BARIX
THE VOICE OF SIMPLICITY

BARIONET

Universal network-enabled automation interface for home automation, commercial control, and monitoring applications

This Manual Describes:	Barionet 100 (Original Barionet)	Firmware Version: 3.02
	Barionet 50	Firmware Version 2.01
Released	January 17, 2013	

Table of Contents

1 Introduction

1.1 What is the Barionet?

The Barionet from Barix is family of programmable network-enabled automation controllers for interfacing a wide variety of devices and systems to IP-based networks in home and industrial automation applications. With the Barionet, most devices can be network-enabled for monitoring and control via a web browser as well as other standards-based automation systems, such as SNMP and Modbus.

The Barionet provides a variety of standard hardware and software interfaces as well as general-purpose inputs and outputs for control and monitoring applications.

Two Barionet models are currently available—the Barionet 100 (the original Barionet) and the new Barionet 50. The two models differ primarily in the number of inputs and outputs provided.

Barionet 100 Only

The Barionet 100 provides:

- 4 digital outputs
- 4 analog or digital inputs
- 4 digital inputs
- 2 relay outputs
- RS-422/485 serial interface
- Wiegand reader Interface

Barionet 50 Only

The Barionet 50 provides:

- 4 relay outputs
- 4 dry contact inputs
- RS-485 serial interface

Both the Barionet 100 and Barionet 50 provide:

- 10/100 Mb auto-sensing Ethernet port
- Integrated web server
- RS-232 serial interface
- Dallas Semiconductor 1-wire® interface with built-in support for temperature sensors.
- Support for Modbus/TCP, SNMP, CGI, HTTP and other standard protocols.
- Barix proprietary dynamic tags for creating custom web pages to display status information.
- Programmability via BASIC-like Barix Control Language (BCL).
- Support for control of external Modbus devices via BCL code.

Barix also offers a variety of I/O expansion modules that can be connected to the Barionet using the RS-485 port for additional I/O capability. Other manufacturers I/O modules and smart sensors can be controlled using the Modbus protocol via the RS-485 port.

1.2 Barionet Applications

Applications for the Barionet are as varied as your imagination. These are just a few simple application ideas:

- Industrial control and monitoring, such as temperature and voltage monitoring, with alarms via email and/or SNMP if voltages or temperatures fall outside an acceptable range.
- Home automation for control of thermostats and lighting as well as web-based remote status monitoring.
- Web-enabling devices with RS-232 or Modbus interfaces for remote control and access.

- The Barionet has a built-in serial gateway function that allows you to control an RS-232 device over a TCP/IP network.
- A sample BCL application ships with the Barionet that mimics digital inputs on one Barionet on the digital outputs of a second Barionet connected by a TCP/IP network.
- Access control and security systems driving horns or lights via relay outputs and providing alarm messages via email or SNMP.
- Remote data collection and logging for digital and analog inputs.

1.3 About This Manual

This manual is intended for Barionet users and developers. The hardware interfaces chapter assumes basic electronics knowledge, and the software interface chapter assumes some familiarity with internet protocols and HTML. You can use the Barionet without advanced electronics or software experience, but interfacing the Barionet to external devices and creating custom applications may require additional skills.

Barix is committed to helping you be successful with the Barionet, so we provide a variety of technical resources in addition to this manual. These resources are listed later in this chapter.

This manual describes both the Barionet 50 and the Barionet 100.

Where the manual describes a function or feature that is specific to the original Barionet (now called the Barionet 100), this symbol appears in the left margin.

Where the manual describes a function or feature that is specific to the Barionet 50, this symbol appears in the left margin.

Note

The original Barionet (now called the Barionet 100) is often referred to without the model number. When the new Barionet 50 was introduced, the original Barionet was renamed the Barionet 100. However, some of the older materials still refer to the original model as the Barionet. For the sake of clarity within this manual, when we refer to the specific Barionet models we will use "Barionet 50" and "Barionet 100". Where we use the term "Barionet" without a model designation, we are referring to the product family and describing characteristics of both models.

This manual assumes the following firmware versions are installed in the Barionet 50 and Barionet 100:

Barionet 100 Firmware: V3.02

Barionet 50 Firmware: V2.0

For instructions on updating the firmware in your Barionet, please refer to Chapter 8.

For the installation of the Barionet please refer to the corresponding “Quick Install Guide”. A printed version is included in the box and can also be downloaded from our web site at www.barix.com.

1.3.1 Notes, Warnings, and Cautions

Throughout this manual, you will find paragraphs that have a special boxed heading of either "Note", "Warning" or "Caution". These paragraphs contain important information defined by the heading.

Note

Note paragraphs describe information that is important for the reader to understand to avoid functional or procedural problems, but the information does not pose a significant risk of damage to the product or personal injury.

Caution

Caution paragraphs describe information that is important for the reader to understand to avoid potential damage to equipment.

Warning

Warning paragraphs describe information that is critical for the reader to understand and abide by to avoid the risk of personal injury.

1.4 Chapter Overview

1.4.1.1 Chapter 1: Introduction

You are currently reading the introduction chapter.

1.4.1.2 Chapter 2: Accessing and Configuring the Barionet

This chapter helps you get started with accessing the Barionet via a web browser and describes the use of the built-in configuration and status web pages. It also describes each of the configuration screens and settings in detail.

1.4.1.3 Chapter 3: Hardware Interfaces

The Hardware Interfaces chapter focuses on information required to connect the Barionet to external devices, such as switches, relays, lamps, etc. This chapter describes the Barionet hardware interfaces as well as considerations for connecting external devices to the inputs and outputs of the Barionet and provides simple schematics for suggested configurations. Separate sections are devoted to the Hardware interfaces for each Barionet model.

1.4.1.4 Chapter 4: Software Interfaces

Chapter 4 looks at each of the software interfaces that the Barionet provides and helps you choose the software interfaces that best suit your application. The chapter begins with a summary table of all the software interfaces and then goes on to describe each one in more detail.

1.4.1.5 Chapter 5: Creating Custom Applications & Web Pages

This chapter provides an overview of the Barionet's features for developing custom web pages and applications. The process for uploading customized web pages and BCL applications is also discussed. Detailed information on the BCL language is provided in a separate manual, the Barix Control Language (BCL) Programmers Manual. The BCL Programmers manual is available for downloading from the Barix web site.

1.4.1.6 Chapter 6: The Sample Digital I/O and Serial Tunnel Application

The Barionet comes pre-loaded with a sample BCL application and a custom application configuration page for the sample application. The source BCL code for the application and the HTML source for the configuration page are both provided in the development kit, which is available for download from the Barix web site. This chapter describes the functionality of the sample application and its configuration page. You can use the sample application as is, or you can use the BCL code and custom HTML pages as an example for

creating your own applications and custom pages. The complete source for the sample application is supplied in the Update, Rescue and Development Kit, which is available for download on the Barix web site.

1.4.1.7 Chapter 7: Troubleshooting

This chapter discusses procedures for troubleshooting and debugging Barionet applications. Procedures for rebooting the Barionet, resetting to factory defaults, using SYSLOG messages for debugging, and the serial rescue procedure are discussed.

1.4.1.8 Chapter 8: Updating Barionet Firmware

This chapter provides detailed instructions for updating the Barionet 50 and Barionet 100 firmware.

1.4.1.9 Appendix A: I/O Addressing

Appendix A provides a concise reference to the I/O addressing scheme in the Barionet. These I/O addresses are used in custom HTML tags, BCL applications, and Modbus/TCP commands.

1.4.1.10 Appendix B: Configuration and Setup Memory Layout

This appendix provides a quick reference for the layout of the configuration and setup memory, including space available for user configuration parameters.

1.4.1.11 Appendix C: Accessories

Appendix C lists and briefly describes accessories available from Barix for the Barionet

1.4.1.12 Appendix D: Mounting the Barionet

Appendix D describes physical mounting considerations for the Barionet.

1.4.1.13 Appendix E: Glossary

Appendix E is a glossary of terms used throughout this manual. Many of the terms in the text are linked directly to a definition in the glossary in the PDF version of the manual. Where a term is linked to a definition in the glossary, it is displayed in underlined dark blue text.

1.4.1.14 Appendix F: Specifications & Warranty

Appendix F lists the specifications and warranty for the Barionet.

1.4.1.15 Appendix G: IP Addresses, Netmasks and Gateways

Appendix G briefly describes IP addressing and the relationship of net masks to addresses and the function of the gateway IP address. This appendix discusses IPv4 addresses (and briefly mentions the differences with IPv6).

1.5 Additional documents

The following additional documents are available from Barix for the Barionet.

1.5.1.1 Barionet Quick Start

This document provides a brief introduction to the Barionet with a quick reference for interface pins. A printed copy of this guide ships with each Barionet and the document is also available for download at: http://www.barix.com/downloads/Barionet_Family/51/ Each Barionet model has a unique quick start guide.

1.5.1.2 Barix Control Language Programmer's Manual

For Detailed information on developing custom applications with the Barix Control Language (BCL), refer to the Barix Control Language (BCL) Programmer's Manual. This manual is available for download at: http://www.barix.com/downloads/Barionet_Family/51/

1.5.1.3 I/O Expansion Module Manuals

The manuals for the Barix expansion modules (models IO12, X8, R6) provide information on connecting and using the expansion modules, including details of Modbus interfacing and use with the Barionet.

1.6 Getting Support

Barix is committed to helping you succeed with the Barionet. The first line of help should always be to read the manuals carefully. If after reading the manuals, you are still having trouble, here are some support resources that should help answer your question:

1.6.1.1 Barionet Distributors

Check with your Barix distributor for technical support resources they offer. A list of distributors is available on the Barix website. Some of the dealers and distributors specialize in the Barionet.

1.6.1.2 Barionet Users Forum

The Barionet user's forum brings together new and experienced Barionet users from all over the world, along with technical support representatives from Barix. Users exchange information, questions and ideas and Barix support staff answer questions as well. This is a great resource for learning from others and sharing experience. The forum also provides a search function that allows you to search for forum posts on a particular topic.

A link to the Barionet User's forum is provided on the Barix support web page at:

http://www.barix.com/Support__FAQs/71/

1.6.1.3 Barix BCL Demo Applications Site

Barix operates a website that allows BCL developers to share the applications they have developed for Barix hardware. A number of Barix-created demo applications are also available for download as examples and a starting point for developing your own applications. You can find this site at:

http://www.bcl-applications.info/

1.6.1.4 Barix Wiki

Barix staff also maintains a wiki for all Barix products, including the Barionet 50 and the Barionet 100. The Wiki contains information on the features and applications for the products as well as hints and tricks that may not be included in the manuals.

1.6.1.5 Email Support

You can email a Barix support technician at support@barix.com.

2 Accessing and Configuring the Barionet

The Barionet includes a built-in web server and comes pre-loaded with web pages that allow you to configure various settings of the Barionet as well as monitor the status of inputs and control outputs.

This chapter provides detailed instructions for connecting the Barionet to a network and accessing the standard status and configuration pages for the first time.

2.1 Accessing the Barionet for the first time

2.1.1 Connecting to an Ethernet/IP network

The first step in preparing to access the Barionet's configuration and status pages is to connect it to an Ethernet network and assign it an IP address.

The Barionet is equipped with a standard Ethernet 10/100 Mbit, full / half duplex, auto negotiation interface. Connect the Barionet to an Ethernet network using an RJ-45 Ethernet cable to a hub or switch. You'll also need a computer connected to the same network. If you are going to connect the Barionet directly to a computer without a hub or switch, you must use a "crossover" cable, which is a specially-wired cable available at many electronics supply stores.

2.1.2 Supplying Power

Before you can access the Barionet's built-in status and configuration web pages, you must connect the Barionet to an appropriate power source. The Barionet does not come with a power supply.

The Barionet 50 and Barionet 100 can be operated on any DC power supply with an output voltage between 9 and 30 volts DC. The overall power consumption is 4 watts maximum (all relays active).

Please refer to Power Supply Inputs (J6) in Chapter 3 for more details on connecting the Barionet 100 to a power supply and Power (J2) in Chapter 3 for more details on connecting the Barionet 50 to a power supply.

2.1.3 Using the Barionet Discovery Tool

The Barionet must have a valid IP address before you can use most of the functions available via the Ethernet interface. The easiest way to find the Barionet's IP address and/or change the Barionet's address assignment, is using the Barix Discovery Tool. Other methods are outlined later in this chapter.

The Discovery tool is supplied with the development and update kits after version 2.30 (for the Barionet 100) and 1.04 (for the Barionet 50). You can also download it from the Barix web site at: http://www.barix.com/downloads.

The Discovery tool is written in the Java programming language, so it requires a Java Runtime Environment (JRE) installed on your computer. If you do not have a JRE installed, you can download and install it from: http://java.com/en/download/manual.jsp. Java run time environments are available for all major operating systems. If you are running the discovery tool on a Linux or UNIX platform, the Discovery tool also requires the X-window graphical user interface.

The Discovery tool is distributed in a Java Archive (.jar) file. On most operating systems you can run the discovery tool by simply double-clicking on the discover.jar file. The tool should start up and the initial screen shown in Figure 1 should appear.

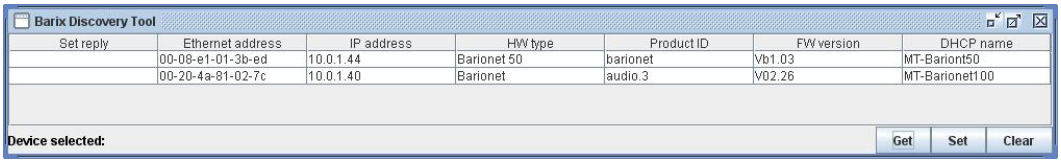


Figure 1. The Barix Discovery Tool helps to find Barionets on the same local network as the computer running the tool.

Click the "Get" button to initiate a search. If the Discovery tool finds one or more Barionets on the network, each device will be listed in the table, including its current IP address, MAC address (labeled "Ethernet Address" in the Discovery tool), firmware version, and other information.

The Discovery tool should find any Barionet devices that are on the same network as the computer running the tool, regardless of their current IP address setting. The tool will not search through a router to another subnet.

Note

With some versions of Barionet 100 firmware prior to V2.30 or Barionet 50 version 1.04, the Discovery tool will only find Barionets if they are on the same subnet as the computer running the discovery tool. For example, with the older firmware, if the computer is set for IP address 10.0.1.5 and the Barionet has an IP address of 192.168.0.6, the Discovery tool will not find the Barionet.

The IP address displayed in the discovery tool may be a dynamic IP address, assigned by a DHCP server or other automatic address assignment protocol. The Barionet comes from the factory with the IP address set to 0.0.0.0, which means that it will automatically attempt to acquire an IP address using one of several methods. See Setting the Barionet IP address automatically later in this chapter for more information on automatic address assignment.

If the Barionet is configured for automatic IP address assignment (IP address = 0.0.0.0), the Discovery tool will show the address it's been assigned, not the 0.0.0.0. You can use the address that appears in the Discovery tool to access the Barionet via a web browser. However, keep in mind that the IP address may change if the Barionet is rebooted.

You can also set the address in the Discovery Tool by double clicking on the IP address field of the Barionet you wish to change. When you double click the address field, the IP address field will become editable. Enter the IP address you want to assign to the Barionet. Then click "Set". The "Set reply" field should say "No error" if the address assignment was successful.

If you decide to assign the Barionet a static IP address, you must be sure to use an address that is outside the range of addresses that are automatically assigned by any DHCP server on the network. See Setting the Barionet IP address manually later in this chapter for more information.

If you've been successful at finding and/or setting the IP address of the Barionet with the discovery tool, you can skip the next few sections and go right to Opening the Configuration and Status Web Pages later in this chapter.

2.1.4 Alternatives to the Discovery Tool

If you are unable to use the Barix Discovery Tool to set the IP address or the Discovery Tool doesn't find the Barionet on your network, there are three alternative methods for finding and/or setting the Barionet's IP address.

Each of the procedures below assumes that the Barionet is physically connected to the Ethernet network and that a computer is connected to the same network with an IP address that is in the same subnet (i.e. address range) as the Barionet. If the network uses dynamic address assignment, such as DHCP, this will happen automatically.

The three alternatives to using the Barix Discovery Tool are:

1. The Barionet can obtain an IP address from an external server running the DHCP protocols. Using the DHCP server's administrative interface, you can find the IP address the DHCP server has assigned to the Barionet.
2. You can find the Barionet's IP address using syslog output when the Barionet boots.
3. The Barionet's IP address can be manually assigned using **arp** and **telnet** commands.

2.1.4.1 Alternative 1: Getting the Barionet's address from the DHCP or BOOTP server

The default method the Barionet uses for obtaining an IP address is to attempt to contact a DHCP or BOOTP server on the network. The DHCP or BOOTP server assigns the Barionet an unused IP address within the range of the subnet of the current network. In many small networks, the router or server computer serves as a DHCP or BOOTP server.

If you have access to the administration functions of your router or DHCP/BOOTP server, you should be able to look up the address that the server assigned to the Barionet. Most servers provide access to a table that lists the addresses assigned (sometimes referred to as "leased") to each device on the network that uses the DHCP or BOOTP protocol. Figure 2 shows a typical address table from a DHCP server built into a router.

If you do not have access to the administrative functions of your router, or the network does not have an active DHCP or BOOTP server, try using one of the other methods described in this chapter.

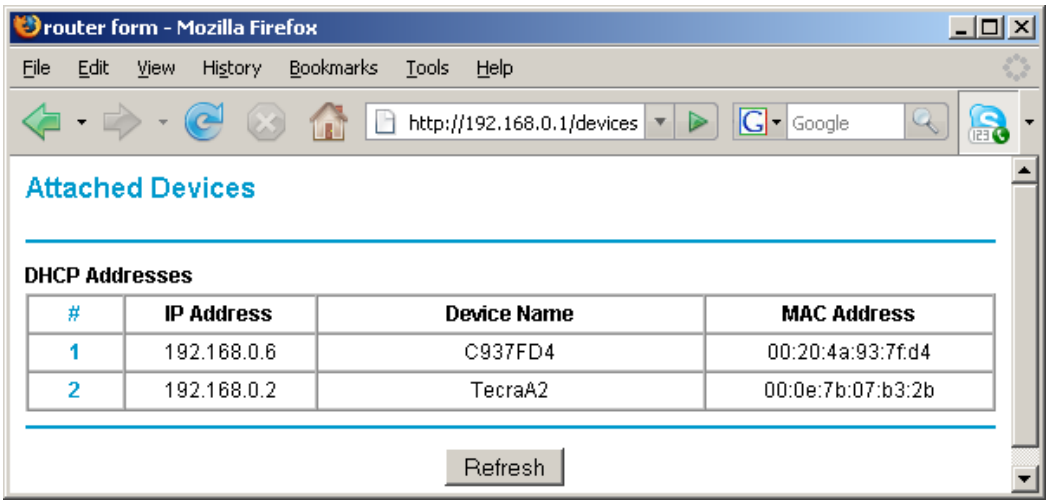


Figure 2. A typical DHCP address table

To find the IP address of the Barionet, locate its hardware MAC Address, which is printed on the label on the bottom of the Barionet. Find the corresponding MAC address in the DHCP address table and write down the IP address that corresponds with the Barionet's MAC address in the table.

If the DHCP Host Name setting in the Barionet's configuration is set, the host name will appear in the "Device Name" column of the table as well. If the Barionet is set to the factory default configuration, the Device Name will be a default value that is derived from the Barionet's MAC address. Note that the table shown above is only an illustration of a typical DHCP server address table. Your DHCP server table may appear somewhat different.

You will use the IP address from this table corresponding to your Barionet to access the Barionet's built-in status and configuration pages. Remember, this IP address may change any time the Barionet is rebooted, so you'll have to either set a static IP address in the configuration web pages, or check the DHCP server table after each Barionet reboot to find the Barionet's IP address.

2.1.4.2 Alternative 2: Getting the IP Address from SYSLOG messages

This method requires you to install a program on your computer that can receive and display the system log (syslog) messages that the Barionet issues by default. For Microsoft Windows computers, we recommend the Kiwi Syslog Daemon available for free download from www.kiwisyslog.com. There is a free syslog viewer dashboard widget available for the Macintosh as well.

1. Install and run the syslog program on your computer connected to the same network as the Barionet.
2. Power up the Barionet. If it was already powered up, remove power for a couple of seconds and then re-connect the power to restart the Barionet.
3. When the Barionet restarts, it should issue at least a couple of syslog messages, which should appear in the Kiwi syslog program. The source IP address of the Barionet is displayed with the messages. This IP address is what you'll use to access the Barionet via the web browser.

If these messages do not appear, either the Barionet was not able to acquire a valid IP address or the syslog setting in the Barionet has been changed from the default broadcast setting for syslog output. You'll need to try one of the other procedures for finding or setting the Barionet IP address.

2.1.4.3 Alternative 3: Setting the IP address manually using ARP

With this method, you will temporarily set an IP address for the Barionet. Using the temporary address, you can access the Barionet's configuration pages, where you can set a permanent address.

Note
The IP address you set with this method is only temporary. In order to access the device after a power-cycle, you must set the IP address permanently in the Barionet's configuration web pages. See the description of the Network Settings Page to set the IP address permanently.

1. Open a command line window. In Microsoft Windows, click "Start" and the "Run". In the dialog box type "cmd" and press enter. On the Macintosh run the Terminal program, normally located in the Utilities folder. On Linux open a terminal window if you are using a graphical user interface or enter the following commands directly in a terminal console session.
2. You must assign a temporary address that is not used by any other devices on the network and is in the same subnet as the computer that will access the Barionet. Contact your network administrator if you are not sure what address is available to assign to the Barionet. You can use the **ping** command to check to the address is free. In the command window, type:

   ping <IP address>

   Where <IP address> is the address you want to check to be sure it's not already assigned. For example, if you want to check if the address 192.168.0.6 is not assigned to any other device, type:

   ping 192.168.0.6

   Figure 3 shows a typical response to the ping command on Windows when the IP address is not assigned to any other device on the subnet.

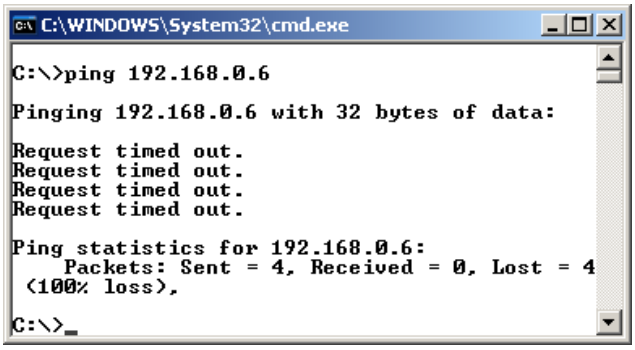


Figure 3. The results of a ping command for an unused IP address

A "Request timed out" or "Destination host unreachable" response indicates that there was no other device on the network at the address 192.168.0.6. If there is any other response, the IP address is in use, so you'll need to use a different address.

3. Write down the Barionet's MAC address, which is printed on the label on the underside of the Barionet. The MAC address is 12 hexadecimal digits (0 – 9 and A – F). When you enter the MAC address in the next step, you'll need to insert either hyphens (Windows) or colons (Mac OS-X or Linux) between each pair of digits, so it might be helpful to write it down in the correct format with hyphens or colons between each pair of digits. For example:

   00-20-4A-93-7F-D4 (Windows)

   00:20:4A:93:7F:D4 (Mac OS-X or Linux)

4. Enter the following command in the command window, followed by the Enter key. Substitute the IP address you want to assign to the Barionet for <IP address> below and substitute the Barionet's MAC address for <MAC address> in the command.

   arp -s <IP address> <MAC address> (Windows)

   sudo arp –s <IP address> <MAC address> (Mac OS-X or Linux)

   For example, to set the Barionet's IP address to 192.168.0.6 with a Barionet whose MAC address is 00-20-4A-93-7F-D4, the command would look like this:

   arp -s 192.168.0.6 00-20-4A-93-7F-D4 (Windows)

   sudo arp –s 192.168.0.6 00:20:4A:93:7F:D4 (Mac OS-X or Linux)

   For OS-X and Linux, you must supply the administrator's password when prompted.

   You can use the –a switch in the arp command (**arp –a)** to check that you entered the correct MAC and IP addresses. Look for an entry that matches the MAC address of the Barionet and confirm that you entered the correct IP address.

5. The arp (address resolution protocol) command tells your computer to associate the Barionet's MAC address with the IP address you specify. Next, we must tell the Barionet to use this IP address by attempting to connect to that IP address on port 1 using the Telnet command. Enter the following command:

   telnet <IP address> 1

   Again, substitute the IP address you used in the **arp** command here for <IP address>. Using the same IP address as our previous example, the command would be:

   telnet 192.168.0.6 1

   This command should generate an error message because the Barionet will refuse a connection on port 1. Don't worry—that's normal. The purpose of this command is simply to set the Barionet to listen to the specified IP address. The error message is not meaningful here.

Note

The error message that results from the telnet command should occur almost immediately. If there is a significant delay (i.e. greater than about 5 seconds), the Barionet may not be reachable. Try cycling power on the Barionet and repeating the procedure. If it still fails, the serial rescue procedure can be used to restore the Barionet.

At this point, the Barionet should be assigned the IP address you specified in the **arp** and **telnet** commands. You can check that the address assignment succeeded by using the ping command again with the IP address you assigned the Barionet. Figure 4 shows the response to the ping command when the address assignment succeeded.

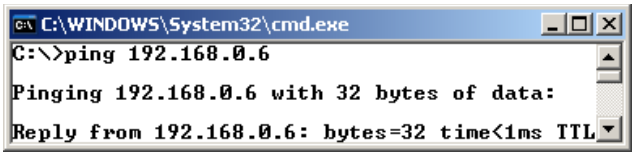


Figure 4. If the address assignment was successful, the response should look like this.

If the response says "Request timed out", the address assignment failed. Try repeating the steps again, starting with step 4 above. Be sure you included the "1" at the end of the telnet command.

If the assignment worked, the Barionet will now respond to web page requests at the assigned IP address. Remember, however, that this IP address assignment will be lost if the Barionet's power is interrupted, so be sure to set the IP address of the Barionet in the configuration pages before removing power or rebooting the Barionet.

2.2 Opening the Configuration and Status Web Pages

You can now access the standard configuration and I/O status web pages using any standard web browser.

Open your browser and in the address bar type:

http://<IP address>/

Substitute the IP address of your Barionet for <IP address>. For example, if your Barionet is assigned to IP address 192.168.0.6, enter the following in your browser's address bar:

http://192.168.0.6/

Your browser should display the Barionet 100's standard I/O status home page as shown in Figure 5. The Barionet 50's I/O status page is shown in Figure 6.

Barionet 50 Only

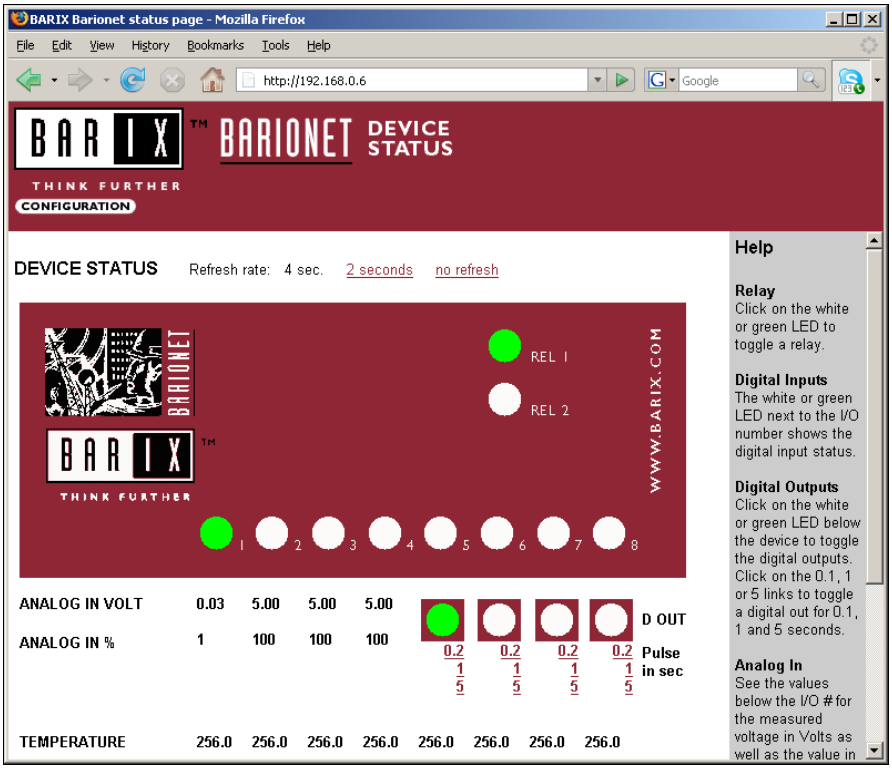


Figure 5. The Barionet 100 I/O status home page.

Barionet 100 Only

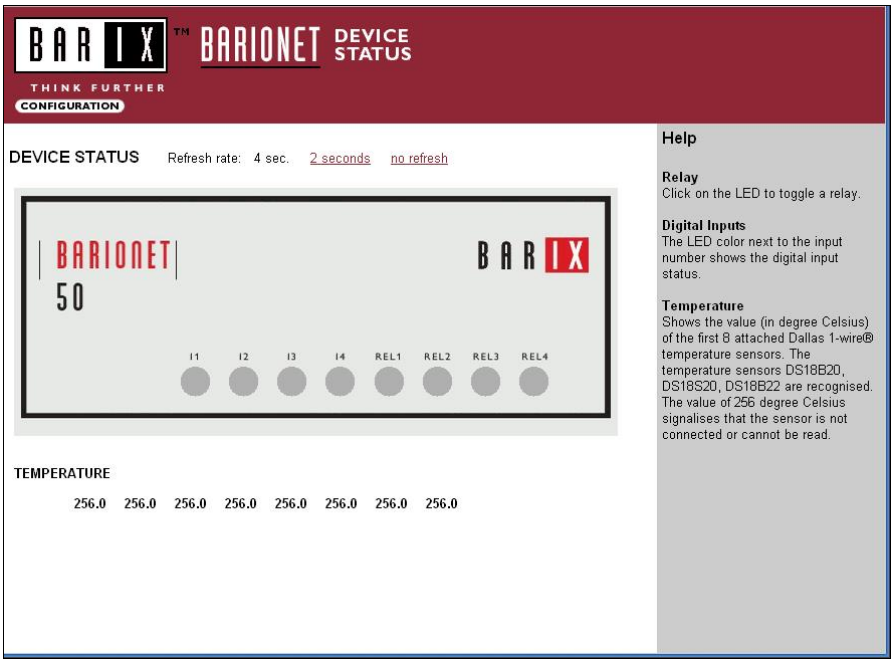


Figure 6. The Barionet 50 I/O status home page

This status page automatically refreshes itself at 4 second intervals. However, you can change the refresh rate to 2 second intervals or disable the refresh entirely by clicking on one of the refresh rate links just below the Barionet banner at the top of the page.

You'll also find a "Help" section on the right edge that defines each of the elements on the page. Use the scroll bar next to this section to scroll down for more help information.

2.3 Using the Home I/O Status and Control Page

The Barionet I/O status home page shows the status of its inputs and outputs and also allows you to control the outputs. The page shows indicators for each of the inputs and outputs. White or gray indicates that the input or output is OFF (inactive), and green indicates that the input or output is ON (active). You can also control the state of each of the outputs by clicking on the indicator to change its state or on the pulse links below the digital outputs to turn the output on for the specified interval and then automatically turn it

back off.

Barionet 100 Only

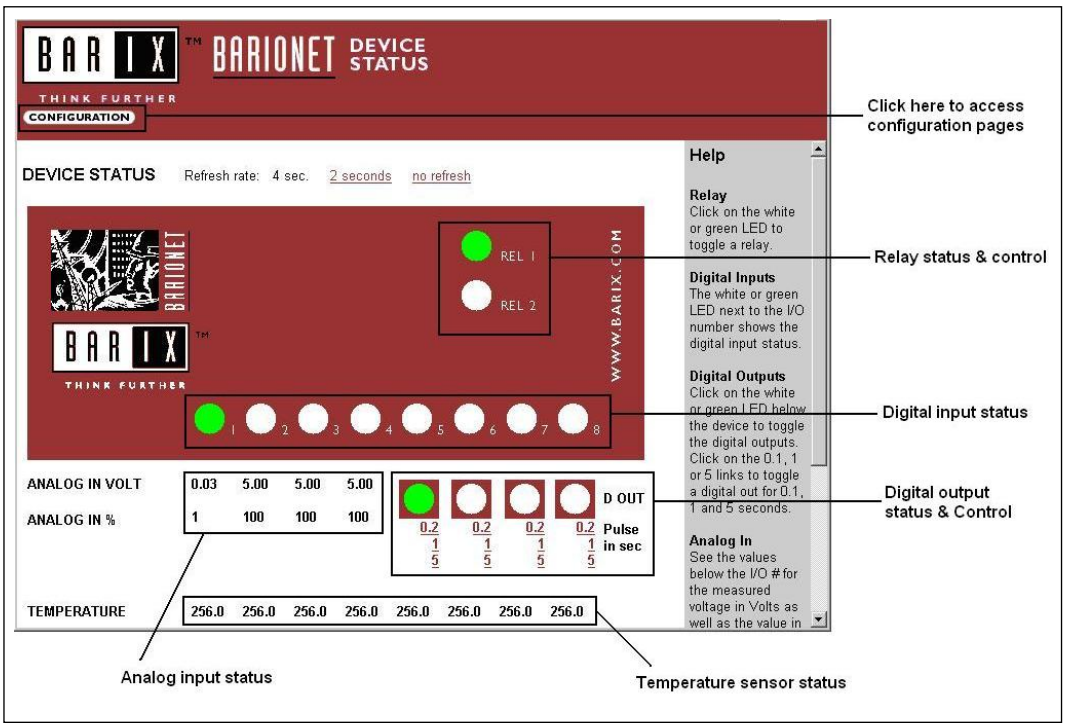
The Barionet 100 has eight inputs, four of which may be used as either digital or analog inputs. It also has four digital outputs as well as two relay outputs. It can also monitor up to 50 1-Wire temperature sensors. Figure 7 shows the Barionet's home I/O status and control page with the various elements on the page labeled.

To turn one of the relay outputs on, click on the white status indicator. If you listen carefully, you can hear the relay output switch and the relay status indicator should turn green at the next refresh. Note that if you have disabled the automatic refresh, the indicator status will not change after you click the indicator, even though the relay will switch on or off.

The eight input status indicators show the status of each of the eight inputs. The first four inputs may be configured as analog inputs, so their corresponding input voltage level is shown below the indicators. The last four inputs are dedicated digital inputs. Clicking on these status indicators has no effect as they are inputs only.

The status of the four general purpose digital outputs is shown to the right of the analog input levels. These outputs can be turned on and off just like the relay outputs. However, because they are not relays, you won't hear any switching sound when they change states. The status indicators will change on the next refresh of the status page.

You can also turn these outputs on for a short period of 0.2 seconds, 1 second, or 5 seconds by clicking on the corresponding links below the output indicators. Remember that the status indicators are only updated when the page refreshes, so you may not even see the status indicator change if you click one of the pulse links, since the output may change faster than the refresh rate of the status screen.



Barionet 100 Only

Figure 7. The Barionet 100 I/O Status and Control Page.

Finally, the status of up to eight 1-wire temperature sensors is displayed near the bottom of the screen. Temperatures are displayed in degrees Celsius. Where a temperature sensor is not present, it displays 256 degrees C on this page.

The Barionet 50 has four digital inputs and four relay outputs and can also monitor up to 50 1-Wire temperature sensors. Figure 8 shows the Barionet 50 home I/O status page.

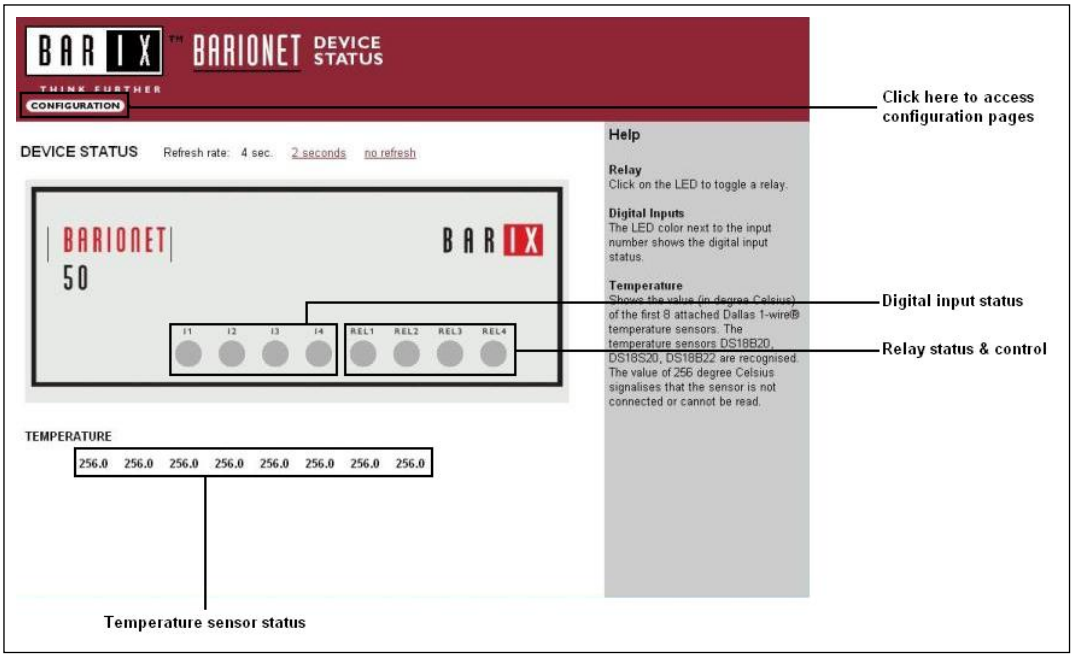


Figure 8. The Barionet 50 I/O Status and Control Page.

To turn one of the relay outputs on, click on the gray status indicator. If you listen carefully, you can hear the relay output switch and the relay status indicator should turn green at the next refresh. Note that if you have disabled the automatic refresh, the indicator status will not change after you click the indicator, even though the relay will switch on or off.

You can create custom pages that display different information, or in different formats. We'll discuss the process of creating custom pages in Chapter 5.

the Configuration Web Pages

Barionet 50 Only

The Barionet's built-in web pages also include pages for setting a wide variety of configuration options. This section shows each of the configuration pages and describes the parameters and options on each page.

To access the configuration pages, click on the Configuration button near the top of the page, just under the Barix logo.

The configuration settings are divided into eight web pages accessed by the tabs at the top of the configuration pages as shown in Figure 9.

The following sections describe each of these settings pages.

1 The Network Settings Web Page

The Network Settings page allows you to set the IP address, Netmask, Gateway, and DNS server for the Barionet. See Figure 9.

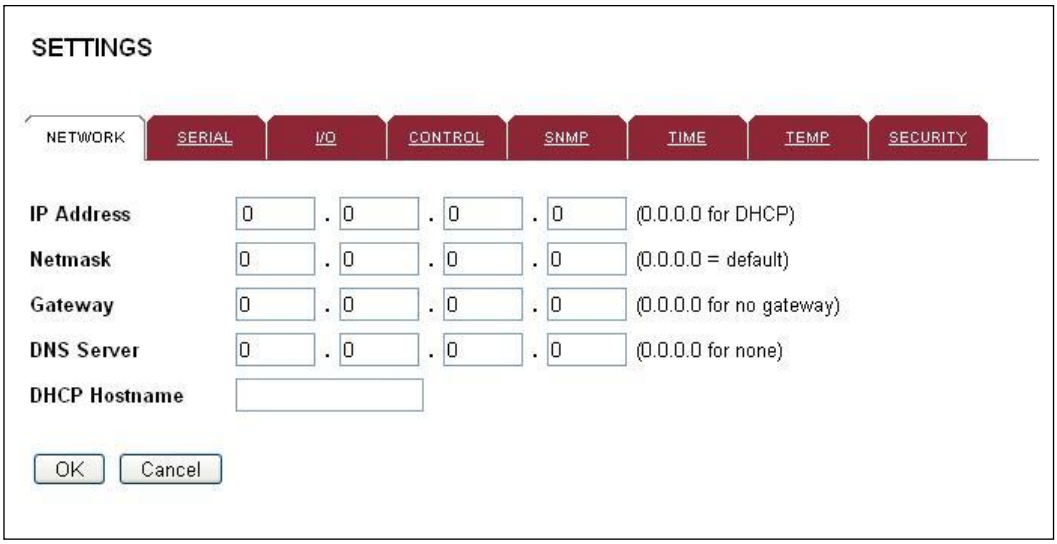


Figure 9. The Network Settings Page.

Note

If you set the IP address of the Barionet temporarily using Alternative 3 described earlier in this chapter, that IP address will not appear in the IP address boxes on this page. If you want to assign a static IP address or control the automatic IP assignment protocols, you must set the IP address in the configuration pages before restarting the Barionet. Otherwise, the temporary IP address you set using the ***arp –s*** *and* ***telnet*** *commands will be lost when the Barionet is rebooted. The following sections discuss how to set the IP address manually or for automatic operation.*

2.4.1.1 Setting the Barionet IP address manually

In some cases, it's desirable to set the IP address of the Barionet manually, instead of allowing a DHCP server or other protocols to automatically set the Barionet's IP address. Setting the IP address manually has the advantage that once the address is set, it does not change unless you reset it, so you always know the IP address to use to access the Barionet. This is sometimes referred to as a "static IP" because the IP address does not change.

However, when you set the IP address manually, you are responsible for insuring that the IP address assigned to the Barionet is not assigned to any other device. If your network uses DHCP or other automatic assignment protocols, you must insure that the IP address assigned to the Barionet is not within the range of addresses that could be automatically assigned by the DHCP server.

The Barionet always checks the IP address that it is assigned (whether manually or automatically) to see if the address is already associated with another device. If a manually assigned address is already assigned to another device, the Barionet reboots and tests the address again. This may result in the Barionet being unreachable. To correct the address conflict you can:

- Temporarily assign an available IP address using Alternative 3 described earlier in this chapter and then change the address assignment in the configuration pages.
- Disconnect or disable the other device that has been assigned the same IP address in order to access the Barionet to change its IP address.

If you are connecting the Barionet to a larger network where you are not the administrator, it is important to check with the administrator before assigning an IP address to the Barionet.

To set the IP address, simply enter the values in the four boxes labeled "IP Address". The

Barionet must be rebooted for the new IP address to take effect. Click the "Reboot" button near the top of the screen and click "OK" to reboot the Barionet. You may want to make all the changes required in the Network Settings screen before rebooting the Barionet.

Refer to Appendix G IP Addresses, Netmasks and Gateways for more information on choosing an appropriate static IP address.

2.4.1.2 Setting the Barionet IP address automatically

The Barionet supports four different methods for automatic IP address assignment. You can configure the Barionet to try each of the four methods to obtain an IP address, or you can control which of the four methods it uses. The methods are executed in a specific order and the first method that succeeds will stop the process. These are the four methods the Barionet attempts to use, in the order of their execution, with a brief explanation of the method's requirements. See the glossary entries for each method for more specific information.

1. BOOTP: The Barionet attempts to retrieve an unused IP address from a "configuration" server. If there is no configuration server that responds to the BOOTP protocol, this protocol will fail, and the Barionet will attempt to use the DHCP protocol.
2. DHCP: The Barionet attempts to retrieve an unused IP address from a DHCP server. If there is no active DHCP server on the network, the Barionet will attempt to use the IPZator method. The Barionet waits for roughly 12 seconds for a response from BOOTP or DHCP before attempting to use the IPZator or AutoIP methods.
3. IPZator: This is a Barix-proprietary protocol that does not require an external server to assign IP addresses. It attempts to locate an unused address within the current subnet by listening to the network traffic. A class-C network is assumed (netmask 255.255.255.0). The Barionet tests sequential addresses starting with 168 in the last octet (e.g. xx.xx.xx.168). It uses the first address it finds in this range that is unused. The IPZator method can be quite fast if there is substantial network traffic. However, it may fail entirely after 3 minutes if it does not find any existing network traffic (excluding traffic in the AutoIP range of 169.254.x.x).
4. AutoIP: The AutoIP protocol also does not require any external server. However, it always assigns the Barionet an address in the range of 169.254.x.x. It randomly tries addresses in this range and finds the first free address. This method should only be used in small networks that are not connected to a larger network or the internet. The AutoIP method assumes a class-B network (netmask 255.255.0.0).

The default IP address setting for the Barionet in the configuration screen is 0.0.0.0. This setting tells the Barionet to try all four of the methods described above in the order shown. If you want to disable any of these methods, you can use a special IP address that is not otherwise valid to control which methods are executed. Set all but the third value to zero according to the following table:

0.0.0.0: Enables all four methods

0.0.1.0: Disables the AutoIP method

0.0.2.0: Disables the DHCP method

0.0.4.0: Disables the BOOTP method

0.0.8.0: Disables the IPZator method

Any combination of these settings can be added together to disable multiple methods. For example:

0.0.5.0: Disables the BOOTP and AutoIP methods

0.0.13.0: Disables all methods except DHCP

Note

If you set the IP address to 0.0.15.0, you disable all four automatic addressing methods and you also don't have a valid static IP address that can be used to communicate with the Barionet. The only way to communicate with the Barionet with this IP address setting is to manually reset the IP address using the technique described in Alternative 3 earlier in this chapter and then change the IP address in the web configuration settings. You can also reset the Barionet to factory defaults using the procedure described in Resetting to Factory Defaults in Chapter 7. However, resetting to factory defaults restores all the Barionet's configuration settings back to factory defaults.

2.4.1.3 Setting the Netmask

The Netmask parameter defines the range of addresses in the current subnet. Addresses outside this range must be "routed" through a device called a "router". Like the IP address, you can choose to either manually set the netmask, or you can set it to 0.0.0.0, which tells the Barionet to automatically acquire a netmask setting from the DHCP server, or assume a netmask based on the range of the IP address it is assigned.

If the Barionet gets its IP address from a DHCP server, it will also get a netmask from the server. However, if the Barionet gets an IP address using the IPZator or AutoIP method, it assumes a netmask from the network address. For example, if your Barionet is assigned an IP address of 192.168.0.6, the IP address is called a "class C" address. If you leave the net mask set to 0.0.0.0, the Barionet will automatically assume a net mask of 255.255.255.0.

In general, it's best to assign a specific netmask, based on the configuration of your network. If you are unsure what netmask is appropriate for your subnet, contact your network administrator. See Appendix G: IP Addresses, Netmasks and Gateways for more information. Assuming that the Barionet is connected to the same subnet as your computer, you can probably use the net mask as your computer.

2.4.1.4 Setting the Gateway IP Address

The gateway settings tells the Barionet the IP address of a router or host that is responsible for forwarding any network traffic that is not within the Barionet's subnet. In a small network connected to the internet, the gateway IP address is typically the address of the router that connects the network to the internet (which is probably also the DHCP server).

The gateway address is not necessary unless the Barionet will need to communicate with devices outside its own network. For example, if the Barionet needs to send SNMP traps to a computer that is outside the Barionet's subnet, the gateway address is required in order for the Barionet to connect to the gateway to forward the traffic to its final destination.

The gateway IP address is always in the same subnet as the Barionet. For example, in a small network where the Barionet is assigned an IP address of 192.168.0.6, the gateway address might be 192.168.0.1.

If the Barionet uses DHCP to automatically acquire its IP address, the DHCP server will typically also supply a gateway IP address. In those cases, you can leave the gateway IP address set to 0.0.0.0, and the Barionet will automatically retrieve the gateway address (if it's configured) from the DHCP server.

If you are unsure what to set the gateway address to, contact your network administrator. Assuming that the Barionet is connected to the same subnet as your computer, you can most likely use the gateway address setting of your computer. See Appendix G: IP Addresses, Netmasks and Gateways for more information.

2.4.1.5 Setting the DNS Server Address

The DNS (Domain Name System) server is essentially an Internet phone book. A DNS server translates a domain name (e.g. barix.com) to an IP address (e.g. 209.197.116.112). All communication in an IP network is eventually done using IP addresses, so if a server name is used in a request, the first step is to translate that name into an IP address. That's the job of the DNS server.

The Barionet needs the address of a DNS server only if it uses domain names in any of its requests. For example, if you write BCL code that connects to a mail server by name, you can use BCL's **resolve()** function to translate a domain name into an IP address. Unless you create a custom BCL application that uses domain names to refer to external computers or devices, you don't need to fill in a DNS server.

If you use DHCP to obtain an IP address, the DHCP server can also supply a DNS server address if it is configured in the DHCP server. Leave the DNS server address set to all zeros if there is no DNS server or if it will be automatically configured by DHCP.

2.4.1.6 Setting the DHCP Host Name

The DHCP host name gives the Barionet a name that will be included in the DHCP server's list of assigned addresses. This name simply makes it easier to find the Barionet's IP address in a list of DHCP-assigned IP addresses. If the DHCP host name field is not filled in, the DHCP server's table will include a host name that is derived from the Barionet's hardware MAC address.

2.4.2 The Serial Settings Web Page

The Serial Settings page allows you to set the configuration parameters for the RS-232 and RS-422/RS-485 interfaces. Changes are saved (but not applied) when you click the "OK" button. The Barionet must be rebooted for the changes to take effect.

Figure 10 shows the Serial Settings page. This page is very similar for the Barionet 100 and the Barionet 50 except that the Barionet 50 does not have the "Interface Type" setting. Its two serial interfaces are always RS-232 and RS-485.

The specific settings for each of these interfaces depend on the requirements of the device that is connected to the interface. For example, to use the RS-485/422 interface to connect to Barix Modbus expansion modules, the RS-485/422 interface must be set according to the requirements of the Modbus expansion device. Refer to the serial interfacing requirements of your specific device for appropriate settings.

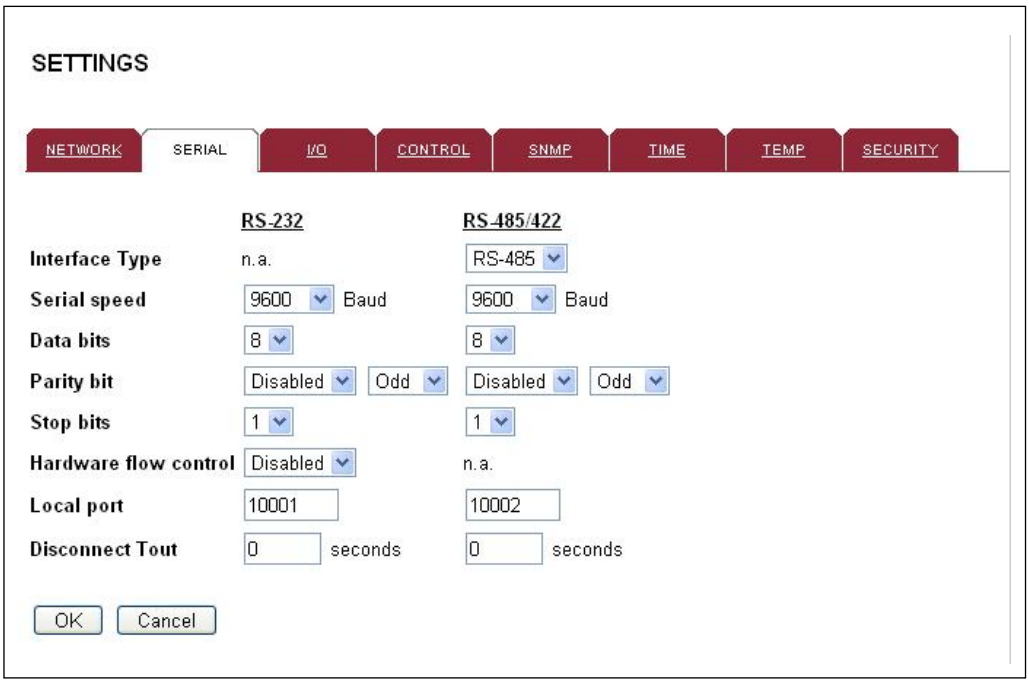


Figure 10. The Barionet 100 Serial Settings Page.

2.4.2.1 Setting the Interface Type

On the Barionet 100, the RS-422/485 interface type can be set for either RS-485 (2-wire interface) or RS-422 (4-wire). The default is RS-485. The RS-232 interface type cannot be changed.

2.4.2.2 Setting the Serial Speed

Barionet 100 Only

Barionet 50 Only

Select the baud rate for the RS-232 and RS-485/422 interfaces using the drop-down boxes for each interface. The default is 9600 baud for both interfaces. Valid serial speed settings are shown in the table below for each Barionet model.

Barionet 100 Baud Rates	Barionet 50 Baud Rates
300	300
600	600
1200	1200
2400	2400
4800	4800
9600	9600
19200	19200
	38400
	57600
	76800
	115200
	230400

2.4.2.3 Setting the Data Bits

The number of serial data bits can be set for 7 bits or 8 bits. The default is 8 bits.

2.4.2.4 Setting the Parity Bit

The parity bit for each interface can be enabled or disabled. If the parity bit is enabled, it can be set for even or odd parity. The default for both interfaces is disabled.

2.4.2.5 Setting the Stop Bits

Select the number of stop bits (1 or 2). The default is 1 stop bit.

2.4.2.6 Setting the Flow Control

This setting applies only to the RS-232 interface. The setting affects the way the Barionet controls data flow on the RS-232 interface. The Barionet 100 only offers hardware flow control, while the Barionet 50 offers software or hardware flow control.

The Barionet 100 offers two settings for hardware flow control: Disabled and Enabled. When Hardware Flow is enabled, the Barionet will only send data as long as the CTS signal is asserted (ON). This is an input to the Barionet.

The RTS signal is an output from the Barionet and it can be controlled from BCL. It is not used for flow control unless explicitly controlled by a BCL program.

Hardware Flow Control is disabled by default. In this state, the CTS signal is not used to control data flow on the RS-232 interface. However, the RTS output can still be controlled by a BCL program, even with Hardware Flow Control disabled.

The Barionet 50 offers three flow control settings: None, Software, and Hardware

None: No flow control is used on the RS-232 interface.

Software (XON/XOFF): The Barionet 50 sends an XOFF character (ASCII 19 decimal) when it's input buffer is approaching full and an XON character

(ASCII 17 decimal) when the input buffer is ready to receive data again. Conversely, the Barionet will stop sending RS-232 data when it receives XOFF, and start again when it receives XON.

Barionet 50 Only

Hardware (RTS/CTS): The Barionet 50 will only send data on the RS-232 port when the CTS signal is asserted (ON). Conversely, the Barionet 50 will assert RTS until its input buffer is nearly full. Then it will clear RTS until the buffer is emptied enough to receive more data.

2.4.2.7 Setting the Local Port

The Barionet has a built-in "serial gateway" function that will allow data sent or received on the RS-232 or RS-485/422 ports to be forwarded to a TCP socket. The Local Port setting specifies the TCP port number that should be used to access this functionality. For example, if the Local Port for the RS-232 interface is set to 10001, and a TCP connection to port 10001 is established, data sent to this TCP port will be transmitted on the RS-232 interface as well. Data received on the RS-232 interface will also be sent from the Barionet to the TCP connection on port 10001.

The built-in serial gateway functionality can be used to control an RS-232 device across a network.

You should set a different port number for each of the interfaces, and set the port number to zero if you do not plan to use the serial gateway function. The default Local Port setting for both interfaces is zero.

Note

If you are using the Barionet's serial gateway function, you should not open TCP connections to the same port in BCL or open the serial ports from within BCL. Otherwise, unpredictable behavior may occur.

2.4.2.8 Setting the Disconnect Timeout

Barionet 100 Only

You can set a disconnect timeout value that will cause the Barionet to automatically close an open TCP connection to the Local Port if there is no activity on the corresponding serial port within the specified interval. The timeout can be set from 0 (which indicates no timeout) to 255 seconds. The default value is zero (no timeout).

3 The I/O Settings Web Page

The I/O settings page controls the configuration of the digital inputs. The page is different on the Barionet and Barionet 50.

The configuration of the Barionet 50's four digital inputs is fixed to active low polarity with a pull-up resistor. This configuration is suitable for a pushbutton contact that connects the input to ground to activate the input. The I/O settings page on the Barionet 50 does not have any user-configurable parameters.

The I/O settings page on the Barionet 100 allows you to set the polarity (active high or active low) of each of the eight digital inputs, and also allows you to select whether the input pull-up resistor is on or off.

Low Act indicates that when the input is low, the input is considered active. High Act indicates that when the input is high, the input is considered active.

Enabling the pull-up resistor on an input ties a 10k ohm resistor from the input to an internal +5 volt supply. The pull-up resistor insures that an open input goes to the high state.

If you want to connect one of the digital inputs to a switch or button, it's usually best to set the polarity to active low and enable the pull up resistor. Then the switch or button connects the input to ground to activate the input.

Barionet 100 Only

If you want to connect one of the digital inputs to a signal that goes to a positive voltage when it is active, change the input polarity to active high and disable the pull up resistor.

If any of the first four inputs are used as analog inputs, the pull up resistor will usually need to be disabled for accurate voltage measurements. However, in certain circumstances, the pull-up resistor can be left enabled to detect an open circuit (i.e. when the input is disconnected). This configuration is often referred to as a "supervised" input.

More details on interfacing to the inputs are provided in Chapter 3.

Figure 11 shows the Barionet 100 I/O Configuration Page.

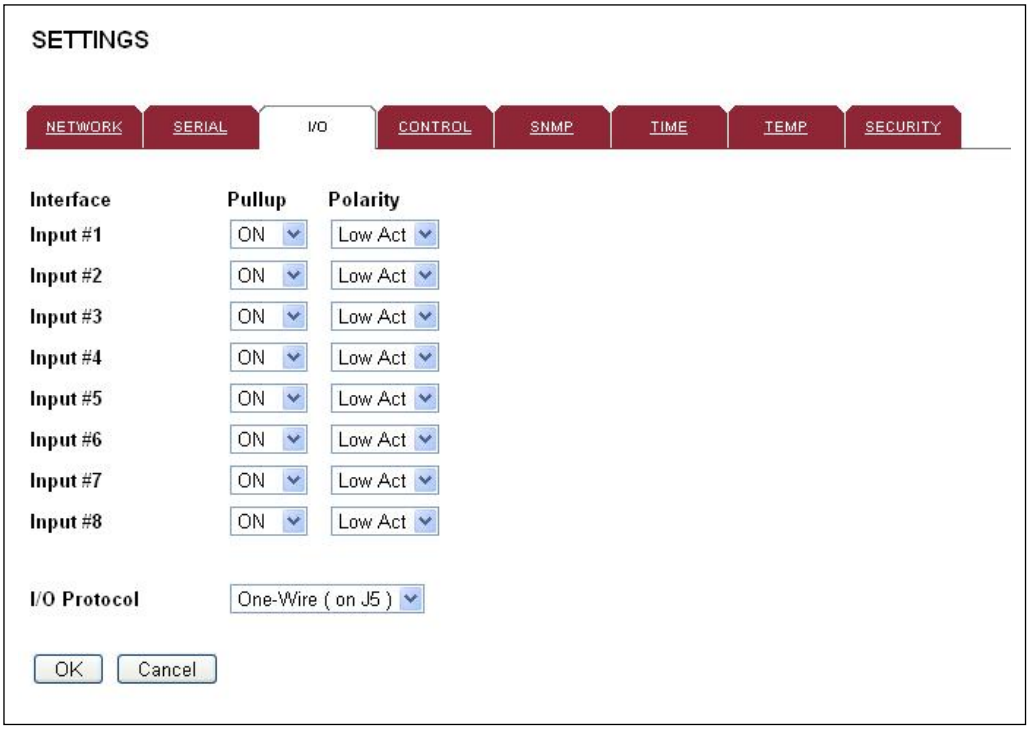


Figure 11. The Barionet 100 I/O Configuration Page.

The I/O protocol setting controls the protocol implemented on the 1-wire interface (J5). The default is to implement 1-Wire®, which allows connection to digital temperature sensors and a real-time clock.

If this setting is changed to Wiegand®, the 1-wire interface on J5 is disabled, and a Wiegand® interface is implemented on the digital inputs on J6. In this mode, the digital temperature sensors cannot be used and the values will always be returned as 256°C.

Note

The Barionet 100 can be connected to a 1-wire Real Time Clock (RTC) device via J5 even when the I/O Protocol is set to Wiegand. The Real time clock device is read only during start up to synchronize the internal Barionet clock with the real time clock value. 1-wire temperature sensors cannot be read in this mode.

Changes are saved when you click OK, but the Barionet must be rebooted for the changes to become effective.

2.4.4 The Control Web Page

The Control page sets a variety of parameters to control the behavior of various software interfaces, including the ASCII/TCP interface, the web interface and UDP interfaces. These various interfaces are described in more detail in Chapter 4.

The Control page, shown in Figure 12 is the same for the Barionet 50 and the Barionet 100.

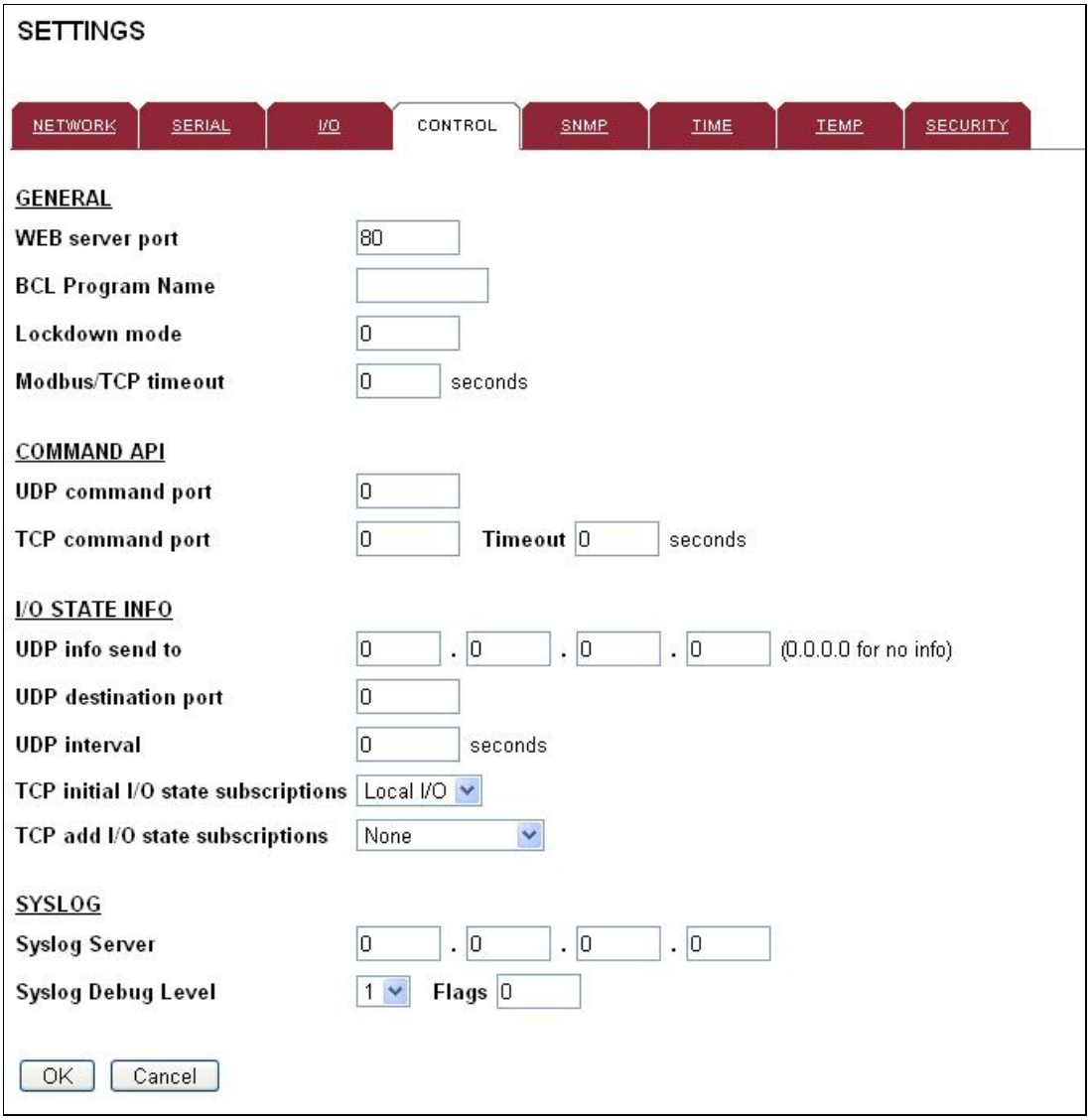


Figure 12. This is the Barionet Control Page. It is identical for the Barionet 50 and Barionet 100.

2.4.4.1 Setting the Web Server Port

This parameter defines the port used to connect to the Barionet's internal web server. The default value of zero sets the web server to the standard HTTP port, which is port 80. A value of zero is functionally equivalent to setting this value to 80.

Note

The Web Server port should not be set to any of the port numbers used by other Barionet services, including Modbus/TCP (port 502), SNMP (port 161), the Serial Gateway (defaults to port 10001), TCP Command or UDP command ports (both are disabled by default by setting to port 0). Setting the web server port to any of these ports can cause the web user interface to become inaccessible. The only way to recover from this situation is to reset the settings to factory defaults. See Resetting to Factory Defaults in Chapter 7 for information on resetting to factory defaults.

2.4.4.2 Setting the BCL Program Name

This parameter specifies the name of the program file that will be executed at start up. Do not include the .tok extension on the file name. If no name is specified, the default name of "barionet" (barionet.tok) is used. For more information on creating a BCL program, refer to

Chapter 5.

Using this setting, you can load more than one program file into the Barionet's memory and then choose which one to run at start up by setting this parameter.

2.4.4.3 Setting the Lockdown mode

This parameter can be used to lock (disable) specific services at start up for security purposes. The services can also be locked or unlocked within a BCL program. The lockdown mode is a single value from 0 to 65,535 where each bit in the 16-bit lockdown value corresponds to a specific function or service as defined in the table below:

Bit Number	Value	Service
0	1	SNMP write
1	2	SNMP read
2	4	Modbus/TCP write
3	8	Modbus/TCP read
4 – 7		Reserved
8	256	rc.cgi
9	512	I/O dynamic tags (&LIO, etc.)
10	1024	setup.cgi
11	2048	Setup dynamic tags
12	4096	BAS.cgi
13	8192	BCL variable dynamic tags (&LBAS)
14	16384	basic.cgi
15	32768	tftp *(Applies only to the Barionet 100)*

To set a lock value, add up the values in the table of the functions you want to disable. For example, to disable Modbus/TCP reads and writes and the setup.cgi function, the lock value would be 1036:

4 (Modbus/TCP write) + 8 (Modbus/TCP read) + 1024 (setup.cgi) = 1036

2.4.4.4 Setting the Modbus/TCP Timeout

The Barionet implements the Modbus/TCP protocol for monitoring and controlling the Barionet via Modbus/TCP commands. The Modbus/TCP timeout parameter sets the time in seconds after which a TCP connection on port 502 (the Modbus port) will be closed due to inactivity. Valid timeout values range from zero to 255 seconds. The default value of zero disables the timeout.

2.4.4.5 Setting the UDP command port

The Barionet implements an ASCII control interface that can be used over a TCP or UDP connection to the Ethernet interface. The UDP command port setting sets the port number that is used for the ASCII command interface over UDP. The port number can be set from 1 to 65535. The same port number is used both for sending ASCII commands and for response messages, including state change messages.

The ASCII command protocol over TCP and UCP is described in more detail in ASCII Command Protocol in Chapter 4.

The default value for the UDP command port is zero, which disables the ASCII interface over UDP.

Note

The UDP command port should not be set to any of the port numbers used by other Barionet UDP-based services, such as SNMP (port 161). The ASCII command protocol will not work if the port is set to the same value as any of these other services.

2.4.4.6 Setting the TCP command port

The TCP command port setting is similar to the UDP command port, in that it sets a port number for using the Barionet's ASCII command protocol over TCP. Opening a TCP connection to the port specified in this setting allows you to send ASCII commands to the Barionet to control I/O functions and to receive response and state change messages from the Barionet.

The ASCII command protocol over TCP and UDP is described in more detail in the section titled ASCII Command Protocol in Chapter 4.

The default value for the TCP command port is zero, which disables the ASCII interface over TCP.

Note

The TCP command port should not be set to any of the port numbers used by other TCP-based Barionet services, including Modbus/TCP (port 502), the web server (defaults to Port 80) or the Serial Gateway (defaults to port 10001). The ASCII command protocol will not work if the port is set to the same value as any of these other services.

2.4.4.7 Setting the TCP command port timeout

This setting defines a timeout in seconds, after which the TCP connection to the command port is closed if there is no activity on the connection. Timeout values can be set from 0 to 255 seconds. The default value of zero disables the timeout so that the connection is never closed by the Barionet for inactivity.

2.4.4.8 Setting the UDP Send Info Address

This parameter specifies the IP address of a host where state change messages should be sent using UDP protocol. You can also specify a broadcast address (e.g. 192.168.0.255) to broadcast the state change messages to all IP addresses on the subnet using UDP.

The default value of 0.0.0.0 disables transmission of state change messages via UDP.

See Unsolicited State Change Messages in Chapter 4 for more information on the state change messages.

2.4.4.9 Setting the UDP Destination Port

This parameter specifies the port number to which state change messages and periodic I/O status messages are sent using UDP.

2.4.4.10 Setting the UDP Interval

The Barionet can be configured to periodically transmit the state of all of its inputs and outputs by setting this parameter to the desired interval in seconds. The default setting is zero, which disables sending periodic status messages. This setting does not affect sending state change messages as described in the next section. Periodic status messages are sent to the IP address and port specified in the UDP Send Info Address and UDP Destination Port parameters.

2.4.4.11 Setting the TCP initial I/O state subscriptions

This setting defines whether or not the Barionet sends state change messages on its digital inputs and outputs. In the default "Local I/O" mode, the Barionet sends state change messages on any digital input, as well as digital and relay outputs as long as a TCP command port is defined and a TCP connection is open to the Barionet on that port. If the setting is changed to "None", no state changes messages are sent, unless they are enabled on individual inputs or outputs using the "add subscriptions" setting described next.

Refer to the description of State Changes Message over TCP in Chapter 4 for more details on how these parameters affect the delivery of state change messages over the ASCII interface.

2.4.4.12 Setting the TCP add I/O state subscriptions

The add subscriptions setting allows you to tell the Barionet to automatically enable state change messages on any input or output that you set or query with the ASCII command interface. For example, if the initial I/O state subscriptions setting is set to NONE, but the add subscriptions setting is set to "With getio/setio", when you query a particular input with the ASCII getio command over a TCP connection, the Barionet will automatically subscribe the TCP connection to receive state change messages for that input. This subscription will remain active until the TCP connection is closed either by the remote host or by the Barionet due to an inactivity timeout.

The add subscriptions function also has one additional benefit in that it allows you to receive state change messages from virtual I/O registers. More information on state change messages and Virtual I/O registers is provided in Chapter 4.

Refer to the description of State Changes Message over TCP in Chapter 4 for more details on how these parameters affect the delivery of state change messages over the ASCII interface.

Note

The first four inputs on the Barionet 100 can be used as analog or digital inputs. If State Change messages are enabled on these inputs (using the "Local I/O" or "with getio/setio" settings and a non-zero command port setting), the Barionet 100 compares the input voltage to a threshold of about 1.2V. When the input transitions through that threshold, state change messages are generated.

Barionet 100 Only

2.4.4.13 Setting the Syslog Server

You can specify the IP address of a computer that runs a syslog program or service (a syslog daemon). There are a variety of programs available for all major operating systems that provide syslog server functionality. For the Microsoft Windows platform, we recommend the Kiwi Syslog server available at www.kiwisyslog.com. The Macintosh and Linux operating systems include syslog servers as part of the standard installation.

Syslog messages are transmitted over the network using the UDP protocol on port 514.

The syslog IP address of 0.0.0.0 causes the Barionet to send syslog messages to the broadcast address of the current subnet (e.g. 192.168.0.255). Messages sent to the broadcast address (where the last number is 255) are available to every computer on the same subnet.

2.4.4.14 Setting the Syslog Debug Level

The Syslog debug level setting controls the number and type of messages that are sent to the syslog server. When you are developing BCL programs, you can insert syslog statements in the program to send messages to the log. The syslog statement also includes a log level parameter. The messages are only sent to the syslog server if the debug level setting is the same or higher than the log level parameter in the syslog statement.

This allows you to include extensive debugging and internal logging information in your programs but disable the syslog output when you no longer need it without going back into the BCL program and actually removing all the syslog statements. You simply set the Debug level to a lower number to disable the messages.

The debug levels are defined as follows:

Debug Level	Debug Messages Sent
0	No debug information is sent to syslog
1	System debug information is sent to syslog
2 – 9	User-defined debug levels for BCL programs using the syslog statement

2.4.4.15 Setting the Debug Flags

The debug flags are intended for use by Barix technical support for more detailed debugging information. We recommend that you do not change this value except under the direction of Barix technical support personnel.

2.4.5 The SNMP Web Page

The SNMP page allows you to setup SNMP traps to notify a remote host when an input state has changed. Figure 13 shows the SNMP page for the Barionet 100. The Barionet 50 SNMP page is identical except that it shows only four inputs instead of eight.

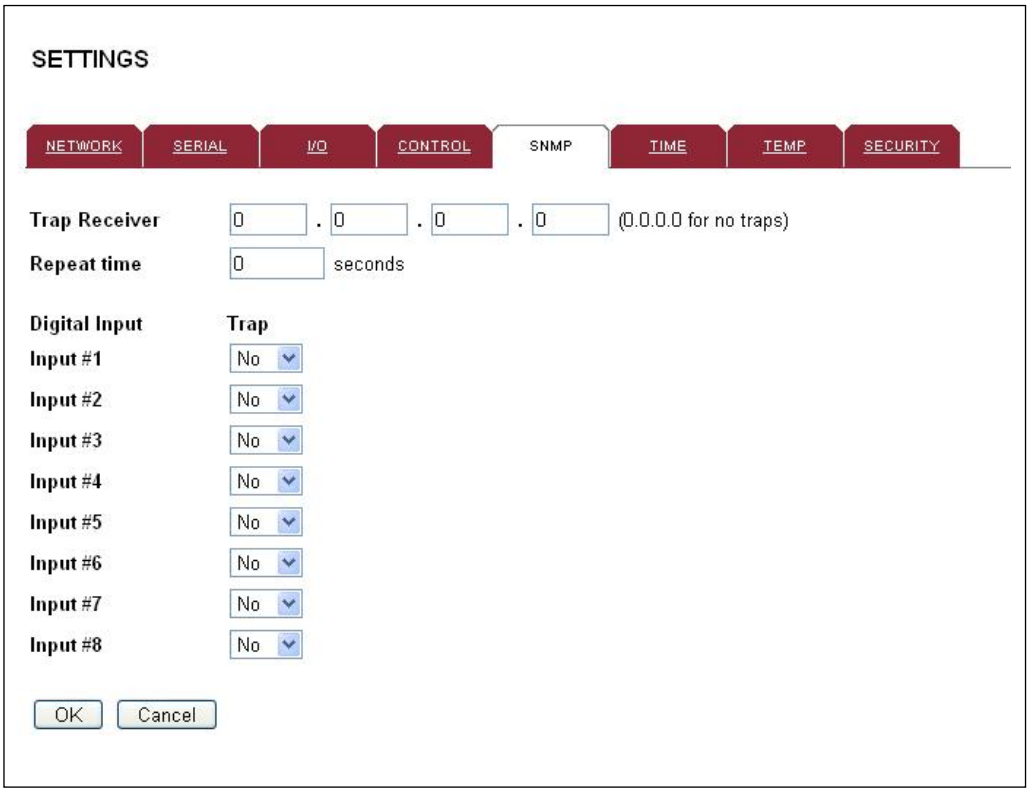


Figure 13. The Barionet 100 SNMP settings page. The Barionet 50 SNMP settings page is identical except that it shows only four inputs, instead of eight.

2.4.5.1 Setting the Trap Receiver IP address

This setting specifies the IP address of the device that will receive traps generated by the Barionet when an input state changes. The default value of 0.0.0.0 disables all traps.

2.4.5.2 Setting the Repeat Time

This setting specifies an interval for sending repeated traps when an input is active. If this value is zero, one trap is sent when the input becomes active and another trap is sent when the input becomes inactive, but no repeated traps are sent during the active time of the

input. The repeat time can be set from zero to 65,535 seconds.

2.4.5.3 Setting the Traps on Inputs

You can enable or disable traps for each input. The default value for each input is "No" (indicating no traps will be sent).

2.4.5.4 Downloading the Barionet MIB file

A MIB file for the Barionet is stored in the Barionet's non-volatile memory. It can be downloaded directly to the browser for viewing or it can be downloaded to a computer on the network in ZIP file format. The ZIP file contains the barionet.MIB file, which can be extracted and loaded into an SMNP manager.

Links for downloading the MIB file are shown in the help panel to the right of the SNMP settings page. Figure 14 shows the SNMP help panel with the links to download the MIB.

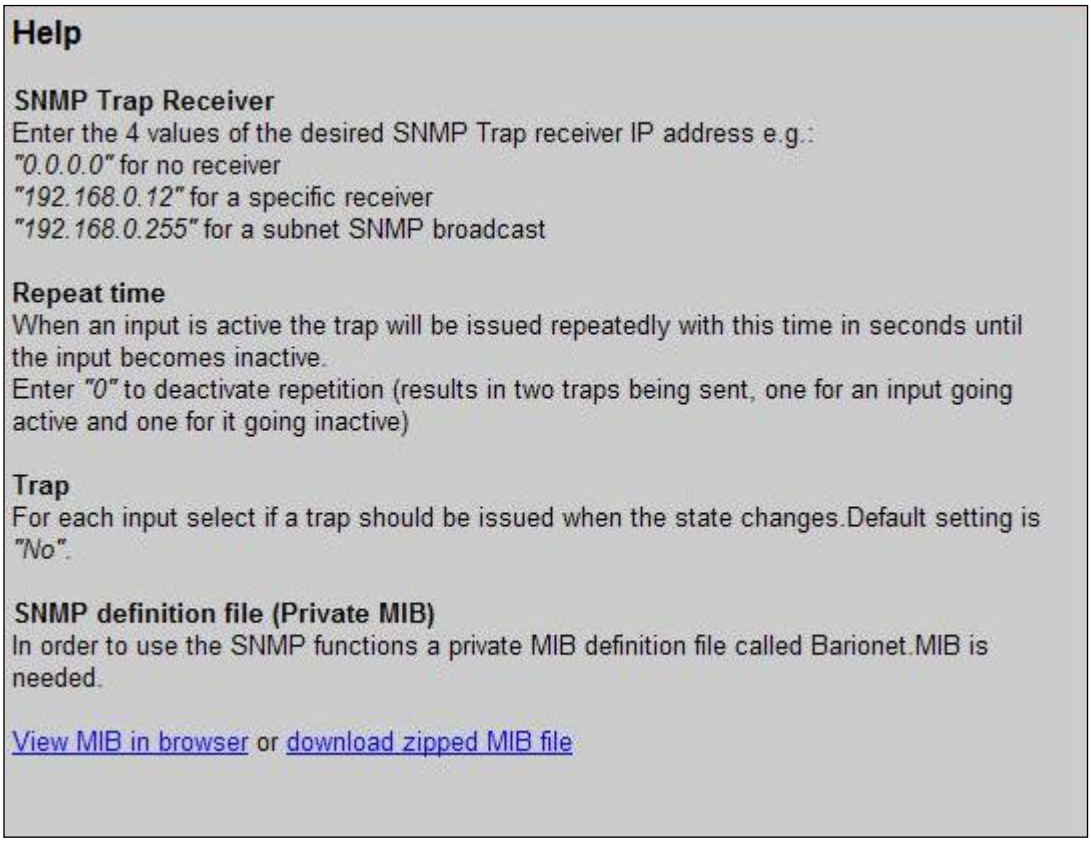


Figure 14. The SNMP Help panel contains links to download the Barionet MIB file.

2.4.6 The Time Web Page

The Time settings page allows you to configure an NTP server to act as a time reference for the Barionet and to set a time zone for the Barionet. Figure 15 shows the Time settings page. This page is identical in the Barionet 100 and the Barionet 50

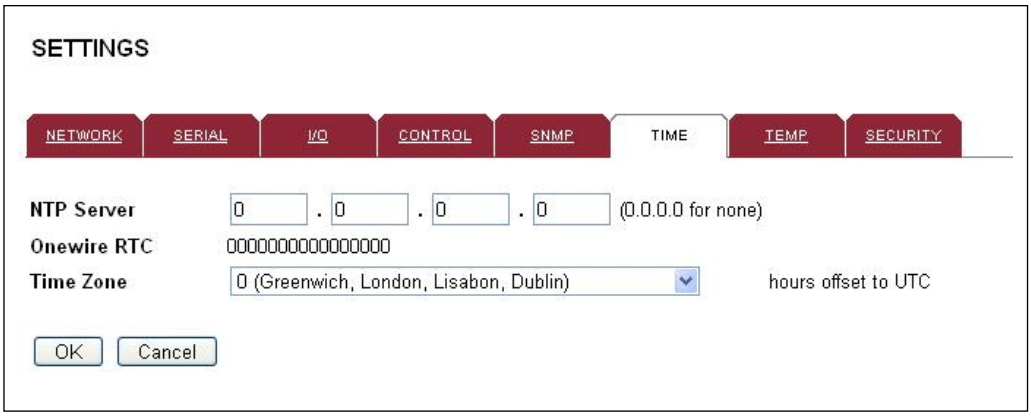


Figure 15. The Time Settings Page. This page is identical in the Barionet 100 and the Barionet 50.

2.4.6.1 Setting the NTP Server

The NTP server field specifies the IP address of an NTP server. If a valid NTP server is found at the specified address, the Barionet will query the NTP server for the current time at start up and then re-synchronize its internal clock to the NTP server roughly every 12 hours.

If a 1-Wire Real Time Clock (RTC) is found on the 1-Wire bus, the unique hardware ID of the RTC is displayed on this page as well.

An RTC is used as a time reference at start up if no NTP server is configured or available.

The NTP server setting is saved when you click OK, but the Barionet must be rebooted for the new setting to become active.

2.4.6.2 Setting the Time Zone

NTP servers report their time in Coordinated Universal time (abbreviated UTC), so it's important to tell the Barionet what time zone it is located in so that it can convert the UTC reported by the NTP server to the local time. Select the Time zone in the Time Zone drop-down box and click "OK" to save the setting. The Barionet must be rebooted for the time zone setting to take effect.

2.4.7 The Temperature Web Page

The Temperature settings page shows the unique hardware IDs of the first eight temperature sensors detected on the 1-wire bus. Up to 50 temperature sensors can be supported by the Barionet, but the hardware IDs of only the first eight are displayed on this page. The IDs of all the sensors are available in registers listed in Table A.5. See Appendix A for more information on these registers.

No settings can be changed on this page. It is for information only.

Figure 16 shows the Temperature settings page, which is identical for the Barionet 100 and the Barionet 50.

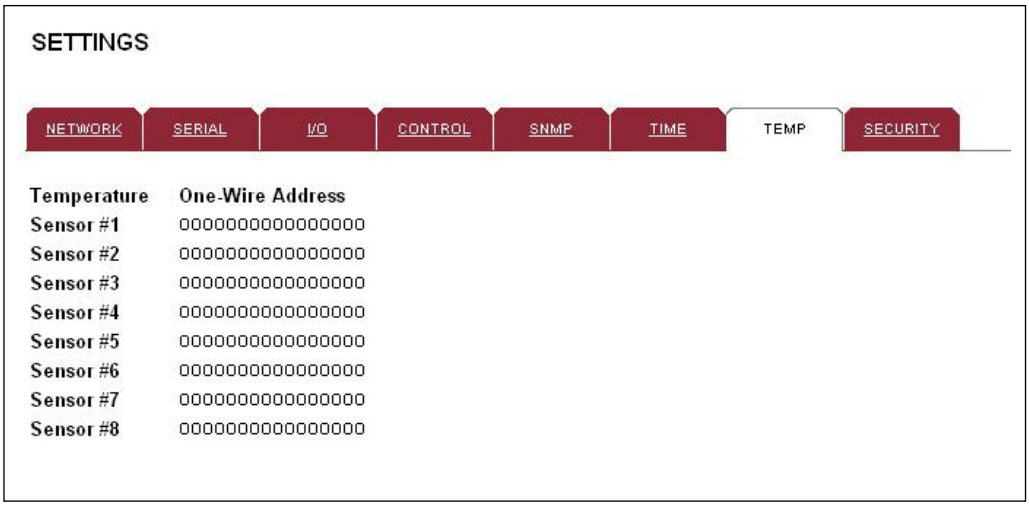


Figure 16. The temperature settings page shows the unique hardware ID of each 1-Wire temperature sensor attached to the Barionet.

For more information on accessing the temperature results returned from the sensors, see Appendix A: I/O Addressing.

2.4.8 The Security Web Page

The Security page allows you to setup a password that can be used to protect the configuration pages as well as any custom HTML pages you develop that you want to protect. By default, no password is required for access to the configuration pages.

Figure 17 shows the Security page, which is identical for the Barionet 100 and Barionet 50. The first time you access the page, the "Old Password" box will not appear, since no password has been set.

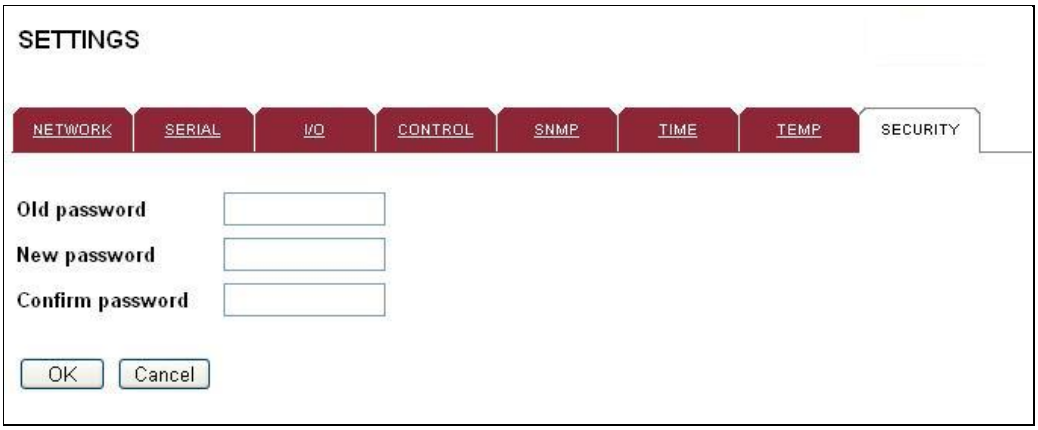


Figure 17. The Security Settings Page allows you to set a password for configuration pages.

2.4.8.1 Setting a Password

To set a password, enter a password with up to 12 characters in the New Password field and re-enter the same password text in the Confirm password box. The two fields must match. Otherwise, an error is generated when you press OK. Click OK to save the password.

If a password was previously set, the Old Password box will also appear. You must enter the current password to change the password. If the old password you enter doesn't match the existing password, an error message is displayed when you click OK.

2.4.8.2 Logging in to the Configuration Pages

After you have set a password, the next time a user attempts to access the configuration page, the web browser will display an authentication dialog box like the one shown in Figure 18. To login, leave the user name field blank and enter the password you set in the

password setting page in the password field of the authentication box.

Figure 18. The authentication dialog appears when a user tries to access configuration pages when a password has been set.

2.4.8.3 Changing the Password

You can change the password that was previously set by going to Security page and entering the current password in the Old Password box. Then enter the new password in the New password and Confirm password boxes and click OK.

2.4.8.4 Clearing the Password

To clear a previously set password and allow access to the configuration pages without entering a password, enter the current password in the Old password box. Then make sure that the New password and Confirm password boxes are empty. When you click OK, the Barionet will prompt you to insure that you want to clear the password. Click OK to clear the password.

2.4.8.5 Password Protecting Custom Web Pages

You can also apply the same password protection to custom web pages you create. We'll discuss the process of creating custom web pages and applying password protection to them in Chapter 5. See the description of the Initialization Tag for information on using the password protection feature on custom web pages.

3 Hardware Interfaces

The Barionet provides flexible I/O capabilities for sensing and controlling a variety of devices. A serial interface is also provided for interfacing to Barix expansion modules as well as other devices with a serial interface. In addition, a built-in 1-Wire® interface connects to temperature sensors and real time clock modules that implement the 1-Wire interface.

This chapter describes the various hardware interfaces the Barionet 100 and Barionet 50 provide along with details on how to properly connect external devices. Section 3.1 describes hardware interfaces specifically for the Barionet 100, while Section 3.2 describes the hardware interfaces for the Barionet 50.

Information on physical mounting options and requirements for the Barionet is provided in Appendix D.

Barionet 100 Only

3.1 Barionet 100 Hardware Interfaces

This section describes the hardware interfaces for the Barionet 100. Please see Section 3.2 for information on the hardware interfaces for the Barionet 50.

Barionet 100 Only

3.1.1 Barionet Connectors Overview

Figure 19 shows the Barionet 100 connectors and indicator lights. Please refer to this diagram in the following sections for the physical location of each of the hardware interfaces and their corresponding connectors.

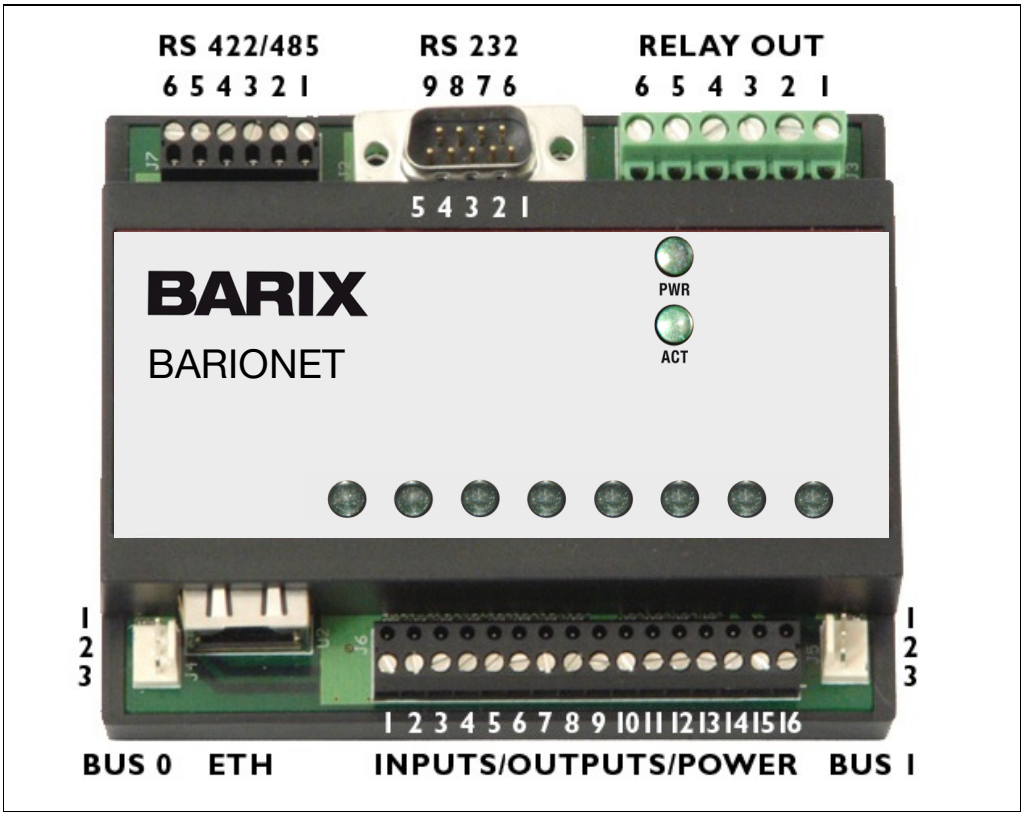


Figure 19. The Barionet 100 connectors and indicator lights.

Barionet 100 Only

3.1.2 Ethernet Interface: (ETH)

The Barionet 100 provides a standard RJ-45 10/100 megabit (Mb), full/half duplex, auto negotiation Ethernet interface. Two indicators on the front face of the RJ-45 connector show the network status. The network status indicators on earlier versions of the Barionet 100 produced in 2005 or earlier are defined in the following table:

Barionet 100 Only

Left LED	Right LED	Function
Orange		Link OK, 10 Mb LAN speed. Blinking indicates Ethernet traffic.
	Green	Link OK, 100 Mb LAN speed. Blinking indicates Ethernet traffic.

The network status indicators on Barionet 100 units produced in 2006 or later are defined in this table:

Left LED	Right LED	Function
Orange		Link OK, 10 Mb LAN speed.
Green		Link OK, 100 Mb LAN speed.
	Orange	Blinking indicates Ethernet traffic with speed selected manually by either end of the connection
	Green	Blinking indicates Ethernet traffic with speed auto-negotiated

Note

If you are unsure which version of Barionet 100 you have, you can identify the version by removing the top cover and checking the product number label on the top of the Ethernet connector. Older Barionet 100's have product number XP1001000-01, while later Barionet 100's have product number XP1001001-03R.

3.1.3 Power Supply Inputs (J6)

Barionet 100 Only

The Barionet 100 can be operated on a power supply from 9 volts through 30 volts DC. Voltages above these limits may cause damage. The positive power supply lead should be connected to pin 15 and the negative lead should be connected to pin 16.

The Barionet 100 uses a maximum of 4 watts from the power supply.

3.1.4 RS-232 Serial Interface (J2)

Barionet 100 Only

The Barionet 100 provides an RS-232 serial interface for connection to a variety of devices that implement this serial interface. The RS-232 interface is supplied via a 9-pin sub D male connector, wired as a DTE. The connector's DTE wiring is similar to a personal computer's serial port, as defined in the following table.

Pin	Direction	Function
1		Not connected
2	To Barionet	RxD Receive Data
3	From Barionet	TxD Transmit Data
4	From Barionet	No connection or Unregulated power supply (See Caution below)
5	From Barionet	Ground
6		Not connected
7	From Barionet	RTS (Ready To Send)
8	To Barionet	CTS (Clear To Send)
9		Not connected

Pin 4 of the Barionet 100 RS-232 connector can be tied to the Barionet's un-regulated power supply input to supply power to external devices through the RS-232 connector. A

pair of solder pads labeled L2 on the Barionet's circuit board must be bridged together in order to connect the Barionet's power supply to pin 4. The L2 pads are located next to "C31", left of pin 16 on the serial driver chip labeled "U8".

Caution

If the solder pads are bridged to connect pin 4 to the unregulated power supply and the Barionet is operated on a supply voltage above 15 volts, the Barionet or external devices may be damaged since pin 4 may be connected to active signals on the external devices. The RS-232 specification allows for a maximum of 15 volts on all RS-232 inputs and outputs.

Barionet 100 Only

The baud rate for the RS-232 port is selectable from 300 baud (bits per second) to 19,200 baud with 7 or 8 data bits, 1 or 2 stop bits, and even, odd, or no parity.

Hardware flow control can also be enabled or disabled. When hardware flow control is enabled, the Barionet will only transmit data on the serial port when the CTS signal (Clear to Send) is asserted (high). The RTS signal can be controlled from BCL to control data being transmitted to the Barionet, but it is not automatically asserted when the Barionet's serial input buffer fills.

Note

The RS-232 baud rate, data bits, stop bits, parity and flow control parameters must be set through the web configuration interface (see "The Serial Settings Page" in Chapter 2 for more details). These parameters are also specified in the BCL OPEN statement, but they are ignored in BCL programs, so the parameters must be set via the web configuration page.

A blinking RS-232 light on the Barionet top cover indicates activity on the RS-232 port.

5 RS-422/485 Serial Interface (J7)

Barionet 100 Only

The RS-422/485 serial interface can be used to interface to Barix expansion modules, other Modbus devices, or for interfacing to other RS-422 or RS-485 devices.

The connector is a removable screw block that carries the RS-485 (2 wire) and RS-422 (4-wire) signals, as well as connections for a reference ground. All pins are ESD (Electro-static discharge) protected to reduce the possibility of damage due to static discharge.

When the interface is set for RS-485 operation, the TX signals (pins 4 and 5) are used for both transmit and receive function. The table below defines the pins and signals on this connector. The configuration of the interface (i.e. RS-422 or RS-485) is set in the web configuration pages (See the description of the Serial Settings Page in Chapter 2 for more details)

Pin	RS-422	RS-485
1	Shield	Shield
2	RXA	Not used
3	RXB	Not used
4	TXA	A
5	TXB	B
6	Shield	Shield

Barionet 100 Only

Note

The shield pins on the RS-485 connector are tied to the Barionet's ground through a 100-ohm resistor. Barix recommends that external devices connected to this port be powered by a DC power supply to reduce interface noise induced by AC power supplies.

The RS-485 indicator on the case flashes when there is activity on the RS-422/485 port.

Barionet 100 Only

Note

If the RS-485 indicator remains constantly lit, it's likely that the A and B signals from the RS-485 port have been swapped.

Barionet 100 Only

6 Rescue Jumper (J4)

This connector is used to initiate the "serial rescue" function that can be used to re-load the Barionet firmware in the event of a failure that prevents updating the firmware via the Ethernet interface. Refer to The Serial Rescue Procedure in Chapter 7 for more details on the serial rescue procedure.

The pins of this connector are defined by the following table:

Pin	Name	Function
1	Vcc (9 – 30 VDC or 12-24 VAC)	Unregulated power supply
2	Vss (Ground)	Supply/rescue jumper
3	Bus 0 (5 VDC)	Rescue jumper

Barionet 100 Only

7 1-Wire® Expansion Port (J5)

This three pin connector provides a Dallas Semiconductor 1-Wire® interface. The Barionet firmware contains built-in support for up to 50 1-Wire temperature sensors. A 1-wire real time clock can also be connected to this port. Other 1-wire devices are not supported.

Note.

The Barionet 100 can also be configured to implement the Wiegand reader protocol on the digital input pins (J6). While the Wiegand interface does not use the J5 Expansion port, enabling the Wiegand interface in the web configuration pages disables the 1-Wire interface on J5. See the description of The I/O Settings Web Page in Chapter 2 for more details.

The pin assignments for the J5 expansion port are defined in the following table.

Barionet 100 Only

Pin	Name	Function
1	Power supply	Unregulated power supply
2	Vss (Ground)	1-Wire ground
3	Bus1	1-Wire data

3.1.7.1 Connecting 1-Wire Temperature Sensors

The Barionet firmware is compatible with the Dallas Semiconductor/Maxim DS18B20 temperature sensors. There are two methods of connecting the sensors to the Barionet. In both cases, the Data (DQ) pin of the sensor is tied to pin 3 of J5. The only difference between the two methods is the ground connection.

Figure 20 shows the first connection method. In this method, pin 14 (Ground for Outputs)

Barionet 100 Only

and pin 16 (Vss Ground) on the main I/O connector, J6, are tied together, and the ground pin of the temperature sensor is tied to pin 2 of the expansion port, J5.

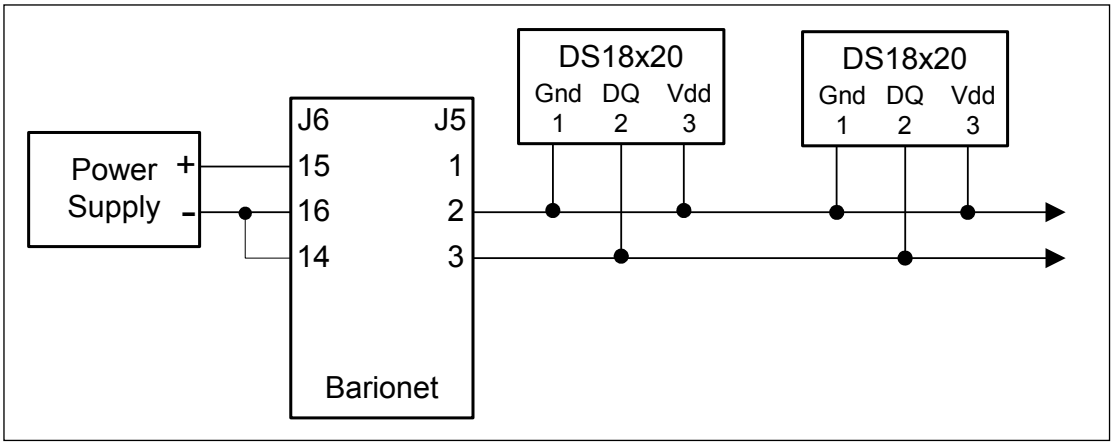


Figure 20. The first connection method for 1-Wire sensors.

This method has the disadvantage that it defeats some internal noise filtering circuitry in the Barionet by bridging the output ground (pin 14) and the power supply ground (pin 16). The second method, shown in Figure 21 is preferred for noise immunity.

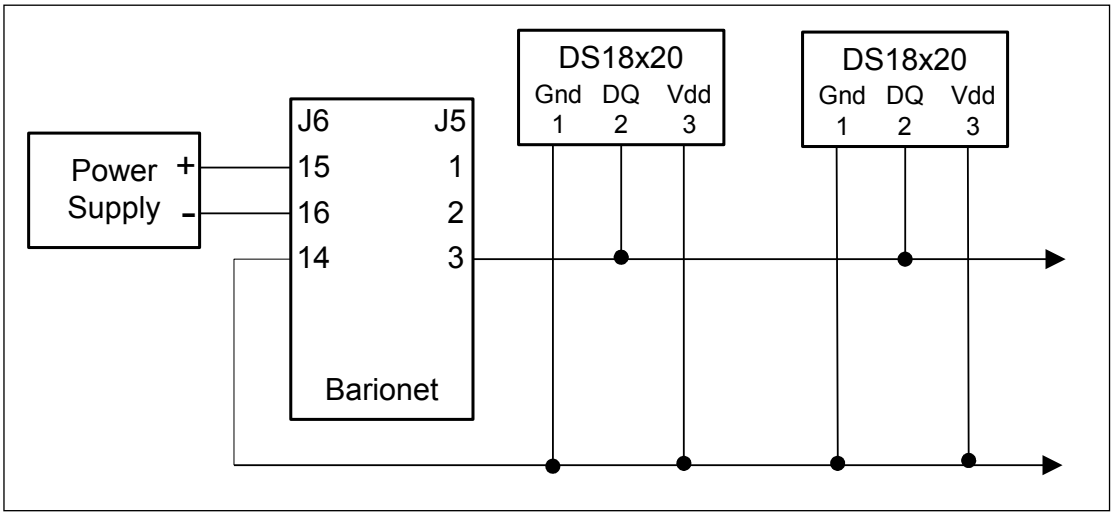


Figure 21. The second connection method for 1-Wire sensors is preferred for improved noise immunity and reliability. Pins 1 and 3 of the sensor are internally tied together in the Barix TS temperature sensors.

Each temperature sensor has a permanently stored unique ID number, which is displayed in the Temperature page of the built-in configuration web pages. See the description of The Temperature Web Page in Chapter 2 for more details. The Barionet scans the 1-Wire bus for temperature sensors at start-up and displays the IDs of all the sensors on the configuration page.

There are two simple methods for identifying individual sensors:

1. Connect individual sensors to the Barionet one at a time. Then restart the Barionet and record the sensor's ID. A small label on the sensor wire can be attached to permanently label it.
2. Connect all the sensors to the Barionet and use cooling spray to cool each sensor one at a time to identify the sensor in the user interface.

The sensors are always displayed in the same order on the user interface until a new sensor is introduced on the bus.

Barix also offers these temperature sensors conveniently packaged with 12" (33 cm) leads and a small mounting hole. The Barix packaged sensors internally tie the Ground (pin 1) and Vdd (pin 3) of the sensors together, so there are only two leads to connect—one for Data and one for ground.

For more information on the Barix "TS" Temperature sensor, refer to Appendix C: Accessories.

Barionet 100 Only

3.1.8 Relay Outputs (J3)

The Barionet 100 provides two single-pole double-throw relay outputs capable of switching up to 5 amps at 250VAC maximum. The relay outputs are provided on the six-position screw terminal block at the top right of the Barionet. Figure 22 shows the relay outputs and their pin assignments.

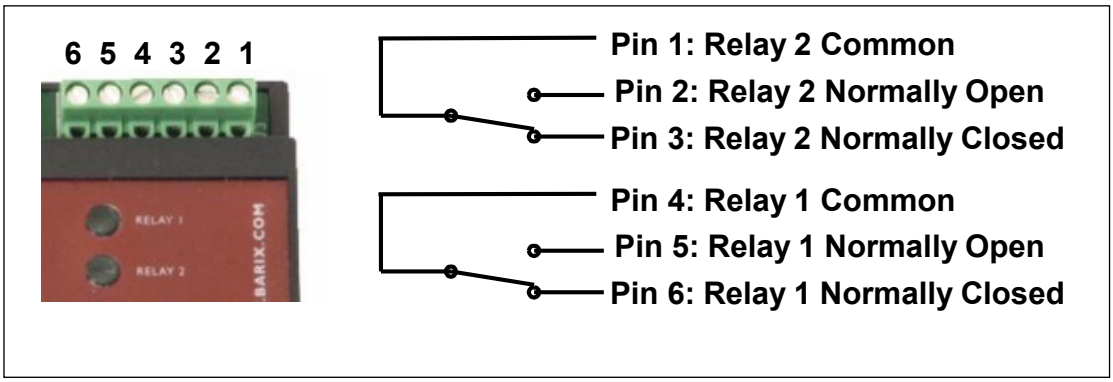


Figure 22. The Barionet Relay Outputs.

The Relay LED indicators next to the relay output terminals are lit when the corresponding relay is active (i.e. the "normally open" contacts are closed).

Barionet 100 Only

3.1.9 Digital/Analog I/O and Power (J6)

The large removable screw-terminal connector near the bottom of the Barionet provides connections to the four digital/analog inputs, the four digital inputs and the four digital outputs. It also includes terminals for an input ground, output ground, and the power supply inputs described earlier in this section. See the description of the Power supply inputs for more details on power supply requirements.

The following table lists the pins on J6 and their functions. See Figure 19 for the physical arrangement of the pins.

Pin	Function	Pin	Function
1	Input 1 (analog/digital)	9	Ground for inputs
2	Input 2 (analog/digital)	10	Output 1 (digital)
3	Input 3 (analog/digital)	11	Output 2 (digital)
4	Input 4 (analog/digital)	12	Output 3 (digital)
5	Input 5 (digital)	13	Output 4 (digital)
6	Input 6 (digital)	14	Ground for outputs
7	Input 7 (digital)	15	+ Power Supply (+9-30VDC)
8	Input 8 (digital)	16	- Ground

Barionet 100 Only

3.1.9.1 Using the Analog Inputs (Pins 1 – 4)

The first four Barionet 100 inputs can be used as analog or digital inputs. No configuration change is required to change the input type. However, if the inputs are used as analog inputs, it's usually best to turn off the pull-up resistors that are provided for use on the digital inputs, and to disable the state change messages for these inputs. Both of these configuration changes are made through the configuration web pages described in the previous chapter. To disable the pull-up resistors, refer to the description of the I/O Settings Page. To disable the state change messages, refer to the description of the Control Page in the previous chapter.

The analog inputs can measure voltages from zero to 5 volts DC. Higher voltages can be

Barionet 100 Only

measured using an external resistor divider network to reduce the voltage at the input to 0 – 5V. The inputs can safely handle up to 24V, but the analog voltage measurement function reaches its full-scale value at 5 volts. Any voltage above 5 volts will measure as full-scale. In addition, above 5 volts, the inputs will sink up to about 2 milliamps of current.

The resolution of the analog-to-digital converter on these four inputs is 10 bits (values of 0 to 1024). However, accuracy is typically limited to about 9 bits and requires a low-impedance source and careful grounding.

3.1.9.2 Creating an Analog Voltage Divider

If you want to measure voltages higher than 5 volts with the Barionet analog inputs, you'll need to create a simple "voltage divider" using resistors to scale the voltage you want to measure down to 0-5 volts. Figure 23 shows how to create a voltage divider using two resistors.

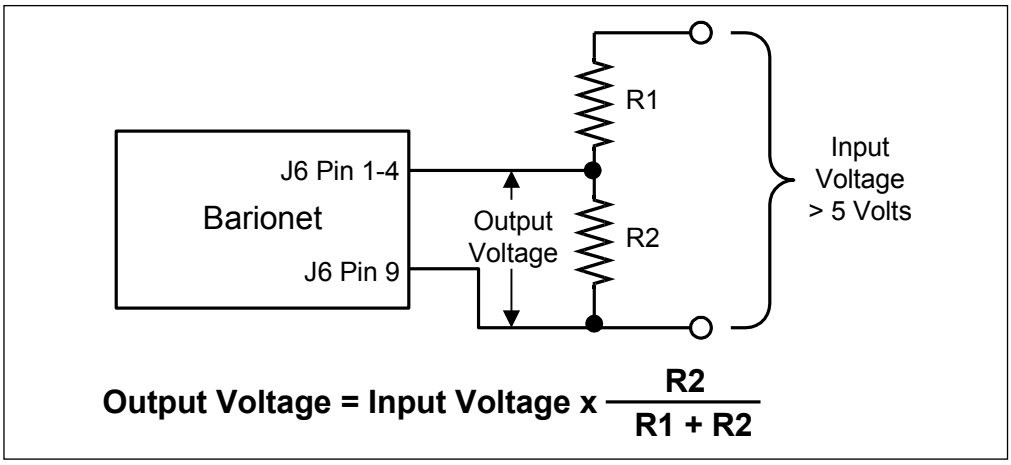


Figure 23. A Voltage divider allows measuring input voltages greater than 5 volts.

Barionet 100 Only

For example, if you're trying to measure a voltage from zero to 24V, we need a divider ratio of approximately 4.8 (24 / 5 = 4.8). If we use a 1000Ω resistor (1kΩ) for R2 and a 3900Ω (3.9kΩ) resistor for R2, we get a ratio of 4.9 (1k / (1k + 3.9k)). With this combination of resistor values in a voltage divider, a 24V input yields an output voltage (input to the Barionet) of 4.898V.

When you display the measured voltage, you can scale the value back up to its original input voltage so that the display shows a voltage from 0 to 24V. The built-in web server on the Barionet includes proprietary dynamic tags that include the ability to scale a value before it is displayed. For more information on these Barix dynamic tags, refer to the discussion of Barix Dynamic Tags in Chapter 5.

For best results, keep several factors in mind when designing the voltage divider circuit:

- The resistor voltage divider technique works well for low-impedance sources, such as battery or power supply voltages. For higher impedance sources, where the current used by the voltage divider may affect the voltage being monitored, an active buffer amplifier or voltage scaling circuit may be required. Designing such active divider circuits is beyond the scope of this manual.
- In most cases, it's best to turn off the input's pull-up resistor in the The I/O Settings Web Page, since the current supplied by the pull-up resistor will affect the measured voltage. One exception would be when a "supervised" input is needed. In this case, the pull-up resistor can be left enabled, and the voltage divider values must account for the 10k pull-up. The voltage divider should be setup so that normal inputs never reach the full 5V input level. If the input is completely disconnected (including the voltage divider), the voltage level will rise to 5V because of the pull-up resistor. Custom BCL software can be written that takes appropriate action if the input is disconnected, indicated by the input voltage reaching 5 volts.
- If you decide to use a voltage divider on the input with the pull-up resistor turned on, the calculation for input voltage vs. output voltage is more complex, since you have to

Barionet 100 Only

account for the pull-up resistor and the 5V power supply it is tied to. Figure 24 shows the equivalent schematic and the formula for calculating the effects of the voltage divider with the 10kΩ pull-up turned on.

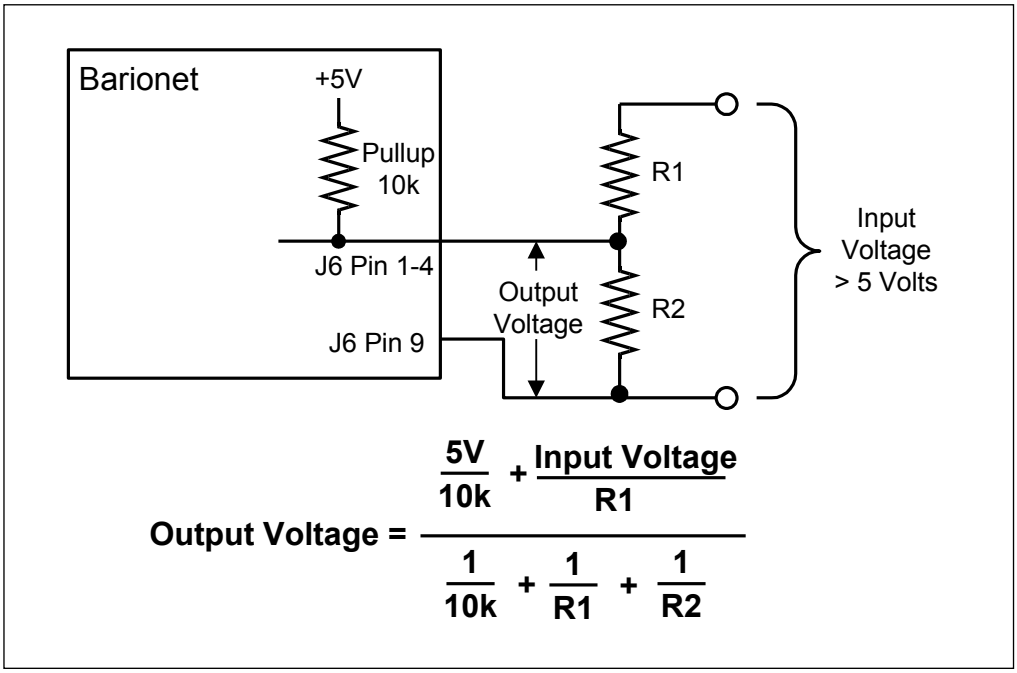


Figure 24. The effects of turning on the 10kΩ pull-up resistor on the voltage divider.

- The accuracy of the measured voltage is heavily dependent on the accuracy of the resistor values chosen. If accurate measurements are required, be sure to use high precision (1%), or use "trim" variable resistors that allow you to precisely set the divider ratio. You can also measure the precise resistor values with a high-quality ohmmeter and adjust the scale factors to compensate for any inaccuracy in the resistor values.

Barionet 100 Only

3.1.9.3 Using the Digital Inputs (Pins 1 – 8)

The first four Barionet 100 inputs (inputs 1 – 4) can be used as either analog or digital inputs. The last four inputs (inputs 5 – 8) can only be used as digital inputs. All inputs can be configured for either active high operation or active low operation and with or without pull-up resistors.

In active high mode, the external device applies a voltage to the input. The switching threshold (from low to high) is different for the first four analog/digital inputs and for the last four digital-only inputs. Refer to the Specifications listed in Appendix F for more information on the input switching thresholds.

The inputs can accept voltage levels as high as 24V, though above 5V levels, the input may sink as much as 2 milliamps of current. Do not exceed 24V on the digital inputs. Inputs should also not be driven below zero volts. In general, it's best to disable the pull-up resistors in this mode, as the external device supplies the voltage to drive the inputs high or low.

In active low mode, the input is considered active when it is connected to ground (or at a voltage below the switching threshold—see the input specifications). This mode is normally used with the pull-up resistor enabled when the input is connected to a switch or other contact that pulls the input low when active. When the input is inactive, the pull-up resistor pulls the input up to near 5 volts. The device driving the input must be capable of sinking at least 200 micro amps (the current supplied by the 10k pull-up resistor tied to 5V).

Figure 25 shows a Barionet 100 digital input with the internal pull-up resistor enabled connected to a push-button, switch or other contact. Notice that there is a separate ground for the inputs (pin 9). This ground does not need to be tied to the power supply ground (pin 16) unless it is required by the external devices that drive the inputs.

Barionet 100 Only

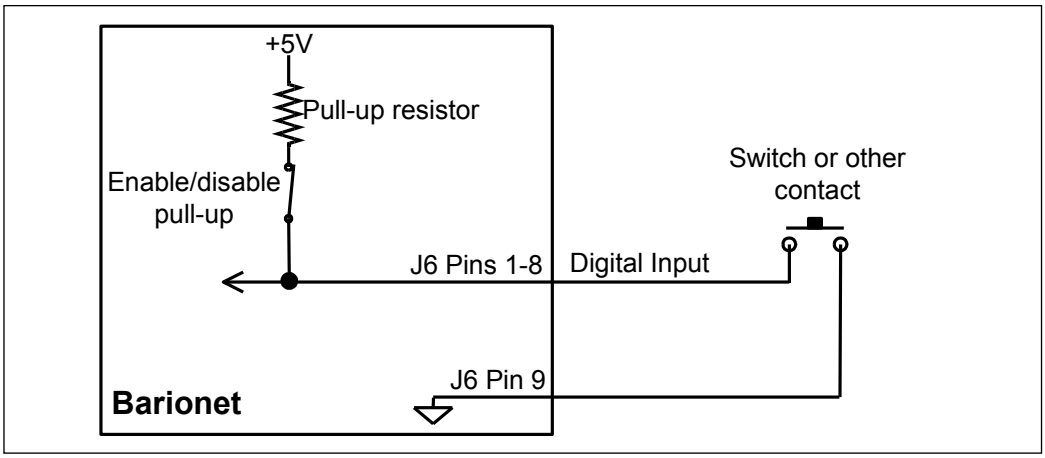


Figure 25. With the pull-up enabled, a digital input can be directly to a switch or other contact.

3.1.9.4 Using the Digital Outputs (Pins 10 – 13)

Barionet 100 Only

The Barionet 100 also provides four open-collector digital outputs. These outputs can switch up to 24 volts and can sink up to 100 milliamps of current. They are suitable for driving LEDs, smaller signaling lamps, and relay coils, up to 24V and 100mA. The outputs have current-limiting circuitry that prevents them from sinking more than 100mA. Figure 26 shows how the digital outputs are used.

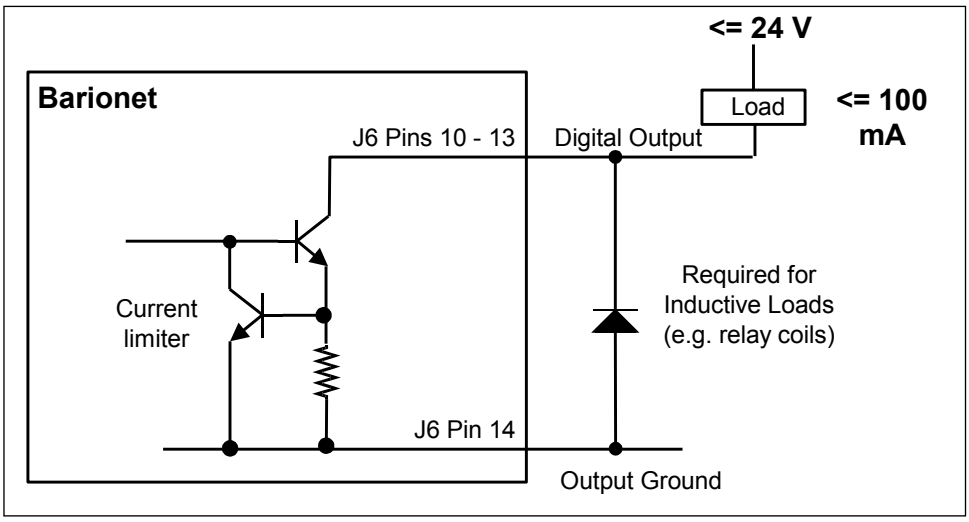


Figure 26. The open-collector digital outputs can sink up to 100 milliamps.

Caution

If the digital outputs (pins 10 – 13) are used to drive inductive loads, such as relay coils, the "snubber" diode shown in Figure 26 is important and should be installed close to the load. Inductive loads will generate short but potentially large negative spikes when the output shuts off. This large pulse can damage the Barionet's digital outputs. The snubber diode shunts the pulses to ground. The diode should be rated with peak reverse voltage of at least 50 volts and be capable of at least 1 amp forward current.

There is a separate ground for the outputs which can be used if the output loads are driven by a separate power supply and ground. However, if the external loads use the same ground as the Barionet 100 power supply, the output ground (pin 14) and the power supply ground (pin 16) can be tied together. Keeping the output ground and power supply ground separate improves noise immunity, especially when using 1-Wire® temperature sensors.

Figure 27 shows two methods of connecting outputs and grounds. The upper diagram shows digital outputs using a separate power supply and ground. This is the preferred

Barionet 100 Only

method for noise immunity. The lower diagram shows digital outputs driven by a common power supply and ground.

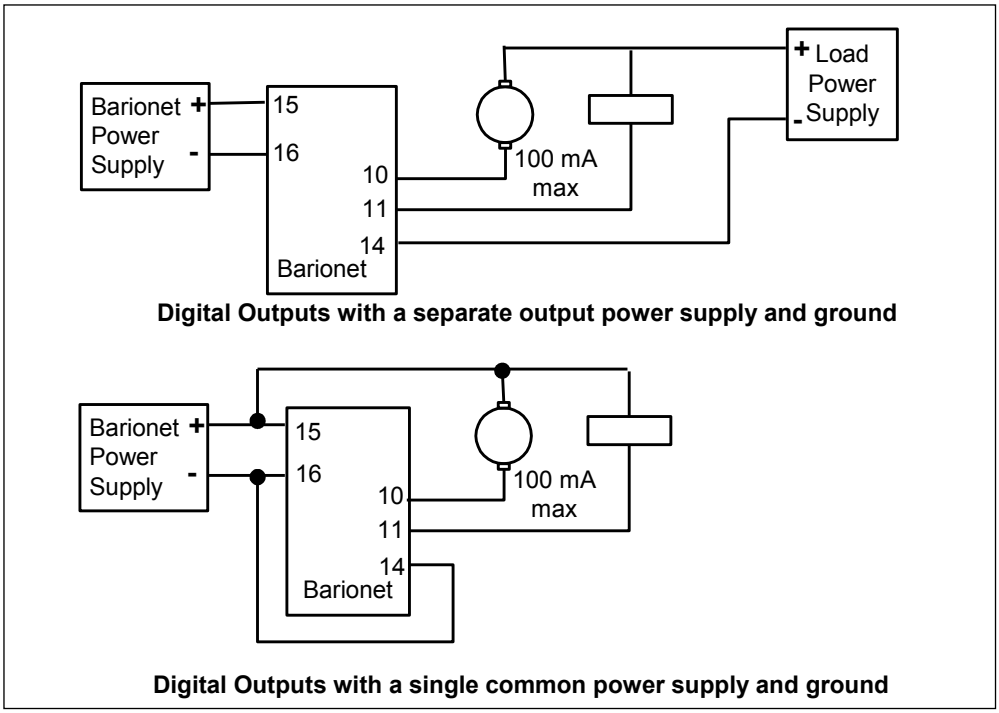


Figure 27. Digital outputs can use a separate power supply and ground or a common power supply and ground.

3.2 Barionet 50 Hardware Interfaces

This section describes the hardware interfaces for the Barionet 50. Please see Section 3.1 for information on the hardware interfaces for the Barionet 100.

Barionet 50 Only

1 Barionet 50 Connectors Overview

Figure 28 shows the Barionet 50 connectors and LEDs. Please refer to this diagram in the following sections for the physical location of each of the hardware interfaces and their corresponding connectors.

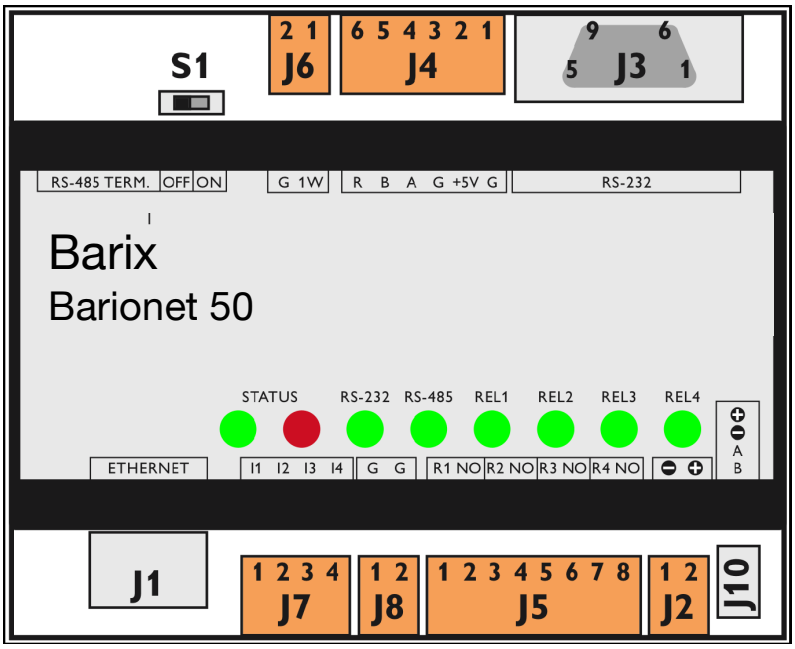


Figure 28. The Barionet 50 connectors and indicators.

Barionet 50 Only

2 Red and Green Status Indicators

The red and green status indicators display the status of the Barionet 50. During the start-up operation, the indicators show the status of the boot up operation and, if the Barionet is configured to acquire its IP address automatically, the lights flash in a specific pattern while the Barionet is acquiring an IP address. If errors are detected during these processes, the lights flash in a pattern that indicates the type of error. The following table summarizes the start-up phases and the light patterns

Mode	Red Status LED	Green Status LED	Comments
Boot up	On	Off	During the brief boot-up sequence, the red LED is on. However, this period is typically 1 second or less, so it appears to be a brief flash, The green LED is also on for an even shorter period, but the time is so short that the green LED appears to be off.
Automatic IP Address Acquisition (DHCP, IPZator, AutoIP)	Alternate blinking (continuous)	Alternate blinking (blink on for 5 cycles then off for 3 cycles	During the IP address acquisition process, the red and green LEDs alternate flash for 5 cycles. Then, the green LED is off for three cycles, so the red flashes alone 4 times. This pattern continues until the Barionet 50 acquires an IP address or until an error is detected. If the Barionet 50 is set for a static IP address, this step is skipped completely
Errors Detected	On	Flashing three or five times	If an error is detected at start-up, the Barionet flashes the green LED as follows: 3 green flashes indicate that the network hardware could not be initialized, or the MAC address is corrupt. 5 green flashes indicate that the loaded BCL application is corrupt or there is an IP address conflict.
Application Running	Off	On	This is the normal condition after boot up and address acquisition is complete.
Boot loader	Blinking	On	When the Barionet 50 is in boot loader mode (used, for example, to upload new firmware), the green LED is on steady and the red LED flashes.

Barionet 50 Only

3 Ethernet Interface: (J1)

The Barionet 50 provides a standard RJ-45 10/100 megabit (Mb), full/half duplex, auto negotiation Ethernet interface. Two indicators on the front face of the RJ-45 connector show the network status as defined in the following table:

Left LED	Right LED	Function
Orange		Blinking indicates Ethernet traffic.
	Green	Link OK

Barionet 50 Only

4 Power (J2)

Barionet 50 Only

The Barionet 50 can be operated from a power supply from 9 to 30 volts DC. Voltages above 30V on the supply inputs may cause damage. The positive lead of the power supply

should be connected to pin 2 of J2 and the negative lead should be connected to pin 1, as indicated on the label on the top of the Barionet 50 case.

The Barionet 50 uses a maximum of 4 watts from the power supply.

5 RS-232 Serial Interface (J3)

Barionet 50 Only

The Barionet 50 provides an RS-232 serial interface for connection to a variety of devices that implement this serial interface. The RS-232 interface is supplied via a 9-pin sub D male connector, wired as a DTE. The connector's DTE wiring is similar to a personal computer's serial port, as defined in the following table.

Pin	Direction	Function
1		No connection
2	To Barionet	RxD Receive Data
3	From Barionet	TxD Transmit Data
4		No connection
5	From Barionet	Ground
6		No connection
7	From Barionet	RTS (Ready To Send)
8	To Barionet	CTS (Clear To Send)
9		No connection

The baud rate for the RS-232 port is selectable from 300 baud (bits per second) to 230,400 baud with 7 or 8 data bits, 1 or 2 stop bits, and even, odd, or no parity.

Hardware flow control can also be enabled or disabled. When hardware flow control is enabled, the Barionet will only transmit data on the serial port when the CTS signal (Clear to Send) is asserted (high). The RTS signal can be controlled from BCL to control data being transmitted to the Barionet, but it is not automatically asserted when the Barionet 50 serial input buffer fills. Software flow control (using Xon and Xoff characters) is also available.

. Note

The default settings for the RS-232 baud rate, data bits, stop bits, parity and flow control parameters are set through the web configuration interface (see "The Serial Settings Page" in Chapter 2 for more details). These parameters can also be specified in the BCL OPEN statement, and the parameters in the BCL statement override the configuration settings on the web page. If different parameters are specified in a BCL OPEN statement, the parameters revert to the default settings specified in the configuration pages when the connection is closed in BCL.

A blinking RS-232 light on the Barionet or Barionet 50 top cover indicates activity on the RS-232 port.

6 RS-485 Serial Interface (J4)

Barionet 50 Only

The RS-485 serial interface can be used to interface to Barix expansion modules, other Modbus devices, or for interfacing to other RS-485 devices. The Barionet always acts as a Modbus master.

This connector carries the RS-485 signals, a connection for reference ground as well as 5V power supply and ground. The 5V power supply pin can supply a maximum of 100 milliamps and is protected by a self-resetting fuse. The following table defines the pins on this connector.

Barionet 50 Only

Pin	Pin Name	Function
1	G	Ground
2	+5V	+5 VDC (100 mA. max)
3	G	Ground
4	A	RS-485 A data signal
5	B	RS-485 B data signal
6	R	Shield (100 ohm resistor to ground)

The RS-485 indicator on the case flashes when there is activity on the RS-485 port.

Note

If the RS-485 indicator remains constantly lit, it's likely that the A and B signals from the RS-485 port have been swapped.

Barionet 50 Only

7 RS-485 Termination (S1)

In many cases where long wire runs or high baud rates are used, terminating the RS-485 interface will improve communication reliability. When the RS-485 termination switch is turned ON (switched to the right), the RS-485 A and B signals are terminated as shown in Figure 29. This termination is designed for twisted pair cables with a characteristic impedance of about 100 ohms. If a cable with substantially different characteristic impedance is used, an external termination network may be more suitable. Turning S1 off (switch to the left) disconnects the termination network from the RS-485 A and B lines.

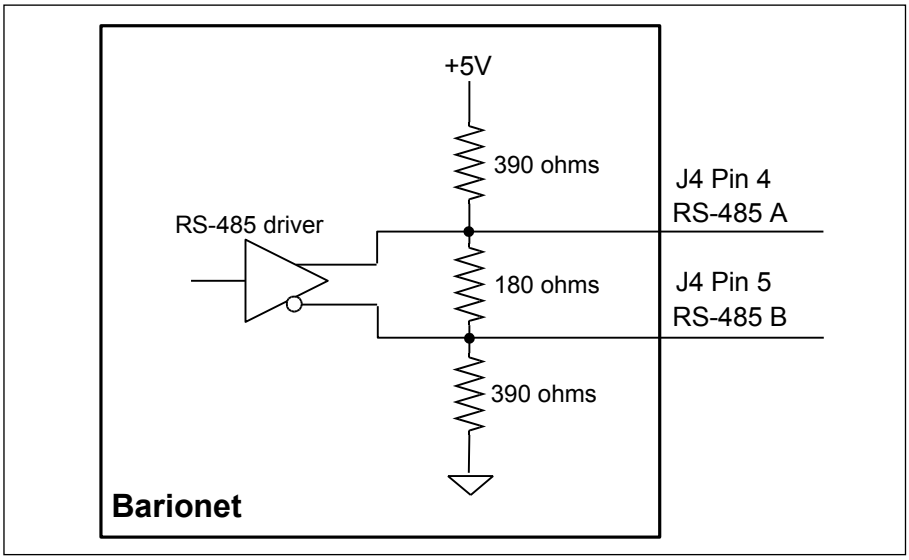


Figure 29. When S1 is turned on, the RS-485 A and B lines are terminated as shown here.

Barionet 50 Only

As a general principle, the higher the baud rate or the longer the cable, the more termination will improve interface reliability. Ideally, both ends of the line should be terminated, but even terminating a single end will improve reliability.

3.2.8 Relay Outputs (J5)

The Barionet 50 provides four single-pole single-throw relay outputs capable of switching up to 0.5 amps at 30 VDC maximum. The following table shows the relay connector pin assignments.

Barionet 50 Only

Pin	Name	Function
1	R1	Relay 1 common
2	NO	Relay 1 normally open
3	R2	Relay 2 common
4	NO	Relay 2 normally open
5	R3	Relay 3 common
6	NO	Relay 3 normally open
7	R4	Relay 4 common
8	NO	Relay 4 normally open

When a relay is active, the common pin is connected to the normally open pin. When the relay is inactive, these pins are disconnected. The four relay LED indicators just above J5 are lit when the corresponding relay is active.

3.2.9 1-Wire® Port (J6)

Barionet 50 Only

This two pin connector provides a Dallas Semiconductor 1-Wire® interface. The Barionet firmware contains built-in support for up to 50 1-Wire temperature sensors and one 1-Wire real time clock.

The pin assignments for the J6 expansion port are defined in the following table.

Pin	Name	Function
1	1W	1-wire data
2	G	1-wire ground

3.2.9.1 Connecting 1-Wire Temperature Sensors

Barionet 50 Only

The Barionet 50 firmware is compatible with the Dallas Semiconductor/Maxim DS18B20, DS18S20, and DS1822 temperature sensors. These are three-pin devices that can be powered either in "parasitic power mode" where the device draws its power from the data stream itself, or they can be powered externally. For parasitic power mode, connect both the Ground and VDD pins together to the Ground pin of J6 (pin 2). Connect the third "DQ" pin to pin 1 ("1W") of the Barionet 50.

If you want to connect multiple temperature sensors to the Barionet 50, wire them in parallel on a single cable connected to J6.

Barix also offers these temperature sensors conveniently packaged with 12" (33 cm) leads and a small mounting hole. The Barix packaged sensors internally tie the Ground (pin 1) and VDD (pin 3) of the sensors together, so there are only two leads to connect—one for Data and one for ground.

For more information on the Barix "TS" Temperature sensor, refer to Appendix C: Accessories.

Each temperature sensor has a permanently stored unique ID number, which is displayed in the Temperature page of the built-in configuration web pages. See the description of The Temperature Web Page in Chapter 2 for more details. The Barionet scans the 1-Wire bus for temperature sensors at start-up and displays the IDs of the first eight sensors on the configuration page. The IDs of all the sensors (up to 50) are available in registers listed in Table A.5 in Appendix A.

There are two simple methods for identifying individual sensors:

1. Connect individual sensors to the Barionet one at a time. Then restart the Barionet and record the sensor's ID. A small label on the sensor wire can be attached to permanently label It.

Barionet 50 Only

2. Connect all the sensors to the Barionet and use cooling spray to cool each sensor one at a time to identify the sensor in the user interface.

The sensors are always displayed in the same order on the user interface until a new sensor is introduced on the bus.

Barionet 50 Only

3.2.10 Inputs (J7)

Four digital inputs are available on the Barionet 50 with a built-in 27K ohm pull-up resistor tied to each input. This allows the inputs to be directly tied to switches or relay contacts that connect the input to ground (pins 1 or 2 of J8). See Figure 30. The input is active when it is tied to ground.

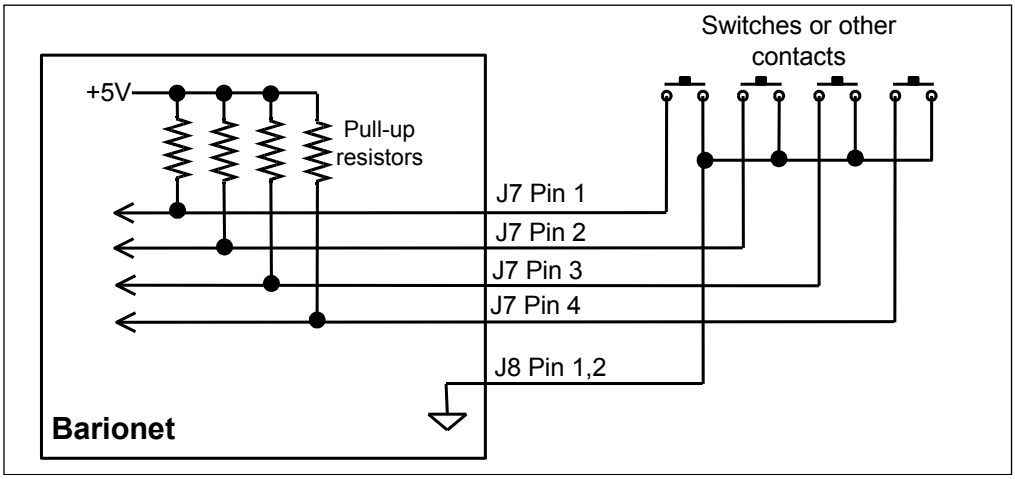


Figure 30. The Barionet 50's four digital inputs are designed to be used with switches or other contacts that connect to ground.

Caution

Do not apply voltage to these inputs. Each input is protected by a 3.3V Zener diode that can be damaged if voltage higher than 3.3V is applied.

Barionet 50 Only

3.2.11 Input Ground (J8)

This connector provides two pins that are the ground reference for the inputs. See Figure 30.

Barionet 50 Only

3.2.12 RS-485 Expansion Port (J10)

This connector provides the same RS-485 data lines as J4 (RS-485 A and B), plus the unregulated power supply and ground. The following table shows the pin assignments.

Pin	Name	Function
1	+	Unregulated power supply
2	-	Power supply ground
3	A	RS-485 A data signal
4	B	RS-485 B data signal

J10 provides a convenient way to connect Barix expansion modules, such as the R6, X8 and IO12, to the Barionet 50 with the 4-pin extension cable supplied with each extension device.

4 Software Interfaces

The Barionet supports a variety of protocols and software interfaces via the Ethernet and serial hardware interfaces. The following table lists the software interfaces and protocols supported by the Barionet. Both models (the Barionet 50 and the Barionet 100 support all of these protocols).

Software Interface/Protocol	Hardware Interface	Application
HTTP	Ethernet	The Barionet comes from the factory with configuration and control web pages. Customized dynamic pages can also be loaded.
ASCII Command	Ethernet	A built-in ASCII command protocol over TCP or UDP, with status change messages sent over a TCP connection or to a specified port using UDP.
SNMP	Ethernet	Allows control and monitoring of the Barionet via Simple Network Management Protocol (SNMP)
Syslog	Ethernet	Allows logging of internal events and messages via the syslog protocol.
Modbus/TCP	Ethernet	Modbus protocol over TCP.
Modbus	RS-485	Control of external Modbus slave modules from the Barionet with custom BCL software.
Serial Gateway	Ethernet, RS-232, RS-485	Built-in serial gateway function forwards data from the Ethernet interface on a defined TCP port to the RS-232 and/or RS-485 interfaces.
Custom BCL Applications	Ethernet, RS-232, RS-485	Custom applications written in the BCL language can interact with any of the hardware interfaces.

This chapter describes each of the software interfaces, but does not go into detail about creating custom applications. Chapter 5 focuses exclusively on creating custom web pages and BCL applications.

4.1 Built-in Web Server (HTTP)

The Barionet has a built-in web server that can serve pages stored in the Barionet's non-volatile memory. By default, the web server listens for HTTP requests on port 80. However, this port number can be changed to a non-standard port number through the web-based configuration interface. See Setting the Web Server Port.

Note

If the web server port is changed from the default value of 80, you'll need to append the new port number to any HTTP requests for web pages from the Barionet, by adding a colon and the port number at the end of the URL. For example, if you change the web server port to 8080 from the default 80, the URL will look something like this: http://192.168.0.40:8080/mypage.html

You can use the standard Barionet and Barionet 50 web pages to monitor the status of the inputs and outputs and to change configuration settings. We'll discuss creating custom web pages in Chapter 5: Creating Custom Applications & Web Pages.

4.1.1 Web Server CGI

The Barionet's built-in web server also implements four CGI commands that allow HTTP requests to initiate a variety of actions, such as setting the status of an output or initiating a subroutine within a BCL program. The Barionet CGI commands that can be invoked by entering them in the browser's address line or in HTML links (as an HTTP “GET”) or embedding them in HTML forms (an HTTP “POST”).

4.1.1.1 The rc.cgi Command

The rc.cgi CGI command can be used to set the status of a Barionet output or relay. It also includes an optional parameter for setting the name of a web page that will be displayed after execution of the rc.cgi command is complete. The format of the rc.cgi command is as follows:

http://<barionet IP address>/rc.cgi?o=<I/O address>,<value>[&L=<response page>]

For example, assuming the Barionet is set to address 192.168.0.32, entering the following line into your browser's address bar will set relay number 2 to the on state (1). It also tells the Barionet to display the index.html page after setting the relay output.

http://192.168.0.32/rc.cgi?o=2,1&L=index.html

Notice that after executing this command, the index.html page that is displayed shows relay number 2 set to the "On" state.

You can set the relay back to the off state by entering the same line but changing the <value> parameter to zero:

http://192.168.0.32/rc.cgi?o=2,0&L=index.html

The <value> parameter can be set to four different ranges of values according to the following table.

Value	Function
0	Set the output to inactive (off)
1	Set the output to active (on)
999	Toggle the output. If it was on, change it to off and vice versa.
2 – 998 or 1000 – 9999	Toggle the output for n * 100 ms. (e.g. 50 = toggle the output for 5 seconds)

See Appendix A: I/O Addressing for information on the I/O addresses of the various Barionet outputs.

4.1.1.2 The BAS.cgi Command

The BAS.cgi CGI command can be used to set the value of one or more variables in a BCL program. The format of the command is as follows:

http://<barionet IP address>/BAS.cgi?<variable name>=<value>

Variable names must already be defined within the currently executing BCL program. Since all variable names in BCL are internally stored in upper case, the variable names in the cgi command line must be in upper case.

Additional variable names and values can be appended to the end of the command using the ampersand character (&) to separate each variable and value pair as follows:

BAS.cgi?<variable name>=<value>&<variable name>=<value>…

For example, the following URL entered in the browser's address bar sets the pre-defined BCL variable named "VAR1" to a value of 9, and the string variable S$ to "Hello".

http://192.168.0.32/BAS.cgi?VAR1=9&S$=Hello

You can also use an HTML form to submit a series of variables and values. Here's a simple HTML form that when submitted sets the same two variables as the example above.

```
<form action="BAS.cgi" method="GET" target="empty">
<input type="hidden" name="VAR1" value="9">
<input type="hidden" name="S$" value="Hello">
<input type="submit" value="Send DATA">
</form>
```

Note

Values submitted directly in a URL using the BAS.cgi command should not be wrapped in quotes. However, when the values are submitted via a form, the HTML value attribute generally expects the values to be wrapped in quotes.

4.1.1.3 The setup.cgi Command

The setup.cgi command is used to store configuration setup values in the Barionet non-volatile setup memory. It can also be used to initiate several operations, such as resetting the Barionet to factory defaults, rebooting, or setting the debug level.

The table below describes the setup.cgi commands and arguments

Setup.cgi command	Arguments	Example	Comments
F	None	setup.cgi?F	Reset the Barionet to factory defaults. See Resetting to Factory Defaults in Chapter 7 for more information.
R	None	setup.cgi?R	Reboot the Barionet. See Rebooting the Barionet in Chapter 7 for more information.
X	0 – 9	setup.cgi?X=0	Set the debug level to the specified value (0 – 9). See Setting the syslog debug level in Chapter 2 for more information.
Y	0 or 1	setup.cgi?Y=1	Set the debug flag. See Setting the debug flags in Chapter 2 for more information.
P1	String	setup.cgi?P1=newpassword	The P1 command sets a new password. The new password is passed as a string argument, and must be 12 characters or less in length. If an existing password is set, the p parameter (described below) must also be included with the old password. Note that the Barionet must be rebooted for the new password to take effect. You can append the R command to initiate a reboot. (e.g. setup.cgi?P1=newpassword&R)
P	String	setup.cgi?P=oldpassword	The P command is used to specify an old password for verification before changing the password with the P1 command. The P command is not necessary if no existing password is set. If the old password specified does match the existing password, the Barionet issues a “You are not Authorized” error.
L	String	setup.cgi?L=newpage.html	Sets the page that setup.cgi should open after completing other operations. This parameter should be specified before a factory reset or reboot commands in a command line.
Setup Variable Name	Depends on the setup variable	setup.cgi?B16=8	Sets the setup memory location specified by the variable name to the specified value. See the Setup memory layout in Appendix B for the setup variable names.

When setup.cgi is called from a web URL, multiple commands can be appended in the URL, separated by ampersands. For example, the following command sets the baud rate (by setting variable B16 to a value of 8). Then it specifies a new page to open after the setup.cgi is complete. Finally, it reboots the Barionet so that the baud rate change takes effect.

http://192.168.1.32/setup.cgi?B16=8&L=newpage.html&R

You can also construct an HTML form that calls setup.cgi. HTML elements within the form must be named to correspond with the setup variable names and setup.cgi commands described in the table above.

For example, here’s a simple form that sets the RS-232 baud rate and displays the “newpage.html” web page after the setup.cgi command is complete. A more realistic form would probably have a drop-down box named "B16" with various valid baud rates the user can select and value properties that correspond to the baud rate setting values specified in the setup memory tables in Appendix B. Note that HTML hidden form elements can be used to specify commands that the user does not change, such as the “L” command that specifies the new page.

```
<form action="setup.cgi" method="GET">
<input type="hidden" name="B16" value="8">
<input type="hidden" name="L" value="newpage.html">
<input type="submit" value="Set Baud Rate">
</form>
```

Note

The setup.cgi command changes the values of configuration parameters in the EEPROM. However, the Barionet must be rebooted for those changes to take effect.

Refer to Appendix B: Configuration and Setup Memory Layout for the names of the setup variables and their layout in setup memory.

Caution

Be very careful using the setup.cgi command because setting inappropriate values in setup memory may cause the Barionet to behave unpredictably or become unreachable.

Setup variable values can also be inserted into HTML pages using the &LSetup dynamic tag. See the description of the &LSetup tag for more information.

4.1.1.4 The basic.cgi Command

The basic.cgi command is designed to initiate a subroutine within a BCL program. When basic.cgi is invoked, a CGI handler subroutine defined within BCL using the **ON CGI** statement is called. For example, assume that the currently running BCL program included the following statement somewhere near the beginning of the program.

ON CGI GOSUB 100

When a basic.cgi command is invoked, the BCL program will suspend the currently executing code and jump to the subroutine that begins at line 100. When this subroutine is complete, control is returned to the original location in the BCL program that was executing before basic.cgi was invoked.

The use of the **ON CGI GOSUB** statement and the basic.cgi CGI command is described in more detail in the BCL Programmers Manual.

4.2 ASCII Command Protocol

The Barionet includes a built-in ASCII command protocol that allows you to send simple English-like commands over TCP/IP. The commands allow you to control Barionet outputs and to get the status of the inputs.

The ASCII command protocol can be used with either TCP or UDP. If you use a TCP connection, only one device (i.e. a computer) can make a connection to the Barionet and send and receive commands. On the other hand, UDP allows multiple devices or computers to send commands to the Barionet.

In order to avoid conflicts with other applications, the ASCII command protocol is disabled by default. To enable the command protocol, you must specify a port number in the Control page of the web-based configuration page for the TCP Command Port and/or the UDP Command Port. See the description of these parameters in the description of The Control Web Page in Section 2.1.

4.2.1 Command Format

All Barionet ASCII commands use a similar format:

command-name[,argument1,argument2,...]<cr>

The command string begins with a command name, followed in some commands by one or more arguments, separated from the command name and each other with commas. The

command is terminated with a carriage return character (decimal value 13, hex 0x0D). Command names are case sensitive. All command names are in lower case.

On the Barionet 100 using a TCP connection, a line feed character may also be appended to the end of the command (decimal value 10, hex 0x0A), though it is not required. Appending a line feed in the Barionet 50 using TCP or UDP or appending a line feed using UDP on the Barionet 100 will cause a command error.

If the command generates a response, the response string will be in a similar format. It will only have a carriage return character (no line feed) at the end.

4.2.2 ASCII Commands

Barionet 100 Only

4.2.2.1 Iolist

The iolist command returns a comma-separated list of the IO capabilities of the Barionet. The return string contains the number of I/O types in each of the following categories:

- Analog inputs (4 on the Barionet 100, 0 on the Barionet 50)
- Digital inputs (8 on the Barionet 100, 4 on the Barionet 50)
- Analog outputs (0 on both the Barionet 100 and Barionet 50—reserved for future use)
- Digital outputs (4 on the Barionet 100, 0 on the Barionet 50)
- IR outputs (0 on both the Barionet 100 and Barionet 50—reserved for future use)
- Relays (2 on the Barionet 100, 4 on the Barionet 50)
- Temperature sensors (zero to 50, depending on the number detected)

The Barionet 50 with no temperature sensors attached responds like this:

io,0,4,0,0,0,4,0

The Barionet 100 with no temperature sensors attached responds like this:

io,4,8,0,4,0,2,0

4.2.2.2 version

This command returns the hardware type and software version of the device using the following syntax:

version,<hardware_type> <command_protocol_version>

Here's a typical response:

version,BARIONET 1.3

Note that the last parameter is the version of the ASCII protocol implemented in this Barionet, not the firmware version of the Barionet.

4.2.2.3 getio,<i/o address>

This command gets and returns the state of the I/O port specified by the parameter. For example, to get the state of the first relay, the command is:

getio,1

The response is in the following format:

state,<i/o address>,<value>

If the first relay is active, the getio,1 command responds with:

state,1,1

For analog values, the getio command returns the un-scaled value. For example, to get the current analog value of analog input number 1 in a Barionet, send the command:

getio,501

With an analog input level of 600 on input 1, the Barionet responds with:

state,501,600

See Appendix A: I/O Addressing for a list of I/O addresses.

4.2.2.4 setio,<i/o address>,<value>

The setio command sets an output to a specified state. The value parameter can be one of three types:

- An absolute value. 1 = active, 0 = inactive
- A special "toggle" value (999), which toggles the output to the opposite of the current state. If the output was active, it goes inactive and vice versa when this value is used in the setio command.
- A pulse value from 2 – 998, which indicates a period of time that the output should be toggled. The output toggles from its current state to the opposite state for the period of time specified in the argument. Then it toggles back to the original state. The pulse parameter is in 100 millisecond units (1/10th second). A parameter of 50 indicates that the specified output will toggle for 5 seconds and then return to the original state.

For example, to set the first relay to active state, send the command:

setio,1,1

To toggle relay 1 for 15 seconds and then return to its previous state, send this command:

setio,1,150

Sending another setio command with a toggle value (i.e. from 2 – 998) while the output is toggled, will not toggle the output again, but simply extend the amount of time that the output is toggled.

4.2.3 Unsolicited State Change Messages

The Barionet can also send unsolicited state change messages when an input or output changes state. A state change message is delivered in the following format:

statechange,<i/o address>,<value>

State change messages are handled a bit differently over TCP and UDP.

4.2.3.1 State Changes Message over TCP

With a TCP connection, state change message behavior is controlled in the Control page of the configuration web pages. If the "TCP Initial I/O State Subscriptions" parameter in the Control settings page is set to "Local I/O", state change messages for any digital input or output or relay are sent to the TCP socket specified in the TCP Command Port parameter.

If the TCP Protocol default Subscriptions parameter is set to "None", state change messages are not sent by default. Instead, the "TCP Add I/O state Subscriptions" parameter affects when state change messages are enabled. If the "TCP Add I/O state Subscriptions" setting is set for "With getio/setio", every time a getio or setio command is sent, state change messages are automatically enabled on the input or output addressed by the getio or setio command.

Refer to the description of the Setting the TCP initial I/O state subscriptions and Setting the TCP add I/O state subscriptions settings in Chapter 2 for more information on these parameters.

The table below summarizes the behavior of state change messages over a TCP connection.

TCP Initial I/O state Subscriptions Setting	TCP Add I/O state Subscriptions Setting	State Change Messages Sent?
None	None	No state change messages sent.
Local I/O	Either setting	All state change messages are sent.
None	With getio/setio	State change messages sent for any input or output previously addressed by a getio or setio command.

In addition, if the TCP Default Subscriptions setting is set to "Local I/O", the Barionet will send one state change message for each of the digital inputs and outputs when a TCP connection on the TCP command port is first established.

4.2.3.2 State Changes Message over UDP

State change messages are also transmitted over UDP in the same format. In order to receive state change messages, the UDP info send to parameter and the UDP destination port must be set. If these parameters are set to non-zero values, state change messages will be sent via UDP to the specified IP address and port whenever one of the digital inputs, outputs, or relays changes state.

Note

A single UDP packet may contain more than one state change message if multiple state changes occur simultaneously or very close in time. Each state change message is terminated by a carriage return.

In addition, if the UDP interval value is set to a value other than zero, a series of state messages will be sent at the interval (in seconds) defined by the UDP interval value. One state message is sent for each digital input, output, and relay. These periodic messages are in the same format as state change messages, but use the command word "state" instead of "statechange" to differentiate the messages.

Here's a typical set of state messages sent from a Barionet 50:

```
state,1,0
state,2,0
state,3,0
state,4,0
state,201,0
state,202,0
state,203,0
state,204,0
```

All the state messages may be sent in a single UDP packet, but each state message is terminated by a carriage return.

Refer to the description of Setting the UDP Send Info Address and Setting the UDP Destination Port in Chapter 2 for more information on these parameters.

4.2.4 Getting Started with the ASCII Command Protocol

A simple way to begin using the ASCII command protocol is to set the TCP port number to the standard telnet port (23) and open a connection to the Barionet 100 or Barionet 50 using a telnet (terminal) application.

For best results, set your terminal application for:

- Local echo. This causes the terminal application to show the characters you type on screen without depending on the Barionet to "echo" the characters back to the terminal application. The Barionet does not echo command characters sent to it.
- Append a line feed to carriage returns. The Barionet terminates all its responses with carriage return (no line feed). As a result, when you get a response from the Barionet,

the terminal program's cursor will still be on the same line as the last response from the Barionet. The next command you type will appear over the top of the previous response. Configuring your telnet application to append a line feed after any carriage return character it receives moves the cursor down a line after each response. This behavior has no effect on the functionality of the ASCII command protocol, but makes the display easier to read.

In Microsoft Windows, the HyperTerminal application that comes standard with the operating system can be configured this way. To use HyperTerminal to connect to the Barionet, follow these steps:

1. In HyperTerminal, create a new connection, and give it any name you choose. Click OK.
2. In the second dialog box choose "TCP/IP (Winsock)" from the "Connect Using" drop down box and click OK.
3. Enter the IP address of the Barionet or Barionet 50 in the "Host Address" box. Make sure the Port number matches the TCP Port number you set in the Barionet's configuration. The default for the telnet application is port 23.
4. Choose "Properties" from the File menu and click the "Settings" tab in the dialog box. Click the "ASCII Setup button near the bottom of the dialog.
5. Check the "Send line ends with line feed", the "Echo typed characters locally" and the "Append line feeds to incoming line ends" check boxes in this dialog and click "OK".
6. Click OK in the Properties dialog box.
7. Check your connection by typing version and pressing enter. You should see a response similar to the one shown in the description of the version command above.

4.3 SNMP

The Barionet implements the Simple Network Management Protocol (SNMP) for monitoring and control. Typically, the Barionet will be monitored or controlled using an SNMP manager program installed on a computer on the network.

The SNMP manager uses a special database file called a Management Information Base (MIB) to define the objects (registers) that can be managed in the device. The Barionet ships with an MIB file stored in its non-volatile memory that defines the Barionet's I/O capabilities. You can download the MIB file by clicking on a link in the help pane on the right side of the SNMP page in the Configuration web pages. The MIB file is available in two forms—a directly viewable form that will appear in your browser when you click the link, and a ZIP file that can be downloaded from the Barionet into your computer. If you download the ZIP file, you'll need a program capable of extracting the MIB file from the ZIP package. Most recent version of Microsoft Windows as well as Mac OS X versions 10.3 and later, have built-in ability to open and unpack ZIP files. The unpacked barionet.MIB file can then be loaded into the SNMP manager.

The SNMP manager can monitor the operational state of the Barionet (i.e. is it up and running?) and also read and write to the entire range of inputs, outputs, and virtual I/O registers. The Barionet SNMP implementation uses the same I/O addresses (i.e. registers) as the CGI commands, the ASCII protocol, Modbus/TCP, and the IOSTATE and IOCTL statements in BCL. See Appendix A: I/O Addressing for a complete listing of the I/O addresses.

The Barionet also implements SNMP traps, which are special SNMP messages initiated by the Barionet to inform an SNMP manager of an alarm condition. You can configure the Barionet 100 and Barionet 50 to generate a TRAP when a digital input changes state. You can also generate a trap using the **TRAP** statement in a BCL program.

To use traps generated by the digital inputs, you must set the trap receiver IP address in the configuration web pages and enable traps on the digital inputs. You can also optionally specify a repeat time to generate traps at the specified interval for as long as the input is active. If the repeat time is set for zero (the default), a trap is generated each time the input

changes state (i.e. goes inactive to active or active to inactive), but no traps are generated except at the input transitions. If the trap receiver IP address is set for 0.0.0.0 (the default), no input state change traps are generated even if traps are enabled on the digital inputs.

The **TRAP** statement in BCL specifies its own receiver IP address within the statement, so it is not dependent on the configuration setting for the trap receiver IP address. The **TRAP** statement ignores this setting.

Refer to the description of the SNMP Configuration page for more details on setting the SNMP parameters and for downloading the MIB file.

Note

If traps are enabled on all of the Barionet 100 inputs (addresses 201 – 208), traps will also be generated on the virtual I/O bits from address 209 to 400. If any of the eight inputs are set to "No" in the SNMP Configuration page, traps will not be generated on these virtual I/O bits. See Table A.5 in Appendix A for more information on I/O addresses.

4.4 Syslog

Barionet 100 Only

The Barionet can send text messages to a standard syslog server using UDP. There are a variety of syslog programs available on the Internet for capturing and viewing syslog messages.

The Barionet generates two types of syslog messages:

- System generated status, debug, and error messages. These messages are built-in to the Barionet firmware. The type of messages generated depends on the Debug Level setting set in the Control page of the configuration web pages.
- User-generated messages embedded in a BCL program using the **syslog** statement. For more information on using the syslog statement, refer to the BCL Programmers Manual. Syslog statements inserted into a BCL program are an important tool for debugging BCL programs. See Chapter 7 for more information on troubleshooting.

Setting the syslog server IP address causes syslog messages to be sent to that specific IP address on port 514. If the syslog server IP address is set to 0.0.0.0, the Barionet sends syslog messages on the subnet's broadcast address, so all devices on the same subnet that listen on port 514 will receive the messages. For more information on setting the syslog server IP address and syslog debug level, refer to the description of these settings in Chapter 2.

4.5 Modbus/TCP

The Barionet can also be controlled and monitored using the industry-standard Modbus/TCP protocol. Support for the Modbus/TCP protocol is built in to the Barionet firmware.

The Modbus/TCP protocol uses standard TCP Port 502. The only configurable setting for the Modbus/TCP protocol is the timeout period. If a timeout period is set, the Barionet closes any open connection to port 502 if no traffic occurs on the port within the timeout period. See Setting the Modbus/TCP Timeout in Chapter 2 for more information.

The Barionet implements most of the standard function codes specified in the Modbus/TCP standard. The following table shows the function codes and their use in the Barionet.

<table>
<tr><th>Function Code</th><th>Description</th></tr>
<tr><td>01</td><td>Read Coils. Reads the current state of one or more inputs, outputs, or registers. The state is returned as a single bit (1 or 0) per input, output, or register packed into one or more bytes in the response.</td></tr>
<tr><td>02</td><td>Read Discrete Inputs. Functionally identical to function code 01 in the Barionet implementation.</td></tr>
<tr><td>03</td><td>Read Holding Registers. Reads the state of one or more 16-bit registers. For registers that map to digital inputs, the response is either 0 or 1. For registers that map to analog values (temperatures or analog inputs on the Barionet), the current analog value is returned.</td></tr>
<tr><td>04</td><td>Read Input Registers. Functionally identical to function code 03 in the Barionet implementation</td></tr>
<tr><td>05</td><td>Write single coil. Writes a single bit (0 or 1) to a Barionet I/O register. If the register maps to an output, the output is set active or inactive based on the value written. If the register maps to an input, the Barionet responds by echoing the request as per the Modbus/TCP specification.</td></tr>
<tr><td>06</td><td>Write single register. Writes a 16-bit value to a Barionet I/O register. If the register corresponds to a digital output or relay the output is set as follows:
<table>
<tr><th>Value</th><th>Output Response</th></tr>
<tr><td>0</td><td>The output is set to inactive (off)</td></tr>
<tr><td>1</td><td>The output is set to active (on)</td></tr>
<tr><td>999</td><td>The output is toggled from its current state to the opposite state.</td></tr>
<tr><td>2-998 or 1000-9999</td><td>The output is toggled for the period specified by the value in 100-millisecond (1/10 second) increments. For example, writing a value of 50 toggles the output for a period of 5 seconds.</td></tr>
</table></td></tr>
<tr><td>15</td><td>Write multiple coils. Similar to function code 05 except that this function can write a single bit (0 or 1) to up to 65,535 registers in a single operation.</td></tr>
<tr><td>16</td><td>Not implemented in the Barionet/Barionet 50</td></tr>
</table>

For more information on the Modbus/TCP and Modbus protocols, please see http://www.modbus.org/specs.php and the Barix Modbus wiki at: http://wisi.barix.com/index.php5/Modbus.

4.6 Modbus via RS-485

Unlike the Modbus/TCP protocol, which is built into the Barionet firmware, Modbus ASCII or Modbus/RTU protocols over the RS-485 interface must be implemented by the user in BCL code. Both Modbus ASCII and Modbus/RTU (binary) protocols are relatively straightforward to implement using BCL. The Barix expansion modules (X8, R6, and IO12) all implement the binary Modbus/RTU protocol.

The RS-485 interface settings can be configured in the Serial Settings page in the Configuration web pages. The Barix expansion modules default to 19200 baud, eight bits, even parity, and one stop bit.

For more information on implementing the Modbus/RTU protocol in BCL and interfacing to the Barionet expansion modules, see the following references:

- The Modbus Specification: http://www.modbus.org/specs.php
- The Barix BCL Programmer's manual, available for download from the Barix web site at: http://www.barix.com/downloads/Barionet_Family/51/
- The Barix Expansion module manuals: http://www.barix.com/downloads/Barionet_Extensions/1441/

4.7 Serial Gateway Function

The Barionet also has a built-in serial gateway function that can be used to forward traffic on one or both of the RS-232 and RS-485 interfaces to TCP/IP. If a TCP connection is made to the port specified in the Local Port setting on the configuration page, any traffic received by the Barionet on this TCP connection will be sent on the corresponding serial interface (either the RS-232 or RS-485, depending on the Local Port settings). Data received on the serial interface will be sent to the TCP connection as well.

Note

Don't confuse the serial gateway function, which is built into the Barionet firmware with the digital I/O and serial tunnel sample application, which is a sample BCL application pre-loaded with the standard firmware. The serial gateway function forwards traffic from one or both of the serial interfaces to TCP/IP. The serial tunnel BCL application allows to Barionets to "tunnel" serial port data across a network via TCP/IP. For more information on the digital I/O and serial tunnel sample application, see Chapter 6.

5 Creating Custom Applications & Web Pages

The Barionet offers a variety of built-in interfaces that you can use to monitor and control the various I/O functions without any custom programming or web page development. These standard interfaces are described in Chapter 4 of this manual.

For more advanced or customized applications, you can create custom HTML web pages including static text, graphics as well as proprietary Barix tags that insert dynamic data, such as the current state of Barionet inputs or outputs into the page. For ultimate application flexibility, you can also create and load custom BCL programs into the Barionet. BCL programs can read Barionet inputs, control outputs, accept input from the web server and send output to the web server, as well as send and receive data on the RS-232, RS-485, and Ethernet interfaces.

This chapter provides details on creating and loading custom HTML pages into the Barionet. It also describes the process of creating, loading, and debugging BCL programs. For more details on the BCL language, refer to the BCL Programmers Manual.

5.1 Development Process Overview

Figure 31 shows a flow chart of the typical development process for creating or modifying custom BCL or HTML applications. The process includes six main steps. You may skip some of these steps if you don't need to create custom HTML pages or you don't need custom BCL code.

1. If you need to create custom HTML pages, write or edit the HTML page file(s) using any convenient editor that generates plain text output. Some editors will even help check your HTML syntax.
2. If you need to create custom BCL code, write the code, again using a convenient text editor. You can use almost any editor that can create plain text files. For the Windows platform, the free Notepad++ editor (http://notepad-plus.sourceforge.net/uk/site.htm) can be set to recognize BASIC syntax, which is very similar to BCL. Versions of Notepad++ are available for a variety of operating systems.
3. Process the BCL text file through the tokenizer program to create a .TOK file. You can skip this step if you don't need a custom BCL program.
4. Package the HTML files, the .TOK file output from the tokenizer (if you are creating a custom BCL program), and some other support files into a .COB file using the web2cob program (called bpkg in Macintosh and Linux).
5. Upload the .COB file to the appropriate user memory area in the Barionet.
6. Test the pages and/or application. If you find bugs or need to make changes, repeat this process.

The next several sections describe these steps in more detail.

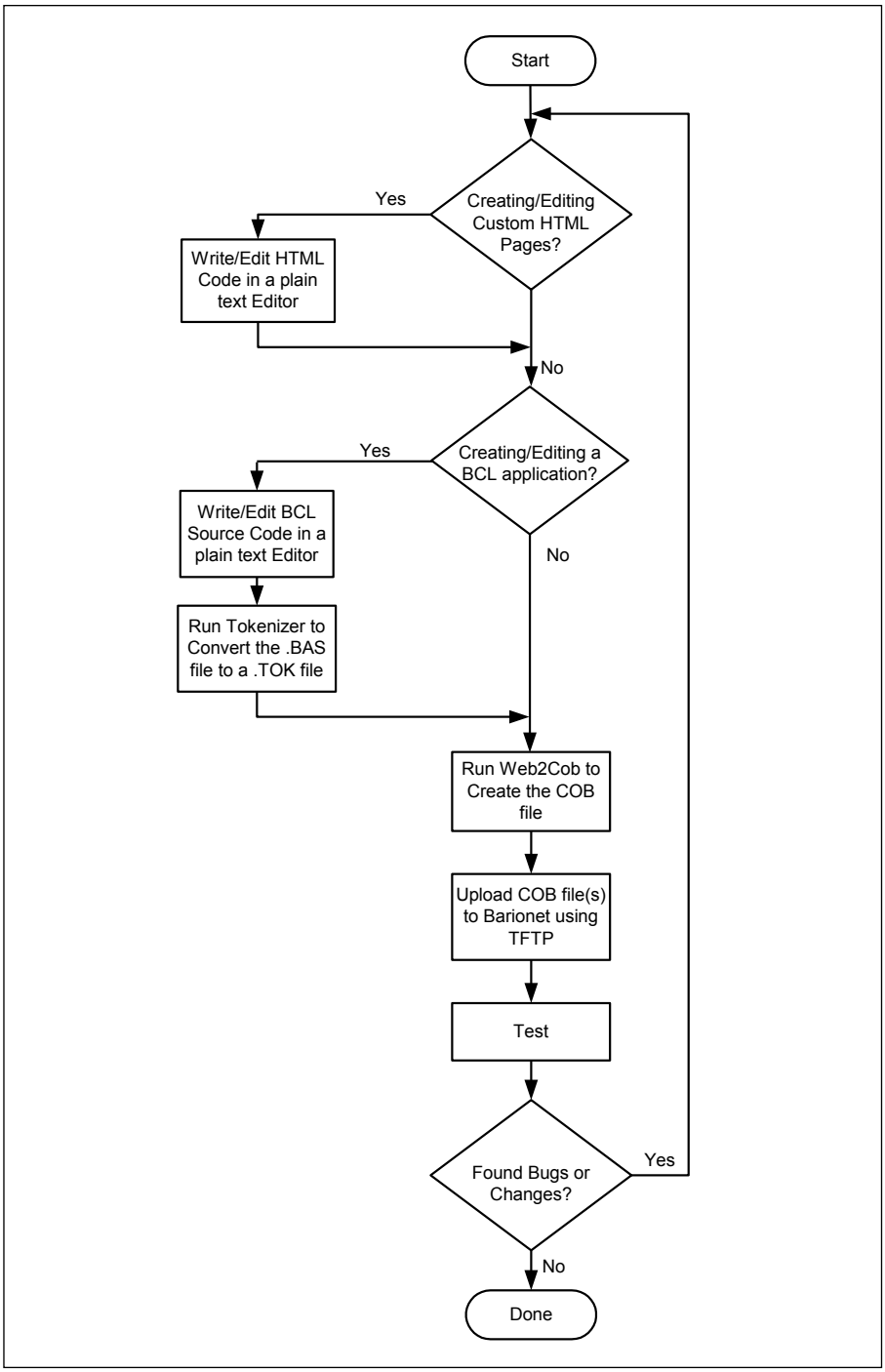


Figure 31. A flow chart of the typical development process for custom HTML pages and BCL applications.

5.2 Creating Custom HTML Pages

Barionet web pages can include four types of elements:

- Text and images
- Standard HTML
- JavaScript
- Barix Proprietary HTML tags

The first three types of elements are common to all standard web pages. These elements are transmitted from the web server to the browser exactly as they are stored in the Barionet web server's memory without any changes. Thus, the text, images, HTML and JavaScript you embed in the pages can contain anything that the browsers you use will support. They do not depend on the Barionet. Note that the Barionet does not support a server-side scripting language like PHP or ASP, though much of the same functionality can be implemented using BCL code and Barix dynamic tags.

This manual assumes that the reader is familiar with standard HTML. There are many excellent tutorials and references available for HTML on the internet.

5.2.1 Barix Dynamic Tags

Imbedding Barix proprietary tags in your Barionet web page allows you to display a variety of dynamic data, including the current status of inputs and outputs as well as BCL variable values, and setup memory values.

Note

Unlike standard HTML, the Barix proprietary tags described in this section are case sensitive. They must appear in the case shown to be recognized by the Barionet's web server. Otherwise, the tag will not be replaced and may appear in the web page unchanged.

5.2.1.1 Barix Tag Summary

The following table lists the Barix proprietary tags and their function. Each tag is described in more detail in the following sections.

Tag	Description
&L(…);	Barix Initialization tag. Required before any other Barix dynamic tags.
&LIO(…)	Inserts the current value of an input or output in the page in place of this tag.
&LBAS(…)	Inserts the current value of a BCL variable in place of this tag or calls a BCL subroutine that generates output to insert in place of this tag.
&LSetup(…)	Inserts the value of a configuration variable in place of this tag.

5.2.1.2 Initialization Tag

To use any of the Barix dynamic tags in your web pages, the web server requires a special initialization tag be present before any other Barix tags and within the first 512 bytes of every web page that uses the dynamic tags. The format of the initialization tag is:

&L(0,"*",<security flag>);

The optional **<security flag>** argument is a security flag that indicates whether the page should be protected by the configuration password feature. Set this flag value to 2 to protect the page or zero to leave the page unprotected. If the third parameter is omitted, it will default to zero (unprotected). See the description of The Security Web Page for more details on setting a password to protect the standard configuration pages as well as any custom pages marked with this parameter. Refer to Password Protecting Custom HTML Pages for more details on using the password protection feature in custom web pages.

This initialization tag can appear anywhere within the first 512 bytes of the page, but it is typically embedded in HTML comment tags somewhere in the <HEAD> section of the page like this:

```
<HTML>
<HEAD>
<!-- &L(0,"*",2); -->
</HEAD>
```

It's not required that you embed the initialization tag within an HTML comment (i.e. within the <!-- and --> tags). However, doing so prevents the tag from appearing as a syntax error in editors that check HTML syntax.

5.2.1.3 I/O Status Tag (&LIO)

The &LIO tag inserts the current value of an input or output into a web page. When the Barionet web server encounters this tag in a web page, it replaces the tag with the current

status of the specified input or output before sending the page to the browser.

There are two forms of the &LIO tag. The syntax of the first form of the tag is:

&LIO(1,<format>,<I/O address>);

<format> is a string that controls the formatting of the output value, similar to the format string used in the "C" programming language for formatted output. See the description of Format Strings later in this section for details in the format string.

<I/O address> specifies the address of the I/O whose current state is to be inserted in place of this tag. See the tables in Appendix A for a list of I/O addresses associated with various inputs and outputs.

For example, the following tag inserts the current state of digital input number 2 into the web page. The Barionet web server replaces this tag with either a "1" or a "0", depending on the state of the digital input at the time that the page is generated.

&LIO(1,"%u",202);

While this tag only inserts a “1” or “0”, the &LIO tag can also be imbedded in other HTML, such as an image tag. Using this technique, you can create a page that displays different images depending on the state of an input. For example, the following HTML displays “image1.jpg” if the input is active, and image0.jpg if the input is inactive:

<IMG src=”image&LIO(1,"%u",202);.jpg”>

Analog values can also be inserted. The tag below inserts the raw value of the first temperature sensor, which ranges from 0 – 4095 (A value of 4096 indicates no sensor is present). However, this tag outputs the value in fixed point format, with a single digit to the right of the decimal point, so a value of 235 would be displayed as 23.5.

&LIO(1,"%0.1F",601).

There is also a second form of the &LIO tag that is particularly useful for displaying analog input values because this form of the tag also includes three scaling factors that make it easy to convert the raw analog values into meaningful values. The syntax of the second form of the tag is:

&LIO(2,<format>,<I/O address>,<multiplier>,<offset>,<divisor>);

<format> A string that controls the formatting of the output value, similar to the format string used in the "C" programming language for formatted output. See the description of Format Strings later in this section for details in the format string.

<I/O address> Specifies the address of the I/O whose current state is to be inserted in place of this tag. See the tables in Appendix A for a list of I/O addresses associated with various inputs and outputs.

The last three arguments specify values that are applied to scale the analog value before it is inserted into the web page. They are applied to the numeric value in the order they appear in the tag.

<multiplier> The numeric value from the input is first multiplied by the multiplier.

<offset> Next, the offset is added to the result after the multiplier is applied.

<divisor> Finally, the resulting value is divided by this constant.

Displaying a temperature sensor value in degrees Celsius provides a good example of using this form of the tag. As previously mentioned, the temperature sensors return a raw value of 0 – 4095 (or -2048 to +2047 in signed notation). The following tag inserts the temperature reading from the first temperature sensor, scaled in degrees Celsius:

&LIO(2,"%.1F",601,5,0,8);

In this tag, the raw signed value from the first temperature sensor (I/O address 601) is multiplied by 5, no offset is added, and the result is divided by 8. The result is a temperature value that is 10 times the actual value in degrees Celsius. However, the **"%.1F"** format string effectively divides the value by ten by inserting the decimal point with one digit to the right of the decimal.

For example, a raw output value from the sensor of 520 would be displayed as 32.5 degrees Celsius. (520 x 5 / 8 = 325, with one digit after the decimal = 32.5).

Scaling the same raw temperature value to degrees Fahrenheit is easy as well:

&LIO(2,"%.1F",601,9,2560,8);

In this example, the raw value from the first temperature sensor is multiplied by 9. Then a value of 2560 is added to account for the 32 degree offset between Fahrenheit and Celsius scales, and the result is divided by eight. The output is a value that is ten times the temperature in Fahrenheit. Again, the **"%0.1F"** moves the decimal point one place to the left, resulting in a value in degrees Fahrenheit with one digit after the decimal place.

Using the same example as above, if the raw sensor value is 520, the output would be 90.5 degrees Fahrenheit. ((520 x 9 + 2560) / 8 = 905, with one digit after the decimal = 90.5).

Barionet 100 Only

The same principle can be applied to scaling values read from the analog inputs on the Barionet. The analog inputs provide a 10-bit (0 – 1023) raw output that corresponds to 0 to 5 volts input. The following tag displays the first analog input scaled to display the output as 0 to 5 volts.

&LIO(2,"%.2F",501,500,0,1023);

This tag multiples the raw analog output by 500, adds no offset, and then divides by 1023. The result is value from 0 – 500. The format string ("%.2F") moves the decimal point two places to the left, resulting in a value from 0 to 5.00 volts.

If the analog input is measuring a higher voltage with a resistive divider to reduce the voltage to 0 – 5 volts, you can scale the value to account for the effects of the voltage divider in this tag as well. For example, if you're measuring a level that goes from 0 to 10V, you would add a 2:1 voltage divider to reduce the 0 – 10V range to 0 – 5V. (see Creating an Analog Voltage Divider in Chapter 3). However, you can compensate for the voltage divider so that the &LIO tag displays 0 – 10V, by adjusting the scaling factors in the tag. Here's a tag that will display values from 0 to 10V.

&LIO(2,"%.2F",501,1000,0,1023);

Note

Keep in mind that the Barionet does all internal math as integers. There is no floating point data type in the Barionet, so all the scaling factors must be integers. To avoid truncation errors in the math, be sure to use a multiplier that scales the value up enough so that you can divide by an integer to get accurate results. All BCL integers are 32 bits (-2147483648 to +2147483647). Take care to avoid scale factors that will result in overflowing this range.

5.2.1.4 BCL Variable Tag (&LBAS)

The &LBAS tag inserts the value of a pre-defined BCL variable in a web page. When the Barionet web server encounters this tag, it retrieves the current value of the specific variable and replaces the tag with the value of the variable before sending the page to the browser. If the variable does not exist in the currently executing BCL program, or a BCL program is not running (e.g. at startup), the string [NO_VAR] is inserted in the page.

There are two forms of the BCL variable tag. The first form is show below:

&LBAS(1,<format>,<variable name>);

<format> is a string that controls the formatting of the output value, similar to the format string used in the "C" programming language for formatted output. See the description of Format Strings later in this section for details in the format string.

<variable name> specifies the name of a BCL variable that must be defined in the currently executing BCL program.

Here's a simple example that displays the current up time in seconds from the last time the Barionet was booted, which is stored in the pre-defined BCL variable **_DTS_**.

&LBAS(1,"%lu",_DTS_);

The variable in the tag may also be a string variable or a numeric array as in these examples:

&LBAS(1,"%fs",Str$);

&LBAS(1,"%ld",Array(5));

If an array element is specified in the &LBAS tag that is out of the range, the tag returns "[NO_VAR]".

The second form of the BCL variable tag invokes a BCL subroutine to return text, instead of simply retrieving a pre-defined BCL variable value. This form of the tag can be useful if special processing or formatting of a value is required. Here is the second form of the &LBAS tag:

&LBAS(2,<CGI string>,0);

When the Barionet web server encounters this form of the tag, it calls the subroutine defined by the BCL statement **ON CGI GOSUB** and loads the pre-defined BCL variable **_CGI_$** with the **<CGI string>** from the &LBAS tag. The web server is blocked from proceeding until the BCL subroutine finishes executing. The BCL subroutine sends any output to be substituted for the &LBAS tag by writing to the pre-defined -1 stream handle (e.g. **write -1,"output"**). The subroutine finishes by clearing the **_CGI_$** variable and executing a **return** statement.

Because the web server is blocked from proceeding until the BCL routine finishes, subroutines that are called by this method must be kept as short and fast as possible.

The third argument in this form of the &LBAS tag is required, but ignored.

For more information on writing BCL subroutines please refer to the BCL Programmers Manual available for download from the Barix web site: http://www.barix.com.

5.2.1.5 Setup Data Tag (&LSetup)

The &LSetup tag inserts configuration values stored in the Barionet's non-volatile configuration memory into a web page. The &LSetup tag has three forms:

This first form of the &LSetup tag is the simplest and most common:

&LSetup(1,<format>,<pos>[,<type>]);

<format>	A string that controls the formatting of the output value, similar to the format string used in the "C" programming language for formatted output. See the description of Format Strings later in this section for details in the format string.
<pos>	Specifies the byte offset position of the setup variable in the Barionet's setup memory. See the setup memory layout tables in Appendix B.
<type>	Specifies the type of setup variable. There are four valid values:

Type	Description
bx	A bit value where x = bit number
B	A byte value (the default)
S	A null-terminated string
W	A word value

If the <type> parameter is omitted, it is assumed to be a byte.

For example, the following HTML displays an editable text box that is pre-filled with the current DHCP Host name:

DHCP Hostname:
<input name=S109 size=15 maxlength=15 value="&LSetup(1,"%s",109,S);">

The DHCP host name is a string that is stored beginning at location 109 in the setup memory. The &LSetup tag above retrieves the host name string from location 109 and inserts it in the **value** attribute of the **input** tag, so the input box appears pre-filled with the current DHCP host name setting.

The **name** attribute in the HTML **input** tag above is not significant in retrieving the DHCP host name. However, the **name** attribute is significant when this text box is part of a form that is submitted using setup.cgi as the action for processing the form to save the results. If the user makes changes to the DHCP host name value, the setup.cgi command can be used to store the new value into setup memory. The **name** attribute tells setup.cgi where to store the value of the input box.

In the HTML above, the name attribute of the **input** tag encapsulates the data type (S for a string) and the bye position (109) of the data in setup memory. If this input box is part of a form that is processed by setup.cgi, the name (S109) tells setup.cgi the type of data and where to store it in setup memory. See the description of The setup.cgi Command for more details on saving setup parameters using setup.cgi.

The second form of the &LSetup tag is designed specifically for netmask values. Netmask values are stored in setup memory as a single integer with a count of the number of zero bits in the netmask. The syntax of the second version of the &LSetup tag is very similar to the first:

&LSetup(2,<format>,<pos>,<octet>);

<format> For this form of the &LSetup tag, the <format> string is generally always "%u" because the net mask values are stored as a 16-bit integer. See the description of Format Strings later in this section for details in the format string.

<pos> Specifies the byte offset position of the setup variable in the Barionet's setup memory. See the setup memory layout tables in Appendix B.

<octet> Specifies which of 4 sets of numbers (called "octets") of the netmask value should be displayed.

A few examples should help clarify the use of this form of the &LSetup tag.

Assume that the Barionet net mask is set for 255.255.255.0. Since the value in setup memory is the count of zeros in the netmask, starting with the last byte, the value would be 8 (there are eight zeros).

A netmask of 255.255.0.0 would be stored as a value of 16.

Here is a set of four &LSetup tags that will display the netmask value:

<input name=N0 size=3 maxlength=3 value=&LSetup(2,"%u",6,0);> .
<input name=N1 size=3 maxlength=3 value=&LSetup(2,"%u",6,1);> .
<input name=N2 size=3 maxlength=3 value=&LSetup(2,"%u",6,2);> .

<input name=N3B6 size=3 maxlength=3 value=&LSetup(2,"%u",6,3);>

Notice that there are four input boxes. Each box retrieves the value from location 6 of setup memory. However, instead of directly displaying the value, the value is broken into four parts based on the count of zeros and each part is displayed as a value from 0 – 255.

For the sake of simplicity in this example, we've omitted some JavaScript code that is included in the Barionet's setup pages to verify the netmask values before they are stored.

The third form of the &LSetup tag conditionally inserts a string if the value in setup memory matches a value specified in the &LSetup tag. This form of &LSetup is particularly useful in the creating HTML list boxes, where one of several options displayed in the box must be marked as "selected", based on the current value of the setup variable. Here's the syntax of the third form:

&LSetup(3,<format>,<pos>,<type>,<compare_value>,<return string>);

<format> A string that controls the formatting of the output value, similar to the format string used in the "C" programming language for formatted output. See the description of Format Strings later in this section for details in the format string. With this form of the &LSetup tag, the format string is usually "%s" because the output value is usually a string, such as "selected"

<pos> Specifies the byte offset position of the setup variable in the Barionet's setup memory. See the setup memory layout tables in Appendix B.

<type> Specifies the type of setup variable. There are four valid values:

Type	Description
bx	A bit value where x = bit number. (e.g b0 = bit zero of the byte)
B	A byte value
S	A null-terminated string
W	A word value

<compare value> Specifies a value that should be compared to the current value of the setup variable. If this compare value matches the current setting, the setup tag is replaced with the <return string>. Otherwise, the setup tag is replaced with nothing.

<return string> If the <compare value> matches the current value of the setup variable, the &LSetup tag is replaced with this string. Otherwise, the setup tag is removed from the HTML output and replaced with nothing.

Here's a simple example of using this form of the &LSetup tag to create a drop-down box with options to enable or disable traps on Digital input #1:

```
<select size=1 name=B65b0>
   <option value=1 &LSetup(3,"%s",65,b0,1,"selected");>Yes</option>
   <option value=0 &LSetup(3,"%s",65,b0,0,"selected");>No</option>
</select>
```

The &LSetup tags in this HTML fragment each compare the value of Bit 0 of the byte at position 65 to the value 1 or 0. If bit 0 of the 65th byte is 1, the first tag returns "selected", which causes the first option in the drop-down select box to be selected. If bit 0 is 0, the first &LSetup tag returns nothing and the second tag returns "selected" so the second value in the drop-down box is selected.

5.2.1.6 Format Strings

The &LIO, &LBAS, &LSetup dynamic tags described in the previous sections all include a

"format" string that controls how the values that the web server substitutes for these tags are displayed. The format strings follow a syntax that is similar to the "printf" format strings used in the "C" programming language. The set of formatting strings that is supported in these Barix dynamic tags is a subset of the standard "C" strings, with some functional differences.

The following table summarizes the supported format strings:

Format	Example	Applies to	Description
%[[–|0]n]u	%u	&LIO, &LBAS, &LSetup	Display the value as an unsigned 16-bit integer (0 – 65,535)
%[[–|0]n]lu	%lu	&LIO, &LBAS, &LSetup	Display the value as an unsigned 32-bit integer (0 – 4,294,967,295)
%[[–|0]n]d	%d	&LIO, &LBAS, &LSetup	Display the value as a signed 16-bit integer (-32,768 - +32,767)
%[[–|0]n]ld	%ld	&LIO, &LBAS, &LSetup	Display the value as a signed 32-bit integer (-2,147,483,648 - +2,147,483,647)
%[[–|0]n]x	%x	&LIO, &LBAS, &LSetup	Display the value as a 16-bit hexadecimal number (0 – FFFF)
%[[–|0]n]lx	%lx	&LIO, &LBAS, &LSetup	Display the value as a 32-bit hexadecimal number (0 – FFFFFFFF)
%[[-|0][n].dF	%.2F	&LIO, &LBAS	Display the integer value in a fixed-point format with a specified number of digits to right of the decimal point.
%c	%c	&LIO, &LBAS	Display the integer value as a single ASCII character
%fs	%fs	&LBAS, &LSetup	Display a string
%v	%v	&LIO, &LBAS	Display the Barionet firmware version

Each of the numeric format strings includes several optional parameters that control specifics of how the output is formatted. The following table describes the optional parameters

Parameter	Applies to	Description
n	%u, %lu, %d, %ld, %x, %lx, %F	Specifies the number of character positions in the output string. If this parameter is omitted, the output string is only as wide as the number of digits required to display the value. If the value is smaller than the number of digits required, the full value is displayed regardless. If this parameter is larger than the number of characters required to display the value, the output is right justified and padded with spaces (or zeros) to the width specified. Keep in mind, however, that browsers typically collapse multiple spaces into a single space for display, so only one space character will typically be displayed before or after the numeric string.
–	%u, %lu, %d, %ld, %F	If the output number requires fewer characters than specified by the "n" parameter, adding the minus sign left aligns the string, adding space characters after the number to fill the field to the specified length. If the minus sign is omitted, space characters are added before the output to fill the field to the specified length.
0	%u, %lu, %d, %ld, %x, %lx, %F	If the output number requires fewer characters than specified by the "n" parameter, the zero parameter specifies that the number should be padded with leading zeros to the length specified by "n". The 0 parameter and the minus sign (–) are mutually exclusive.
d	%F	Specifies the number of digits to be displayed to the right of the decimal point. Numeric values in the Barionet are always stored as integers, but the %f format scales the value by inserting a decimal point "d" digits from the left.

The table below shows several examples of format strings and the output they generate.

Barix Dynamic Tag	Example Output	Description
&LIO(1,"%lu",601);	3192	The output of the first temperature sensor is displayed as an unsigned integer.
&LIO(1,"%.1f",601);	319.2	The same raw integer value from the temperature sensor is displayed in floating point format with one digit to the right of the decimal.
&LIO(1,"%lx",601);	C78	The temperature sensor value is displayed in hexadecimal.
&LIO(1,"%05lx",601);	00C78	The hexadecimal value is displayed with leading zeros in a field width of 5 characters
&LIO(1,"%8lx",601);	C78	The hexadecimal value is displayed in a field width of 8 characters with 5 spaces padded on the left. Note that web browsers will typically collapse the 5 spaces and only show a single space to preceding the numeric value.
&LBAS(1,"%fs",S$);	Hello, world	The BCL string variable, S$, which contains "Hello, world" is displayed.
&LBAS(1,"%c",Var1);	A	The BCL numeric variable Var1, which contains the value 65 is displayed as an ASCII character "A".
&LBAS(1,"%V",Var1);	B1.03	The Barionet firmware version is displayed. The Var1 argument is required, but ignored.

5.2.2 Password Protecting Custom HTML Pages

The Barionet provides a password protection feature that can optionally require a password to access the built-in configuration web pages. You can use this same feature to password protect custom HTML web pages you create as well.

There are two steps required to add password protection to your custom web pages:

1. Set the optional security parameter in the Barix dynamic initialization tag to a value of 2. The tag must appear in the first 512 bytes of your custom page. See the description of the Barix Initialization Tag earlier in this chapter for more information.

2. Use the Security page in the pre-loaded configuration web pages to set, clear, or reset the password. If you want to create your own page to manage the password, copy the HTML from the pre-loaded page in the file **uisecurity.html** and modify it as required for your application. The **uisecurity.html** file can be found in the **webuidevkit/bario50web** folder in the Barionet 50 update, rescue and development kit and the **webuidevkit/barionetweb** folder in the Barionet 100 update, rescue and development kit. These update and rescue kits are available for download on the Barix web site.

5.3 Creating Custom BCL Applications

In many cases, a few HTML web pages with Barix dynamic tags will be all that's required to customize the Barionet to a specific application. However, if your application requires unique input or output processing, you need to control one of the Barix extension modules, or you need to implement more complex functions such as sending email on a specific condition of inputs, you'll probably need to create a custom BCL program. BCL gives you ultimate flexibility to create an application that does precisely what you need.

Fortunately, creating BCL applications is a simple process if you're familiar with most any other programming language. BCL is very similar to the BASIC language with specific extensions for monitoring and controlling the Barionet's inputs and outputs. Refer to the BCL Programmer's Manual for more details on the BCL Language.

This section describes the process of creating a BCL application, packaging the application with other required files and uploading the application to the Barionet.

5.3.1 Creating and Editing a BCL File

BCL programs can be created and edited with any text editor, as long as the editor can save the BCL text in plain text files. On Windows, the standard Notepad editor application will work fine for creating BCL files. However, a variety of other free editors are available that offer more flexible searching and some even recognize language syntax and highlight the source appropriately. The free Notepad++ editor (http://notepad-plus.sourceforge.net/uk/site.htm) can be set to recognize BASIC syntax, which is very similar to BCL.

BCL source program files typically have a .bas file extension, though you can also create special "include" files that have segments of BCL code or macros that will be included in the main BCL source file. These include files must have a .bcl extension. Refer to the BCL Programmers manual for more information on using include files and macros.

5.3.2 The Tokenizer

Before a new BCL application can be loaded into the Barionet, it must be converted into a "token" file. The token file is more compact and efficient for the Barionet's BCL interpreter to execute. The token file is created with a "tokenizer" program supplied as part of the Barionet development kit. You can download the development kit from the download section of the Barix web site at: http://www.barix.com/downloads/Barionet_Family/51/

The Tokenizer is supplied in three forms:

1. A windows command-line executable program called **tokenizer.exe**.

2. A Macintosh OS-X command-line executable called **tokenizer-mac-x86.**

3. A Linux command-line executable called **tokenizer-static-linux-x86.**

Note

Some earlier versions of the tokenizer program were named "tokenize" (without the trailing "r"). They are the same program. However, be sure to use a version of tokenizer that is compatible with the firmware loaded in your Barionet. Check the readme file included with the development kit for details on the correct version or use the version of the tool supplied with the development kit.

The tokenizer is a command-line tool, so it needs to be executed in a command line window. In Windows, you can open a command line window by clicking "Start" and then "Run" and typing "cmd". Press enter to open the command window.

In the Macintosh, run the terminal program which is normally located in the Utilities folder.

tokenizer <target> <source file>

<**target**>	This is the type of device that the tokenizer is creating the TOK file for. The tokenizer can produce TOK files for a variety of different Barix devices. For our purposes, <target> should be either “barionet100” or “barionet50”. The quotes are not part of the syntax.
<**source file**>	This parameter specifies the .BAS file that is to be processed by the tokenizer.

For example, to tokenize a file called “myapp.bas” for a Barionet 100, the syntax would be:

tokenizer barionet100 myapp.bas

The tokenizer program performs some syntax checking on the BCL source file and issues error and warning messages. Warning messages don't prevent the tokenize process from completing, but the warnings may point out errors that may create subtle problems later, such as misspelled variable names. Syntax errors will cause the tokenizer to stop without generating an output file.

These preliminary syntax checks won't catch all types of errors. Additional errors will be caught and reported by the BCL interpreter via syslog when the program is loaded and executed on the Barionet. See Using Syslog Messages in Chapter 7 for more information on syslog output.

The output of the tokenizer is a token file with the same base name as the source file but with a .TOK extension. The tokenizer will also create a special error message file called ERRORS.HLP if it doesn't find this file in the directory where the source file and .TOK file are located. The ERRORS.HLP file is used by the BCL interpreter to report errors in syslog with English readable messages. Thus, the ERRORS.HLP file should be included with the .TOK file for packaging and uploading to the Barionet. If you want error messages in syslog to also include the line of source where the error was detected, the source .BAS file must also be included in the package. We'll discuss creating the package .COB file in the next section.

5.4 Loading Custom Web Pages and BCL Applications

5.4.1 Creating a COB Package File

Before you can upload your tokenized BCL application or HTML pages, they need to be packaged in a special file called a COB file. The program used to create this package file is supplied as part of the development kit which can be downloaded from the Barix web site at: http://www.barix.com/downloads/Barionet_Family/51/.

This program is provided in three forms:

1. A windows command-line executable program called **web2cob.exe**
2. A Macintosh OS-X command-line executable called **bpkg-mac-x86**

3. A Linux command-line executable called **bpkg-static-linux-x86**

This packaging program assembles your HTML page files, tokenized BCL program (the .TOK file), the ERRORS.HLP, and optionally the .BAS source file, into a single COB file that can be uploaded to the Barionet.

The command line syntax for all versions of the program is the same except for the program name itself:

web2cob [/o <output file>] [/d <input directory>]

For the other operating systems, simply substitute the appropriate program name for web2cob in the above syntax.

<**output file**>	specifies the name of the output file. The default output file is called cobox.cob. You can name the output file anything you like, but the output extension should be .cob.
<**input directory**>	specifies the directory that web2cob will package into the COB file. Everything in this directory is packaged into the COB file, so be sure you specify a directory that has only the files you want to include in the package. The default (if the /d switch is not specified) is the current directory.
/v	This option tells the bpkg program to output additional details on its processing and statistics for the output file. This option applies to bpkg (Mac and Linux versions) only.
/n	This option tells the bpkg program to omit HTTP headers on any files included in the COB file that are known MIME types. This option applies to bpkg (Mac and Linux versions) only.

In Windows, you can open a command line window to run web2cob by clicking "Start" and then "Run" and typing "cmd". Press enter to open the command window.

In the Macintosh, run the terminal program which is normally located in the Utilities folder.

The output of the program is a COB file that can be uploaded to the Barionet.

Note

COB files are limited in size only by the available memory in the Barionet. However, no single file contained with a COB file can be larger than 64K bytes.

5.4.2 Uploading the COB File

The completed COB file can be uploaded to the Barionet using the TFTP (Trivial File Transfer Protocol), which is a variant of the standard file transfer protocol (FTP). Windows includes a built-in TFTP client. A TFTP client is available for Linux platforms at: http://freshmeat.net/projects/atftp/

The tftp client on the Windows platform uses the following syntax:

tftp [-i] <host> [GET | PUT] <source> [<destination>]

-i	This option specifies that the file should be transferred in binary mode. This option must be included when transferring COB files to the Barionet.
<**host**>	The IP address of the Barionet (e.g. 192.168.0.40).
PUT	Since you are transferring a COB file to the Barionet, specify the PUT function in the tftp command line. The Barionet does not support the GET function. You cannot download files from the Barionet using tftp.
<**source**>	The name of the COB file you want to transfer to the Barionet/Barionet 50.

<**destination**> The memory location in the Barionet where the COB file should be loaded. We'll discuss the memory layout and where to load a new COB file in the next section. This parameter is required by the Barionet.

The atftp client on Linux uses the following syntax:

atftp –l <source> -r <destination> -p <host>

-l <**source**> Specifies the local source file (the .COB file) that you want to transfer.

-r <**destination**> Specifies the destination for the file on the Barionet. This is a memory location name. We'll discuss the memory layout and where to load the COB file in the next section.

-p Indicates that this is a "PUT" operation, transferring the COB file from the local computer to the Barionet.

<**host**> The IP address of the Barionet (e.g. 192.168.0.40).

Note

The Barionet 50 only accepts tftp file transfers when it is in the boot loader mode. This prevents accidental or malicious updates to the device. To enter the boot loader mode, click the "Update" menu function in the Configuration web page. Then click the "Please Click here to continue" link near the bottom of the page. Note that the Barionet 50 must be re-booted to exit the boot loader mode.

Barionet 100 Only

3 The Barionet Memory Map

The Barionet 100 and Barionet 50 each have non-volatile flash memory that stores the firmware as well as the standard web configuration pages. The flash memory is divided into 64K byte pages that are assigned names, which are the "destination" parameter in the tftp file transfer.

The Barionet 100 has a total of 512K of non-volatile memory divided into eight 64kB pages.

The following table shows the memory map for the Barionet 100.

Name	Size	Usage
X1	64kB	Barionet firmware
WEB1	64kB	Web UI application, Help, firmware
WEB2	64kB	Web UI application, Help
WEB3	64kB	Reserved for Web UI application extensions
WEB4	64kB	Sample BCL Digital I/O tunnel application.
WEB5	64kB	Free
WEB6	64kB	Free
WEB7	64kB	Free

Your custom pages and BCL applications can be loaded into any location from WEB4 through WEB7. Loading a COB file into WEB4 overwrites the Sample BCL application, but the application is included in the development kit and can be re-loaded if you want it.

Barionet 50 Only

The Barionet 50 has considerably more memory for user pages and applications. The following table shows the memory layout for the Barionet 50.

Name	Size	File Name	Usage
8K	64kB	Bn50.rom	Barionet firmware
WEB1	64kB	Bario50web.cob	Web UI application, Help
WEB2	64kB		Web UI application, Help
WEB3	64kB		Reserved for Web UI application extensions
WEB4	64kB	Barionetbcl.cob	Sample BCL Digital I/O tunnel application.
WEB5	64kB		Free
…	1152kB total free space (WEB5 – WEB 23)		
WEB23	64kB		Free
WEB24	64kB		Reserved for future use
…	Reserved for future use		
WEB28	64kB		Reserved for future use
WEB28	64kB	Sg.bin	Firmware extension
WEB30	64kB	Bclio.bin	I/O drivers
Boot loader			

In the Barionet 50, you can upload new COB files to any location from WEB5 through WEB24. You can even use WEB4 if necessary if you do not need the Digital I/O tunnel application.

You can upload multiple COB files to the Barionet 100 or Barionet 50, but each COB file must occupy at least one WEB memory location. If the COB file is bigger than 64K it will occupy more than one location. You cannot load more than one COB file into a single memory location

Note.

If you load an COB file that contains an HTML page or a .TOK program file with the same name as one of the existing pages (e.g. index.html), your custom page or program most likely won't be accessible, since the web server searches for file names staring in low memory and working to higher memory (from the top of the memory map table toward the bottom). If you name your custom page "start.html", the web server will open that page by default, instead of the normal index.html default page.

5.5 Development Tools and Scripts Summary

We've already discussed two of the programs (the tokenizer and web2cob) used in the development process. Barix also supplies a variety of other tools that help automate the various steps of the development process as well as aiding in managing Barionet memory and the serial rescue procedure, which is discussed more in the next chapter on Troubleshooting.

The following tables summarize the various tools and scripts and their uses for each of the three operating systems supported (Windows, Macintosh OS-X and Linux).

Note that the batch and script files make some assumptions about the location of the executable files (e.g. tokenizer, web2cob, etc..) If these files are not in the locations referenced in the batch/shell script files, either they need to be copied to those locations or the batch/shell script files need to be modified.

5.5.1 Windows Tools and Batch Files

File	Applies To	Development Kit Folder	Type	Description
tokenizer.exe	Barionet 50/100	\tools	Binary executable	Converts a BCL source file to a token file (TOK)
web2cob.exe	Barionet 50/100	\tools	Binary executable	Packages TOK file(s), HTML files, and other files into a COB file for uploading to the Barionet using tftp.
bcl.bat	Barionet 50/100	\bcldevkit	Batch file	Runs the tokenizer and web2cob (Bpkg) to create the factory default COB file and uploads the resulting COB file to the barionet
erasebcl.bat	Barionet 50/100	\bcldevkit	Batch file	Erases the WEB4 page in the Barionet by uploading the empty.cob file.
erase.bat	Barionet 100	\bcldevkit	Batch file	Erases all WEB pages from WEB1 – WEB7
erasefree.bat	Barionet 50	\bcldevkit	Batch file	Erases all WEB pages from WEB1 – WEB23
sleep.exe	Barionet 100	\tools	Binary executable	Used to insert delays required in the erase.bat and erasebcl.bat files between erase operations.
web.bat	Barionet 50/100	\webuidevkit	Batch file	Runs web2cob to create a COB file with the factory default web pages, and uploads the COB file to the Barionet using tftp. Similar to bcl.bat, but does not run the tokenizer first.
barionet.bat	Barionet 100	\update_rescue	Batch file	Updates or reloads the Barionet 100 by loading the factory default COB files.
gen.bat	Barionet 50	\update_rescue	Batch file	Calls load_win.exe to create the compound.bin file required to update or reload the Barionet 50 via the web interface.
load_win.exe	Barionet 50/100	\update_rescue	Binary executable	Creates the compound.bin file required to update or reload the Barionet . It can also load firmware into the Barionet via a serial port in the serial rescue process. See The Serial Rescue Procedure in Chapter 7 for more information.
serial.bat	Barionet 50	\update_rescue	Batch file	Manages the Barionet 50 serial rescue process using load_win.exe. See The Serial Rescue Procedure Chapter 7 for more information.
rescue.bat	Barionet 100 (EX versions manufactured 2006 or later)	\update_rescue	Batch file	Manages the serial rescue process for later (EX) versions of the Barionet 100 . See The Serial Rescue Procedure Chapter 7 for more information.
lxrescue.bat	Barionet 100 (LX versions manufactured 2005 and earlier)	\update_rescue	Batch file	Manages the serial rescue process for earlier (LX) versions of the Barionet 100 . See The Serial Rescue Procedure Chapter 7 for more information.

5.5.2 Macintosh OS-X Tools and Scripts

File	Applies To	Development Kit Folder	Type	Description
tokenizer-mac-x86	Barionet 50/100	/tools	Binary executable	Converts a BCL source file to a token file (TOK)
bpkg-mac-x86	Barionet 50/100	/tools	Binary executable	Packages TOK file(s), HTML files, and other files into a COB file for uploading to the Barionet using tftp.
bcl.sh	Barionet 50/100	/bcldevkit	shell script	Runs the tokenizer and bpkg to create the factory default COB file and uploads the resulting COB file to the barionet
erasebcl.sh	Barionet 50/100	/bcldevkit	Shell script	Erases the WEB4 memory page in the Barionet by uploading the empty.cob file.
erase.sh	Barionet 100	/bcldevkit	Shell script	Erases all WEB memory pages from WEB1 – WEB7
erasefree.sh	Barionet 50	/bcldevkit	Shell script	Erases all WEB memory pages from WEB1 – WEB23
cpcrlf.sh	Barionet 50/100	/tools	Shell script	Copies a file, converting Linux/Macintosh line endings (LF only) to Windows/DOS line endings (CR/LF), as necessary
crlf-mac-x86	Barionet 50/100	/tools	Binary executable	Used by cpcrlf to handle line ending changes
web.sh	Barionet 50/100	/webuidevkit	Shell script	Runs bpkg to create a COB file with the factory default web pages, and uploads the COB file to the Barionet using atftp. Similar to bcl.sh, but does not run the tokenizer first.
Barionet.sh	Barionet 100	/update_rescue /linux_mac	Shell script	Updates or reloads the Barionet by re-loading the factory default COB files.
gen.sh	Barionet 50	/update_rescue /linux_mac	Shell script	Creates the compound.bin file required to update or reload the Barionet 50 via the web interface.
load_mac	Barionet 50/100	/update_rescue /linux_mac	Binary executable	Used in the serial rescue process to load new firmware into the Barionet. See The Serial Rescue Procedure in Chapter 7 for more information on the serial rescue procedure.
seriald.sh	Barionet 50	/update_rescue /linux_mac	Shell script	Manages the Barionet 50 serial rescue process. See The Serial Rescue Procedure Chapter 7 for more information.
rescue.sh	Barionet 100 (EX versions manufactured 2006 or later)	/update_rescue /linux_mac	Shell script	Manages the serial rescue process for later (EX) versions of the Barionet 100 . See The Serial Rescue Procedure Chapter 7 for more information.
lxrescue.sh	Barionet 100 (LX versions manufactured 2005 and earlier)	/update_rescue /linux_mac	Shell script	Manages the serial rescue process for earlier (LX) versions of the Barionet 100 . See The Serial Rescue Procedure Chapter 7 for more information.
atftp-mac-emul.sh	Barionet 50/100	/tools	Shell script	A shell script that emulates the Linux atftp program using the Mac's native tftp program.

5.5.3 Linux Tools and Scripts

File	Applies To	Development Kit Folder	Type	Description
tokenizer-linux-static-x86	Barionet 50/100	/tools	Binary executable	Converts a BCL source file to a token file (TOK)
bpkg-static-linux-x86	Barionet 50/100	/tools	Binary executable	Packages TOK file(s), HTML files, and other files into a COB file for uploading to the Barionet using atftp.
bcl.sh	Barionet 50/100	/bcldevkit	shell script	Runs the tokenizer and bpkg to create the factory default COB file and uploads the resulting COB file to the barionet
erasebcl.sh	Barionet 50/100	/bcldevkit	Shell script	Erases the WEB4 memory page in the Barionet by uploading the empty.cob file.
erase.sh	Barionet 100	/bcldevkit	Shell script	Erases all WEB memory pages from WEB1 – WEB7
erasefree.sh	Barionet 50	/bcldevkit	Shell script	Erases all WEB memory pages from WEB1 – WEB23
cpcrlf.sh	Barionet 50/100	/tools	Shell script	Copies a file, converting Linux/Macintosh line endings (LF only) to Windows/DOS line endings (CR/LF), as necessary
crlf-static-linux-x86	Barionet 50/100	/tools	Binary executable	Used by cpcrlf to handle line ending changes
web.sh	Barionet 50/100	/webuidevkit	Shell script	Runs bpkg to create a COB file with the factory default web pages, and uploads the COB file to the Barionet using atftp. Similar to bcl.sh, but does not run the tokenizer first.
Barionet.sh	Barionet 100	/update_rescue /linux_mac	Shell script	Updates or reloads the Barionet by re-loading the factory default COB files.
gen.sh	Barionet 50	/update_rescue /linux_mac	Shell script	Creates the compound.bin file required to update or reload the Barionet 50 via the web interface.
load_lin	Barionet 50/100	/update_rescue /linux_mac	Binary executable	Used in the serial rescue process to load new firmware into the Barionet. See The Serial Rescue Procedure in Chapter 7 for more information on the serial rescue procedure.
seriald.sh	Barionet 50	/update_rescue /linux_mac	Shell script	Manages the Barionet 50 serial rescue process. See The Serial Rescue Procedure Chapter 7 for more information.
rescue.sh	Barionet 100 (EX versions manufactured 2006 or later)	/update_rescue /linux_mac	Shell script	Manages the serial rescue process for later (EX) versions of the Barionet 100 . See The Serial Rescue Procedure Chapter 7 for more information.
lxrescue.sh	Barionet 100 (LX versions manufactured 2005 and earlier)	/update_rescue /linux_mac	Shell script	Manages the serial rescue process for earlier (LX) versions of the Barionet 100 . See The Serial Rescue Procedure Chapter 7 for more information.

6 The Sample Digital I/O and Serial Tunnel Application

The Barionet comes pre-loaded with a sample BCL application that allows two Barionet 100's or Barionet 50's to "tunnel" one Barionet's inputs to another Barionet's outputs using UDP and to bi-directionally tunnel one Barionet's serial data across the network to/from a second Barionet's serial port using TCP. The I/O tunnel application also supports connecting an external Barix R6 relay expansion module to one or both Barionets via the RS-485 port.

You can configure the application so that when an input on one Barionet goes active, an output or relay on the other Barionet goes active as well. The association between inputs on one Barionet and outputs on the opposite Barionet is completely configurable via an application setup screen.

6.1 The Application Setup Screen

The sample Digital I/O tunnel application includes an "Application Setup" page that is linked to the Barionet's main configuration user interface page. You can access this page by clicking the Configuration button on the Barionet's main home page and then clicking "Application Setup" in the top menu on the Configuration page. Figure 32 shows the application setup screen.

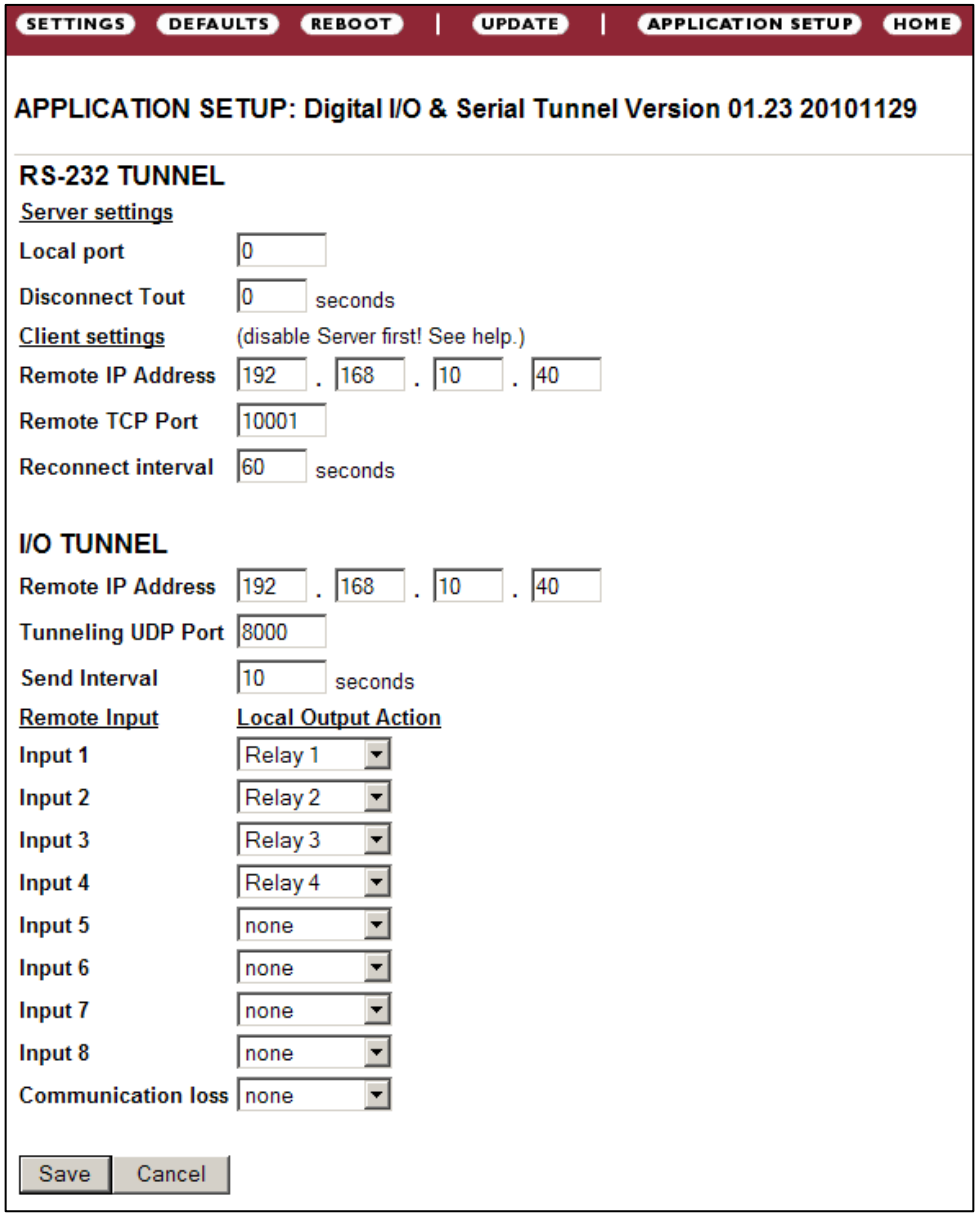


Figure 32. The Sample BCL Digital I/O and Serial Tunnel Application Setup Screen.

6.1.1 Setting up the Serial Tunnel

To use the serial tunnel function, one Barionet must be setup as the server and a second must be setup as a client. The Barionets do not need to be the same model, but they do need to have compatible firmware and application versions. See the README file in the \bcldevkit\serial_io_tunnel folder (Barionet 50) or \bcldevkit\barionet folder (Barionet 100) for more details.

Data is transferred bi-directionally between the two Barionets, so it's not important which end of the interface is the client and which is the server unless there are firewalls between the Barionets. The client Barionet is responsible for establishing the TCP connection with the remote (server) Barionet.

The serial settings for both RS-232 ports must be set in the Serial Settings configuration web page. The serial speed settings for both Barionets should be the same to avoid data overrun problems.

6.1.1.1 Setting up the Client Barionet

The Barionet that will act as a client must be disabled as a server by setting the "Local Port" setting to zero. Then the IP address and the local port setting on the server Barionet is entered in the Remote IP address and Remote TCP Port settings.

The Reconnect interval parameter set the time in seconds between attempts to reconnect to the remote Barionet when the TCP connection is lost.

6.1.1.2 Setting up the Server Barionet

The Local Port should be set for the Barionet that will act as the server. The default setting is 10001. The Disconnect Tout parameter sets an inactivity timeout in seconds. If no data is sent or received across the connection for the specified interval, the server closes the TCP connection. The default value for this parameter is zero (no timeout).

6.1.2 Setting up the I/O Tunnel

6.1.2.1 Setting the Remote IP Address

The remote IP address specifies the IP address of the remote Barionet 100 or Barionet 50. You should set the IP address of the opposite Barionet in this page on both Barionets.

6.1.2.2 Setting the Tunneling UDP Port

The tunneling application uses UDP to transmit data about input and output changes. Choose a UDP port number for this communication. The default value of zero disables the application. The port number must be the same on both Barionets.

6.1.2.3 Setting the Send Interval

The application sends status change messages immediately whenever an input changes. However, it can also optionally send a status message at specified intervals even when no inputs change. This status message helps detect communication loss between the two Barionets. You can configure an output or relay to close in the event of communication loss.

The Send Interval parameter should be the same on both Barionets. Leaving the parameter set to zero disables the periodic status messages, though state change messages are always sent immediately when an input changes. This value can range from 1 to 65535. A smaller Send Interval allows the remote Barionet to detect a communication loss more quickly. A communication loss event will be triggered at double the send interval plus one second. Thus, if the send interval is set for 5 seconds, a communication loss will be detected at 11 seconds after the last status message.

6.1.2.4 Setting the Output Action

For each input on the remote Barionet, you can choose an output action on the local Barionet. Since the application supports both the Barionet 100 and Barionet 50, it shows inputs and relays that may not exist on either model. The Barionet 100 Only has two relay outputs, while the Barionet 50 has four relay outputs. The Barionet 100 has eight inputs, while the Barionet 50 has only four inputs. Nonexistent inputs (i.e. inputs 5 – 8 on a Barionet 50) will always be inactive for the purposes of this application.

The "Ext. Relay" outputs assume that a Barix R6 expansion module is connected to the local Barionet via RS-485.

The Communication loss option allows you to specify an output or relay that should be turned on when a communication loss is detected.

6.2 The Sample Application Source Code

The source code for the Digital I/O tunnel application is included in the Barionet Development kit in the bcldevkit\barionet folder (Barionet 100) or bcldevkit\serial_io_tunnel folder (Barionet 50). The source code includes the BCL application code (barionet.bas) as well as the application setup HTML page. These source files are useful for re-loading the application if it is overwritten. In addition, the application provides helpful examples of many of the concepts required for using the Barix dynamic tags and BCL to create custom applications.

7 Troubleshooting

If you're having trouble communicating with the Barionet, or encountering problems developing and debugging applications for the Barionet, this chapter provides a variety of procedures and resources to help you identify and correct issues.

7.1 Common Problems and Solutions

The following table lists a number of common problems with possible causes and solutions.

Symptom	Possible Cause and Solutions
Can't access Barionet Web Pages	1. Be sure you're using the correct IP address and that the Barionet is properly connected to the network. See Accessing the Barionet for the first time in Chapter 2. 2. Invalid configuration settings have made the Barionet unreachable. Try restoring the default settings. See Resetting to Factory Defaults later in this chapter. 3. The default configuration web pages may have been over-written by another application or become inaccessible due to another index.html file being loaded in a lower memory page. Try reloading the firmware and standard configuration pages. Refer to The Serial Rescue Procedure later in this chapter. 4. Check the syslog for error messages. See Using Syslog Messages in this chapter.
The Discovery Tool doesn't find the Barionet	1. Be sure the Barionet is powered up and running. Is the Green Power or Status light on? 2. The Barionet must be on the same physical subnet. The Discovery tool will not find a Barionet that is separated from the computer running the Discovery tool by a router. 3. If the Barionet has an older version of firmware (prior to V1.04 for the Barionet 50 and V2.30 for the Barionet 100), the Discovery tool will only find the Barionet if it's IP address is in the same "logical" subnet (i.e. IP address range) of the computer running the Discovery Tool. See Appendix G: IP Addresses, Netmasks and Gateways for more information.
The Discovery Tool doesn't run.	1. A Java run-time engine must be installed on the computer. See http://java.com/en/download/manual.jsp for downloading and installing the Java run-time. 2. If you are running Linux or Unix, the Discovery Tool requires X-windows, since it creates a graphical user interface. You cannot run the Discovery tool from a command line.
Dynamic HTML tags appear in the web output instead of the I/O or variable values (e.g. &LIO... instead of the state of the input or output)	1. Check that the Barix dynamic initialization tag is included before any other Barix dynamic tags and in the first 512 bytes of the web page. See the description of the Initialization Tag in Chapter 5. 2. If output from some of the tags on a page is properly displayed while others are not, the problem is most likely a syntax error in the tags that are not properly displayed. Check for proper spelling and arguments. Also make sure that all tags have the closing semicolon. See the section on Barix Dynamic Tags in Chapter 5.
Dynamic tags display the contents of the format string (e.g. %zz) instead of the correct value.	1. If the format string is invalid, the contents of the format string will be displayed instead of the value. Check the syntax of the format string carefully. See the discussion of Format Strings in Chapter 5.

Symptom	Possible Cause and Solutions
&LBAS tags display [NO_VAR] instead of the value of a variable	1. The variable referenced in the &LBAS tag is not defined in the currently executing BCL program. Check the syntax of the tag and the spelling of the variable name. See BCL Variable Tag (&LBAS) in Chapter 5 2. The program you loaded is not executing. If the sample digital I/O tunnel application is loaded and the BCL Program Name parameter in the Control section of the configuration pages is not set, the sample application is probably running, instead of your BCL application. 3. The program has a fatal error that is causing the BCL interpreter to abort and re-start the program. Check the syslog output for error messages. See Using Syslog Messages in this chapter.
Web server response is very slow or doesn't respond.	1. The program has a fatal error that is causing the BCL interpreter to abort and re-start the BCL interpreter. Check the syslog output for error messages. See Using Syslog Messages in this chapter. 2. The BCL application or HTML page is calling an ON CGI subroutine that takes a long time to return. During the processing of ON CGI routines, the web server is blocked, so these routines should be kept as short and fast as possible.
Communication with a Barix expansion module connected to the RS-485 interface fails	1. Check that the interface is properly connected. Swapping the RS-485 A and B signals is a common problem. If the RS-485 indicator light remains constantly lit, this is likely the problem. 2. Check the RS-485 serial port settings (baud rate, data bits, stop bits and parity). Note that the settings in the BCL "Open" statement override the configuration port settings. Because the IO Tunnel application sets the parameters using a BCL Open statement, configuration parameters are ignored in the IO tunnel application. 3. Check the BCL code for proper message structure and checksum values. Example BCL code for communicating with a Barix Modbus expansion module is available for download on the Barix web site.
The Web Server returns a PAGE NOT FOUND error.	1. Be sure that the page you expect is loaded in a COB file in memory. Check the file names and re-build and load the COB file.
The wrong page is returned when you request a page from the Barionet web server.	1. First, try re-loading the page in your browser without using the browser's cache. In Internet Explorer, holding down the Ctrl key while pressing the Reload button (or F5) forces the browser to retrieve the page from the web server and not simply re-display a cached version of the page. With Safari, hold down shift and click reload on the toolbar. For other browsers, check the documentation. 2. If the wrong page still appears, it's possible that there is more than one file loaded in the Barionet memory with the same name. The Barionet web server starts in low memory (e.g. WEB1) looking for pages, so if another page with the same name is loaded in lower memory than your page, the other page will be displayed instead of yours. Either rename your page and re-load it, or remove the other page loaded in lower memory.

7.2 Using Syslog Messages

One of the most important tools for troubleshooting the Barionet and debugging BCL programs is the syslog function. The Barionet sends internal status information and user-generated syslog messages to a standard syslog viewer program installed on a computer that is accessible via the network from the Barionet. The Barionet sends syslog messages via UDP on standard port 514.

You can use any syslog viewer program to see the syslog messages. For Windows, Barix recommends the Kiwi Syslog Viewer program (www.kiwisyslog.com). You can download a free fully functional 30-day trial version of the syslog viewer program that will remain

functional (with some advanced features disabled) even after the 30-day trial period. This free version has the basic functionality required for troubleshooting the Barionet and BCL programs. Similar programs are available for Macintosh and Linux platforms.

7.2.1 Barionet Internal Syslog Messages

When the Barionet starts up, it normally sends at least one syslog message. The first message indicates that the BCL interpreter has started up normally. If the latest version of the standard digital I/O tunnel application is loaded and setup to run (i.e. another BCL program name is not entered in the BCL Program Name setting in the configuration web page), a message from the I/O tunnel application is also displayed at start up. Figure 33 shows the syslog display (using the Kiwi syslog program described later in this section).

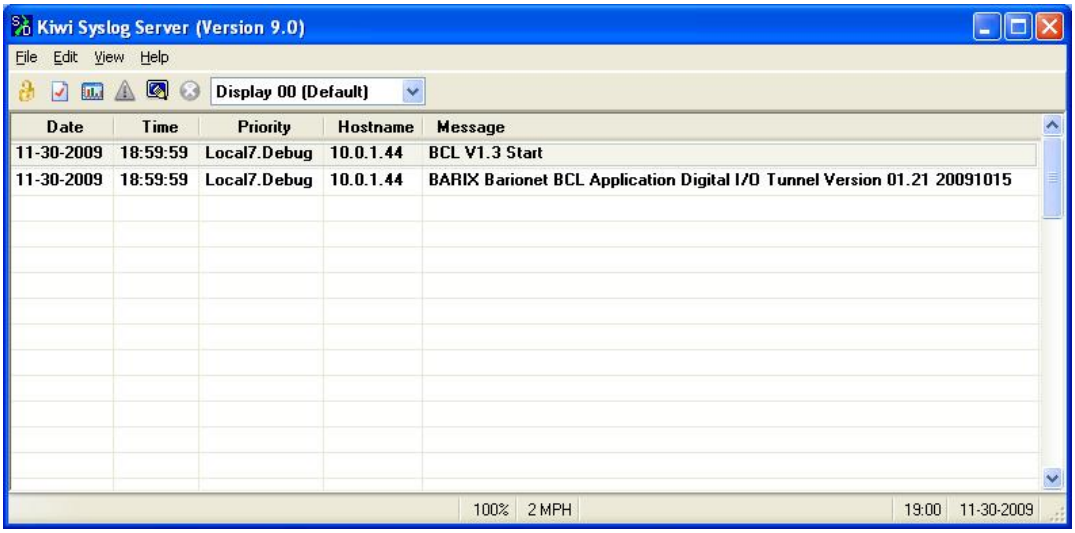


Figure 33. Two syslog messages are normally sent at start up.

The number and detail of syslog messages displayed is controlled by the Debug Level setting in the Configuration web pages. The messages shown in Figure 33 are generated with the debug level setting of 1 or higher. Setting the debug level to a higher value generates more detailed output and potentially many more messages.

Note

Note that the Hostname column shows the IP address of the Barionet, which is also useful in determining the IP address of the Barionet if it uses DHCP to obtain a dynamic IP address, or if the static IP address of the Barionet is unknown.

Most syslog applications, including the Kiwi syslog viewer, also create a log file in addition to the display. The log file is useful for reviewing syslog messages, since some of the messages may scroll out of the view window if many messages appear quickly.

7.2.2 BCL Error Syslog Messages

The BCL interpreter also generates syslog messages when it encounters a problem in executing BCL code. Some messages are simply warnings that do not terminate execution of the program. For example, if a BCL program attempts to assign a string value to a variable that is not dimensioned to a large enough size to hold the string, the BCL interpreter will issue a warning message to syslog, but the string will be truncated and assigned to the variable and execution will continue.

Some BCL errors are more serious and will cause the program execution to abort. In these cases, the BCL interpreter will send the fatal error message to syslog and then the BCL interpreter will re-start, beginning execution of the program again. This can result in a continuous loop of the Barionet encountering the fatal error, issuing the error message in syslog, and re-starting the interpreter. This loop can, under some circumstances, cause the Barionet to respond very slowly or even not at all to web requests. If the Barionet does not seem to be responding as it should, checking the syslog should be one of your first steps.

7.2.3 User-generated Syslog Messages

In addition to the built-in messages from the BCL interpreter, you can insert syslog statements in your own BCL code that will generate messages you define.

The syntax of the BCL syslog statement is:

syslog "<message text>"[,<debug level>]

<**message text**> is any BCL string expression, which could include the current state of numeric or string variables. Numeric variables must be converted to strings using the **str$()** function and concatenated to the string.

<**debug level**> is an optional parameter from 1 – 9, representing a debug level. If the current debug level is equal to or higher than the level specified in this parameter, the syslog message is sent. If the debug level is set lower than this value, the debug message is not sent. (Setting the debug level to zero in the web configuration page disables all debug output. The default level is 3).

For example, here's a syslog statement that sends the value of two variables (Var1 and Var2) to the syslog if the current debug level is set to 5 or higher:

syslog "Var1=" + str$(Var1) + " Var2=" + str$(Var2),5

If the current debug level is set for 5 or higher, the syslog output text will look something like this:

Var1=29 Var2=17

Including the optional debug level parameter in your syslog statements makes turning various syslog messages on and off convenient. You can set the debug level parameter very low for serious errors or other information that you want to always send to syslog. Other less important syslog statements can use a higher debug level parameter so that the debug level has to be set higher for these messages to appear in the log.

For more information on syslog and other BCL statements, refer to the BCL Programmer's Manual, available for download from the Barix web site at: http://www.barix.com/downloads/Barionet_Family/51/.

7.3 Rebooting the Barionet

If the Barionet has stopped responding and you've checked the syslog output, you can try rebooting the Barionet while watching the syslog output (assuming that the syslog parameters were set in the configuration web pages). There are three methods of rebooting. You can use the preloaded configuration web pages, you can use the setup.cgi command, and the Barionet 50 can be rebooted using a hardware jumper. The hardware jumper method only applies to the Barionet 50.

7.3.1 Rebooting from the Configuration Web Page

If the standard configuration web pages are still loaded in the Barionet and the web server is responding, you can reboot the device by clicking the "Configuration" button on the status home page. (See Figure 7 for the Barionet 100 and Figure 8 for the Barionet 50) From the settings page that appears, click the "Reboot" button near the top of the page and click the "Reboot" link on that page.

After approximately 5 seconds, a message saying "Device has successfully rebooted" should appear.

If the message does not appear, or you are unable to access the web configuration pages, you can also reboot using an internal hardware jumper (Barionet 50 only), or you can use the Serial Rescue procedure described later in this chapter to re-load the Barionet firmware and reset to factory defaults.

7.3.2 Rebooting using the Hardware Jumper (J9)

Barionet 50 Only

The Barionet 50 also has a hardware jumper under the snap-on case lid that can be used to both reboot the device and, optionally, reset it to factory default parameters. You will need a standard 5 mm jumper to short the pins of this reset jumper together.

Follow these steps to reboot the device using the reset jumper:

1. Disconnect the Barionet 50 from its power supply.
2. Remove the snap-on cover by inserting a small flat-blade screwdriver into one of the two latches on either end of the cover and gently prying the cover up. Then release the second latch on the same side and lift the lid off.
3. The reset pins (J9) are located under the cover just above RS-232 LED. (See Figure 34).

Note

Some Barionet 50 units were manufactured without the pins installed. In this case, the two pads where the pins go can be shorted with a paperclip or a short piece of wire.

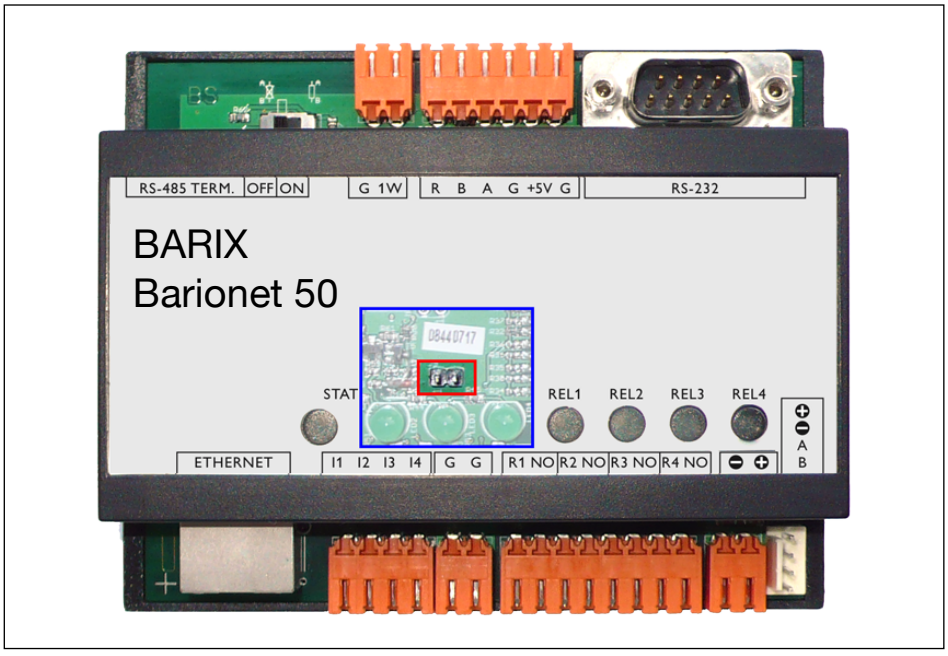


Figure 34. The Barionet 50 Reset jumper is under the lid near the RS-232 LED

4. Connect the Barionet 50's power supply and wait for the green status LED (the left most LED) to come on. Install the jumper on the reset pins or short the pads with a short piece of wire or paper clip.
5. Remove the shorting jumper within the first 3 – 4 seconds (if the jumper is not removed within approximately 10 seconds, the Barionet 50 will also reset all settings to factory

defaults.

7.4 Resetting to Factory Defaults

There are three ways to reset the Barionet to factory default settings. The first method uses the Configuration Web pages and is the same for the Barionet 100 and Barionet 50. The second reset jumper procedure is different for the Barionet 100 and Barionet 50. The third procedure uses a form of the setup.cgi command. See the description of the setup.cgi command for more information on this third method. The first procedure should be used whenever possible.

7.4.1 Resetting Using the Configuration Web Page

The first method requires that you have access to the Configuration web pages. If you have access to the configuration web pages, go to the standard home page and click the "Configuration" button at the top of the home page. (See Figure 7 for the Barionet 100 and Figure 8 for the Barionet 50). From the Configuration Page, click the "Defaults" button at the top of the page. Then click the "Factory Defaults" link to reset the device to the factory default settings.

Resetting to factory defaults affects all settings except the network settings. The IP address, netmask, and gateway settings are not affected by resetting to defaults. The reset also does not erase the BCL program(s) that are installed.

A few seconds after you click the Factory Defaults, link the web page will refresh with the message "Settings Reverted to Factory Defaults." A few seconds later, the page will show a "Reboot" link. After the settings are restored to factory defaults, you must reboot the Barionet to activate the new settings.

7.4.2 Resetting Using the Reset Jumper (Barionet 50 Only)

Barionet 50 Only

If you are unable to access the Configuration Web pages, you can reset the Barionet 50 to factory default settings, including the network settings using the reset jumper under the snap-on case lid.

Note

Before using this reset jumper procedure, Barix recommends you attempt to determine the IP address of the device using the procedures described in Using the Barionet Discovery Tool in Chapter 2 and perform the reset using the Web configuration pages, if possible.

You will need a standard 5 mm jumper to short the pins of this reset jumper together.

Follow these steps to reboot the device using the reset jumper:

1. Disconnect the Barionet 50 from its power supply.
2. Remove the snap-on cover by inserting a small flat-blade screwdriver into one of the two latches on either end of the cover and gently prying the cover up. The release the second latch on the same side and lift the lid off.
3. The reset pins (J9) are located under the cover just above RS-232 LED. (See Figure 34).

Note

Some Barionet 50 units were manufactured without the pins installed. In this case, the two pads where the pins go can be shorted with a paperclip or a short piece of wire.

4. Connect the Barionet 50's power supply and wait for the green status LED (the left-most LED) to come on.
5. Short the reset pins by installing the jumper (or using a short piece of wire) and wait about 10 seconds for the red status LED to start blinking fast.

Barionet 50 Only

6. Remove the jumper from J9.

Unlike the Configuration web page reset procedure described above, this procedure resets all the configuration settings, including the network settings (i.e. the IP address, netmask, and gateway IP address). As a result, you'll need to determine and/or reset the IP address of the Barionet 50 using the procedures described in Using the Barionet Discovery Tool in Chapter 2.

7.4.3 Resetting Using a Serial Cable and Terminal Program (Barionet 100 Only)

Barionet 100 Only

If you are unable to access the Configuration Web pages, you can reset the Barionet to factory default settings, including the network settings, using the following procedure.

Note

Before using this reset procedure, Barix recommends you attempt to determine the IP address of the device using the procedures described in Using the Barionet Discovery Tool in Chapter 2 and perform the reset using the Web configuration pages, if possible.

You will need a serial "Null Modem" cable or adapter to complete this procedure. See Null Modem Cable Wiring later in this chapter for details of the null modem wiring. Null modem serial cables or adapters are commonly available at many electronics stores. You will also need a standard 5mm shorting jumper.

1. Disconnect the Barionet from its power supply.
2. Install the jumper on pins 2 and 3 of the rescue jumper (J4). See Figure 35.

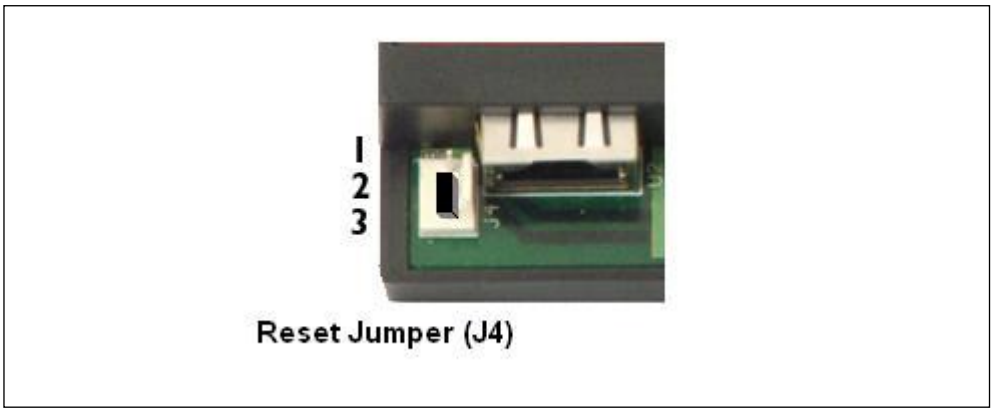


Figure 35. The shorting jumper should be installed on the reset jumper (J4)

3. Connect the null modem serial cable to the Barionet's RS-232 port and to the serial port on a computer.
4. Open a terminal program. In Microsoft Windows, the Hyper Terminal program is typically located in the Start Menu under "Programs→Accessories→Communications". The Macintosh terminal program is typically located in the Utilities folder. Set the program to open the serial port you've connected the null modem cable with the following settings: 9600 baud, 8 bits, No parity, 1 stop bit, and no flow control.
5. Make sure that the terminal program has "focus" (i.e. your keyboard presses are going to the terminal program) and hold the "x" key down.
6. Connect the Barionet to its power supply.
7. Continue holding down the "x" key until you see the message "DEF" in your terminal program. This indicates that that Barionet has successfully restored the factory default settings. Release the "x" key when this message appears.

Barionet 100 Only

Note

If the terminal program is not set for "local echo", you will not see the "x" character on the screen. However, the reset procedure will still work even if local echo is turned off. You will see "DEF" when the Barionet 100 has been restored to factory defaults. The "Local Echo" setting in Windows Hyper Terminal is in the "File" menu under "Properties" on the "Settings" tab of the dialog box. Click the "ASCII Setup" button and check the "Echo typed characters locally" box.

8. Disconnect the Barionet's power supply.
9. Remove the jumper from J4.
10. Re-connect the Barionet's power supply.

Unlike the Configuration web page reset procedure described above, this procedure resets all the configuration settings, including the network settings (i.e. the IP address, netmask, and gateway IP address). As a result, you'll need to determine and/or reset the IP address of the Barionet using the procedures described in Using the Barionet Discovery Tool in Chapter 2.

7.5 The Serial Rescue Procedure

When the Barionet 100 or Barionet 50 is not reachable over the network, and the other procedures and steps described in this chapter have not resolved the problem, the serial rescue procedure will re-load the firmware and restart the device.

Barionet 100 Only

On the Barionet 100, the serial rescue procedure loads the firmware and configuration web user interface pages in WEB1, WEB2 and WEB3. The procedure does not load the default BCL application, so any HTML pages or BCL applications loaded in WEB4 – WEB7 are unaffected.

Barionet 50 Only

On the Barionet 50, the serial rescue procedure erases the entire flash memory. Then it reloads the firmware, and the configuration web user interface pages in 8K, WEB1 and WEB2. It also loads the sample BCL program, in WEB4. Any BCL programs or HTML pages loaded in WEB5 – WEB23 are erased. All settings are returned to their factory default parameters as well.

See the description of The Barionet Memory Map in Chapter 5 for more details on the memory locations.

7.5.1 Null Modem Cable Wiring

You will need a cross-over or "null modem" serial cable for this procedure. Null modem cables or null modem adapters that can be plugged onto a standard 9-pin serial cable are commonly available from many electronics stores. Figure 36 show the minimum required wiring of a null modem cable that can be used with the serial rescue procedure. Other pins may also be cross-wired in many null modem cables and adapters.

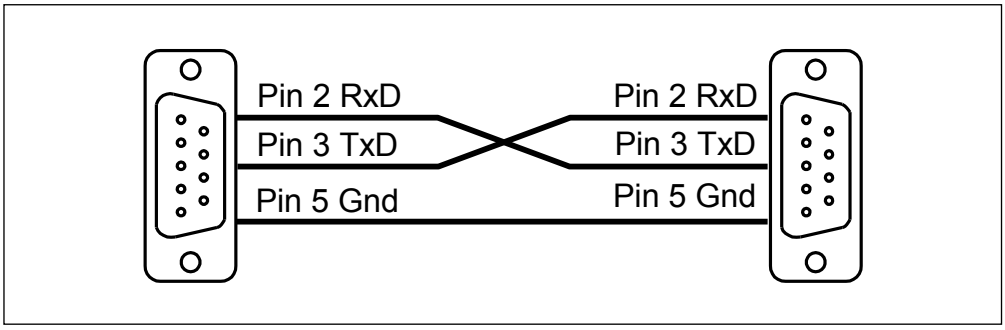


Figure 36. Pins 2, 3, and 5 must be wired as shown in the null modem cable or adapter.

If the computer you are using does not have a serial port but it does have USB ports, you may need a USB-to-serial adapter. Be sure to install the USB-serial adapter, including any

required drivers prior to beginning the serial rescue procedure.

7.5.2 Identifying the Serial Port

You'll also need to know what serial port on your computer you are connected to. If the computer has built-in serial ports, they may be labeled or described in the computer's documentation. The steps below will help you identify the serial port number in Windows or the serial port name in the Macintosh or Linux. You'll need to know the port number or name when you begin the serial rescue procedure.

7.5.2.1 Finding the Serial Port number in Windows

Serial ports in Windows are called "COM ports." The COM ports are listed in the device manager. To open the device manager click:

Start Menu→Settings→Control Panel→System

In the System dialog box, click the "Hardware" tab and click the "Device Manager" button.

Look for the entry in the Device Manager labeled "Ports (COM & LPT)". Click the small + next to the Ports. A list of serial ports, labeled COMx will appear. If you are using a USB-serial adapter, the name of the adapter will appear with the COM port number after the name.

If your computer has one or more built-in serial ports, you should find those listed in the same section of the device manager. If there are multiple built-in ports, check the documentation for your computer to identify the port numbers.

Figure 37 shows a typical device manager screen with a USB-serial converter connected as COM5 in this case.

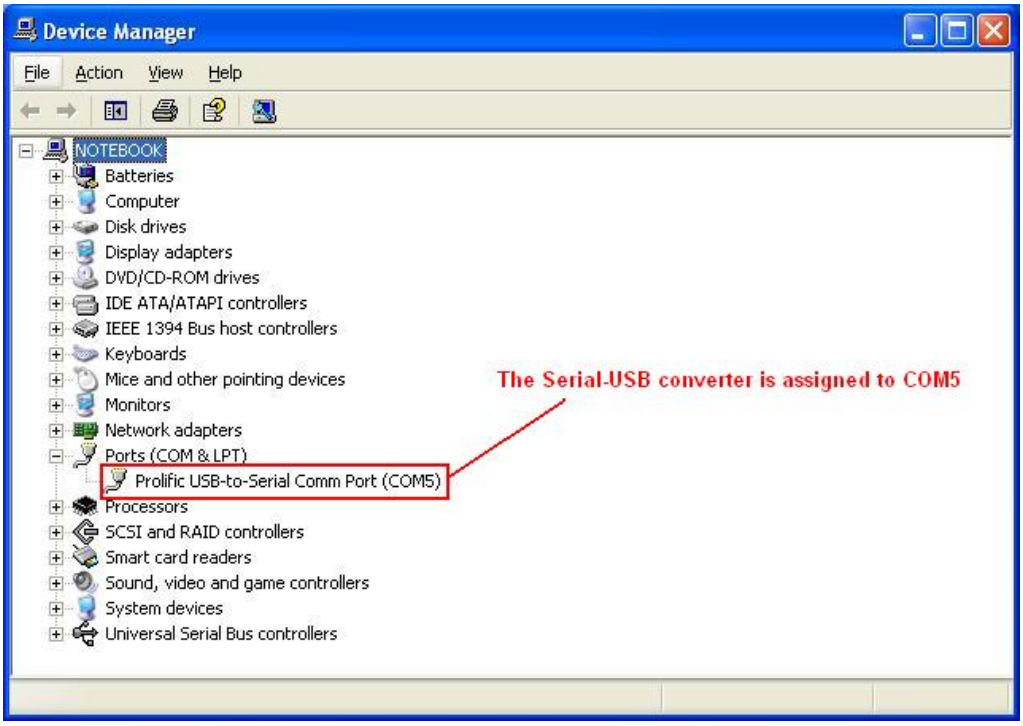


Figure 37. The Device Manager showing a USB-Serial adapter on COM5

Record the number of the COM port for use in the serial rescue procedure described below.

7.5.2.2 Finding the Serial Port Name in the Macintosh

Most modern Macintosh computers don't have built-in serial ports, so you'll need a USB-to-serial adapter with appropriate drivers for the Macintosh operating system. You'll also need a null modem cable or adapter. These instructions assume the Macintosh is running OS-X.

1. Install the USB-serial converter and any required drivers.
2. Open the Terminal application, which is normally located in the "Utilities" sub-folder in the "Applications" folder.
3. In the Terminal window, type:

   ls /dev/tty.*

   See Figure 38 for a typical display.

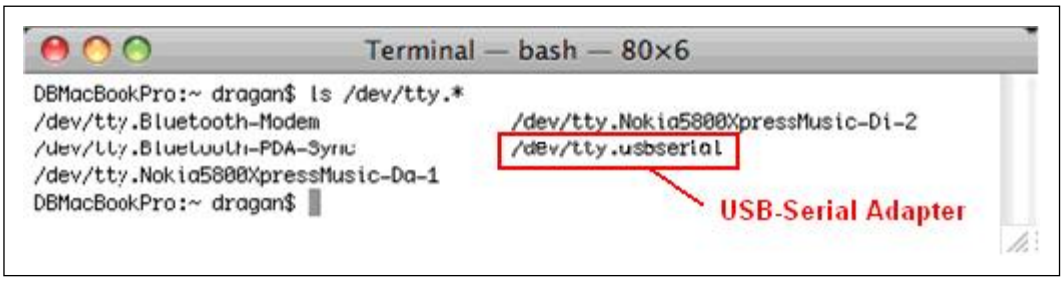


Figure 38. The Terminal window lists serial ports with the ls /dev/tty. command.*

4. The output will list a number of devices as /dev/tty.<some name>. Look for the entry that includes the name of your serial-USB converter. In this case, the last entry is /dev/tty.usbserial, which is the usb-serial adapter.
5. Record the serial port name (in this case "/dev/tty.usbserial"). You'll use this later in the serial rescue procedure.

7.5.2.3 Finding the Serial Port Name in Linux/Unix

There are many variants of Linux and Unix available with minor variations in the steps for determining the serial port used. You can use the "ls" command to list the serial ports by typing:

ls /dev/tty*

This will list all the serial ports that are available on the system. Some of the "ports" in the list may not represent physical serial ports, but terminal sessions connected via the network. Look for ports that begin with **/dev/ttyS** followed by a number. These will, in general, be the hardware serial ports.

If the computer has a USB-serial adapter connected, there may also be devices listed such as **/dev/ttyUSB0**

If there is more than one physical serial port in the computer, you will have to consult the documentation for the computer or experiment with various ports to discover the port name for the port you use in the serial rescue procedure.

Barionet 50 Only

3 Barionet 50 Serial Rescue Procedure

The serial rescue procedure is different for the Barionet 100 and Barionet 50. The steps outlined here apply only to the Barionet 50. If you are working with the Barionet 100, refer to the Barionet 100 Serial Rescue Procedure in this chapter.

1. Disconnect the Barionet 50 from its power supply.
2. Connect the Null modem serial cable to the Barionet 50 RS-232 port and to the serial port on the computer.
3. Start the serial rescue script for your operating system. The script requires an argument that specifies the serial port name you identified in the previous section. The following table lists the scripts for each operating system and shows an example of how the script should be used.

Barionet 50 Only

Operating System	Script syntax	Example
Windows	serial <com port>	serial COM5
Macintosh OS-X	./seriald /dev/tty.<com port>	./seriald /dev/tty.usbserieal
Linux	./seriald /dev/tty<com port>	./seriald /dev/ttyS0 (COM port 1)

Note

If you are executing the rescue procedure on a computer running a POSIX-compliant version of UNIX, you can re-compile the loader program. You will need a C compiler and Make program installed. Remove the executable file called "load" and type "make" to rebuild the load program. If the program compiles successfully, you can then follow the same steps above described for Linux to complete the serial rescue procedure.

4. Within a few seconds of re-connecting the power supply, you should see a message "**Got a reply from the device.**" This indicates that the script has successfully established communication with the Barionet 50.

Note

If you do not see any further messages after "Waiting for the device" within several seconds of applying power to the Barionet 50, the most likely cause is that you have not entered the correct serial port. Go back and check that you are using the correct serial port and that you entered the name of the serial port in the command correctly. If you did not enter it correctly, abort the currently running script and start over at the beginning of this procedure.

5. Over the next few minutes, you'll see a succession of messages as various parts of the firmware are transferred to the Barionet. The progress is shown in kilobytes sent.

6. The device will automatically reboot at the end of the procedure and you should see a line that says "**SUCCESSFUL**" in the terminal window. The device is now set to factory default configuration. Refer to Accessing the Barionet for the first time in Chapter 2 for information on determining or setting the IP address and establishing communication with the device over the network.

The serial rescue procedure erases the Barionet 50's entire flash memory and re-loads the firmware and the example digital I/O tunnel BCL application in WEB4. If you have loaded other BCL programs or HTML pages in WEB4, you'll need to re-load the COB file with the application. Refer to Loading Custom Web Pages and BCL Applications in Chapter 5 for more details on loading COB files.

7.5.4 Barionet 100 Serial Rescue Procedure

Barionet 100 Only

The serial rescue procedure is different for the Barionet 100 and Barionet 50. The steps outlined here apply only to the Barionet 100. If you are working with the Barionet 50, refer to the Barionet 50 Serial Rescue Procedure in this chapter.

You will need a null modem serial cable or adapter, described earlier in this chapter as well as a standard 5mm shorting jumper to complete this procedure.

The Barionet 100 has been manufactured with two different versions of the Ethernet port. You must identify which version of the Ethernet port your Barionet has in order to run the proper script. In general, older Barionets, produced in 2005 or earlier come with the "LX" version of the Ethernet port, while newer Barionets manufactured in 2006 or later have the "EX" version.

If you are unsure which version of port your Barionet has, you can identify the version by removing the top cover of the Barionet and examining the product number on the Ethernet connector. To remove the Barionet cover, insert a small flat-blade screwdriver into one of

Barionet 100 Only

the two latches on either end of the cover and gently pry the cover up. Then release the second latch on the same side and lift the lid off.

Compare the product number on the Ethernet port to the following table:

Product Number	Barionet Manufacture Date	Ethernet Port Version
XP1001001-03R	2006 or later	EX
XP1001000-01	2005 or earlier	LX

You'll need the Ethernet port version for running the serial rescue script below.

1. Disconnect the Barionet from its power supply.
2. Install the shorting jumper on pins 2 and 3 of the reset jumper (J4). See Figure 35 above for the location of J4.
3. Connect the Null modem serial cable to the Barionet RS-232 port and to the serial port on the computer.
4. Start the serial rescue script for your operating system and Ethernet port version from the table above.

The following table lists the scripts for each operating system and shows an example of how the script should be used.

Operating System	Port Version	Script Name	Example
Windows	EX	rescue.bat <com_port>	rescue COM5
	LX	lxrescue.bat <com_port>	lxrescue COM5
Macintosh OS-X	EX	rescue.sh <serial port>	./rescue /dev/tty.usbserial
	LX	lxrescue.sh <serial port>	./lxrescue /dev/tty.usbserial
Linux	EX	rescue.sh <serial port>	./rescue /dev/ttyS0
	LX	lxrescue.sh <serial port>	./lxrescue /dev/ttyS0

5. When the appropriate rescue script is started, re-connect the Barionet to its power source.
6. Within a few seconds, the script should begin to transfer the firmware to the Barionet. The script displays progress messages as it transfers the data.
7. At the end of the process, the script says "Rebooting the device...The device should have been rebooted now."
8. Disconnect the Barionet's power supply and remove the rescue jumper from J4.

The serial rescue procedure does not reset configuration parameters to factory defaults, unless you are also updating to a new version of firmware which changes the configuration memory layout. See the next section on updating the firmware for more details. If you need to reset the device to factory defaults, see Resetting to Factory Defaults earlier in this chapter.

The serial rescue procedure erases WEB1 – WEB3, but it does not affect WEB4 – WEB7, so any applications or HTML pages loaded in WEB4 – WEB7 are not affected.

8 Updating Barionet Firmware

Barix is constantly working to improve our products. New releases of firmware for the Barionet 100 and Barionet 50 are always freely available for download. This Appendix describes the process of updating the firmware in the Barionet 100 and Barionet 50.

8.1 Checking the Firmware Version

Before attempting to update the firmware in the Barionet, check the current version by clicking the Configuration button on the main status home page (For the location of the Configuration button, see Figure 7 for the Barionet 100 and Figure 8 for the Barionet 50). From the Configuration screen, click the "Update" button near the top of the screen. A version screen should appear showing the versions of the various elements of the firmware. Figure 39 shows a typical screen from the Barionet with the current firmware version information.

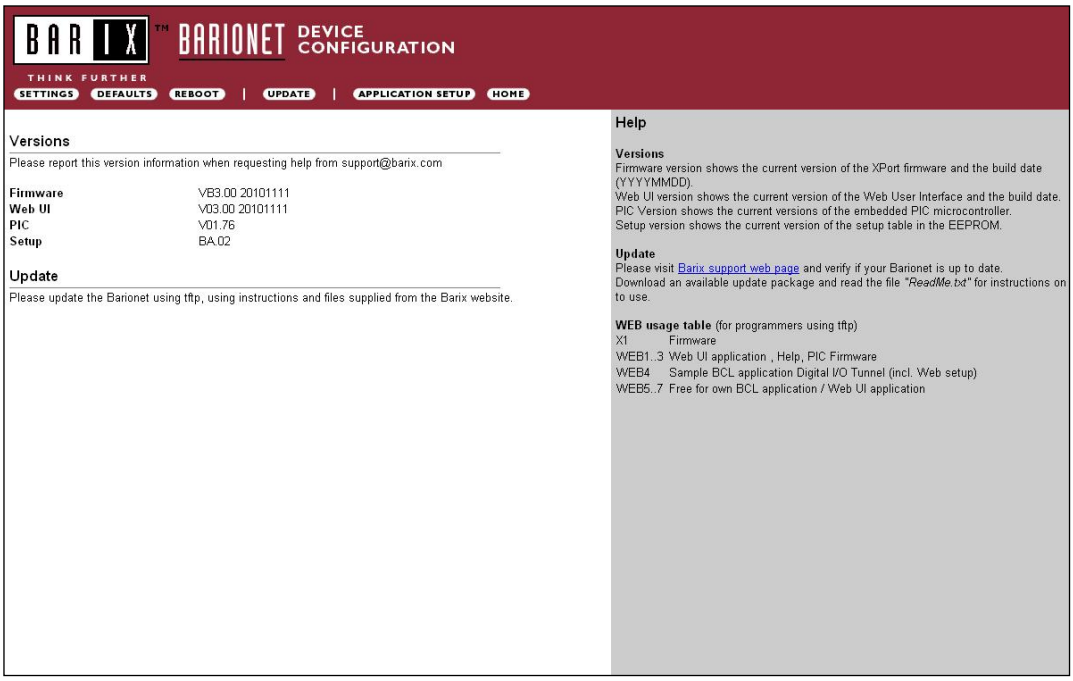


Figure 39. A typical Barionet update screen showing the currently installed firmware versions.

The main firmware version, which is listed first on this screen is the most important version. Update kits from Barix will include all the necessary firmware elements with the version numbers referenced to the primary firmware version (V3.00 in this example).

Check the Downloads section of the Barix web site to see if an updated firmware release is available: http://www.barix.com/downloads/Barionet_Family/51/.

Caution
Be sure to download the firmware for the version of the Barionet (the Barionet 100 or the Barionet 50) that you are updating. Loading firmware for the wrong version will have unpredictable results.

8.2 Updating the Barionet 100 Firmware

Barionet 100 Only

There are two methods for updating the Barionet 100 firmware. The first method, updating via the network, is simpler and should be used whenever possible. The second method uses the serial rescue procedure. Both procedures are described here.

8.2.1 Updating the Barionet 100 via the Network

Barionet 100 Only

1. Download the update package from the Barix web site. The package is delivered as a ZIP file, so the contents of the ZIP file must be extracted into a folder that is accessible to the computer that will be performing the update process.
2. Open a command line window in Windows by clicking Start→Run and typing "cmd" in the dialog box. Click OK to open the command line window. On the Macintosh, start the terminal application from the Utilities folder.
3. Change the current directory to the folder where you extracted the update kit ZIP file.
4. Within that folder, you should find a batch file called barionet.bat (barionet.sh for the Mac and Linux platforms). Run this script file, supplying the IP address of the Barionet as an argument. For example, if the Barionet's IP address is 192.168.0.40, type following command:

   barionet 192.168.0.40 (Windows)

   ./barionet 192.168.0.40 (Mac and Linux)
5. This script will transfer the necessary files to the Barionet. When the process is complete, the script prints "Done" on the screen.

Caution

If power is removed from the Barionet during this process or the network connection is lost, the Barionet may be left in an unreachable state. In that case, you'll have to use the Serial Rescue procedure to re-load the firmware.

8.2.2 Updating the Barionet via the Serial Rescue Procedure

Barionet 100 Only

Use the Serial Rescue procedure to update the Barionet's firmware only if the network update procedure failed, or the Barionet has become unreachable on the network. Check to be sure that you are attempting to contact the Barionet with the correct IP address before you conclude that it is unreachable on the network. See Accessing the Barionet for the first time in Chapter 2.

To update the firmware using the Serial Rescue procedure, download the latest firmware that is compatible with your version of the Barionet. Extract the ZIP file into a folder. Then follow the steps outlined in Chapter 7 under The Serial Rescue Procedure using the tools and firmware package downloaded from the Barix web site.

Note

The update kit will have a README file in the top-level directory extracted from the ZIP file. The readme file will be named README1ST.TXT or something similar. Check the readme file for updated instructions on installing the firmware, since new packages may also include changes to the tools or procedures that have occurred since the time this manual was written.

8.3 Updating the Barionet 50 Firmware

Barionet 50 Only

There are two methods for updating the Barionet 50 firmware. The first method, updating via the network, is simpler and should be used whenever possible. The second method uses the serial rescue process. Both procedures are described in this section.

8.3.1 Updating the Barionet 50 via the Network

Barionet 50 Only

This update method assumes that you are able to connect to the Barionet 50 using a web browser. If you are unable to connect, be sure you are using the correct IP address using the procedures outlined in Chapter 2 under Accessing the Barionet for the first time.

Note

In most cases, updating the firmware in a Barionet 50 does not affect the current configuration settings, such as IP address or other settings. However, some firmware update packages may include updates to the configuration memory layout, which could necessitate resetting the configuration parameters to factory defaults. In addition, if errors occur during the update process, configuration settings may be lost. Barix recommends making a record of all configuration settings prior to updating the Barionet 50 firmware.

1. Open the main I/O status home page of the Barionet 50 in a web browser.
2. Click the "Configuration" button in the upper left corner of the main screen. Then click the "Update" button near the top of the main Configuration screen. See Figure 40.
3. Check the firmware version (the first line of the version information) against the most recent firmware available for the Barionet 50 on the Barix web site at: http://www.barix.com/downloads/Barionet_Family/51/

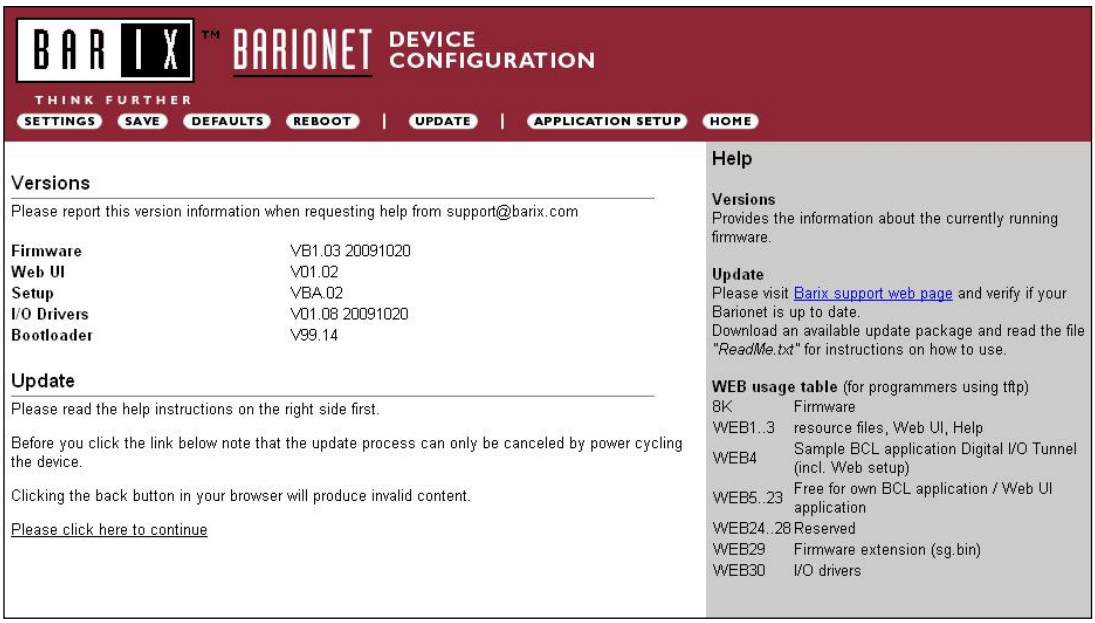


Figure 40. The Barionet 50 Update Screen shows the current firmware version.

4. If a later version of firmware is available, download the update package from the Barix web site. The package is delivered as a ZIP file, so the contents of the ZIP file must be extracted into a folder that is accessible to the computer that will be performing the update process.

Barionet 50 Only

Note

The update kit will have a README file in the top-level directory extracted from the ZIP file. The readme file will be named README1ST.TXT or something similar. Check the readme file for updated instructions on installing the firmware, since new packages may also include changes to the tools or procedures that have occurred since the time this manual was written.

5. At the bottom of the Update screen is a link that says "Please click here to continue". Click this link to start the update process. The Barionet 50 will restart in a special "boot loader" mode. The first screen that appears includes a count-down timer that counts down until the device is restarted in the boot loader mode.
6. When the boot loader process is started, an update screen appears (see Figure 41).
7. Click the "Choose File" button to select a file to load. Navigate to the folder you extracted in step 4 above, and open the "update_rescue" folder within the update kit.
8. There are several .bin files in this folder. This update procedure updates all the firmware components in a single step, so select the "compound.bin" file and click "Open" in the file select dialog box. The compound.bin file contains all the update components.
9. Click the "Upload" button to begin the update process. The process will typically require about 2 minutes to execute, but may take longer depending on the network connection.

Caution

If power is removed from the Barionet 50 during this process or the network connection is lost, the Barionet 50 may be left in an unreachable state. In that case, you'll have to use the Serial Rescue procedure to re-load the firmware.

10. When the upload and update process completes, you will see a page that says "compound.bin successfully loaded." Click the Update link on that page. This will return you to the same Update page as in step 7.
11. Click the "Reboot" button on the update page. After a few seconds, click the link that appears to reload the main Barionet 50 I/O status home page.

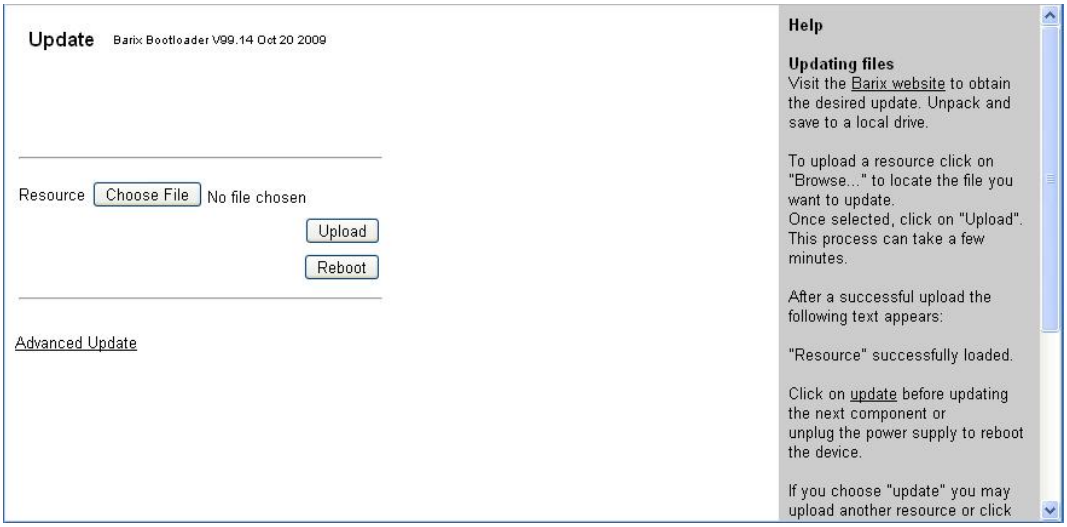


Figure 41. When the Barionet 50 restarts in Boot Loader mode, this Update screen appears.

Barionet 50 Only

2 Using the Advanced Update Process

In the update screen (Figure 41) there is also a link to the "Advanced Update" page. This update process is very similar to the normal network update process described in the previous section except that it allows you to update individual firmware components and

Barionet 50 Only

select the memory destination for the various components.

The Advanced Update screen is identical to the normal update screen except for the addition of a "Target" box, where you can enter the name of a memory location where individual components should be stored in the Barionet 50. The following table describes the components and the typical location of that component. Some of the components must be loaded in the specified location. Others can be moved to alternate locations as indicated in the table.

Resource File	Description	Default Target	Alternatives
Bn50.rom	Barionet 50 firmware	8K	Must be loaded in 8K
Bario50web.cob	Configuration web pages	WEB1	WEB1 – WEB28
barionetbcl.cob	Sample BCL digital I/O tunnel application and app setup pages	WEB4	Not required. May be loaded in WEB2 – WEB28
Sg.bin	Firmware extensions	WEB29	Must be loaded in WEB29
Bclio.bin	I/O drivers	WEB30	Must be loaded in WEB30

Note

You'll also find a file called "empty.cob" which can be uploaded to any of the memory pages from WEB1 through WEB28 to clear that page.

Each time you update a particular component, follow these steps:

1. Click Choose File and choose the .bin, .cob, or .rom file you wish to update.
2. Enter the name of the target memory location for the component.
3. Click the "Upload" button.
4. When the upload and update is complete, you should see a page that indicates that the file you chose was successfully uploaded. Click the "Update" link on that page.
5. If you are going to update another component, choose the "Advanced Update" link again and upload the next component.
6. Otherwise, click "Reboot" when you're done updating components.

Barionet 50 Only

3 Updating the Barionet 50 with the Serial Rescue Process

Use the Serial Rescue procedure to update the Barionet 50 firmware only if the network update procedure failed or the Barionet has become unreachable on the network. Before starting the serial rescue process, check to be sure that you are attempting to contact the Barionet with the correct IP address before you conclude that it is unreachable on the network. See Accessing the Barionet for the first time in Chapter 2.

The serial rescue process reloads the firmware, the configuration web user interface, and the example digital I/O tunnel application. The configuration parameters are also reset to factory defaults by this procedure, so you'll have to restore specific settings after the procedure is complete.

To update the firmware using the Serial Rescue procedure, download the latest Barionet 50 firmware from the Barix web site at http://www.barix.com/downloads/Barionet_Family/51/.

The firmware comes in a ZIP file, so you will need to extract the contents of the ZIP file into a folder. Then follow the steps outlined in Chapter 7 under the Barionet 50 Serial Rescue Procedure using the tools and firmware package downloaded from the Barix web site.

Barionet 50 Only

Note

The update kit will have a README file in the top-level directory extracted from the ZIP file. The readme file will be named README1ST.TXT or something similar. Check the readme file for updated instructions on installing the firmware, since new packages may also include changes to the tools or procedures that have occurred since the time this manual was written.

Appendix A I/O Addressing

All input and output functions in the Barionet 100 and Barionet 50 are addressed using a common I/O address map. Table A.5 shows the register map for the Barionet 100. Table A.6 shows the register map for the Barionet 50.

A.1 Universal Analog/Digital Inputs

The first four inputs of the Barionet 100 can serve as both analog and digital inputs. The Barionet 100 hardware always treats these inputs as analog inputs, and the input voltage can be read from the analog value registers at address 501-504. However, the Barionet 100 firmware also compares the analog input to a fixed threshold value (approximately 1.2 volts) and reports the state of the inputs as digital states in registers 201-204. If the input voltage is above the threshold, the input is considered "high", and if it is below, the input is considered "low".

A.2 Virtual I/O

The I/O address map for both the Barionet 100 and the Barionet 50 includes a number of "virtual I/O" addresses. These are I/O addresses that are not tied directly to an input or output, but are tied to memory locations referred to as "registers".

One of the most common uses of virtual I/O registers is to simplify I/O with Modbus extension modules, like the Barix X8, IO12, and R6. Simple BCL code can check the input status of an external I/O device, such as the X8 or IO12, and copy the input status to a virtual I/O register, or check the status of a virtual I/O register and copy its status to the outputs of an external I/O device, such as the R6. With that simple BCL code to make the connection between the virtual I/O register and the physical inputs or outputs, the rest of the system can use the virtual I/O register just as if it were a physical I/O register, including using &LIO tags to display the input or output status on a web page. Virtual I/O registers are also accessible over all control interfaces (ASCII commands over UDP and TCP, HTTP/CGI, SNMP, Modbus/TCP and BCL).

A.3 Input Pulse Counters

Each digital input of the Barionet 100 and Barionet 50 also has a 32-bit counters associated with it that can count active input edges (i.e. transitions from inactive to active), up to a frequency of 30 Hz (15 pulses per second). The 32-bit counters can count up to 2,147,483,647 pulses before they wrap around back to zero (about 4-1/2 years of pulses at the maximum pulse rate!).

Since all the I/O registers, except the temperature sensor registers (registers 601-650) are 16 bits in length, the counters are accessed with two I/O registers. The lower register contains the lower 16-bits of the counter, and the upper register contains the upper 16 bits. Thus, the total register count is computed by multiplying the upper register by 65536 (16 bits) and adding the lower register.

Total Count = (High Register Value * 65536) + Low Register Value

For example, the counter for digital input 1 is stored in registers 401 and 402. The total count for digital input 1 is:

Total Count Input 1 = (Register 402 Contents * 65536) + Register 401 contents

The lower 16-bits of these input counters can also be preset by writing a preset value to the lower register. Any write access to the lower 16-bits of the counter register automatically clears the upper 16 bits, so the maximum preset value is 65535.

A.4 Using the I/O Addresses

The I/O addresses defined in the tables that follow apply to all of the I/O access methods:

- &LIO Barix dynamic tags (see the description of the I/O Status Tag (&LIO) in Chapter 5.
- IOSTATE and IOCTL BCL functions, which get or set the state of an input or output.
- Modbus/TCP register and coil reads and writes. Keep in mind that Modbus "coil" read and write operations are single-bit operations, which applies to an individual input or output, while register operations can access 16-bits. See Modbus/TCP in Chapter 4 for more information.
- HTTP/CGI writes using the rc.cgi command. See The rc.cgi Command in Chapter 4 for more information.
- SNMP reads and writes. See the section on SNMP in Chapter 4 for more information.
- TCP and UDP ASCII command system. See ASCII Command Protocol in Chapter 4 for more information.

A.5 Barionet 100 I/O Address Table

I/O Address	Read/Write	Size	Barionet 100 I/O
1	Read/Write	1 bit	Relay 1
2	Read/Write	1 bit	Relay 2
3 - 8			Reserved for future expansion
9	Read/Write	1 bit	RS-232 RTS output
10 – 100	Read/Write	1 bit	Virtual I/O bits
101	Read/Write	1 bit	Digital output 1
102	Read/Write	1 bit	Digital output 2
103	Read/Write	1 bit	Digital output 3
104	Read/Write	1 bit	Digital output 4
105 – 200	Read/Write	1 bit	Virtual I/O bits
201	Read only	1 bit	Digital input 1
202	Read only	1 bit	Digital Input 2
203	Read only	1 bit	Digital input 3
204	Read only	1 bit	Digital input 4
205	Read only	1 bit	Digital input 5
206	Read only	1 bit	Digital input 6
207	Read only	1 bit	Digital input 7
208	Read only	1 bit	Digital input 8
209 – 400	Read/Write	1 bit	Virtual I/O bits
401 – 402	Read/Write	32 bits	Digital Input 1 counter (odd address = low 16 bits; even address = high 16 bits)
403 – 404	Read/Write	32 bits	Digital Input 2 counter (odd address = low 16 bits; even address = high 16 bits)
405 – 406	Read/Write	32 bits	Digital Input 3 counter (odd address = low 16 bits; even address = high 16 bits)
407 – 408	Read/Write	32 bits	Digital Input 4 counter (odd address = low 16 bits; even address = high 16 bits)
409 – 410	Read/Write	32 bits	Digital Input 5 counter (odd address = low 16 bits; even address = high 16 bits)
411 – 412	Read/Write	32 bits	Digital Input 6 counter (odd address = low 16 bits; even address = high 16 bits)
413 – 414	Read/Write	32 bits	Digital Input 7 counter (odd address = low 16 bits; even address = high 16 bits)
415 – 416	Read/Write	32 bits	Digital Input 8 counter (odd address = low 16 bits; even address = high 16 bits)
417 – 500			Reserved for future expansion
501	Read only	16 bits	Analog Input 1 value
502	Read only	16 bits	Analog Input 2 value
503	Read only	16 bits	Analog Input 3 value
504	Read only	16 bits	Analog Input 4 value
505 – 600	Read/Write	16 bits	Virtual I/O register
601-650	Read only	16 bits	Temperature sensor values
651 – 700	Read only	32 bits	Temperature sensor addresses (lower 32 bits only)
701 – 1000	Read/Write	16 bits	Virtual I/O registers

I/O Address	Read/Write	Size	Barionet 100 I/O
1001 – 1200	Read only	64 bits	Temperature sensor hardware ID (odd addresses = low 32 bits; even addresses = high 32 bits)
60000[1]	Read only	16 bits	Hardware type identifier
60001	Read only	16 bits	Number of serial ports
60002	Read only	16 bits	Number of relays
60003	Read only	16 bits	Number of digital outputs (excluding RTS/CTS)
60004	Read only	16 bits	Number of digital inputs (excluding RTS/CTS)
60005	Read only	16 bits	Number of analog outputs
60006	Read only	16 bits	Number of analog inputs

[1] Addresses 60000 – 60006 can only be accessed using the BCL IOSTATE() function. They are not accessible using the &LIO tag or the ASCII command protocol.

A.6 Barionet 50 I/O Address Table

I/O Address	Read/Write	Size	Barionet 50 I/O
1	Read/Write	1 bit	Relay 1
2	Read/Write	1 bit	Relay 2
3	Read/Write	1 bit	Relay 3
4	Read/Write	1 bit	Relay 4
5 – 8			Reserved for future expansion
9	Read/Write	1 bit	RS-232 RTS output
10 – 200	Read/Write	1 bit	Virtual I/O bits
201	Read only	1 bit	Digital input 1
202	Read only	1 bit	Digital Input 2
203	Read only	1 bit	Digital input 3
204	Read only	1 bit	Digital input 4
205 - 300	Read/Write	1 bit	Virtual I/O bits
301 – 400			Reserved for future expansion
401 – 402	Read/Write	32 bits	Digital Input 1 counter (odd address = low 16 bits; even address = high 16 bits)
403 – 404	Read/Write	32 bits	Digital Input 2 counter (odd address = low 16 bits; even address = high 16 bits)
405 – 406	Read/Write	32 bits	Digital Input 3 counter (odd address = low 16 bits; even address = high 16 bits)
407 – 408	Read/Write	32 bits	Digital Input 4 counter (odd address = low 16 bits; even address = high 16 bits)
409 – 500			Reserved for future expansion
501 – 600	Read/Write	16 bits	Virtual I/O register
601 - 650	Read/Write[2]	16 bits	Temperature sensor values
651 – 700	Read only	32 bits	Temperature sensor addresses (lower 32 bits only)
701 – 1000	Read/Write	16 bits	Virtual I/O registers
1001 – 1200	Read only	64 bits	Temperature sensor hardware ID (odd addresses = low 32 bits; even addresses = high 32 bits)
60000[3]	Read only	16 bits	Hardware type identifier
60001	Read only	16 bits	Number of serial ports
60002	Read only	16 bits	Number of relays
60003	Read only	16 bits	Number of digital outputs (excluding RTS/CTS)
60004	Read only	16 bits	Number of digital inputs (excluding RTS/CTS)
60005	Read only	16 bits	Number of analog outputs
60006	Read only	16 bits	Number of analog inputs

[2] As of Barionet 50 Firmware version V2.01, BCL programs can write to the temperature sensor registers (addresses 601-650). These registers can be used as general-purpose registers when no temperature sensor is connected.

[3] Addresses 60000 – 60006 can only be accessed using the BCL IOSTATE() function. They are not accessible using the &LIO tag or the ASCII command protocol.

Appendix B Configuration and Setup Memory Layout

This Appendix provides the layout of the Barionet 100 and Barionet 50's non-volatile setup memory. This memory stores the configuration parameters for the device. The Barionet 100 has 1024 bytes of non-volatile memory and Barionet 50 has 512 bytes of non-volatile memory devoted to storing configuration data. On both devices, the first 256 bytes of the configuration memory is reserved for Barix-defined configuration parameters. The remaining configuration memory is available for user-defined configuration parameters.

The setup memory can be read and written to using three different methods:

1. The BCL **OPEN STP:** statement which opens the setup memory for read and write access. See the BCL Programmer's Manual for more information on using this method to read and write setup memory.
2. The setup.cgi command. The built-in setup.cgi command can be used to store values from an HTML form into setup memory by invoking setup.cgi in the action attribute of the HTML <form> tag. The entity names in the HTML form are used by setup.cgi to determine where each piece of data is to be stored in setup memory. It's extremely important, therefore, to be sure that you use correct names for HTML elements in forms that use setup.cgi, since incorrect names can cause improper data to be written to the setup memory, which may render the Barionet 100 or Barionet 50 unreachable. See The setup.cgi Command for more details.
3. The &LSetup tag can be used to retrieve the current value of setup parameters and display them in a web page. See Setup Data Tag (&LSetup) for more details.

B.1 The Setup Memory Map Tables

The next two sections show the layout of the setup memory for the Barionet 100 and the Barionet 50. There are several important points to keep in mind when reading these tables:

- IP addresses, which are stored in four consecutive bytes are always stored with the highest byte (the first octet) stored at the lowest address. For example, if an IP address of 192.168.0.40 is stored in setup memory, the 192 byte is stored in the first location, followed by 168 in the second byte, 0 in the third byte, and 40 in the fourth byte.
- Strings in setup memory are stored in ASCII and terminated with a null byte (0x00). The length in the table includes the terminating null.
- Some parameters are stored as single bit values. Eight single-bit parameters are stored in a single byte. These values have a lower case "b" in the length column because they are stored in a single bit. The HTML name indicates the byte number and the bit number where the value is stored, with bit numbers starting at zero. For example, the number of stop bits configured for the RS-232 interface is stored in bit 7 of byte 17. The HTML name for this entity is B17b7.
- Word values are stored in little endian (Intel) format, with the lowest byte first.
- All numeric values are integers. Signed values are stored in 2's complement notation.

B.2 Barionet 100 Setup Memory Layout

The following table applies only to the Barionet 100 (the original Barionet). Refer to Section B.3 for the same information for the Barionet 50.

Parameter	Offset	HTML Name	Length[4]	Description/Link
IP Address	0	B0 – B3	4	B0.B1.B2.B3
Product ID	4		1	0xBA for Barionet
Setup Version	5		1	0x02 for version 02
Netmask	6	N0, N1, N2, N3B6	1	The value is the count of the zero bits in the net mask starting from the lowest byte. (e.g. 255.255.255.0 is stored as 8).
Reserved	7		1	
Web server port	8	W8	2	Default value of 0 indicates port 80
Lockdown mode	10	W10	2	Sets the lockdown mode. See Setting the Lockdown mode
Gateway IP	12	B12 – B15	4	See Setting the Gateway IP Address
RS-232 baud rate	16	B16	1	
RS-232 stop bits	17	B17b7	b	0 = 1 stop bit, 1 = 2 stop bits
RS-232 data bits	17	B17b6	b	0 = 8 bits, 1 = 7 bits
RS-232 Parity	17	B17b5	b	0 = Disabled, 1 = Enabled
RS-232 Parity mode	17	B17b4	b	0 = Odd, 1 = Even
RS-232 Flow control	17	B17b2	b	0 = None, 1 = RTS/CTS
RS-232 buffer flush	17	B17b0	b	0 = No, 1 = Flush buffer
RS-232 Local port TCP	18	W18	2	The port number for the built-in serial gateway function. 0 = disabled. See Setting the Local Port.
Reserved	20		2	
Reserved	22		2	
RS-232 Serial Gateway Disconnect timeout	26	B26	1	0 = disabled; 1 – 255 = timeout in seconds. See Setting the Disconnect Timeout.
Reserved	27		5	
RS-422/485 baud rate	32	B32	1	4 = 19200, 8 = 9600, 16 = 4800, 32 = 2400, 64 = 1200, 128 = 600
RS-422/485 stop bits	33	B33b7	b	0 = 1 stop bit, 1 = 2 stop bits
RS-422/485 data bits	33	B33b6	b	0 = 8 bits, 1 = 7 data bits
RS-422/485 parity	33	B33b5	b	0 = Disabled, 1 = Enabled
RS-422/485 parity mode	33	B33b4	b	0 = Even, 1 = Odd
Interface type	33	B33b3	b	0 = RS-485, 1 = RS-422
RS-422/485 Buffer flushing	33	B33b0	b	0 = No, 1 = Flush buffer
RS-422/485 Local Port TCP	34	W34	2	The port number for the built-in serial gateway function. 0 = disabled. See Setting the Local Port
Reserved	36		2	

[4] This column indicates the length of the data element in bytes. "b" indicates that this data element occupies a single bit in a byte of data. The HTML name includes the bit number in the last two characters: bx, where x is the bit number within the byte.

Parameter	Offset	HTML Name	Length	Description/Link
Reserved	38		2	
RS-422/485 Serial Gateway Disconnect timeout	42	B42	1	0 = disabled; 1 – 255 = timeout in seconds. See Setting the Disconnect Timeout.
Reserved	43		5	
SNMP Trap receiver IP address	48	B48 – B51	4	See Setting the Trap Receiver IP address
UDP send Info IP	52	B52 – B55	4	See Setting the UDP Send Info Address
UDP Command Port	56	W56	2	See Setting the UDP command port
UDP Destination Port	58	W58	2	See Setting the UDP Destination Port
UDP Interval	60	W60	2	See Setting the UDP Interval
TCP Command Port	62	W62	2	See Setting the TCP command port
Pull-up for Input 1-8	64	B64b0 - B64b7	b	0 = On, 1 = Off
SNMP Trap Input 1-8	65	B65b0- B65b7	b	0 = No trap, 1 = Send trap
Polarity for Input 1-8	66	B66b0- B66b7	b	0 = Active low, 1 = Active high
Input 1 – 4 Type	67	B67b0- B67b3	b	0 = Digital, 1 = Analog
SNMP Trap Repeat Time	68	W68	2	0 = Disabled; 1 – 65535 = Trap repeat time in seconds.
I/O Protocol	70	B70	1	0 = 1-wire®, 1 = Wiegand
Reserved	71	B71	9	Reserved for OEM
Syslog Server IP	80	B80 – B83	4	See Setting the Syslog Server
NTP Server IP	84	B84 – B87	4	See Setting the NTP Server
DNS Server IP	88	B88 – B91	4	See Setting the DNS Server Address
Syslog Debug Level	92	B92	1	See Setting the Syslog Debug Level
Debug Mode/Flags	93	B93	1	For Barix internal use only
Time Zone	94	B94	1	Time zone in 30 minute intervals from GMT (signed, +/- 24).
Reserved	95	B95	1	DST rule flags
TCP Command Port Disconnect timeout	96	B96	1	0 = No timeout; 1 – 255 = timeout in seconds.
Reserved	97		3	
BCL Program Name	100	S100	9	Filename of BCL program without .TOK extension. 8 characters max plus null string terminator.
DHCP Host Name	109-124	S109	16	Barionet's name for DHCP server. 15 characters max plus null string terminator.
Password	125-137	P1	13	md5-hashed or plain text password

B.3 Barionet 50 Setup Memory Layout

The following table applies only to the Barionet 50. Refer to Section B.2 for the same information for the Barionet 100.

Parameter	Offset	HTML Name	Length	Description/Link
IP Address	0	B0 – B3	4	B0.B1.B2.B3
Product ID	4		1	0xBA for Barionet
Setup Version	5		1	0x03 for version 03
Netmask	6	N0, N1, N2, N3B6	1	The value is the count of the zero bits in the net mask starting from the lowest byte. (e.g. 255.255.255.0 is stored as 8).
Reserved	7		1	
Web server port	8	W8	2	Default value of 0 indicates port 80
Lockdown mode	10	W10	2	Sets the lockdown mode. See Setting the Lockdown mode
Gateway IP	12	B12 – B15	4	See Setting the Gateway IP Address
RS-232 baud rate	16	B16	1	7 = 300, 6 = 600, 5 = 1200, 4 = 2400, 3 = 4800, 2 = 9600, 1 = 19200, 0 = 38400, 9 = 57600, 11 = 76800, 8 = 115200, 12 = 230400
RS-232 stop bits	17	B17b7	b	0 = 1 stop bit, 1 = 2 stop bits
RS-232 data bits	17	B17b6	b	0 = 8 bits, 1 = 7 bits
RS-232 Parity	17	B17b5	b	0 = Disabled, 1 = Enabled
RS-232 Parity mode	17	B17b4	b	0 = Odd, 1 = Even
RS-232 Flow control	17	B17b2	b	0 = None, 1 = Software (Xon/Xoff) 2 = Hardware (RTS/CTS)
RS-232 buffer flush	17	B17b0	b	0 = No, 1 = Flush buffer
RS-232 Local port TCP	18	W18	2	The port number for the built-in serial gateway function. 0 = disabled. See Setting the Local Port.
Reserved	20	W20	2	Serial gateway destination port
Reserved	22	B22-B25	4	Serial gateway destination IP
RS-232 Serial Gateway Disconnect timeout	26	B26	1	0 = disabled; 1 – 255 = timeout in seconds. See Setting the Disconnect Timeout.
Reserved	27		5	
RS-422/485 baud rate	32	B32	1	7 = 300, 6 = 600, 5 = 1200, 4 = 2400, 3 = 4800, 2 = 9600, 1 = 19200, 0 = 38400, 9 = 57600, 11 = 76800, 8 = 115200, 12 = 230400
RS-422/485 stop bits	33	B33b7	b	0 = 1 stop bit, 1 = 2 stop bits
RS-422/485 data bits	33	B33b6	b	0 = 8 bits, 1 = 7 data bits
RS-422/485 parity	33	B33b5	b	0 = Disabled, 1 = Enabled
RS-422/485 parity mode	33	B33b4	b	0 = Even, 1 = Odd
Interface type	33	B33b3	b	Always 0 = RS-485
RS-422/485 Buffer flushing	33	B33b0	b	0 = No, 1 = Flush buffer

Parameter	Offset	HTML Name	Length	Description/Link
RS-422/485 Local Port TCP	34	W34	2	The port number for the built-in serial gateway function. 0 = disabled. See Setting the Local Port
Reserved	36	W36	2	Serial gateway destination port
Reserved	38	B38-B41	4	Serial gateway destination IP
RS-422/485 Serial Gateway Disconnect timeout	42	B42	1	0 = disabled; 1 – 255 = timeout in seconds. See Setting the Disconnect Timeout.
Reserved	43		5	
SNMP Trap receiver IP address	48	B48 – B51	4	See Setting the Trap Receiver IP address
UDP send Info IP	52	B52 – B55	4	See Setting the UDP Send Info Address
UDP Command Port	56	W56	2	See Setting the UDP command port
UDP Destination Port	58	W58	2	See Setting the UDP Destination Port
UDP Interval	60	W60	2	See Setting the UDP Interval
TCP Command Port	62	W62	2	See Setting the TCP command port
Reserved	64	B64	1	Pull-up resistor configuration for other Barionet models.
SNMP Trap Input 1-4	65	B65b0 - B65b3	b	0 = No trap, 1 = Send trap
Reserved	66	B66	1	Input polarity for other Barionet models
Reserved	67	B67	1	Select Analog or Digital input types for other Barionet models.
SNMP Trap Repeat Time	68	W68	2	0 = Disabled; 1 – 65535 = Trap repeat time in seconds.
Reserved	70	B70	1	0 = 1-wire, 1 = Wiegand
Reserved	71-77	B71 – B77	9	Reserved for OEM
Reserved	78-79	B78 – B79	2	
Syslog Server IP	80	B80 – B83	4	See Setting the Syslog Server
NTP Server IP	84	B84 – B87	4	See Setting the NTP Server
DNS Server IP	88	B88 – B91	4	See Setting the DNS Server Address
Syslog Debug Level	92	B92	1	See Setting the Syslog Debug Level
Debug Mode/Flags	93	B93	1	For Barix internal use only
Time Zone	94	B94	1	Time zone in 30 minute intervals from GMT (signed, +/- 24).
Reserved	95	B95	1	DST rule flags
TCP Command Port Disconnect timeout	96	B96	1	0 = No timeout; 1 – 255 = timeout in seconds.
Modbus/TCP timeout	97	B97	1	0 = no timeout; 1 – 255 = timeout in seconds.
BCL Program Name	100	S100	9	Filename of BCL program without .TOK extension. 8 characters max plus null string terminator.
DHCP Host Name	109-124	S109	16	Barionet's name for DHCP server. 15 characters max plus null string terminator.
Password	125-137	S125	13	md5-hashed or plain text password

Appendix C Accessories

Barix offers a variety of accessories and expansion modules that are designed to work with the Barionet 100 and Barionet 50 to add additional functionality. This section briefly lists and describes the accessories available at the time of the publication of this manual. Check the Barix web site for new products, accessories, and expansion modules.

C.1 Barix TS Temperature Sensors

Barix offers a version of the Dallas Semiconductor/Maxim DS18B20 1-wire digital temperature sensor in a convenient rubber-encapsulated package with approximately 12" leads. Figure 42 shows the TS Temperature sensor and the unpackaged DS1820B sensor.

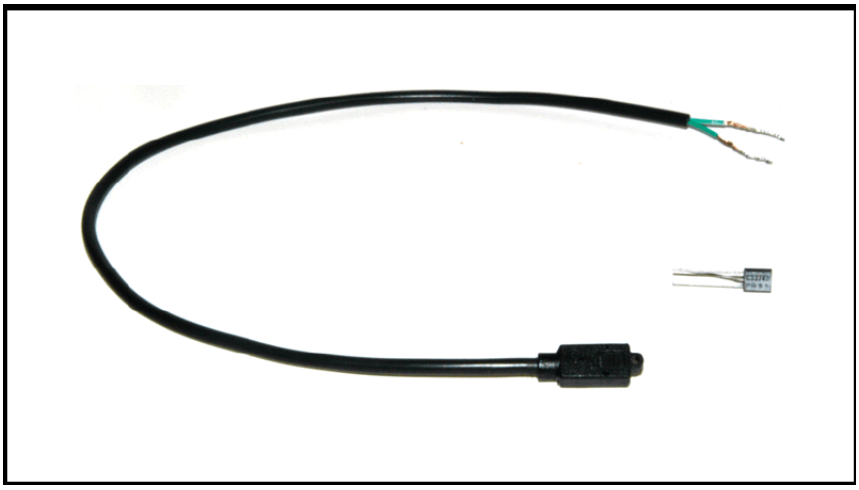


Figure 42. The Barix TS Temperature Sensor is a DS1820B 1-wire sensor in a rubber-encapsulated package.

The Barix package is designed to operate the temperature sensor on parasitic power—meaning that the sensor derives its power from the data bus. No external power source is required. There are two leads with bare ends from the package:

- Ground: Black on later units and Green on earlier units
- Data (DQ): Yellow on later units and Green with a white stripe on earlier units.

C.1.1 Features

- Digital temperature measurement with 12-bit resolution
- Wide temperature range: -55ºC to +125ºC (-67ºF to +257ºF)
- 0.5ºC accuracy from -10ºC to +85ºC
- Rubber encapsulated package with mounting hole

Refer to Connecting 1-Wire Temperature Sensors in Chapter 3 of this manual for more information on connecting the TS Temperature sensors to the Barionet 100 or Barionet 50.

For more information on ordering the Barix TS temperature sensor, please visit the Barix web site (www.barix.com) to find a distributor in your area.

C.2 Barix X8 Expansion Module

The Barix X8 is an RS-485 Modbus I/O module designed to interface directly to the Barionet 100 and Barionet 50's RS-485 interface. See Figure 43. The X8 provides eight independently configurable inputs or outputs. The X8 can be driven by any Modbus master, capable of implementing the Modbus-RTU protocol on RS-485.

Example BCL code for driving the X8 and other Barix expansion modules is available for download on the Barix web site.

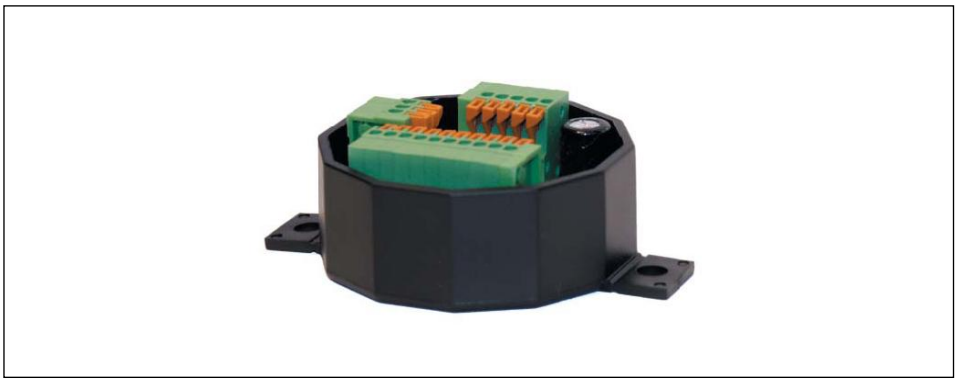


Figure 43. The Barix X8 is a RS-485 Modbus Input/output module that can be connected directly to the Barionet or Barionet 50's RS-485 interface.

C.2.1 Features

- Eight configurable TTL-level inputs or outputs
- Presets for power-on output states
- RS-485 interface
- Modbus/RTU protocol
- Internal pull-up resistors can be enabled or disabled on inputs
- Connects directly to LED displays, buttons, 1-wire temperature sensors

C.2.2 Ordering Information

For more information on ordering the Barix X8 Modbus I/O module, please visit the Barix web site (www.barix.com) to find a distributor in your area.

C.3 Barix IO12 Expansion Module

The Barix IO12 is an RS-485 Modbus I/O module designed to interface directly to the Barionet 100 and Barionet 50's RS-485 interface. See Figure 44. The IO12 provides 12 ESD-protected optically isolated inputs with de-bounce and pulse counting capability plus 12 solid-state current sourcing outputs with thermal and over-current protection. The IO12 can be driven by any Modbus master, capable of implementing the Modbus-RTU protocol on RS-485.

Example BCL code for driving the IO12 and other Barix expansion modules is available for download on the Barix web site.

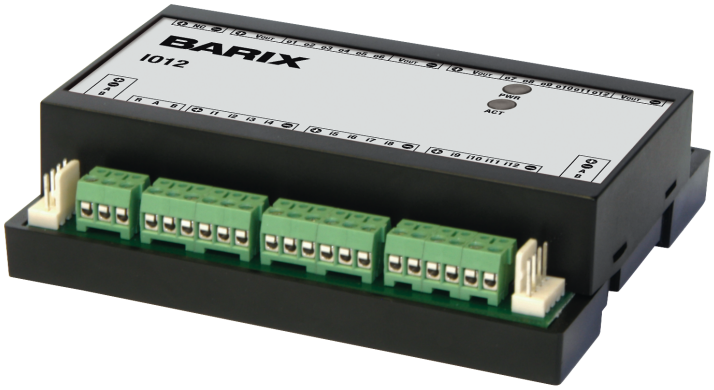


Figure 44. The Barix IO12 is a RS-485 Modbus Input/output module that can be connected directly to the Barionet or Barionet 50's RS-485 interface.

C.3.1 Features

- Twelve solid state current-sourcing outputs with over-current and thermal protection.
- Twelve ESD-protected optically isolated inputs with de-bounce and pulse counting capability.
- RS-485 interface
- Modbus/RTU protocol
- DIN rail mounting capability, with a package very similar to the Barionet 100 and Barionet 50.
- Separate detachable screw terminal blocks for inputs, outputs, and power supplies

C.3.2 Ordering Information

For more information on ordering the Barix IO12 Modbus I/O module, please visit the Barix web site (www.barix.com) to find a distributor in your area.

C.4 Barix R6 Relay Expansion Module

The Barix R6 is an RS-485 Modbus I/O module designed to interface directly to the Barionet 100 and Barionet 50's RS-485 interface. See Figure 45. The R6 provides 6 single-pole double throw relays (Normally open and Normally closed contacts for each relay). Each relay can switch up to 250VAC and up to 16 amps of resistive load. The R6 can be driven by any Modbus master, capable of implementing the Modbus-RTU protocol on RS-485.

Example BCL code for driving the R6 and other Barix expansion modules is available for download on the Barix web site.

Figure 45. The Barix R6 is a RS-485 Modbus Relay Output module that can be connected directly to the Barionet or Barionet 50's RS-485 interface.

C.4.1 Features

- Six single-pole double-throw relays. Each relay has a common, normally open and normally closed contact available.
- Capable of switching up to 250VAC and 16 amps (resistive).
- RS-485 interface
- Modbus/RTU protocol
- DIN rail mounting capability, with a package very similar to the Barionet 100 and Barionet 50.
- Separate detachable screw terminal blocks for inputs, outputs, and power supplies

C.4.2 Ordering Information

For more information on ordering the Barix R6 Modbus relay module, please visit the Barix web site (www.barix.com) to find a distributor in your area.

Appendix D Mounting the Barionet

The Barionet and Barionet 50 cases are designed to snap onto a standard DIN rail. To mount the unit, start by engaging the bottom of the track on the rear of the Barionet with the bottom edge of the rail. Press upward and in against the rail gently until the Barionet snaps into place on the rail.

Mounting holes are also provided in the bottom of the plastic case. To use these holes, the top cover must be removed by inserting a small screwdriver into one of the two latches on the side of the cover and prying up gently. Once the first latch is released, insert the screwdriver into the other latch on the same side and release the cover. Pull up gently and release the cover from the latches on the opposite side.

The main circuit board can be removed from the bottom part of the plastic case to reveal the mounting holes.

Caution

Static electricity can permanently damage the Barionet circuit board. Be sure to wear an anti-static wrist strap and/or discharge any static electricity to a grounded metal object before handling the Barionet circuit board. The Barix warranty does not cover electro-static damage to the Barionet 100 or Barionet 50.

Appendix E Glossary

AutoIP

AutoIP is a process that allows a device to acquire an IP address on its own, without the use of an external server, as required by DHCP and BOOTP. AutoIP always assigns addresses in the range 169.254.0.0 through 169.254.0.16. This protocol is primarily useful in very small networks and was designed to facilitate very easy network setup for inexperienced or non-technical users.

BCL

Barix Control Language is a programming language implemented in the Barionet for creating custom applications. It is very similar to the BASIC language, but contains specific extensions for the advanced I/O capabilities of the Barionet. Refer to the BCL Programmer's Reference Manual for more information on BCL.

BOOTP

Sometimes referred to as the Bootstrap Protocol is an older protocol used to obtain an IP address from a configuration server. This protocol has been replaced on most networks with the more flexible dynamic host configuration protocol, DHCP.

CGI

Common Gateway Interface. CGI requests over HTTP execute an operation on the web server (in this case, the Barionet). The routine that is executed should return valid HTML for display, though some routines may return blank pages or nothing at all.

DHCP

Short for Dynamic Host Configuration Protocol, a protocol used to automatically assign an IP address to a device connected to a Network.

DNS

DNS is an acronym for Domain Name System, which is the system used on the Internet and other IP networks to translate a domain name (e.g. barix.com) to an IP address (e.g. 209.197.116.12). The DNS server is a computer that is responsible for this name translation. The DNS system on the Internet is actually a distributed system. Individual DNS servers typically store only a small fraction of the DNS entries for the entire internet. If a server is asked to translate a domain name that it doesn't have a record for, it asks other servers that are higher in a hierarchy of servers for the information it needs. For more details on the DNS system, see http://en.wikipedia.org/wiki/Domain_Name_System.

DTE

DTE stands for Data Terminal Equipment, as defined by the RS-232 standard. A DTE device is one end of a point-to-point RS-232 serial interface. The opposite end of the connection is a DCE (Data Communications Equipment) device. The DTE device receives data on pin 2 of the interface and transmits data on pin 3, while the DCE device receives data on pin 3 and sends data on pin 2.

Gateway

A "gateway" is a term for a computer or router that typically connects two subnets on a network, or a local area network to the internet. The purpose of the gateway is to forward any network traffic that is destined for a device outside the local subnet to another subnet, or the internet so that it can reach its final destination. If the Barionet needs to send network traffic to device that is not on the same subnet as the Barionet, the gateway IP address must either be manually filled in, or supplied automatically by a DHCP server, so that the Barionet can contact the gateway to send the traffic.

HTML

Hypertext Markup Language (HTML) is the "language" of web pages. The HTML language defines a rich set of "tags" that describe the formatting and semantics for data. The HTML language is interpreted by the web browser to render the page. Barix extends standard HTML by adding special proprietary tags that allow inserting dynamic data from the current status of Barionet inputs or outputs or BCL variables.

IP

Short for Internet Protocol. Every device on an IP-based network requires an IP address to identify its location or address on the network. Example: **192.168.2.10**
See also Appendix G: IP Addresses, Netmasks and Gateways for more details.

IPzator

The IPZator function allows Barix devices to detect the network address and find a free IP address. If DHCP and BOOTP fail, IPzator searches for a free IP address within the subnet starting with **x.x.x.168**.

MAC address

Abbreviation for Medium Access Control, a MAC is a unique address for each Ethernet device. The MAC address is typically written in hexadecimal format. Example: **00:08:E1:00:D8:24.**

MIB

MIB stands for Management Information Base. An MIB is a database whose format is defined SNMP protocol standard. The MIB data is defined by the manufacturer to describe the capabilities of the device to be monitored using SNMP. The Barionet comes pre-loaded from the factory with an MIB file called Barionet.MIB. This file can be downloaded and installed in an SNMP manager program to define the SNMP capabilities of the Barionet.

MIME

MIME stands for Multi-purpose Internet Mail Extensions. MIME was introduced to extend the standard email transfer protocol, which was originally designed only to send standard ASCII text email. MIME allows email attachments of a variety of types, including images, sound files, and other files to be transferred as email attachments with information to tell the receiver what the attachment contains. MIME types are defined by the non-profit Internet Assigned Number Authority: http://www.iana.org/assignments/media-types/.

Modbus

Modbus is a communication protocol originally created in 1979 by Modicon for use with its programmable logic controller (PLC). The protocol has since become a widely accepted standard for monitoring and controlling a variety of industrial and automation devices. Modbus can be implemented on both a serial interface, typically RS-485, or a variant of the standard, called Modbus/TCP, exists that is specifically adapted to operate on TCP/IP networks. There are two versions of the serial interface protocol: Modbus ASCII, which uses strictly human readable ASCII characters for communication, and a more compact binary version called Modbus/RTU (Remote Terminal Unit).

Netmask

A netmask differentiates between two parts of an IP address. The upper part of the IP address, called the "network prefix" is common to all devices on a particular subnet. The lower part, called the "host part" is unique to each device on that subnet. The netmask defines how much of the 32-bit address is a "network prefix" and how much is a "host part". See Appendix G: IP Addresses, Netmasks and Gateways for more details.

NTP

Network Time Protocol. NTP servers provide a very precise time reference for other computers and devices on the Internet. See http://www.ntp.org/ for more information on NTP servers.

Ping

Ping is a basic Internet program that lets you verify that a particular IP address exists and can accept requests. Example: ping **192.168.2.10**

SNMP

SNMP is a UDP protocol for monitoring and managing devices on a network. It is a component of the Internet Protocol suite. SNMP further defines a database called a Management Information Base (MIB) for defining the capabilities of the device being managed.

Socket

In a TCP/IP network, a "socket" refers to a TCP/IP connection to a particular IP address and port number. The combination of the host IP address and the port number is often

written in the form <IP Address>:<Port Number>.

Static IP

A Static IP is a fixed IP address that you assign manually to a device on the network, as compared to an address that is dynamically assigned to a device using a protocol such as DHCP or BOOTP. A static IP address does not change unless the user or network administrator changes the address assignment. In contrast, a dynamically assigned address can change. The network administrator is responsible for ensuring that no two devices on the same subnet have the same static IP address.

Subnet

A large TCP/IP network is often logically divided into "subnets", which are parts of the network that share a common netmask (see the definition of Netmask). Even a small home network is considered a subnet because all the IP addresses in the network fall in a range defined by the netmask.

Telnet

Telnet is a user command and an underlying TCP/IP protocol for logging in to command-line access on a remote computer. With Telnet, you log in to a remote computer just as if you were sitting at a locally-connected terminal on that computer. Example: telnet **192.168.2.10.**

TFTP

TFTP stands for Trivial File Transfer Protocol. TFTP is a simple protocol for transferring files. The Barionet contains a built-in TFTP server that is used to transfer COB files to the Barionet.

Trap

A Trap is a notification message, defined by SNMP (Simple Network Management Protocol) that is sent on a specified condition, such as an input state change.

Virtual I/O

The Barionet allows you to use special locations in memory as if they were real I/O registers. The most common use for virtual I/O registers is to make an external input or output that is physically contained in a Barix expansion module, such as an X8 or R6 appear as if it was a built-in input or output. You write BCL code that polls or the external input and updates the virtual I/O register, or polls the virtual I/O register and updates the physical output. Virtual I/O registers offer several advantages including simplified programming and the ability to setup state change notification on these virtual I/O registers.

Wiki

A Wiki is a type of collaborative website that allows readers to create and edit articles online via a web browser. Wikis are often used for online encyclopedias or for sharing information.

Appendix F Specifications & Warranty

F.1 Barionet 100 Specifications

I/O Interfaces

- 2 Relay outputs (240VAC 5A)
- 4 digital inputs (0 – 24V), configurable pull-up resistors
- Digital input switching thresholds (approximate, not guaranteed):
  Low-to-high: 1.4 V
  High-to-low: 3.0 V
- 4 universal inputs (analog 0 – 5V or digital 0 – 24V)
- Universal input switching thresholds (approximate, not guaranteed)
  Low-to-high: 1.2V
  High-to-low: 1.2V
- 4 digital outputs (open collector 24V 0.1A)
- 2 Barix extension connectors with Dallas 1-Wire® capability

Serial Interfaces

- 1 RS-232 (DSub-9 male connector, 5-wire interface: RxD, TxD, RTS, CTS, Ground)
- 1 RS-422 (4 wire) or RS-485 (2 wire) interface
- 600 – 19200 baud 7 or 8 bits, 1 or 2 stop bits, odd, even, or no parity

Network Interface

- RJ-45 10/100 Mbit auto-detect Ethernet interface
- Protocols supported: TCP/IP, UDP, ICMP, DHCP, AutoIP, IPZator™, Modbus/TCP, SNMP, CGI, HTTP
- Built-in web server for control, status, and configuration with proprietary dynamic tags for I/O, BCL variables, and setup parameters.

Miscellaneous

- LED indicators for:
  - Power
  - RS-232 activity
  - RS-422/485 activity
  - Digital inputs 5 – 8 status
  - Relay 1 status
  - Relay 2 status

Power Supply

- 9 – 30V DC
- 4 watts maximum (all relays active)

Physical Dimensions

- 4.13" x 3.34" x 1.1" (or 2.83" with optional tall cover)
- 105mm x 85 mm x 28 mm (or 72 mm with optional tall cover)
- Din rail mount

Operating Temperature & Humidity

- 0 - 50° C (32 - 122° F)
- 0 – 70% relative humidity, no condensing

Certifications

- FCC (A and B), CE (A and B)

F.2 Barionet 50 Specifications

I/O Interfaces

- 4 relay outputs (30 VDC, 0.5A)
- 4 digital inputs suitable for dry contacts with internal pull-up resistors.
- Dallas 1-Wire® Interface
- All I/O interfaces supplied via pin connectors with removable screw blocks provided

Serial Interfaces

- 1 RS-232 (DSub-9 male connector, 5-wire interface: RxD, TxD, RTS, CTS, Ground)
- 1 RS-485 (2 wire) interface on removable screw terminal blocks
- 300 – 230400 baud 7 or 8 bits, 1 or 2 stop bits, odd, even, or no parity

Network Interface

- RJ-45 10/100 Mbit auto-detect Ethernet interface
- Protocols supported: TCP/IP, UDP, ICMP, DHCP, AutoIP, IPZator™, Modbus/TCP, SNMP, CGI, HTTP
- Built-in web server for control, status, and configuration with proprietary dynamic tags for I/O, BCL variables, and setup parameters.

Miscellaneous

- LED indicators for:
  - Power
  - CPU Activity
  - RS-232 activity
  - RS-422/485 activity
  - Relay 1 status
  - Relay 2 status
  - Relay 3 status
  - Relay 4 status

Power Supply

- 9 – 30V DC
- 4 watts maximum (all relays active)

Physical Dimensions

- 4.13" x 3.34" x 1.25"
- 105mm x 85 mm x 31 mm
- Din rail mount

Operating Temperature & Humidity

- 0 - 50° C (32 - 122° F)
- 0 – 70% relative humidity, no condensing

Certifications

- FCC A, CE A, RoHS compliant (lead-free)

F.3 Barionet Warranty

The warranty terms are part of the General Terms and Conditions of Barix AG. The warranty period for defects due to material flaws, design errors or careless work is two years from time of purchase by the end customer. The contractual partner undertakes to notify supplier immediately in writing of any hidden defects identified during the warranty period.

The warranty does not apply for defects for which the supplier is not responsible, such as natural wear and tear, force majeure, handling and usage of the delivered product by the contractual partner or third parties which is inappropriate and contrary to regulations and purpose, interventions by the contractual partner or third parties, excessive loads, unsuitable equipment, faulty maintenance or extreme environmental influences, as well as in the event of modifications made by the contractual partner to products before reselling them to third parties.

The warranty also does not apply in the event of defects which would have been identified during the course of a proper inspection of the products, as specified in the General Terms and Conditions of Barix. The supplier undertakes at his discretion to repair as quickly as possible, or replace products which are demonstrably defective, during the warranty period as a result of the use of bad-quality materials, faulty design or poor workmanship or which do not exhibit the features contractually guaranteed. The contractual partner must provide the supplier with all relevant information for this. Replaced parts are the property of the supplier. If it transpires that the supplier is not responsible for a particular defect, the product shall be repaired by the supplier against payment.

The supplier is only ever liable for direct damage to the delivered item which has occurred at the contractual partner’s site. Liability for indirect or consequential damage, such as loss of profits, third party claims and damage caused by the contractual partner’s failure to comply with contractual obligations, is explicitly excluded.

The contractual partner does not have any further rights arising out of product defects other than those explicitly mentioned above.

F.3.1 RMA:

If a device is deemed defective, the following procedure shall apply:

a) Customer claiming for defective device shall send a failure report to the distributor

b) Distributor shall review the failure report and provide any tips or suggest rectification measures

c) After investigation, distributor may authorize the customer for returning material (distributor RMA)

d) Any product found to be defective within 30 days of receipt at the customer shall be deemed Dead on Arrival (DOA) and advance replaced by the distributor.

e) Upon receipt of the failed device, the distributor shall attempt to operate a serial rescue with standard firmware, confirm failure and eventually attempt to determine the cause.

f) If failure is confirmed, the distributor shall contact Barix support (support@barix.com) providing the failure report in accordance with (a) and (e) and all relevant information.

g) Barix support reviews the failure report and provides tips or suggests rectification measures

h) After verification Barix support may authorize the distributor for returning material (Barix-RMA). For Barix-RMA products, returns to Barix are regarded as the provision of the products at the supplier's office.

i) In the event the defect is confirmed to have been caused by user mishandling (non-intended operation, incorrect connection, etc.), the device shall be repaired on a cost basis

j) For devices demonstrably defective during the warranty period, Barix shall authorize replacement of the defective device free of charge with the next regular shipment. Replaced parts are the property of the supplier.

k) If it transpires the supplier is not responsible for a particular defect, the product shall be repaired by the supplier against payment.

Appendix G IP Addresses, Netmasks and Gateways

Communication between devices on a network running the Internet Protocol (often abbreviated IP) is based each device on the network having a unique IP address. An IP address is a 32-bit value divided into four "octets" of eight bits each. The 32-bit IP address is commonly written with each of the four octets written as decimal numbers separated by dots[5].

For example: 192.168.0.40

One of the important features of the Internet protocol addressing scheme is its ability to divide a network into "subnets" that share some portion of the 32-bit IP address for all devices on the subnet. For example, all the devices on a small office or home network, often share the same first three octets in their IP addresses. Only the last octet is different for each device on such a network. Thus, all the devices on this type of network might have 192.168.0.xx as the first three octets and then each individual computer or device on the network has a unique last octet. For example, a particular computer on this subnet might have the IP address 192.168.0.40.

The part of the IP address that is common to all the devices on a particular subnet is called the "network prefix", since that part of the address defines a unique subnet. The part of the IP address that is unique to each individual device on the network is called the "host part".

In our first example, 192.168.0 is the "network prefix" while 40 is the "host part".

This scheme of dividing a network into subnets makes managing network traffic between devices on a network much more efficient, since much of the traffic between computers and other devices occurs within a subnet and does not affect devices on other subnets.

G.1 The Netmask Parameter

Every device on an IP network has a "netmask" parameter in addition to an IP address. The netmask defines which portion of the IP address is a "network prefix" and which portion is a "host part". Netmasks are written in the same notation (four octets separated by dots) as IP addresses.

Zeros in a netmask value represent portions of the IP address that are the host part, while 1's in the netmask (expressed in binary notation) represent portions of the IP address that are the network prefix.

In our example, the netmask for a subnet where all the devices share the first three octets would be 255.255.255.0. The first three values are all 1's in binary, so the first three octets are the network prefix. The last octet is zero, so all the bits of the last octet are the host part.

The following table shows several valid netmasks and the size (in bits) of the network address portion and the host address portion of the IP address.

[5] This appendix describes IPv4 addresses which are the most common on the Internet. A newer IPv6 standard, using 128 bit addresses, instead of 32-bit addresses has been defined and is gradually supplanting the IPv4 scheme. However, most devices still implement IPv4 and most computers running modern operating system versions support both. This appendix does not address IPv6. The Barionet only supports IPv4.

Netmask	Network Prefix Bits	Host Part Bits	Maximum Number of Devices in the Subnet[6]
255.255.255.252	30	2	2
255.255.255.248	29	3	6
255.255.255.240	28	4	14
255.255.255.224	27	5	30
255.255.255.192	26	6	62
255.255.255.128	25	7	126
255.255.255.0	24	8	253
255.255.254.0	23	9	510
255.255.252.0	22	10	1022
255.255.248.0	21	11	2046
…	…	…	…
255.128.0.0	9	23	8,388,606
255.0.0.0	8	24	16,777,214

G.2 The Gateway Parameter

Whenever a device is attempting to make a connection to another device on the network, the device compares the address of the remote device to its own netmask and IP address. If the remote device's IP address is in the same subnet (i.e. it shares the same network address), a direct connection to the remote device can be made.

Going back to our example, assume that the Barionet has an IP address of 192.168.0.40 and a netmask of 255.255.255.0. For the sake of this example, assume that the Barionet wants to send an SNMP trap message to a device at 192.168.0.32. Since this device is on the same subnet, (it shares the 192.10.0 portion of the IP address) the trap message can be sent directly to the device at 192.10.0.32.

If, however, the trap message is going to a device at 192.10.12.15, the remote device is not in the same subnet (remember, all devices in the Barionet's subnet share the same first three octets, because the netmask is 255.255.255.0). When the remote device is not in the same subnet, the Barionet must communicate through the "gateway", which is a device that routes traffic between two subnets. Thus, the Barionet contacts the gateway and asks it to send the message to the remote IP address. The gateway handles the routing required to deliver the message to the remote subnet.

The gateway parameter specifies the IP address of the gateway that routes traffic to another subnet. The gateway IP address must always be in the same subnet as the device. The gateway device actually has at least two IP addresses—one on the local subnet, and one on the other subnet(s) it routes to).

If you are statically assigning an IP address to the Barionet, and it will need to connect to other computers or devices outside its own subnet, you must fill in the gateway IP address. If the Barionet is not communicating with any devices outside its own subnet, you can leave the gateway parameter blank.

If the Barionet is obtaining its IP address automatically from a DHCP server, the DHCP server will assign the IP address and netmask, and may also supply the gateway address.

G.3 Special IP Addresses

Some ranges of IP addresses have special meaning, and therefore aren't available for

[6] Keep in mind that two of the host addresses in the range (all zeros and all ones) are not valid for assignment to devices, so the maximum number of hosts is actually two less than the number of host addresses in the host address range. The next section discusses these special IP addresses that are not available for assignment to a device.

assignment to a device on the subnet.

An IP address with a host address portion of all 1's implies a "broadcast" to all devices on the subnet. If the Barionet has an IP address of 192.10.0.40 and a netmask of 255.255.255.0, the broadcast address for this subnet is 192.10.0.255. No device on the network, including the Barionet can have an IP address of 192.10.0.255.

The Barionet's syslog function is one common example of the use of a broadcast address. If the Barionet's syslog server parameter is set to all zeros (0.0.0.0), it will use the broadcast address for its own subnet to transmit syslog messages. That means that any device on the same subnet that is listening to the defined syslog port (Port 514) will receive the syslog messages. This is useful when you are first establishing communication with the Barionet, because it will transmit the start-up syslog messages to any computer on its subnet without defining the syslog server IP address.

Addresses with all zeros in the host portion are also reserved for special network functions. These addresses cannot be assigned to a network device, including the Barionet. Keep in mind, however, that some of the Barionet network parameters use a special default value of 0.0.0.0 to indicate automatic address assignment or other special settings. This is not the actual address—it's just a special value indicating to the Barionet that it should attempt to acquire a valid IP address automatically. Refer to the sections that define these specific configuration parameters for more information on the valid values and defaults.

The address 127.0.0.1 is also a special address that is called a "loopback" address. This is not a valid address to assign to the Barionet.

G.4 Private IP addresses and the Internet

Because of the limited range of IP addresses in the most common version of the Internet protocol[5], a series of "private" IP address ranges have been defined that allow any number of private networks to use the same range of IP addresses. These "private" IP ranges are translated to one or more "public" IP addresses for access to the internet using a protocol called "Network Address Translation" or NAT. Most modern routers used in small home or office networks implement NAT so that a single public IP address can be shared by an entire network of computers using a range of these private IP addresses.

The same principles apply to the assignment IP address, netmask, and gateway IP address in a private network. The primary difference is that private network IP addresses don't have to be unique across the entire internet, while public IP addresses must be unique.

If the network you're connecting your Barionet to is not connected to the internet, you can choose any address range you like. However, if the network is, or will be connected to the internet in the future, it's wise to use one of the private IP address ranges and a NAT-capable router/firewall for connecting to the Internet. If you have questions about these networks, and the appropriate address to assign to a Barionet, contact your network administrator.

The table below shows the range of private IP addresses defined for each of the three "classes" of private networks.

Class	IP Addresses	Netmask	Max number of devices
A	10.x.x.x	255.0.0.0	16,777,214
B	172.16.x.x	255.255.0.0	65,534
C	192.168.0.x	255.255.255.0	254