection of registers that are accessible to other masters when operating as slave, and accessible to other Modbus TCP devices as a collection of holding registers.

7.1 Modbus RTU Device Configuration

Modbus device configuration for RTU really consists of port configuration, and includes setting the slave address if the BB2-6040 is functioning as Modbus slave.



This page displays configuration parameters for the Modbus RTU serial port.

Baud Rate: 19200 Parity: No Parity, 1 Stop bit

I am the Master I am a Slave

Parameters for RTU Master:

Default Poll Rate: 2.000 Seconds

Timeout: 0.500 Seconds

Use FC 5/6 instead of 15/16 for unit numbers (slave addresses) starting at: 0

Parameters for RTU Slave:

My Address or Unit #: 0

Double registers are swapped

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

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

If your slave/server device only supports function codes 5 and 6 for writing, check the Use FC 5/6 box. The default function codes are 15 and 16, which are most widely used. If you check the box, you should also enter a "starting at" unit # or slave address. This allows supporting both types of devices at the same time provided you assign slave addresses in two non-overlapping groups. (These settings do not apply if the BB2-6040 is the slave.)

The double register swap on this page only applies when the local device (the gateway you are configuring here) is functioning as a Modbus RTU slave.

The term "swapped" only applies to double or float formats. Modbus registers are, by definition, 16 bits of data per register. Access to 32-bit data, either 32-bit integer ("double"), or IEEE 754 floating point ("float"), is supported by the use of two consecutive registers. Modbus protocol is inherently "big endian", therefore, Modbus by the Module defaults to having the high order register first for double and float. If the low order register comes first on the device being accessed, check the "swapped" box.

If you have "swapped" turned around, you will quickly recognize it. If floating point data is reversed, a 1.0 becomes 2.2779508e-41, which simply rounds to zero. The pattern is not as predictable as the 1.0 example would suggest. A floating point 1.1 becomes negative 107609184. If 32-bit integer data is reversed, 1 becomes 65536.

7.2 Modbus RTU Slave Operation

The term "server" is often used to describe the Modbus TCP version of a Modbus slave. A server will provide data when a client asks for it. The concept of master/slave is less significant in Modbus TCP because any TCP device can be both master and slave at the same time, and there can be multiple "masters" on the network. That is in contrast with Modbus RTU where there can be only one master and multiple slaves, and each device must be one or the other.

The Modbus TCP server is simply a collection of registers that may contain data. The source of that data in the case of Babel Buster BB2-6040 can be any of several possible sources. It may be read from another Modbus device. Another Modbus device could have put it there by writing to the BB2-6040. The data could have been received by the ZigBee client.

Local register numbers held by the server are 1-999 for integer values, and 1001-1999 accessed as register pairs for floating point. If you try to enter an even number above 1001, you will get an error message. All floating point register pairs start on odd boundaries. All local registers are accessed via Modbus as holding registers.

7.3 Modbus RTU Master Read Maps

The screenshot shows the web interface for the Babel Buster 2 Modbus-ZigBee Network Gateway. The page is titled "Modbus RTU Read Map" and includes a navigation menu with options like "Data Objects", "ZigBee", "Modbus", "System Setup", "Modbus RTU Data", "Modbus RTU Setup", "Modbus TCP Data", "Modbus TCP Setup", "Local Device", "RTU Read Map", and "RTU Write Map". Below the navigation, there is a descriptive text: "Read remote registers into local registers. This page creates a map entry that reads data from one or more remote Modbus RTU serial devices for processing here. Click on map number to see more detail and insert/delete rules." A "Showing" box displays "1 to 3 of 3" with "Update", "Prev", and "Next" buttons. A table lists three map entries:

Map #	Remote Type	Remote Register Format	Remote Register #	Remote Unit #	Scale	Local Register #	Name
1	Holding Register	Unsigned	1	1	0.000000	1	External Sensor 1
2	Holding Register	Unsigned	2	1	0.000000	2	External Sensor 2
3	None	Integer	0	0	0.000000	0	

Rule number simply tells you where you're at on the list of register maps. Click "next" and "prev" to scroll through the list. To advance directly to a specific map, enter the desired number in the "Showing" box, then click Update.

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

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

For each remote register to be read, enter the register type, format, number, and remote unit (slave address).

When the remote register is read, the data will be multiplied by the scale factor and written to the local register number given. The name is optional and used only for display purposes.

Selecting "none" for remote type effectively deletes the rule even though it will still appear in the list until deleted. Unused rules at the end of the list will always show none as the type.

Local register numbers are 1-999 for integer values, and 1001-1999 accessed as register pairs for floating point. If you try to enter an even number above 1001, you will get an error message. All floating point register pairs start on odd boundaries. All local registers are accessed via Modbus as holding registers.

Click on the map number in the left column of the tabular read map page (above) to get the expanded view of one read map at a time (below).

Babel Buster 2
MODBUS-ZIGBEE
NETWORK GATEWAY
MODEL BB2-6040

CONTROL SOLUTIONS, INC.
MINNESOTA

Data Objects | ZigBee | **Modbus** | System Setup

Modbus RTU Data | **Modbus RTU Setup** | Modbus TCP Data | Modbus TCP Setup

Local Device | **RTU Read Map** | RTU Write Map

This page creates a map entry that reads data from a remote Modbus RTU serial device for processing here.

Map #

Read as from register # at Unit # with doubles swapped

Apply bit mask if applicable: then apply scale: and offset:

Save in local register # named Repeat this process every seconds.

Apply this default value: after read failure(s).

RTU Read Maps Enabled:

Rule number simply tells you where you're at on the list of register maps. Click "next" and "prev" to scroll through the list. To advance directly to a specific map, enter the desired number in the "Map #" box, then click Update.

For each remote register to be read, enter the register type, format, number, and remote unit (slave address).

When the remote register is read, data may be manipulated before being written to the local register. If a bit mask is entered (in hexadecimal), and the remote register type is signed or unsigned (16-bit data), the mask will be bit-wise logical AND-ed with the data, and the retained bits will be right justified in the result. The result will then be multiplied by the scale factor. The offset is then added and this final result is written to the local register number given. The name is optional and used only for display purposes.

The periodic poll time determines how often the remote register will be read. This number, if nonzero, will override the default poll time given in the Devices page for the remote device being read.

The default value will be stored into the local register after the given number of read failures if the fail count is non-zero. Setting the count to zero will disable the default, and the object will retain the most recent value obtained.

Delete will remove the rule number shown in the "Map #" box. Insert will insert a new rule before the rule number shown, and is used for placing rules between existing rules. It is not necessary to use Insert to add rules to the bottom of the list or to define any rule presently having zero for a source object or "none" for remote type.

Selecting "none" for remote type effectively deletes the rule even though it will still appear in the list until deleted. Unused rules at the end of the list will always show none as the type. If you wish to prevent these from being displayed, reduce the number of rules enabled.

The number of rules enabled simply limits the scope of rule review so that you do not have to review a lot of unused rules. If the displayed rules are used up and you need more, increase the enabled number.

7.4 Modbus RTU Master Write Maps

Map #	Local Register #	Scale	Remote Type	Remote Register Format	Remote Register #	Remote Unit #	Name
1	3	0.000000	Holding Register	Integer	3	1	Actuator 1
2	0	0.000000	None	Integer	0	0	

Rule number simply tells you where you're at on the list of register maps. Click "next" and "prev" to scroll through the list. To advance directly to a specific map, enter the desired number in the "Showing" box, then click Update.

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

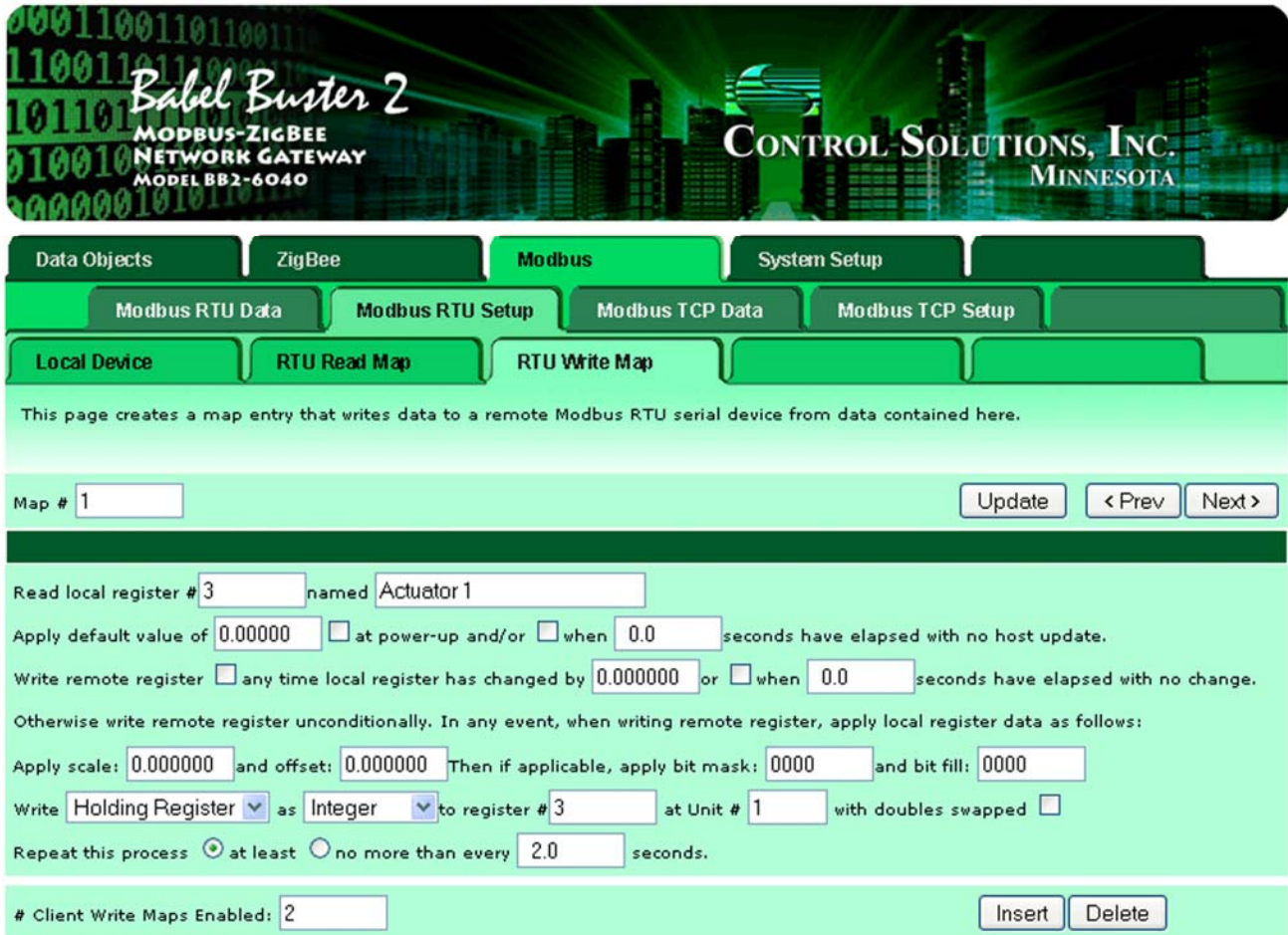
An abbreviated version of a list of rules is shown on this page. Any of the parameters shown may be changed here and registered by clicking the Update button. To view and/or modify the complete set of parameters, click on the map number in the left most column.

Data from the local register given will be multiplied by the scale factor before being written. For each remote register to be written, enter the register type, format, number, and remote unit (slave address).

Selecting "none" for remote type effectively deletes the rule even though it will still appear in the list until deleted. Unused rules at the end of the list will always show none as the type.

Local register numbers are 1-999 for integer values, and 1001-1999 accessed as register pairs for floating point. If you try to enter an even number above 1001, you will get an error message. All floating point register pairs start on odd boundaries. All local registers are accessed via Modbus as holding registers.

Click on the map number in the left column of the tabular write map page (above) to get the expanded view of one write map at a time (below).



Rule number simply tells you where you're at on the list of register maps. Click "next" and "prev" to scroll through the list. To advance directly to a specific map, enter the desired number in the "Map #" box, then click Update.

The local register data may be written periodically, or when it changes, or both. To send upon change (send on delta), check the first box and enter the amount by which the local register must change before being written to the remote device. To guarantee that the remote register will be written at least occasionally even if the data does not change, check the second box and enter some amount of time. This time period will be referred to as the "maximum quiet time".

Data from the local register may be manipulated before being written to the remote register. The local data is first multiplied by the scale factor. The offset is then added to it. If a bit mask is entered, and the remote register type is signed or unsigned (16-bit data), the mask will be bit-wise logical AND-ed with the data. The mask is right justified, then AND-ed with the data. The result is then left shifted back to the original position of the mask. In other words, the least significant bits of the original data will be stuffed at the position marked by the mask.

After the scaling and masking, the bit fill will be logically OR-ed into the result, but only if the mask was nonzero and was used. Both mask and fill are entered in hexadecimal.

Multiple local registers may be packed into a single remote register. To accomplish this, define two or more rules in sequence with the same remote destination. If the destination is the same, data types are 16-bit (integer or unsigned), bit

masks are nonzero, and the rules are sequential, the results of all qualifying rules will be OR-ed together before being sent to the remote destination.

For the remote register to be written, enter the register type, format, number, and remote unit (slave address).

The repeat time may determine how often the remote register will be written. If send on delta and maximum quiet time are not checked above, clicking the "at least" button will establish a periodic update time. If send on delta is used and you wish to limit the network traffic in the event changes are frequent, click the "no more than" button and enter the minimum time that should elapse before another write to the remote device.

Delete will remove the rule number shown in the "Map #" box. Insert will insert a new rule before the rule number shown, and is used for placing rules between existing rules. It is not necessary to use Insert to add rules to the bottom of the list or to define any rule presently having zero for a source register or "none" for remote type.

Selecting "none" for remote type effectively deletes the rule even though it will still appear in the list until deleted. Unused rules at the end of the list will always show none as the type. If you wish to prevent these from being displayed, reduce the number of rules enabled.

The number of rules enabled simply limits the scope of rule review so that you do not have to review a lot of unused rules. If the displayed rules are used up and you need more, increase the enabled number.

7.5 Modbus RTU Master Data Displayed Per Slave

This page displays data to and from registers in devices accessed via the Modbus RTU serial port.

RTU Unit # 1 Showing 1 to 3 of 3 Update < Prev Next >

Dir.	Reg. Type	Remote Reg. #	Register Name	Local Object #	Hex	Update	Register Data	Time since Last update
From	Holding Reg	00001	External Sensor 1	00001	<input type="checkbox"/>	<input type="checkbox"/>	55	1.740
From	Holding Reg	00002	External Sensor 2	00002	<input type="checkbox"/>	<input type="checkbox"/>	66	1.760
To	Holding Reg	00003	Actuator 1	00003	<input type="checkbox"/>	<input type="checkbox"/>	0	1.510

RTU Unit # 1 + Unit - Unit

The values of Modbus registers that have been read from remote RTU serial devices is displayed here. One remote unit at a time is displayed. To display a different unit, change the RTU Unit #.

Simply click the Update button to view the most recent data. Enter a new value and check the Update box if the value should be changed when you click the Update button. Check the Hex box if you wish to view or enter values in hexadecimal (not recommended for floating point).

Click Update to view the most recent data values. Click "Prev" or "Next" to scroll through the list of registers. You may also enter a number in the "Showing" box to jump directly to a given register when Update is clicked.

7.6 Modbus RTU Errors

This page displays error codes encountered in processing reads and writes via the Modbus RTU serial port.

Showing devices from

Unit #	Reset -->	Read Error	Offending Read Map #	Reset -->	Write Error	Offending Write Map #	Reset -->	Total Messages	No Responses	CRC Errors	Exceptions
1	<input type="checkbox"/>	0/0	0	<input type="checkbox"/>	0/0	0	<input type="checkbox"/>	771	0	0	0
2	<input type="checkbox"/>	0/0	0	<input type="checkbox"/>	0/0	0	<input type="checkbox"/>	0	0	0	0
3	<input type="checkbox"/>	0/0	0	<input type="checkbox"/>	0/0	0	<input type="checkbox"/>	0	0	0	0
4	<input type="checkbox"/>	0/0	0	<input type="checkbox"/>	0/0	0	<input type="checkbox"/>	0	0	0	0
5	<input type="checkbox"/>	0/0	0	<input type="checkbox"/>	0/0	0	<input type="checkbox"/>	0	0	0	0
6	<input type="checkbox"/>	0/0	0	<input type="checkbox"/>	0/0	0	<input type="checkbox"/>	0	0	0	0
7	<input type="checkbox"/>	0/0	0	<input type="checkbox"/>	0/0	0	<input type="checkbox"/>	0	0	0	0
8	<input type="checkbox"/>	0/0	0	<input type="checkbox"/>	0/0	0	<input type="checkbox"/>	0	0	0	0
9	<input type="checkbox"/>	0/0	0	<input type="checkbox"/>	0/0	0	<input type="checkbox"/>	0	0	0	0
10	<input type="checkbox"/>	0/0	0	<input type="checkbox"/>	0/0	0	<input type="checkbox"/>	0	0	0	0
11	<input type="checkbox"/>	0/0	0	<input type="checkbox"/>	0/0	0	<input type="checkbox"/>	0	0	0	0
12	<input type="checkbox"/>	0/0	0	<input type="checkbox"/>	0/0	0	<input type="checkbox"/>	0	0	0	0
13	<input type="checkbox"/>	0/0	0	<input type="checkbox"/>	0/0	0	<input type="checkbox"/>	0	0	0	0
14	<input type="checkbox"/>	0/0	0	<input type="checkbox"/>	0/0	0	<input type="checkbox"/>	0	0	0	0
15	<input type="checkbox"/>	0/0	0	<input type="checkbox"/>	0/0	0	<input type="checkbox"/>	0	0	0	0

The first occurrence of read and write errors are shown along with the map number that was being processed when the error occurred. Check the reset box and click update to clear it and possibly show the next error if there are more than one active error conditions.

A total count of all errors is also shown. This total is the sum of errors for all maps for this device. Check the reset box and click update to reset the counts. Click Update to view the most recent data values.

Error code indications of A/B indicate the following errors with the first number:

- 1 = Transaction ID out of sync
- 2 = Exception code returned by remote device
- 3 = Function code mismatch (bad packet)
- 4 = Insufficient data (bad packet)
- 5 = No response from remote device, timed out
- 6 = CRC error in received packet

When A is code 2 indicating an exception code was returned, B indicates the exception as follows:

- 1 = Illegal function code

2 = Illegal data address (the requested register does not exist in the device)

3 = Illegal data value

8. Setting Up Modbus TCP

The BB2-6040 can be a Modbus client or server. As a client (master) you can read Modbus data from, or write Modbus data to, other Modbus servers (slaves). The BB2-6040 will periodically poll the other Modbus devices according to register maps you set up. The Modbus server (slave) devices that you will read/write are defined on the Devices page. To read from a remote Modbus device, configure a Read Map. To write to a remote Modbus device, configure Write Map.

Data read from a remote device is stored in a local register when received. Data written to a remote device is taken from a local register when sent. The local registers are the same collection of registers that are accessible to other clients via the server map 'virtual device', and accessible to other Modbus TCP and RTU devices as a collection of holding registers.

8.1 Modbus TCP Device Configuration

The screenshot shows the configuration interface for the Babel Buster 2 Modbus-ZigBee Network Gateway (Model BB2-6040) by Control Solutions, Inc. The interface is titled "Modbus TCP Setup" and is part of a larger "Modbus" configuration section. The main navigation tabs include "Data Objects", "ZigBee", "Modbus", and "System Setup". Under "Modbus", there are sub-tabs for "Modbus RTU Data", "Modbus RTU Setup", "Modbus TCP Data", and "Modbus TCP Setup". The "Modbus TCP Setup" sub-tab is active, showing a "Devices" section with sub-tabs for "Client Read Map", "Client Write Map", and "Server Map".

The "Devices" section contains the following information:

- Device #: 1
- IP Address: 192.168.1.111
- Local Name: ModSim
- Unit (optional): 1
- Use FC 5/6 instead of 15/16:
- Swap Double Registers:
- Default Poll Period: 2.0 Seconds
- Connection Status: 0

Buttons for "Update", "< Prev", and "Next >" are visible at the top right of the device configuration area.

The Modbus Devices page is illustrated above. Device number simply shows you where you are on the device list. Click "next" and "prev" to scroll through the list.

Remote Modbus/TCP devices to be accessed by this device are specified here. Enter the IP address of the remote device, a name to reference in other pages, a unit number, poll rate, and check "swapped" if appropriate. Then click "update".

If your slave/server device only supports function codes 5 and 6 for writing, check the Use FC 5/6 box. The default function codes are 15 and 16, which are most widely used.

The term "swapped" only applies to double or float formats. Modbus registers are, by definition, 16 bits of data per register. Access to 32-bit data, either 32-bit integer ("double"), or IEEE 754 floating point ("float"), is supported by the use of two consecutive registers. Modbus protocol is inherently "big endian", therefore, Modbus by the Module defaults to having the high order register first for double and float. If the low order register comes first on the device being accessed, check the "swapped" box.

If you have "swapped" turned around, you will quickly recognize it. If floating point data is reversed, a 1.0 becomes 2.2779508e-41, which simply rounds to zero. The pattern is not as predictable as the 1.0 example would suggest. A floating point 1.1 becomes negative 107609184. If 32-bit integer data is reversed, 1 becomes 65536.

Connection status will show a non-zero error code if there is a socket error. Possible errors include:

- 5 = Connection failed, unable to bind (usually means remote device not connected or not reachable)
- 81 = Connection in progress (means unsuccessful connect attempt, still trying)
- 95 = Network is unreachable
- 97 = Connection aborted
- 98 = Connection reset by peer
- 103 = Connection timed out
- 104 = Connection refused
- 107 = Host is unreachable

8.2 Modbus TCP Client Read Maps

The screenshot shows the web interface for the Babel Buster 2 Modbus-ZigBee Network Gateway. The navigation menu includes Data Objects, ZigBee, Modbus, and System Setup. Under Modbus, there are options for Modbus RTU Data, Modbus RTU Setup, Modbus TCP Data, and Modbus TCP Setup. The 'Client Read Map' option is selected. Below the navigation, there is a description: 'Read remote registers into local registers. This page creates a map entry that reads data from one or more remote Modbus/TCP servers for processing here. Click on map number to see more detail and insert/delete rules.' A 'Showing' box displays '1 to 3 of 3' with 'Update', '< Prev', and 'Next >' buttons. A table lists three map entries:

Map #	Remote Type	Remote Register Format	Remote Register #	Remote Device	Scale	Local Register #	Name
1	Holding Register	Integer	4	ModSim	0.000000	4	Sensor 4
2	Holding Register	Integer	5	ModSim	0.000000	5	Sensor 5
3	None	Integer	0	None	0.000000	0	

Rule number simply tells you where you're at on the list of register maps. Click "next" and "prev" to scroll through the list. To advance directly to a specific map, enter the desired number in the "Showing" box, then click Update.

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

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

For each remote register to be read, enter the register type, format, number, and location (device). The names in the device list are defined in the Devices page.

When the remote register is read, the data will be multiplied by the scale factor and written to the local register number given. The name is optional and used only for display purposes.

Selecting "none" for remote type effectively deletes the rule even though it will still appear in the list until deleted. Unused rules at the end of the list will always show none as the type.

Local register numbers are 1-999 for integer values, and 1001-1999 accessed as register pairs for floating point. If you try to enter an even number above 1001, you will get an error message. All floating point register pairs start on odd boundaries. All local registers are accessed via Modbus as holding registers.

Click on the map number in the left column of the tabular read map page (above) to get the expanded view of one read map at a time (below).

The screenshot shows the configuration interface for the Babel Buster 2 Modbus-ZigBee Network Gateway. The interface is divided into several sections:

- Navigation:** A top bar with tabs for "Data Objects", "ZigBee", "Modbus" (selected), and "System Setup". Below this, there are sub-tabs for "Modbus RTU Data", "Modbus RTU Setup", "Modbus TCP Data", and "Modbus TCP Setup".
- Map Selection:** A row of buttons for "Devices", "Client Read Map" (selected), "Client Write Map", and "Server Map".
- Map Information:** A text box stating "This page creates a map entry that reads data from a remote Modbus/TCP server for processing here." Below this is a "Map #" input field containing "1", and buttons for "Update", "< Prev", and "Next >".
- Configuration Fields:**
 - "Read" dropdown: "Holding Register"
 - "as" dropdown: "Integer"
 - "from register #" input: "4"
 - "at" dropdown: "ModSim"
 - "Apply bit mask if applicable:" input: "0000"
 - "then apply scale:" input: "0.000000"
 - "and offset:" input: "0.000000"
 - "Save in local register #" input: "4"
 - "named" input: "Sensor 4"
 - "Repeat this process every" input: "2.0" seconds.
 - "Apply this default value:" input: "0.000000" after "0" read failure(s).
- Map Management:** A "# Client Read Maps Enabled:" input field containing "3", and buttons for "Insert" and "Delete".

Rule number simply tells you where you're at on the list of register maps. Click "next" and "prev" to scroll through the list. To advance directly to a specific map, enter the desired number in the "Map #" box, then click Update.

For each remote register to be read, enter the register type, format, number, and location (device). The names in the device list are defined in the Devices page.

When the remote register is read, data may be manipulated before being written to the local register. If a bit mask is entered (in hexadecimal), and the remote register type is signed or unsigned (16-bit data), the mask will be bit-wise logical AND-ed with the data, and the retained bits will be right justified in the result. The result will then be multiplied by the scale factor. The offset is then added and this final result is written to the local register number given. The name is optional and used only for display purposes.

The periodic poll time determines how often the remote register will be read. This number, if nonzero, will override the default poll time given in the Devices page for the remote device being read.

The default value will be stored into the local register after the given number of read failures if the fail count is non-zero. Setting the count to zero will disable the default, and the object will retain the most recent value obtained.

Delete will remove the rule number shown in the "Map #" box. Insert will insert a new rule before the rule number shown, and is used for placing rules between existing rules. It is not necessary to use Insert to add rules to the bottom of the list or to define any rule presently having zero for a source object or "none" for remote type.

Selecting "none" for remote type effectively deletes the rule even though it will still appear in the list until deleted. Unused rules at the end of the list will always show none as the type. If you wish to prevent these from being displayed, reduce the number of rules enabled.

The number of rules enabled simply limits the scope of rule review so that you do not have to review a lot of unused rules. If the displayed rules are used up and you need more, increase the enabled number.

8.3 Modbus TCP Client Write Maps

Map #	Local Register #	Scale	Remote Type	Remote Register Format	Remote Register #	Remote Device	Name
1	6	0.000000	Holding Register	Integer	6	ModSim	Actuator 2
2	0	0.000000	None	Integer	0	None	

Rule number simply tells you where you're at on the list of register maps. Click "next" and "prev" to scroll through the list. To advance directly to a specific map, enter the desired number in the "Showing" box, then click Update.

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

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

Data from the local register given will be multiplied by the scale factor before being written. For each remote register to be written, enter the register type, format, number, and location (device). The names in the device list are defined in the Devices page. The name is optional and used only for display purposes.

Selecting "none" for remote type effectively deletes the rule even though it will still appear in the list until deleted. Unused rules at the end of the list will always show none as the type.

Local register numbers are 1-999 for integer values, and 1001-1999 accessed as register pairs for floating point. If you try to enter an even number above 1001, you will get an error message. All floating point register pairs start on odd boundaries. All local registers are accessed via Modbus as holding registers.

Click on the map number in the left column of the tabular write map page (above) to get the expanded view of one write map at a time (below).

Babel Buster 2
MODBUS-ZIGBEE
NETWORK GATEWAY
MODEL BB2-6040

CONTROL SOLUTIONS, INC.
MINNESOTA

Data Objects | ZigBee | **Modbus** | System Setup

Modbus RTU Data | Modbus RTU Setup | Modbus TCP Data | **Modbus TCP Setup**

Devices | **Client Read Map** | Client Write Map | Server Map

This page creates a map entry that writes data to one or more remote Modbus/TCP servers from data contained here.

Map #

Read local register # named

Apply default value of at power-up and/or when seconds have elapsed with no host update.

Write remote register any time local register has changed by or when seconds have elapsed with no change.

Otherwise write remote register unconditionally. In any event, when writing remote register, apply local register data as follows:

Apply scale: and offset: Then if applicable, apply bit mask: and bit fill:

Write as to register # at

Repeat this process at least no more than every seconds.

Client Write Maps Enabled:

Rule number simply tells you where you're at on the list of register maps. Click "next" and "prev" to scroll through the list. To advance directly to a specific map, enter the desired number in the "Map #" box, then click Update.

The local register data may be written periodically, or when it changes, or both. To send upon change (send on delta), check the first box and enter the amount by which the local register must change before being written to the remote device. To guarantee that the remote register will be written at least occasionally even if the data does not change, check the second box and enter some amount of time. This time period will be referred to as the "maximum quiet time".

Data from the local register may be manipulated before being written to the remote register. The local data is first multiplied by the scale factor. The offset is then added to it. If a bit mask is entered, and the remote register type is signed or unsigned (16-bit data), the mask will be bit-wise logical AND-ed with the data. The mask is right justified, then AND-ed with the data. The result is then left shifted back to the original position of the mask. In other words, the least significant bits of the original data will be stuffed at the position marked by the mask.

After the scaling and masking, the bit fill will be logically OR-ed into the result, but only if the mask was nonzero and was used. Both mask and fill are entered in hexadecimal.

Multiple local registers may be packed into a single remote register. To accomplish this, define two or more rules in sequence with the same remote destination. If the destination is the same, data types are 16-bit (integer or unsigned), bit masks are nonzero, and the rules are sequential, the results of all qualifying rules will be OR-ed together before being sent to the remote destination.

For the remote register to be written, enter the register type, format, number, and location (device). The names in the device list are defined in the Devices page.

The repeat time may determine how often the remote register will be written. If send on delta and maximum quiet time are not checked above, clicking the "at least" button will establish a periodic update time. If send on delta is used and

you wish to limit the network traffic in the event changes are frequent, click the "no more than" button and enter the minimum time that should elapse before another write to the remote device.

Delete will remove the rule number shown in the "Map #" box. Insert will insert a new rule before the rule number shown, and is used for placing rules between existing rules. It is not necessary to use Insert to add rules to the bottom of the list or to define any rule presently having zero for a source register or "none" for remote type.

Selecting "none" for remote type effectively deletes the rule even though it will still appear in the list until deleted. Unused rules at the end of the list will always show none as the type. If you wish to prevent these from being displayed, reduce the number of rules enabled.

The number of rules enabled simply limits the scope of rule review so that you do not have to review a lot of unused rules. If the displayed rules are used up and you need more, increase the enabled number.

8.4 Modbus TCP Errors

This page displays error codes encountered in processing Modbus Client reads and writes via the Modbus TCP connection(s).

Device	Reset -->	Read Error	Offending Read Map #	Reset -->	Write Error	Offending Write Map #	Reset -->	Total Messages	No Responses	Exceptions
1	<input type="checkbox"/>	0/0	0	<input type="checkbox"/>	0/0	0	<input type="checkbox"/>	822	0	0
2	<input type="checkbox"/>	0/0	0	<input type="checkbox"/>	0/0	0	<input type="checkbox"/>	0	0	0
3	<input type="checkbox"/>	0/0	0	<input type="checkbox"/>	0/0	0	<input type="checkbox"/>	0	0	0
4	<input type="checkbox"/>	0/0	0	<input type="checkbox"/>	0/0	0	<input type="checkbox"/>	0	0	0
5	<input type="checkbox"/>	0/0	0	<input type="checkbox"/>	0/0	0	<input type="checkbox"/>	0	0	0
6	<input type="checkbox"/>	0/0	0	<input type="checkbox"/>	0/0	0	<input type="checkbox"/>	0	0	0
7	<input type="checkbox"/>	0/0	0	<input type="checkbox"/>	0/0	0	<input type="checkbox"/>	0	0	0
8	<input type="checkbox"/>	0/0	0	<input type="checkbox"/>	0/0	0	<input type="checkbox"/>	0	0	0
9	<input type="checkbox"/>	0/0	0	<input type="checkbox"/>	0/0	0	<input type="checkbox"/>	0	0	0
10	<input type="checkbox"/>	0/0	0	<input type="checkbox"/>	0/0	0	<input type="checkbox"/>	0	0	0
11	<input type="checkbox"/>	0/0	0	<input type="checkbox"/>	0/0	0	<input type="checkbox"/>	0	0	0
12	<input type="checkbox"/>	0/0	0	<input type="checkbox"/>	0/0	0	<input type="checkbox"/>	0	0	0

The first occurrence of read and write errors are shown along with the map number that was being processed when the error occurred. Check the reset box and click update to clear it and possibly show the next error if there are more than one active error conditions.

A total count of all errors is also shown. This total is the sum of errors for all maps for this device. Check the reset box and click update to reset the counts. Click Update to view the most recent data values.

Error code indications of A/B indicate the following errors with the first number:

- 1 = Transaction ID out of sync
- 2 = Exception code returned by remote device

- 3 = Function code mismatch (bad packet)
- 4 = Insufficient data (bad packet)
- 5 = No response from remote device, timed out
- 6 = CRC error in received packet

When A is code 2 indicating an exception code was returned, B indicates the exception as follows:

- 1 = Illegal function code
- 2 = Illegal data address (the requested register does not exist in the device)
- 3 = Illegal data value

8.5 Modbus TCP Server (Slave) Operation

The term “server” is often used to describe the Modbus TCP version of a Modbus slave. A server will provide data when a client asks for it. The concept of master/slave is less significant in Modbus TCP because any TCP device can be both master and slave at the same time, and there can be multiple “masters” on the network. That is in contrast with Modbus RTU where there can be only one master and multiple slaves, and each device must be one or the other.

The Modbus TCP server is simply a collection of registers that may contain data. The source of that data in the case of Babel Buster BB2-6040 can be any of several possible sources. It may be read from another Modbus device. Another Modbus device could have put it there by writing to the BB2-6040. The data could have been received by the ZigBee client.

Local register numbers held by the server are 1-999 for integer values, and 1001-1999 accessed as register pairs for floating point. If you try to enter an even number above 1001, you will get an error message. All floating point register pairs start on odd boundaries. All local registers are accessed via Modbus as holding registers.

8.6 Modbus TCP Server Virtual Device Configuration

You may access any of the Modbus registers as holding registers using the local register number. You also have the option of creating a “virtual Modbus device” using the server map. Furthermore, you have the option of using Modicon notation (40001 for holding register 1, etc) to create your virtual device. You set this map up on the Server Map page under Modbus TCP Setup.

Create remote client's custom view of local registers. This page sets up the register map for the virtual Modbus/TCP server. This map is also referred to as the "user map". This allows you to remap the default server register map to match any layout you wish, including matching the map found in other equipment.

Showing 1 to 3 of 3

Map #	Mapped Register #	Mapped Register Format	Local Register #	Scale Factor	Offset	Bit Field	Fill	Name
1	40001	Signed Integer	101	0.000000	0.000000	0000	0000	
2	40002	Unsigned Integer	102	0.000000	0.000000	0000	0000	
3	0	None	0	0.000000	0.000000	0000	0000	

Custom Registers Enabled: 3

User Map Enabled Map is Exclusive
 Swap Double Registers Zero fill null registers
 Use Modicon mapping

Update < Prev Next > Insert Delete

For each register to be mapped into the custom map, enter the server address where this register should appear, the format it should be presented in, and the source of the data. Scale factor is optional. The source data will be multiplied by this to produce the data in the mapped server register. Offset is optional. This value will be added to the source data after multiplying by the scale factor.

Bit field and fill allow compiling register contents derived from multiple sources if the bit field is defined (nonzero). The source data will be limited to the number of bits represented in the bit field (which is a hexadecimal value), and shifted into the position represented by '1' bits in the field. Fill bits will be logically OR'ed into the result before being presented by the server. Consecutive server map entries that reference the same server address will all be OR'ed together and presented at that address. Duplicate map entries that reference the same server address but are not listed in consecutive order following the first instance will be skipped. No special bit field processing takes place if the bit field is set to zero. Bit fields apply to 16-bit integer or unsigned integer server registers only.

The name is optional and is used for display purposes only.

Delete will remove the rule number shown in the "Showing" box. Insert will insert a new rule before the rule number shown, and is used for placing rules between existing rules. It is not necessary to use Insert to add rules to the bottom of the list or to define any rule presently having "none" for register format.

Selecting "none" as the register format effectively deletes the rule even though it will still appear in the list until deleted. Unused rules at the end of the list will always show "none" as the format. If you wish to prevent these from being displayed, reduce the number of rules enabled.

Enter the number of Modbus registers that should be available in your customized register mapping and check "User Map Enabled" to begin using a customized map. Check "Map is Exclusive" if access to registers outside of this map should be prohibited. If exclusive is not selected, all local registers not overlapped by the custom map will also be accessible to the remote client.

By default, double registers in Control Solutions products are "big endian" meaning the most significant bytes are in the first register and least significant bytes are in the second register. If remote clients accessing this server at this IP address expect "little endian", check the swap box. Modbus protocol by definition is "big endian" within each register, but the "endian" order of the registers for 32-bit values is less standardized.

Normally an attempt to read an undefined register will return an exception (error) code. To enable reading of large data packets without nuisance errors, you have the option of zero filling null registers. This means that reading an undefined register in between valid defined registers will simply return zero data rather than an error.

Check "Use Modicon mapping" to map 0X, 1X, 3X and 4X registers anywhere in i.CanDoIt register space. When you use Modicon mapping, the Mapped Register number should be in the following ranges:

Mapped Register #	Read (Write) as	Function codes expected
0-9999	Coil	1, 5, 15
10001-19999	Discrete Input	2
30001-39999	Input Register	4
40001-49999	Holding Register	3, 6, 16

Any of the Modicon register types may be mapped to any local register, except that coils and discrete inputs cannot map to floating point registers. When a local register is read as coil or discrete input, any nonzero value in the local register will return a set bit, and zero in the local register will return a clear bit. Local registers written as coils will be set to 0 or 1.

To use Modicon mapping, you must check the Use Modicon box, and also check User Map Enabled. It is also highly recommended that you check the Map is Exclusive box when using Modicon mapping. Remember to go to the Config File page and save your changes.

This page displays data as mapped by the virtual server "user map" defined under Modbus->Setup->Server Map. This is a snap shot of what the remote client will see when the user map is enabled.

Showing 1 to 2 of 2

Map #	Mapped Register #	Register Name	Register Data
1	40001		0
2	40002		0

Diagnostic Info

- 1: 0/0
- 2: 0/0
- 3: 0/0
- 4: 0/0
- 5: 0/0

The values of Modbus registers that have been created by the virtual server mapping are displayed on this page. These are the values that a remote client would see. (The remote client acts as Modbus master, and this server acts as a Modbus slave having the registers shown here.)

Click Update to view the most recent data values. Click "Prev" or "Next" to scroll through the list of registers. You may also enter a number in the "Showing" box to jump directly to a given register when Update is clicked.

The diagnostic info shows the connection status for each of the available connections. A code "a/b" where a=0 is an available connection and b is a code indicating its reason for closing (may be normal TCP close). A code where a>0 and b=0 is an active connection.

9 System Setup

9.1 Configuration Files

The screenshot displays the 'System Setup' interface for the Babel Buster 2. At the top, there is a header with the product name 'Babel Buster 2', the model 'BACNET-ZIGBEE NETWORK GATEWAY MODEL BB2-7040', and the company logo 'CONTROL SOLUTIONS, INC. MINNESOTA'. Below the header is a navigation menu with tabs for 'Data Objects', 'ZigBee', 'BACnet', 'System Setup', and 'User'. Under 'System Setup', there are sub-tabs for 'Setup', 'Programming', and 'User'. The 'Setup' sub-tab is active, and within it, the 'Config File' sub-tab is selected. The main content area contains the following elements:

- A heading: 'Store configuration to Flash file selected from directory, or to new file if checked.'
- A 'Load' button.
- A 'Local file directory' dropdown menu showing 'BootConfig.xml', with a 'View' button to its right.
- A 'Delete' button.
- A 'Save' button.
- A checkbox labeled 'Create new file' followed by an empty text input field.
- A 'Boot' button.
- A 'Boot configuration' dropdown menu showing 'BootConfig.xml'.
- A checkbox labeled 'Confirm' followed by a 'Restart' button.
- A section titled 'Upload Configuration File' with an 'Upload' button, an empty text input field, and a 'Browse...' button.

IMPORTANT: Configuration changes will be lost the next time you cycle power if you did not click the "save" button to place those changes in non-volatile memory (Flash file).

Click "Save" to store the present configuration to a Flash file. The configuration will overwrite the selected file in the local file directory unless you check "create new file" and enter a new file name.

Click "Load" to load the currently selected file in the local file directory. This means reconfigure the system from this file. Changes take effect immediately and could have consequences. Be certain to understand the changes you are invoking before clicking "load".

Click "Boot" to select a different boot configuration file. This is the Flash file that is automatically loaded upon power-up.

Click "View" to look at the selected file. It should be an XML file, and your browser will recognize it as such if properly formatted.

Click "Delete" to remove a file from the local file directory.

Click "Browse" to select a file for upload from your PC. Once selected, then click "Upload" to complete the process.

Check "Confirm" and click "Restart" to re-boot the system. This will not cycle power to the hardware, but will reboot the processor as if power had been cycled. Use this to force an IP address change remotely. Only the root user is allowed to do this. Upon restart, this page will not reload and the HTTP connection will be lost.

Note: There may be a delay of several seconds while Flash memory is updated by any of the above actions.

Note: Your browser may cache files. If you view a file, make configuration changes, save the file, then view the file again, you may see the old file cached by the browser. To see the updated file, go to "Internet Options" in your browser's "Tools" menu, and delete temporary Internet files (or delete cache files). Also, if you upload a file, make changes on your PC, and re-upload the same file, the browser may send the old file. Again, you will need to find the button inside your browser options that lets you delete the cached files from your PC.

9.2 Local Port Settings



The screenshot shows the web interface for the Babel Buster 2 BACnet-ZigBee Network Gateway. The header includes the product name and logo for Control Solutions, Inc. The navigation menu is divided into 'Data Objects', 'ZigBee', 'BACnet', and 'System Setup'. Under 'System Setup', there are sub-menus for 'Setup', 'Programming', and 'User'. The 'Local Host' sub-menu is selected, showing fields for IP Address (192.168.1.60), Subnet Mask (255.255.255.0), Gateway (192.168.1.1), and HTTP Port (80). Buttons for '- Refresh -', 'Change IP', and 'Set Port' are visible.

IP Address	<input type="text" value="192.168.1.60"/>	192.168.1.60	<input type="button" value="- Refresh -"/>
Subnet Mask	<input type="text" value="255.255.255.0"/>	255.255.255.0	<input type="button" value="Change IP"/>
Gateway	<input type="text" value="192.168.1.1"/>	192.168.1.1	
HTTP Port	<input type="text" value="80"/> (default 80)		<input type="button" value="Set Port"/>

To change the IP address of this device, enter the address, subnet mask, and gateway, then click "change IP". Set the IP address to 255.255.255.255 to specify that DHCP should be used to obtain an IP address upon power-up. IP address change will take effect upon next power cycle.

The default port for web page serving is 80. If you wish to change it, enter the port number and click Set Port. This change will take effect upon the next power-up. If the port is anything other than 80, you must include the port number in the URL. For example, if you would normally use `http://10.0.0.101/` to get here and you change the port to 8215, you would now use `http://10.0.0.101:8215/`. (Note: The port change is only accepted if you are logged in as the root user.)

9.3 User Accounts and Passwords

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

You are logged in as the root user if you see this page. You may enter user names, their passwords, set their privilege levels, and enter an optional IP filter. If the IP filter is a non-zero valid IP address, the user may only log in from that IP address.

Changes will take at least a full minute before this page will refresh.

10 Data Objects

This page displays data as presently found in the local registers maintained by this device.

Showing registers from

Local Register #	Register Name	Hex	Update	Register Data	Unsigned	Register Type	Default Data	Server Timeout (S)
01001	Room 101 Temp	<input type="checkbox"/>	<input type="checkbox"/>	72.0529	<input type="checkbox"/>	Float	0.00000	0.0
01003	Room 102 Temp	<input type="checkbox"/>	<input type="checkbox"/>	74.7166	<input type="checkbox"/>	Float	0.00000	0.0
01005	Room 103 Temp	<input type="checkbox"/>	<input type="checkbox"/>	73.5402	<input type="checkbox"/>	Float	0.00000	0.0
01007	Room 103 Humidity	<input type="checkbox"/>	<input type="checkbox"/>	39.4363	<input type="checkbox"/>	Float	0.00000	0.0
01009	Room 104 Temp	<input type="checkbox"/>	<input type="checkbox"/>	71.8511	<input type="checkbox"/>	Float	0.00000	0.0
01011	Room 104 Humidity	<input type="checkbox"/>	<input type="checkbox"/>	38.4497	<input type="checkbox"/>	Float	0.00000	0.0
01013	Room 105 Temp	<input type="checkbox"/>	<input type="checkbox"/>	71.8511	<input type="checkbox"/>	Float	0.00000	0.0
01015	Room 105 Humidity	<input type="checkbox"/>	<input type="checkbox"/>	37.3398	<input type="checkbox"/>	Float	0.00000	0.0
01017	Room 106 Temp	<input type="checkbox"/>	<input type="checkbox"/>	73.1180	<input type="checkbox"/>	Float	0.00000	0.0
01019	Room 106 Humidity	<input type="checkbox"/>	<input type="checkbox"/>	38.2031	<input type="checkbox"/>	Float	0.00000	0.0
01021	Room 107 Temp	<input type="checkbox"/>	<input type="checkbox"/>	74.5959	<input type="checkbox"/>	Float	0.00000	0.0
01023	Room 107 Humidity	<input type="checkbox"/>	<input type="checkbox"/>	39.6829	<input type="checkbox"/>	Float	0.00000	0.0
01025	Room 108 Temp	<input type="checkbox"/>	<input type="checkbox"/>	71.0065	<input type="checkbox"/>	Float	0.00000	0.0
01027	Room 108 Humidity	<input type="checkbox"/>	<input type="checkbox"/>	39.0663	<input type="checkbox"/>	Float	0.00000	0.0
01029	Room 109 Temp	<input type="checkbox"/>	<input type="checkbox"/>	75.8628	<input type="checkbox"/>	Float	0.00000	0.0
01031	Room 109 Humidity	<input type="checkbox"/>	<input type="checkbox"/>	38.3264	<input type="checkbox"/>	Float	0.00000	0.0

The contents of the set of local registers found within the BB2-6040 is displayed here. You have the option of forcing a new value by entering a value, checking the Update box, and clicking Update. If hexadecimal display format is more useful for packed 16-bit registers, check the Hex box. The Unsigned check box simple applies to how the data is formatted when displayed on this page.

Default data and server timeout apply only when the BB2-6040 is treated as a Modbus server or slave. Timeout means time since some other Modbus master wrote to this register. If a non-zero time is given for timeout, and the other master does not write within this time, then the default data is placed in the local register.

Out of Service means any polling of the slave device will stop. While out of service, the present value may be written by the BACnet client. Data may be forced via this web page at any time, but will be overwritten by the next poll unless the object is out of service.

11 ZigBee Diagnostic Pages

The screenshot shows the 'Babel Buster 2' interface for a 'BACNET-ZIGBEE NETWORK GATEWAY MODEL BB2-7040'. The 'ZigBee' tab is selected, and the 'Diagnostics' sub-tab is active. The 'Local API' button is highlighted. The main content area displays the following information:

```
Command
API Identifier: 0x08
API Frame ID: 0x02
API Frame: 4F 50
Response
API Reply: 88 02 4F 50 00 00 00 00 00 00 02 34
```

The BB2-6040 contains a Digi International/Maxstream XBee PRO (Ember 250 chip) operating as a coordinator. You can do a variety of things with the BB2-6040 using API commands for the XBee PRO as found in XBee PRO documentation at www.digi.com. You can also find links to applicable documentation on the BB2-6040 product page at www.csimn.com.

The XBee PRO ZigBee coordinator is programmed to operate in API mode (rather than AT mode) as defined in XBee PRO documentation. The API mode allows access to a much greater range of capability and is required for BB2-6040 use.

Local API commands are constructed and sent on this page. API commands to remote devices are constructed and sent from the Remote API page. Local API commands affect the local radio.

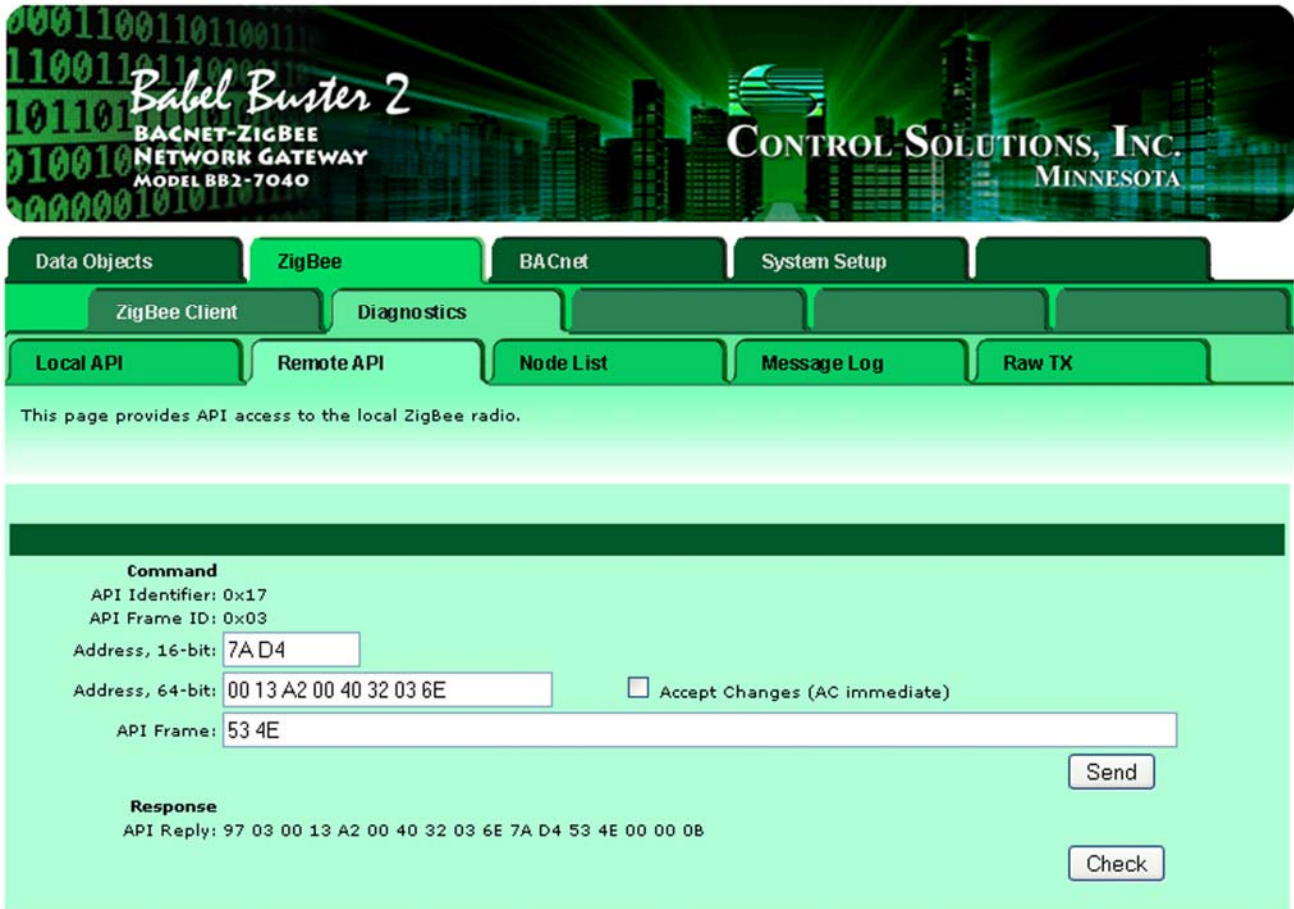
The local API frame input window above provides interpretation and translation of ASCII strings for the construction of raw API packets for transmission. This provides a simple way to create the API version of AT commands.

You may perform a very simple test on the local radio to see if the XBee PRO is responding to commands. the "SL" command should always succeed since this is simply returning the local device address. Enter "/SL" in the API Frame window and click Send. The response that appears shortly for API Reply should be something like: 88 03 53 4C 00 40 30 9E C3 where 88 will always be 88, the 03 in this example will be whatever your next frame ID happened to be, the 53 4C should always be 53 4C, and the next 4 bytes will be the low order half of your local 64-bit address. The last byte is the CRC.

Command interpretation is as follows:

'/' -- results in accepting any number of ASCII alphanumeric characters (0-9,A-Z) with ASCII codes placed in transmit packet.

' ' (quote) -- results in accepting a quote delimited string, required for including blanks or punctuation
NN -- literal number, always interpreted as hexadecimal, delimited by blanks, creates a single byte with this hex code.



Remote API commands are constructed and sent on this page. Remote API commands work exactly like local API commands, except that they are sent to a remote ZigBee device. These API commands are specific to XBee PRO/Ember 250. Commands sent to ZigBee devices using a stack other than Ember will require construction of raw packets on the Raw Tx page.

The remote API frame input window above provides interpretation and translation of ASCII strings for the construction of raw API packets for transmission. This provides a simple way to create the API version of AT commands.

The API command will be sent to the device whose address appears in the address windows. You can enter these manually, or have them filled in automatically by clicking the API Target button on the Node List page.

Command interpretation is as follows:

/' -- results in accepting any number of ASCII alphanumeric characters (0-9,A-Z) with ASCII codes placed in transmit packet.

' ' (quote) -- results in accepting a quote delimited string, required for including blanks or punctuation
NN -- literal number, always interpreted as hexadecimal, delimited by blanks, creates a single byte with this hex code.

An example command that sets the Network Interface name of a remote device would be:

```
/NI "Temp Sensor 1"
```


Configuration changes at the remote device will not take effect immediately unless you check the Accept Changes box. However, there are circumstances where you do not want to do this. If changing a number of parameters affecting how the remote device transmits, you may wish to make several changes, followed by sending the /AC command to accept the changes at that point.

This page provides API access to the local ZigBee radio.

Dev #	Cfg	Addr 16-bit	Addr 64-bit	Node ID	
1	Y	7A D4	00 13 A2 00 40 32 03 6E	Temp Sensor 2	<input type="checkbox"/>
2	Y	9D 8E	00 13 A2 00 40 32 2F 93	Digi Sensor 3	<input type="checkbox"/>
3	Y	BE C1	00 13 A2 00 40 48 5A 2B	Smart Plug 2	<input type="checkbox"/>
4	Y	45 34	00 13 A2 00 40 2D A0 F5	Digi Sensor 4	<input type="checkbox"/>
5	Y	56 0A	00 13 A2 00 40 32 2F A0	Digi Sensor 9	<input type="checkbox"/>
6	N	99 04	00 13 A2 00 40 33 4E D5	Router 1	<input type="checkbox"/>
7	Y	11 4E	00 13 A2 00 40 3E 10 99	Temp Sensor 3	<input type="checkbox"/>
8	Y	35 62	00 13 A2 00 40 32 2F A8	Digi Sensor 6	<input type="checkbox"/>
9	Y	1A CE	00 13 A2 00 40 2D A0 EA	Digi Sensor 7	<input type="checkbox"/>
10	Y	CC B3	00 13 A2 00 40 32 2F A1	Digi Sensor 8	<input type="checkbox"/>
11	Y	CD AE	00 13 A2 00 40 2D A0 E2	Digi Sensor 2	<input type="checkbox"/>
12	Y	39 C1	00 13 A2 00 40 5C 66 BD	Digi Sensor 5	<input type="checkbox"/>
13	Y	E5 6C	00 13 A2 00 40 32 2F 9D	Digi Sensor 10	<input type="checkbox"/>
14	Y	26 D7	00 13 A2 00 40 32 2F 9B	Digi Sensor 1	<input type="checkbox"/>
15	Y	7C 23	00 13 A2 00 40 2D 37 7A	Digi Sensor 11	<input type="checkbox"/>

Refresh Clear Discover API Target

This page shows the list of devices that have associated themselves with the BB2-6040's ZigBee radio functioning as a coordinator. Click Discover to send the /ND command to the local radio. It will respond by sending its list of devices which are then displayed here. Click Refresh after waiting a short time for all of the responses. Click Clear to clear the page and repopulate the list.

Cfg Y/N column indicates whether this device is on our list of target devices for gateway functionality (ZigBee Client Devices page).

To select a target device for Remote API access, check the corresponding box and click the API Target button. Then proceed to the Remote API page to send remote commands to this device.



[Data Objects](#)
[ZigBee](#)
[BACnet](#)
[System Setup](#)

[ZigBee Client](#)
[Diagnostics](#)

[Local API](#)
[Remote API](#)
[Node List](#)
[Message Log](#)
[Raw TX](#)

This page provides API access to the local ZigBee radio.

1,11:55:52 Refresh Clear

Up Time	Status	API Packet
1,11:55:49	->Gateway	92 00 13 A2 00 40 32 2F A1 CC B3 41 01 08 00 0E 08 00 00 30 02 83 02 15
1,11:55:48	->Gateway	92 00 13 A2 00 40 32 2F 9D E5 6C 01 01 08 00 0E 08 00 00 2E 02 65 02 02
1,11:55:42	->Gateway	92 00 13 A2 00 40 2D A0 F5 45 34 01 01 08 00 0E 08 00 00 26 02 77 02 0D
1,11:55:39	->Gateway	92 00 13 A2 00 40 2D A0 E2 CD AE 01 01 08 00 0E 08 00 00 2F 02 6F 02 12
1,11:55:38	->Gateway	92 00 13 A2 00 40 2D A0 EA 1A CE 01 01 08 00 0E 08 00 00 47 02 84 02 09
1,11:55:37	->RemAPI	97 03 00 13 A2 00 40 32 03 6E 7A D4 53 4E 00 00 0B
1,11:55:35	RemAPI->	17 03 00 13 A2 00 40 32 03 6E 7A D4 00 53 4E
1,11:55:27	->Gateway	92 00 13 A2 00 40 32 03 6E 7A D4 01 01 3D D0 0F 18 D0 03 FF 02 FC 02 16 02 14
1,11:55:21	->Gateway	92 00 13 A2 00 40 5C 66 BD 39 C1 41 01 08 00 0E 08 00 00 3C 02 7C 02 17
1,11:55:21	->Gateway	92 00 13 A2 00 40 32 2F A8 35 62 01 01 08 00 0E 08 00 00 3C 02 6B 02 06
1,11:55:19	->Gateway	92 00 13 A2 00 40 32 2F 93 9D 8E 41 01 08 00 0E 08 00 00 30 02 70 02 04
1,11:55:18	->Gateway	92 00 13 A2 00 40 3E 10 99 11 4E 41 01 3D D0 0F 18 D0 03 FF 02 F9 02 10 02 10
1,11:55:18	->Gateway	92 00 13 A2 00 40 32 2F 9B 26 D7 41 01 08 00 0E 08 00 00 26 02 71 02 04
1,11:55:15	->Gateway	97 AF 00 13 A2 00 40 48 5A 2B BE C1 44 34 00
1,11:55:15	Gateway->	17 AF 00 13 A2 00 40 48 5A 2B BE C1 02 44 34 04
1,11:55:14	->Gateway	92 00 13 A2 00 40 32 2F A0 56 0A 41 01 08 00 0E 08 00 00 3D 02 6B 02 1B
1,11:55:03	----	92 00 13 A2 00 40 2D 37 7A 7C 23 41 01 08 00 06 08 00 00 2B 02 71
1,11:55:03	->LocAPI	88 02 4F 50 00 00 00 00 00 00 00 02 34
1,11:55:03	LocAPI->	08 02 4F 50
1,11:54:59	----	92 00 13 A2 00 40 48 5A 2B BE C1 01 01 00 10 0E 00 00 00 13 02 B5 00 87

This page provides a log of the last 64 packets (API frames) sent or received by the BB2-6040. It does not show all ZigBee RF traffic in the air, only that which was addressed to or broadcast to the BB2-6040 (which operates as a coordinator).

Packet status indicates how the packet was processed, as follows. The designation XXX-> means outgoing, and ->XXX means incoming.

LocAPI-> ->LocAPI	Packets processed by the Local API web page
RemAPI-> ->RemAPI	Packets processed by the Remote API web page
Gateway-> ->Gateway	Packets processed by gateway functionality defined in ZigBee Client Read Map and Write Map pages.
Basic-> ->Basic	Packets processed by the user's Script Basic program.
----	Packet received but not recognized for processing for any of the above.



Quick Help

The difference between Raw TX (raw transmission) and Remote API is that nothing is done automatically here and you can send any raw string you like to the local radio, some of which might be transmitted to another radio depending on what you constructed.

WARNING: IT IS VERY POSSIBLE TO CONSTRUCT AN API FRAME THAT RESULTS IN MAKING YOUR RADIO NON-FUNCTIONAL IN A NON-RECOVERABLE MANNER. THE BB2-7040 WILL DO NOTHING TO TRY TO STOP YOU. DO NOT USE THIS PAGE IF YOU ARE NOT A ZIGBEE EXPERT. Misconfiguration of your radio is not considered a warranty covered item.

The difference between Raw TX (raw transmission) and Remote API is that nothing is done automatically here and you can send any raw string you like to the local radio, some of which might be transmitted to another radio depending on what you constructed. Do note the warning about Raw TX.

API frames created here require that you explicitly provide the API frame type code, the 64-bit and 16-bit addresses of the remote device as applicable, and any packet data as applicable. Only the header, length, and checksum bytes used to move the packet from the BB2-6040 to the ZigBee radio are automatically generated here. All bytes given here are sent to the radio exactly as-is. By using raw transmit (API codes 0x10 and 0x11), it is possible to send packets to non-Ember ZigBee devices.

Generally, you should use whatever tools were provided by the sensor manufacturer to configure that sensor and get it to start sending data. Once data starts showing up at the BB2-6040, you only then need concern yourself with with creating a data parsing mask on the ZigBee Client Read Map page.

Command interpretation is as follows:

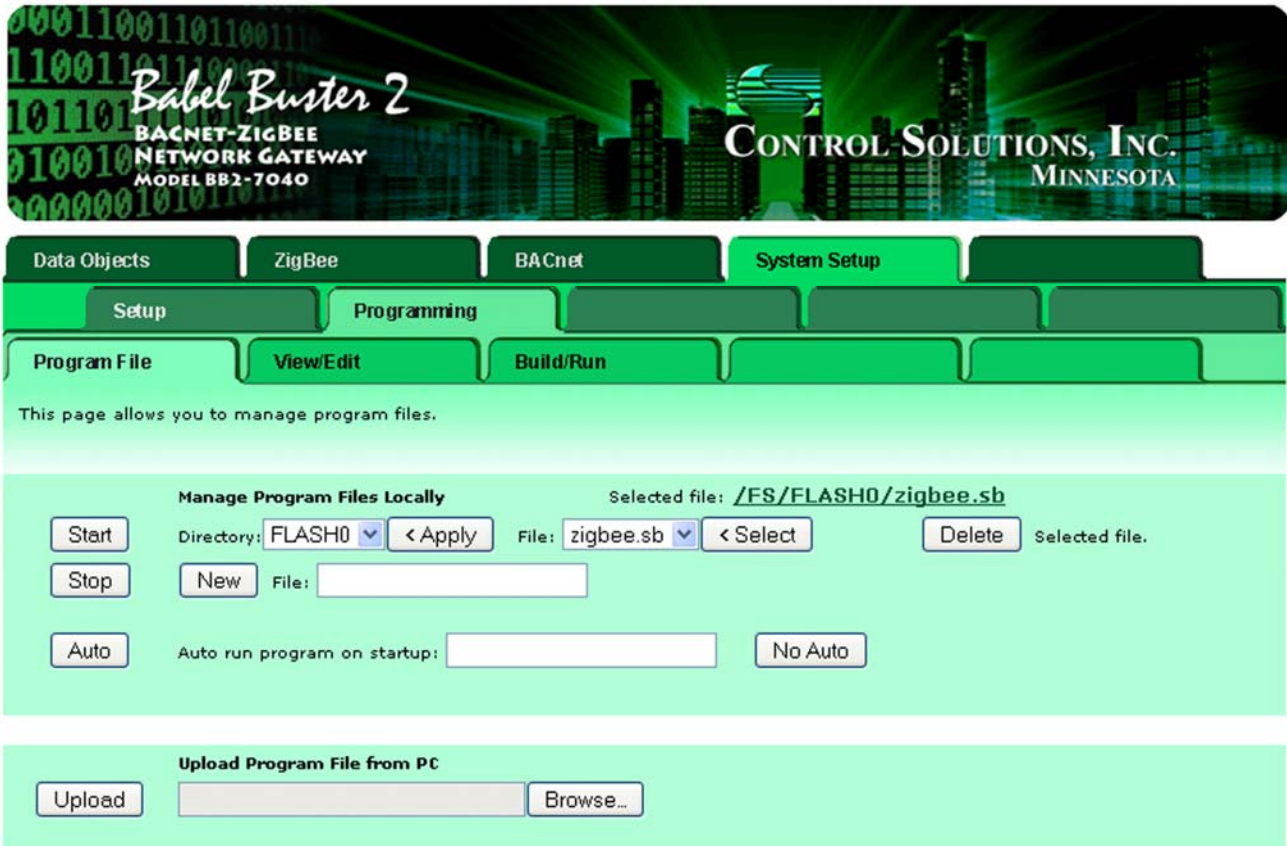
'/' -- results in accepting any number of ASCII alphanumeric characters (0-9,A-Z) with ASCII codes placed in transmit packet.

'"' (quote) -- results in accepting a quote delimited string, required for including blanks or punctuation

NN -- literal number, always interpreted as hexadecimal, delimited by blanks, creates a single byte with this hex code.

12 Programming ZigBee I/O in Basic

12.1 Web Interface Pages for Programming



Click "Apply" to change directory and refresh the file list from that directory. Only files with a suffix of .sb will appear in this list.

Click "Select" to select the program file. This selection will apply to "Start", and also carry over to the edit and virtual terminal pages.

Click "New" to create a new file having the name entered in the window.

Click "Start" to run the selected program in the background. Click "Stop" to terminate this program if it is an endless loop that will not terminate on its own. If you are implementing a control loop, you will want the program to be an endless loop.

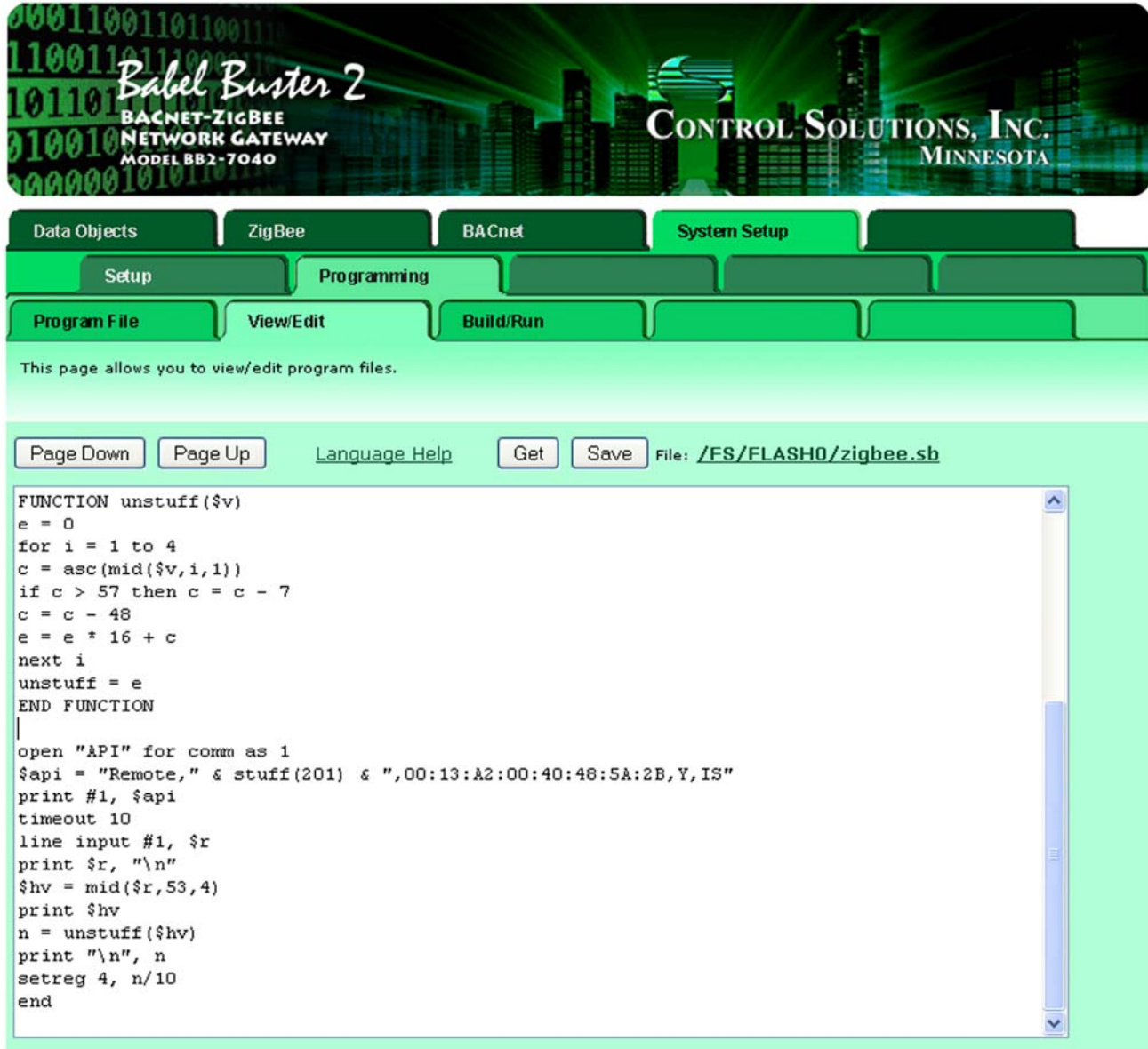
Click "Auto" to select the file that will be started automatically at boot-up. Clear the name window and click "Auto" to disable any previously selected auto-run program. This file name is saved in your boot configuration file, therefore you must save your configuration file in order to retain this auto run file name through power-up.

Click "Delete" to remove a file from the local file directory.

Click "Browse" to select a file for upload from your PC. Once selected, then click "Upload" to complete the process.

To save a copy of the selected program back to your PC, right click on the selected file link (upper right), and then select "save link as" or "save target as" (varies by browser).

Note: There may be a delay of several seconds while Flash memory is updated by some of the above actions.

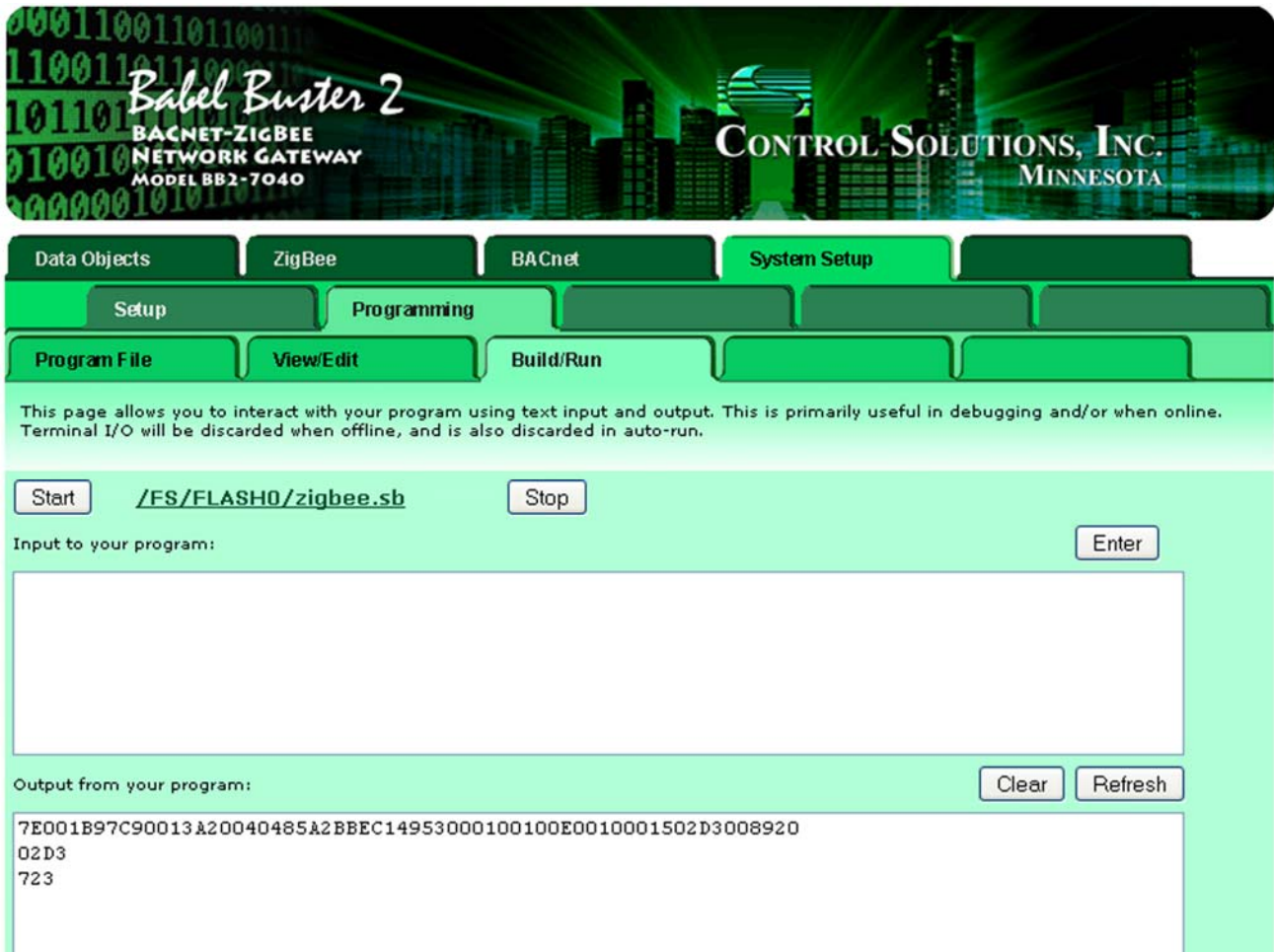


Click "Get" to get the source code for the program shown. Edit using standard text editing (cut, paste, etc). Click "Save" to save changes back to selected file shown.

To create a new file, start by creating a new file in the Program File page. This file will initially be empty, but is needed as a place holder so you have somewhere to save the text on this page when you click "Save".

To erase the scratch pad (shown here), check the erase box and click "Clear".

HTML cannot post large text buffers, therefore larger text files will be displayed for editing in text "pages" within this web page. Use the Page Down and Page Up buttons to scroll through the file. Within each text page, use the scroll bars to navigate through the current text page.



This page emulates terminal I/O interaction with your Basic program as best an HTML page can.

Start the program by clicking "Start". While running, input you type in the input window will be made available to your program each time you click "Enter". Each time you click "Refresh", the output window will be updated with any output generated by your program since the last time you clicked "Refresh". Click "Clear" to discard old output.

Note: The "Enter" and "Refresh" buttons have the same effect as they are both submit buttons. Input present in the input box will be submitted when you click any button.

12.2 Script Basic Enhancements Specific to ZigBee

The ZigBee-specific statements are found only in BB2-6040 version of Script Basic. Communication with a ZigBee device is done via the file interface in Basic. The ZigBee API channel is opened as a file, and ASCII strings are printed to and input from this file.

Opening the ZigBee COM port: Any file number may be used, but file #1 is used in the following example(s).

Program lines:

```
open "API" for comm as 1
open "API+" for comm as 1
```

The first example using "API" as file name will expect API packets to all use the 0x17 remote API command, and will be filtered such that received packets are only returned to Basic when they contain the 0x97 reply code and reference a frame ID generated by Basic. The second example using "API+" will result in Basic receiving unqualified packets, including 0x92 packets generated voluntarily by sensors, etc.

A program code snippet that will send the IS command to a remote device and print its reply is as follows:

```
$api = "Remote,C9,00:13:A2:00:40:2D:37:7A,Y,IS"  
print #1, $api  
timeout 10  
line input #1, $r  
print $r, "\n"
```

The format for the API string is as follows:

```
"Local,FF,CC,hhhhh..hh"  
"Remote,FF,Vnn,Y,CC,hhhhh..hh"  
"Remote,FF,00:00:00:00:00:00:00:00,Y,CC,hhhhh..hh"  
"Raw,hhhhh..hh"
```

The first comma separated field must be one of Local, Remote, or Raw. The FF must be a frame number, in hexadecimal, in the range of 201-255. Frame numbers outside this range will not be routed back to Basic. The next field, for Remote API only, is a device designation, either a 64-bit device address formatted as illustrated, or a device number Vnn to look up on the ZigBee Client Devices page where nn is 1 to 50. Following the device, again in Remote API only, is a field which must be either Y or N. If 'Y', then the AC immediate bit is sent (accept changes). For Local or Remote API, the CC field is the 2-character AT command to be sent. In all cases, the last field(s) represent the data payload (which can be omitted for API commands with no parameters).

The hhhhhh.hhh may actually be one or more fields. For each comma separated field following CC, the value will be interpreted as one or more bytes of data in ASCII hexadecimal form, or be interpreted as multiple bytes of an ASCII string with one byte per character. If the first character of the string is not a hexadecimal digit, it will be taken as an ASCII string. If the first character in the comma separated field is a hexadecimal digit, the string will be parsed as hexadecimal data with one packet byte per two ASCII characters. Note that this interpretation differs from the API web page behavior.

An example of a string that will set the Network Interface name of a remote device is as follows:

```
$api = "Remote,c9,00:13:a0:00:4d:45:fa:af,Y,NI,Temp Sensor 1"
```

For a complete discussion of the API command set, refer to the Digi International/Maxstream XBee PRO documentation.

The following program example sends an on-demand sample command to a Digi SmartPlug and parses a data value from the reply.

```
FUNCTION stuff(a)  
if (a > 255) then  
$h = "FF"  
else  
$k = hex(a)  
if len($k) = 1 then  
$h = "0" & $k  
else  
$h = $k  
end if  
end if  
stuff = $h  
END FUNCTION  
  
FUNCTION unstuff($v)  
e = 0  
for i = 1 to 4  
c = asc(mid($v,i,1))  
if c > 57 then c = c - 7  
c = c - 48  
e = e * 16 + c  
next i  
unstuff = e  
END FUNCTION
```

```
open "API" for comm as 1
$api = "Remote," & stuff(201) & ",00:13:A2:00:40:48:5A:2B,Y,IS"
print #1, $api
timeout 10
line input #1, $r
print $r, "\n"
$hv = mid($r,53,4)
print $hv
n = unstuff($hv)
print "\n", n
setreg 4, n/10

end
```


13 Trouble Shooting

13.1 Modbus RTU Trouble Shooting

This discussion assumes you want the Babel Buster BB2-6040 to be the Modbus Master (most common use). Let's review the setup procedure for a single Modbus read map. We suggest starting with one register. Once you get that working, proceed to fill up the table.

First, go to the Local Device page and make sure you have the baud rate set, and parity (if any) selected. If you do not know what baud rate your Modbus device is set to, consult that manufacturers documentation before proceeding.

Make sure the Master button is clicked. Start with a liberally slow timeout, like 0.5 second just to be rather certain you do not have timeout problems. It is rare to see a piece of working equipment take longer than half a second to respond to a Modbus master. *Setting the timeout to zero, however, will guarantee failure* since the master will miss every reply by not waiting at all for it.

The screenshot shows the configuration interface for the Babel Buster 2 Modbus-ZigBee Network Gateway (Model BB2-6040). The interface is divided into several sections:

- Navigation:** Data Objects, ZigBee, Modbus (selected), System Setup.
- Modbus Setup:** Modbus RTU Data, Modbus RTU Setup (selected), Modbus TCP Data, Modbus TCP Setup.
- Local Device:** Local Device, RTU Read Map (selected), RTU Write Map.

The main configuration area is titled "This page displays configuration parameters for the Modbus RTU serial port." and includes the following settings:

- Baud Rate:** 19200
- Parity:** No Parity, 1 Stop bit
- Update:** Button
- I am the Master:** Selected (radio button)
- I am a Slave:** Unselected (radio button)
- Parameters for RTU Master:**
 - Default Poll Rate: 2.000 Seconds
 - Timeout: 0.500 Seconds
- Parameters for RTU Slave:**
 - My Address or Unit #: 0
 - Double registers are swapped:
 - Use FC 5/6 instead of 15/16 for unit numbers (slave addresses) starting at: 0

Next, go to the RTU Read Map page (below) . To get started, select a register type and format, a register number, a unit # (aka slave ID or slave address), and a local register number to store the data in. If any of the red check marks shown below are "none" or zero, you will get no action even attempted. Make sure the Unit # (slave ID or slave address) matches whatever you have your Modbus device set to. If you are uncertain what address it is set to, you need to consult the manufacturer's documentation for that equipment before proceeding.

The following example shows the only non-zero entries required (the 5 check marks) to successfully read holding register #1 from unit #1 and store the data in local register #1. Once these (or comparable) entries have been made, click the Update button.



Data Objects | ZigBee | **Modbus** | System Setup

Modbus RTU Data | **Modbus RTU Setup** | Modbus TCP Data | Modbus TCP Setup

Local Device | **RTU Read Map** | RTU Write Map

Read remote registers into local registers. This page creates a map entry that reads data from one or more remote Modbus RTU serial devices for processing here. Click on map number to see more detail and insert/delete rules.

Showing 1 to 3 of 3 Update < Prev Next >

Map #	Remote Type	Remote Register Format	Remote Register #	Remote Unit #	Scale	Local Register #	Name
1	Holding Register	Unsigned	1	1	0.000000	1	External Sensor 1
2	Holding Register	Unsigned	2	1	0.000000	2	External Sensor 2
3	None	Integer	0	0	0.000000	0	

At this point, you can go to the data page (below) and see if you have data showing up. If you get no data, there is a problem. The confirmation that you are probably getting no data is the "time since last update". In this example, we see 126 seconds have elapsed. We are attempting to update every 2 seconds, so obviously data retrieval is not happening.

Data Objects | ZigBee | **Modbus** | System Setup

Modbus RTU Data | **Modbus RTU Setup** | Modbus TCP Data | Modbus TCP Setup

RTU Registers | **Error Codes**

This page displays data to and from registers in devices accessed via the Modbus RTU serial port.

Showing 1 to 10 of 10 Update < Prev Next >

Dir.	Reg. Type	Remote Reg. #	Register Name	Local Reg. #	Hex	Update	Register Data	Time since Last update
From	Holding Reg	00001		00001	<input type="checkbox"/>	<input type="checkbox"/>	0	126.720

If you are getting no data, check the Error Codes page (below). Here we see that the "No Responses" is about equal to the "Total Messages". This means we are not getting anything back from the Modbus slave. If you are certain all of the above setup is correct, the only conclusion you (or we) can come to at this point is that there is a wiring problem, or the slave is not responding or not configured correctly. Review wiring information, and check the slave configuration. If you get a high number of CRC errors, this is also an indication of likely wiring problem.

Data Objects | ZigBee | **Modbus** | System Setup

Modbus RTU Data | **Modbus RTU Setup** | Modbus TCP Data | Modbus TCP Setup

RTU Registers | **Error Codes**

This page displays error codes encountered in processing reads and writes via the Modbus RTU serial port.

Showing devices from 1 Update < Prev Next >

Unit #	Reset -->	Read Error	Offending Read Map #	Reset -->	Write Error	Offending Write Map #	Reset -->	Total Messages	No Responses	CRC Errors	Exceptions
1	<input type="checkbox"/>	5/0	1	<input type="checkbox"/>	5/0	1	<input type="checkbox"/>	70	69	0	0

If, instead of No Responses, the count you see is Exceptions, this means you are communicating just fine, but the slave is telling you that your request is incorrect. You are asking for a register number that does not exist, using the incorrect register type, etc. Something about configuration is not right if you get Exception errors.

13.2 Modbus TCP Trouble Shooting

This discussion assumes you want the Babel Buster BB2-6040 to be the Modbus TCP Master. Let's review the setup procedure for a single Modbus read map. We suggest starting with one register. Once you get that working, proceed to fill up the table.

First, go to the IP Network Devices page and make sure you have the IP of the intended Modbus TCP slave entered, along with a local name by which it will be referenced.



Next, go to the Click Read Map page (below). To get started, select a register type and format, a register number, a device (from list created above), and a local register number to store the data in. If any of the red check marks shown below are "none" or zero, you will get no action even attempted. Make sure the IP address (in the device list) matches whatever you have your Modbus device set to. If you are uncertain what IP address it is set to, you need to consult the manufacturer's documentation for that equipment before proceeding.

The following example shows the only non-zero entries required (the 5 check marks) to successfully read holding register #4 from "ModSim" and store the data in local register #4. Once these (or comparable) entries have been made, click the Update button.



Data Objects ZigBee **Modbus** System Setup

Modbus RTU Data Modbus RTU Setup Modbus TCP Data Modbus TCP Setup

Devices Client Read Map **Client Write Map** Server Map

Read remote registers into local registers. This page creates a map entry that reads data from one or more remote Modbus/TCP servers for processing here. Click on map number to see more detail and insert/delete rules.

Showing 1 to 3 of 3 Update < Prev Next >

Map #	Remote Type	Remote Register Format	Remote Register #	Remote Device	Scale	Local Register #	Name
1	Holding Register	Integer	4	ModSim	0.000000	4	Sensor 4
2	Holding Register	Integer	5	ModSim	0.000000	5	Sensor 5
3	None	Integer	0	None	0.000000	0	

At this point, you can go to the data page (below) and see if you have data showing up. If you get no data, there is a problem.

Data Objects ZigBee Modbus **System Setup**

Data

Local Registers

This page displays data as presently found in the local registers maintained by this device.

Showing registers from 1001 Update < Prev Next >

Local Register #	Register Name	Hex	Update	Register Data	Unsigned	Register Type	Default Data	Server Timeout (s)
01001	Room 101 Temp	<input type="checkbox"/>	<input type="checkbox"/>	72.0529	<input type="checkbox"/>	Float	0.00000	0.0
01003	Room 102 Temp	<input type="checkbox"/>	<input type="checkbox"/>	74.7166	<input type="checkbox"/>	Float	0.00000	0.0

If you are getting no data, check the Error Codes page (below). Here we see that the "No Responses" is some number greater than total messages. Zero total messages means we never succeeded in making a TCP connection.

Data Objects ZigBee **Modbus** System Setup

Modbus RTU Data Modbus RTU Setup Modbus TCP Data Modbus TCP Setup

Modbus Server **Error Codes**

This page displays error codes encountered in processing Modbus Client reads and writes via the Modbus TCP connection(s).

Update

Device	Reset -->	Read Error	Offending Read Map #	Reset -->	Write Error	Offending Write Map #	Reset -->	Total Messages	No Responses	Exceptions
1	<input type="checkbox"/>	5/0	1	<input type="checkbox"/>	0/0	0	<input type="checkbox"/>	0	3	0

Check the error codes registers for further definition of what the problem is. In the example below, we see that the error code for our device, TCP device #1 (from list above), is 9516. Decoding this from the information in the Quick Help section of that page in the device, we see that we have a socket error, and the socket error code is 516, which means "timeout" (resulting from no TCP connection).

Local Register #	Register Name	Hex	Update	Register Value	Unsigned	Register Type	Default Data	Server Timeout (S)
09011	-TCP error, device 1	<input type="checkbox"/>	<input type="checkbox"/>	9516	<input type="checkbox"/>	Error Code	0.000000	0.0

Error code registers are automatically named **-Sys Error, Fn #n** for system errors, **-TCP error, device #n** for TCP device errors, and **-RTU error, device #n** for Modbus RTU errors or BACnet device errors. In each instance of "device", the "n" is device number as referenced on other pages.

Error encoding for devices: ABCCC where A=exception codes 1..3 (4), B=error code as follows, and CCC=rule number:

Error code B indicates the following errors:

- 1 = Transaction ID out of sync
- 2 = Exception code returned by remote device
- 3 = Function code mismatch (bad packet)
- 4 = Insufficient data (bad packet)
- 5 = No response from remote device, timed out
- 6 = CRC error in received packet
- 9 = Socket error (in this case, CCC=error code, see below)

Error code A indicates the following exception codes only when B is code 2 indicating an exception code was returned:

- 1 = Illegal function code
- 2 = Illegal data address (the requested register does not exist in the device)
- 3 = Illegal data value
- 4 = other/out of range code

Some example error codes that are common: 5001 (05001) means no response from device at map rule #1. Example 2: 22005 means exception code 2 occurred on map rule #5. This generally means the register you attempted to access is not available at that remote device. Example 3: 6039 means a CRC error was received at map rule #39, and generally indicates a transmission error due to noise on the line.

Special case of socket error: Code will be >9000, and 9xxx will indicate socket error code xxx. Common socket codes include the following:

- 104 - Connection reset by peer
- 105 - No buffer space available
- 111 - Connection refused
- 112 - Address already in use
- 113 - Connection aborted
- 114 - Network unreachable
- 116 - Connection timed out
- 117 - Host is down
- 118 - Host is unreachable
- 401 - General socket error
- 516 - Timeout, no response from remote device

13.3 ZigBee Trouble Shooting

Refer to Section 11, ZigBee Diagnostic Pages. The first thing to do is see if you have anything at all showing up on the message log. If not, re-check the PAN ID of the BB2-6040 and the devices you are expecting to receive data from. Make sure they match. Until you see something showing up on the message log, you are stuck on getting radio configurations to get in sync. Once you have data showing up, it is a matter of reviewing what you see in the message log against the data parse masks you have set up.

Appendix A Connecting Control Solutions Wireless Devices

Control Solutions uses Digi XBee Series 2 ZigBee OEM modules with standard Digi endpoint firmware. Complete details about the XBee modules may be found on Digi's web site. A link to the XBee manual may be found on Control Solutions web site, on the BB2-6040 support page.

A.1 Configuring the Temperature Sensor

The solid state sensor is located on ADI 1. Use the command "/D1 2" to enable that input. Use the /ID command to set the PAN ID. Set sleep mode as follows for reporting every 5 minutes (see OEM Manual for settings to arrive at longer periods for longer battery life).

```
/NI "----name----"  
/SN 00 3B  
/SP 01 F4  
/D1 02  
/IR FF FF  
/ST 03 E8  
/WR
```

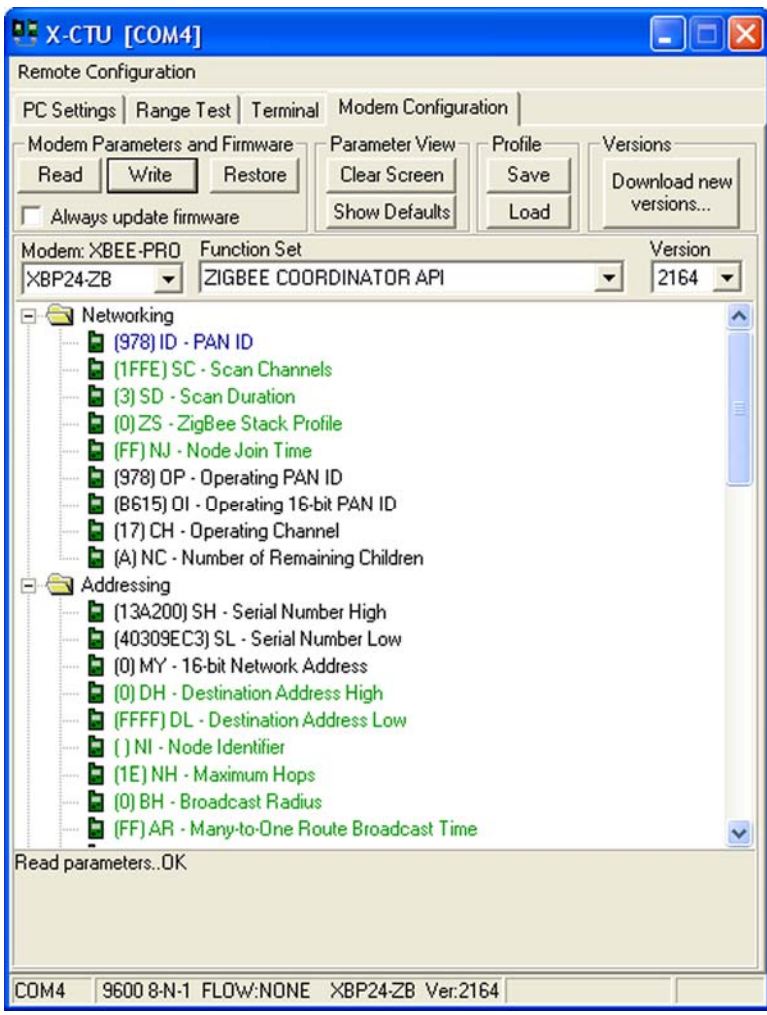
The data parse mask, assuming only D1 is enabled, will be: "92-----02XXXX". Use scale = -0.38053 and offset = 364.30.

A.2 Configuring the Router

No configuration is needed other than setting the PAN ID and sleep period. Use the /SP command to set the sleep period to a value at least as large as the longest period set in any sensor (or endpoint). The router never sleeps, but needs to know the sleep period used by endpoints so that the router knows how long to wait for a reply from the endpoint.

A.3 Generic Configuration and the X-CTU Profile

If you are configuring a generic XBee device using the X-CTU, pictured in the screen shot below, you should select the appropriate firmware version (XB24-ZB ZIGBEE ENDPOINT). Beyond that, you can use the default settings that come with that version of firmware. You can pre-program the above commands via X-CTU if you wish.



Appendix B Connecting Digi International Wireless Devices

The Digi devices mentioned here may be purchased from Digi International.

B.1 Configuring the Digi XBee Sensor

Send the following commands to the Digi Sensor, substituting your own PAN ID for the example shown.

```
/ID 00 00 00 00 00 02 34
/NI "----name----"
/SN 00 3B
/SP 01 F4
/D1 02
/D2 02
/D3 02
/IR FF FF
/ST 03 E8
/WR
```

The data parse masks will be as follows:

```
92-----0E----XXXX----- Light
92-----0E-----XXXX----- Temperature
92-----0E-----XXXX----- Humidity
```

Read device: Digi Sensor 1 applying parse mask: 92-----0E----XXXX-----
Apply binary mask: 0 Then apply scale: 0.21114 and offset: -58.0000
Save in local register: 1001 locally named: Digi Temperature 1
Apply this default value: 0.00000 after 0 minutes with no received data.

Read device: Digi Sensor 1 applying parse mask: 92-----0E-----XXXX-----
Apply binary mask: 0 Then apply scale: 0.12332 and offset: -25.8000
Save in local register: 1003 locally named: Digi Humidity 1
Apply this default value: 0.00000 after 0 minutes with no received data.

B.2 Configuring the Digi XBee SmartPlug

Set the PAN ID and Network Identifier. The write map's Data Format Mask will be as follows:

```
[17][F][A][N][02]D4[X]
```

This assumes the BACnet object mapped will have a value of 0 or 1, and further assumes an AND mask of 1 and an OR mask of 4.

Read local register: 1 named: Smart Plug 2 Relay
Write remote data: any time local register has changed by 0.00000 or when 1 minutes have elapsed with no change.
Apply scale: 0.00000 offset: 0.00000 AND mask: 1 OR mask: 4 in format string.
Format string: [17][F][A][N][02]D4[X] Write to device: Smart Plug 2

B.3 Configuring the Digi Wall Router

You will only need to set the PAN ID and sleep period in the router. Use the /SP command to set the sleep period to a value at least as large as the longest period set in any sensor (or endpoint). The router never sleeps, but needs to know the sleep period used by endpoints so that the router knows how long to wait for a reply from the endpoint.

Appendix C Connecting Point Six Wireless Devices

ZigBee sensors from Point Six Wireless (Point Six, Inc.) are compatible with the BB2-6040 but require some special handling. The Point Six devices use pin sleep mode under control of a second small microprocessor, and therefore will not recognize the remote API commands the BB2-6040 is capable of sending. You must make sure the Point Six sensor is configured properly at the factory, and/or be able to use the Point Six configuration tools provided by Point Six. The standard ZigBee BB2-6040 is unable to configure the Point Six ZigBee devices. (If you are a ZigBee/XBee expert and have a Digi USB/XBee programming board, you can remove the XBee module from the Point Six unit and reprogram its parameters that way.) (OEM versions of the BB2-6040 for Point Six Wireless devices, including XSC 900 MHz, may be available. Contact sales@csimn.com.)

C.1 Changes Required in the BB2-6040

As of this writing, Point Six ZigBee sensors were being shipped configured for stack profile 2, while as of this writing Digi sensors are shipped configured for stack profile 0, and XBee modules in general default to stack profile 0. The BB2-6040 is currently shipped configured for profile 0.

Since reconfiguring the Point Six sensor might not be convenient, you can reconfigure the BB2-6040 to match it instead. To configure the BB2-6040 for stack profile 2, use the following commands from the Local API web page of the BB2-6040:

```
/ZS 2  
/WR
```

Then restart the gateway. You may also want to reconfigure the PAN ID before restarting. To set the PAN ID of the gateway, include the following command before the /WR command:

```
/ID 00 00 00 00 00 00 12 34
```

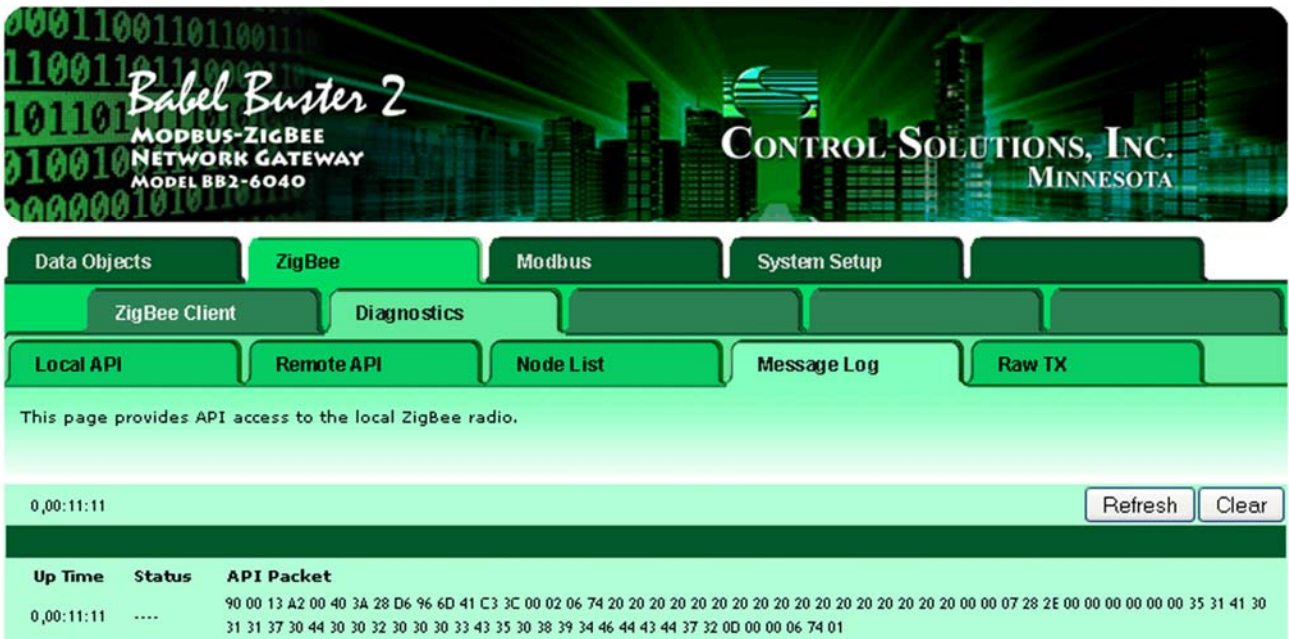
This example will set the PAN ID to 1234 (hex).

C.2 Compatibility Between Point Six and Digi Sensors

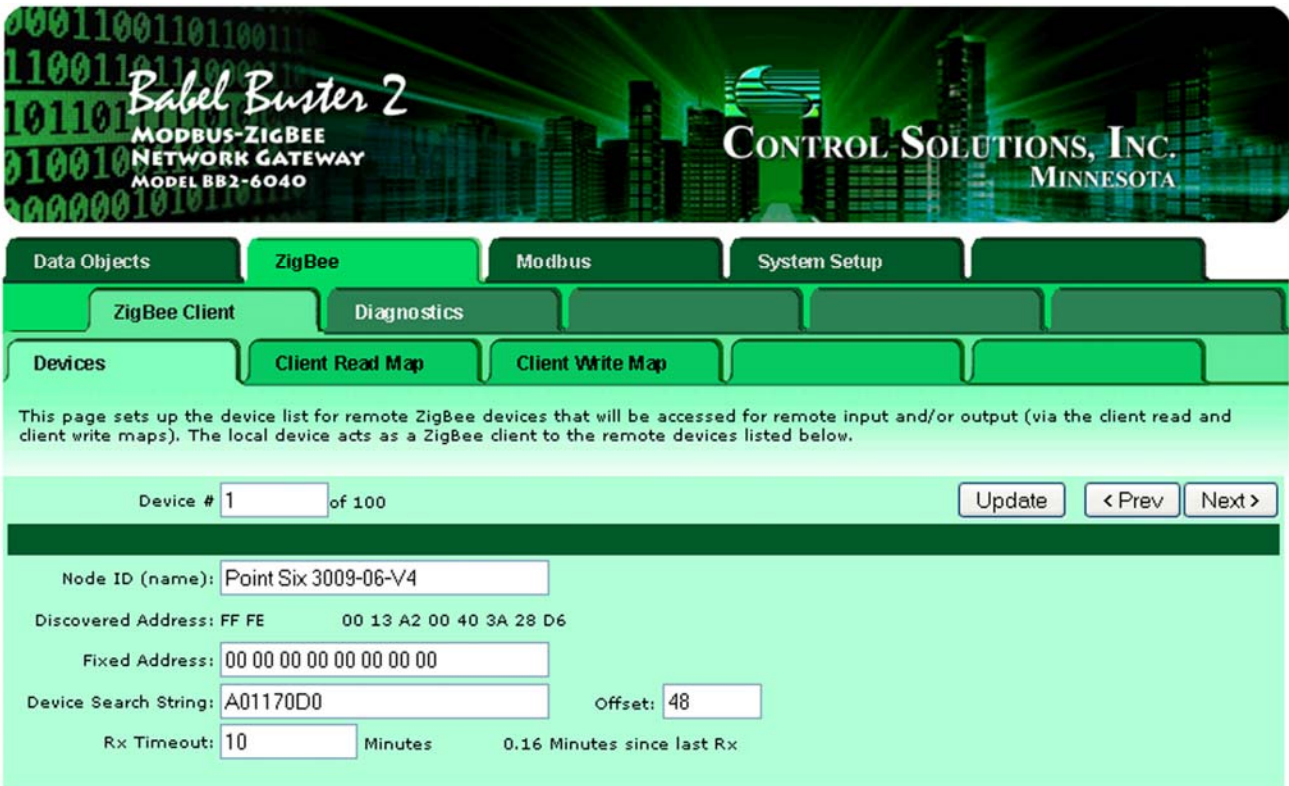
Using both Point Six and Digi sensors on the same network, or any other generic XBee product, will require that you reconfigure one or the other for the same stack profile. All stack profiles must be the same on the same network in order for devices to communicate. You can use the /ZS command in the Remote API page to set the stack profile of remote devices. Be sure to follow it with /WR to save the change past the next power cycle.

C.3 Configuring BB2-6040 to Recognize Point Six Sensors

Once the BB2-6040 is configured to the same stack profile and PAN ID as the Point Six sensor, you will see a data packet as illustrated below appear, after the sleep period has expired. The Point Six sensor data packet includes the sensor's serial number in ASCII code. The search string method of device recognition may be used to identify these sensors.



The serial number taken from the label on the back of the Point Six sensor may be entered as the device search string (with offset 48) in the device list. Once this string is recognized, the device's 64-bit address will be listed here.



The screen shots below illustrate the data parse mask for temperature and humidity data fields from the Point Six sensor. The expanded view also includes the slope and intercept for obtaining relative humidity as a percent, and temperature in degrees Fahrenheit.



This page creates a map entry that receives data from one or more ZigBee devices for processing here and presentation to Modbus via local registers. Click on map number to see more detail and insert/delete maps.

Showing 1 to 3 of 3

Map #	Remote Device	Data Parse Mask for API Frame	Local Register	Local Name
1	Point Six 3009-06-V4	90[11]C33C[50]HHHHHHHH[12]	1001	Point Six Temp F
2	Point Six 3009-06-V4	90[11]C33C[46]HHHHHHHH[16]	1003	Point Six Humidity
3	None		1005	—



This page creates a map entry that reads data from a remote BACnet IP server for processing here.

Map #

Read device applying parse mask

Apply binary mask: Then apply scale: and offset:

Save in local register locally named

Apply this default value: after minutes with no received data.

Client Read Maps Enabled:

The conversion factor for degrees Celsius is scale (slope): 0.03053, offset (intercept): -40.00.

Map #

Read device applying parse mask

Apply binary mask: Then apply scale: and offset:

Save in local register locally named

Apply this default value: after minutes with no received data.

Client Read Maps Enabled:

Appendix D Gateway Setup Quick Start

There are just a couple of settings that need to be made to the ZigBee radio in the BB2-6040 to get started. One is to set the PAN ID, and the other is to set the sleep period.

The BB2-6040 is shipped with its ZigBee radio set as a coordinator, and with PAN ID set to 0, which means it will automatically pick a random PAN ID at power-up.

To set the PAN ID to a chosen fixed ID, go to the ZigBee Diagnostics tab set, and select Local API. Enter the command:

```
/ID 00 00 00 00 00 00 12 34
```

if you want the PAN ID to be hex 1234 (for example). The PAN ID is 8 bytes, and these are entered in the API command window as 1 or 2 digit hex numbers separated by spaces.

After setting the PAN ID, send the /WR command to save this ID to non-volatile memory.

You will never see actually "OK" come back as noted in the command set document. What you will get is the API equivalent, which will be the ASCII code for the command, followed by a byte of 00 if all is well. The last byte will be an error code if it did not execute right. You need to click the Check button to see the reply at least some of the time (most of the time on the Remote API page).

Status codes for XBee endpoint devices using API mode are as follows:

00	No error
01	Error (unspecified)
02	Invalid Command
03	Invalid Parameter
04	Remote command transmission failed.

ZigBee devices that are set to the same PAN ID as your BB2-6040, or are set to "join any" PAN, will associate themselves with the the BB2-6040. To see what devices are known to the BB2-6040, go to the Node List tab, click Discover, followed by Refresh a time or two (after brief delay). The addresses of the nodes will be listed (after some longer delay if sleep periods are set long).

Important note about sleep period:

Use the /SP command to set the sleep period to a value at least as large as the longest period set in any sensor (or endpoint). The coordinator never sleeps, but needs to know the sleep period used by endpoints so that the router knows how long to wait for a reply from the endpoint. If you get errors, in particular error code 04 in a reply, it may indicate the /SP setting is shorter than the endpoint's /SP setting. If the suggested setting of /SP 01 F4 has been used on an endpoint, then /SP 01 F4 should be issued to the local radio via the Local API page. After using /SP, use the /WR command to make the setting persistent.

Revision History

1.0 October 2010 Initial Release