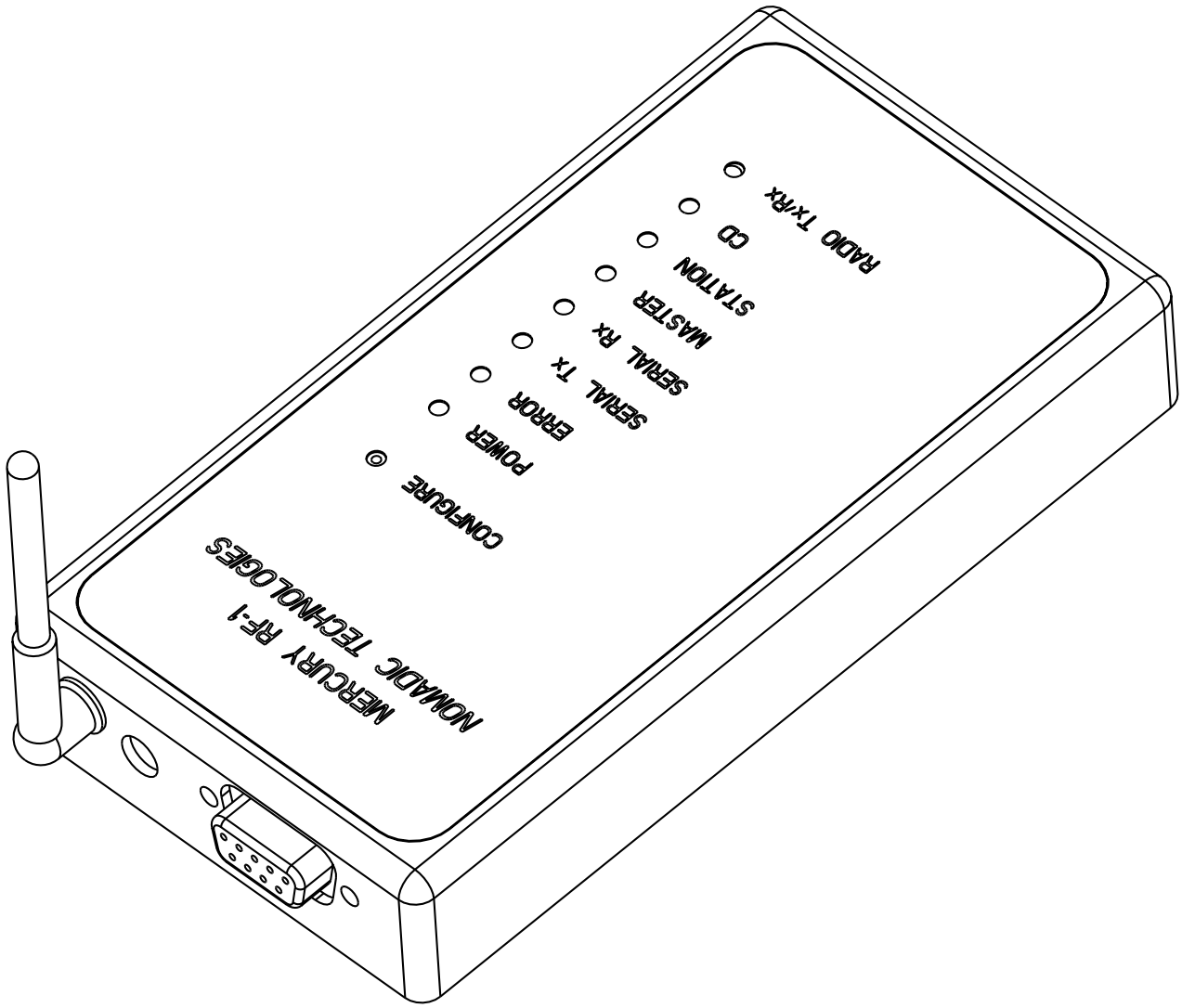
Mercury

User's Guide

Version 2.3



Copyright

© 1996-99 Nomadic Communications, Inc., Mountain View, CA, USA.
All rights reserved. This manual and the software described in it are copyrighted with all rights reserved. No part of this publication may be reproduced, transmitted, transcribed, stored in a retrieval system or translated into any language in any form by any means without the written permission of Nomadic Communications, Inc.

Trademarks

Nomadic Communications, Mercury-RF1, Mercury-EN, and the Pyramid and Lightning Bolt logos are trademarks of Nomadic Communications, Inc. RangeLAN, RangeLAN2, RangeLINK, ProxLink, and Proxim are trademarks of Proxim, Inc. All other trademarks are the property of their respective owners.

Limited Warranty, Disclaimer, Limitation Of Liability

For a period of one (1) year from the date of purchase by the retail customer, Nomadic Communications, Inc. (Nomadic) warrants Mercury products against defects in materials and workmanship. Nomadic will not honor this warranty if there has been any attempt to tamper with or remove the foil FCC label on the bottom of the unit.

This warranty does not cover and Nomadic will not be liable for any damage or failure caused by misuse, abuse, acts of God, accidents, or other causes beyond Nomadic's control, or claim by other than the original purchaser.

If, after inspection, Nomadic determines there is a defect, Nomadic will repair or replace your Mercury product at no cost to you. To return defective merchandise to Nomadic please call Nomadic Communications Customer Service at: 650/988-7200 to obtain a Return Merchandise Authorization (RMA) Number.

In no event shall Nomadic Communications, Inc. be responsible or liable for any damages arising:

- From the use of the product;
- From the loss of use, revenue or profit of the product; or
- As a result of any event, circumstance, action, or abuse beyond the control of Nomadic Communications, Inc.;

whether such damages be direct, indirect, consequential, special or otherwise and whether such damages are incurred by the person to whom this warranty extends or a third party.

WARRANTY RETURN POLICY

If you have a problem with your Mercury product, please call Nomadic Communications Technical Support at 650/988-7200. Nomadic Communications Technical Support will assist with resolving any technical difficulties you may have with your Mercury product.

After calling Nomadic Communications Technical Support, if your product is found to be defective, you may return the product to Nomadic after obtaining an RMA (Return Merchandise Authorization) number from Nomadic Communications Customer Service. The product must be returned in its original packaging. The RMA number should be clearly marked on the outside of the box. Nomadic cannot be held responsible for any product returned without an RMA number, and no product will be accepted without an RMA number.

RIGHT TO CHANGE

Nomadic Communications, Inc. reserves the right to make changes without notice to any products herein for any reason at any time, including but not limited to improving the reliability, form, fit, function or design. Nomadic Communications does not assume any liability arising out of use, misuse or application of any product or circuit describe herein, nor does it convey any license under its patent rights nor the rights of others.

Information and specifications in this manual are checked, however no responsibilities for inaccuracies can be assumed by Nomadic Communications. Please consult a Nomadic Communications salesperson to obtain the latest specifications before placing your order for Mercury products.

LIFE SUPPORT POLICY

Nomadic Communications' products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Nomadic Communications product could create a situation where personal injury or death may occur. Authorization for such use may only be given in the form of express written approval of the president of Nomadic Communications.

Should the buyer or user purchase or use Nomadic Communications products for any such unintended or unauthorized application, both buyer and user shall indemnify and hold harmless Nomadic Communications and its officers, employees, subsidiaries, affiliates, suppliers, and distributors against all claims, costs, damages, and expenses, and attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Nomadic Communications was negligent regarding the design or manufacture of the device, or was aware of a defect that could cause malfunction.

FCC WARNING

This equipment has been tested and found to comply with the limits for a Class B digital device, pursuant to Part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation. If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

- ❑ Reorient or relocate the receiving antenna.
- ❑ Increase the separation between the equipment and the receiver.
- ❑ Connect the equipment into an outlet on a circuit different from that which the receiver is connected.
- ❑ Consult the dealer or an experienced radio/TV technician for help.

EUROPEAN TELECOMMUNICATIONS STANDARDS INSTITUTE

Statement of Compliance
Information to User

This equipment has been tested and found to comply with the European Telecommunication Standard ETS 300.328. This standard covers Wideband Data Transmission Systems referred to in the CEPT recommendation T/R 10.01. This type of accepted equipment is designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications.

Table of Contents

CHAPTER 1 - INTRODUCTION

Thank you for your purchase of the Mercury Serial Wireless Network Interface. The Mercury is a small communications device that replaces RS-232 and/or 10Base-T Ethernet cables with wireless RF (Radio Frequency) technology.

In many environments, the Mercury is a "plug-and-play" product. All you need to do is attach a pair of Mercury units to any two devices with three wire asynchronous RS-232 or 10Base-T Ethernet ports and you can transmit and receive data without the use of wires between the devices (See Figure 1-1).

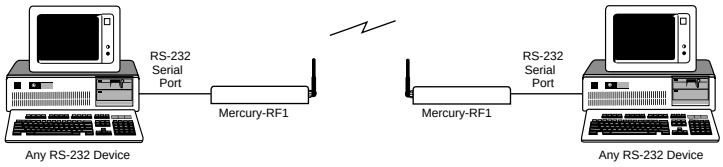


Figure 1-1 Mercury units used to replace an RS-232 or Ethernet cable connection

The Mercury uses a frequency hopping, spread spectrum radio for performing wireless communication. It operates within the FCC's guidelines for spread spectrum systems, and unlike conventional (narrowband) systems, its use does not require an FCC site license.

The Mercury supports a wide variety of configurations that can easily be changed to fit your application requirements. All configuration information is saved in non-volatile memory called FLASH. Both the Mercury-RF1 and Mercury-EN support RS-232 serial data rates from 112.5 up to 230,400 bps (bits per second) full duplex, and the Mercury-EN additionally supports 10Mbps Ethernet. Mercury units also offer unit-to-unit configuration, multiple network domains, radio channels, and software subchannels to allow for non-interfering overlapping systems.

System Requirements

To begin using your Mercury radio device, you will need the following minimum system requirements:

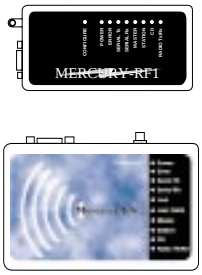
- ❑ Two or more Mercury radio units, and
- ❑ Two or more devices (with RS-232 or 10Base-T Ethernet communications ports) that need to communicate with each other.

Packing List

In this package, you will find the following components:

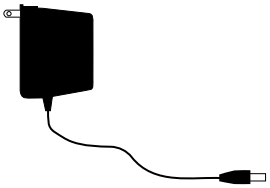
- ❑ One Mercury Serial Wireless Network Interface,
- ❑ One 2.4 GHz antenna with custom connector,
- ❑ One 7.5V AC to DC power adaptor,
- ❑ One RS-232 serial cable, and
- ❑ One Mercury User's Guide (the one you are presently reading).

If any of these items are missing or damaged, please contact the place of purchase or Nomadic Communications Customer Service.

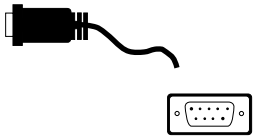


Mercury-RF1 *or* Mercury-EN

2.4 GHz Antenna



Power Adaptor



RS-232 Serial Cable

Figure 1-2 Included components

CHAPTER 2 - SERIAL WIRELESS NETWORK TOPOLOGIES

Numerous serial wireless network topologies can be established with the Mercury. Although many exist, all can be simplified into either Point-to-Point or Broadcast topologies.

Note: When using a Mercury-EN strictly for Ethernet communication, these topologies do not apply: they describe RS-232 serial communication only. Ethernet communications behave as though all computers are connected to the same network, where data is mixed between broadcast and directed.

POINT-TO-POINT TOPOLOGY

In this topology, pairs of devices in the system are configured to communicate with each other exclusively.

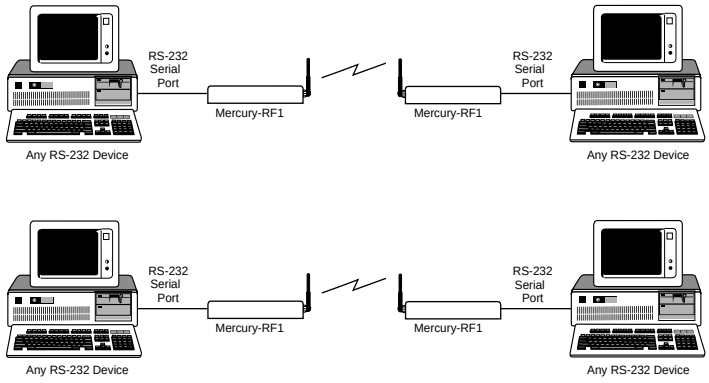


Figure 2-1 Point-to-Point Topology

In Figure 2-1, devices 1 and 2 can communicate with each other, as can devices 3 and 4. Even though all four units may be in range and may all hear each other's messages, each unit will filter out messages that are not intended for it. In this topology, each pair of units simply replaces an RS-232 cable. This topology is the most commonly used in a wide variety of applications.

When configured in a Point-to-Point topology, a Mercury unit that receives a message on its serial port will turn the message into a network packet. To this packet the unit adds its own address and the destination unit's address (as stored in the configuration or passed along in the serial message), as well as other critical information.

Note: With a Mercury-EN, a network packet can be generated on any computer that has TCP/IP support and is connected to a wired network. The Mercury-EN will process it as though it were a serial data packet (as follows).

Next, it "contends" for time on the airwaves, then transmits the packet over the radio. Finally, the unit waits for an acknowledgment from the destination unit that the packet was received without error. If it does not receive such an acknowledgment and it has not exceeded its maximum retry count, the unit will retransmit the packet. This acknowledge and retry mechanism provides for error-free communication.

Any unit that receives packets over its radio will filter them based on the destination address. Only the unit with the correct address will save the packet and send an acknowledgment back to the source unit. Upon receiving a packet the unit also extracts the original message out of the packet and sends it out its serial port. If there was an error in the packet, the unit will ignore it causing the sender to retry. This guarantees the receiving device only receives error-free transmissions.

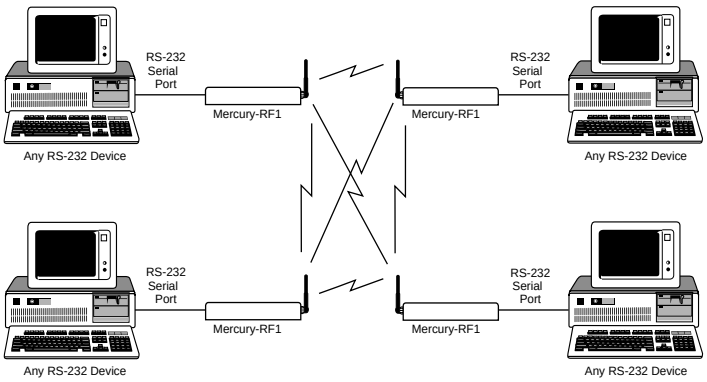


Figure 2-2 Broadcast Topology

BROADCAST TOPOLOGY

In this topology a group of Mercury units is configured to have the same specified broadcast address. In Figure 2-2, whenever a message is sent from any unit in this group, all other units in the group that correctly receive the message send it out their serial port. This is also sometimes referred to as a "party-line" or "multi-drop" topology.

As in Point-to-Point topology, the device that is transmitting a message over its radio forms a packet, however the packet specifies a broadcast address rather than a specific unit's port address. Broadcast topology is generally used only when one device needs to send messages to multiple units, something that is usually not possible with conventional RS-232 communications. Broadcast topology is not as common as Point to Point topology, because most existing software that use RS-232 communication are designed for communication with a single device on the other end of the serial line.

Unlike with Point-to-Point topology, after the Mercury sends the packet it does not wait for an acknowledgment. Broadcast mode is an unacknowledged service for two reasons: 1) because the broadcasting unit doesn't know who is going to be receiving the data, and 2) even if it did know, it can be extremely inefficient to have every unit acknowledge a message once the packet is received. Since broadcasts are unacknowledged, a unit cannot retry transmissions while configured for this topology.

Any Mercury unit that correctly receives the packet compares the packet's broadcast address to the unit's own broadcast address. If the two addresses match, the unit will extract the message and send it out its serial port. This guarantees that the user's devices only receives error-free transmissions.

Note: Since broadcasting is an unacknowledged service, if a message was lost or damaged, the message will not get through. Therefore, the user's devices should contain software to perform message retransmissions should this be necessary. Often devices that can use wired networks already contain such software.

Note: The TCP/IP network protocol is not capable of broadcast operation, due to the connection-based design of the protocol. If broadcast operation is desired, use of the RMP protocol is necessary.

HYBRID TOPOLOGIES

Although Point-to-Point and Broadcast topologies are the most common, the Mercury allows for a great variety of hybrid topologies. A common type of hybrid topology is the client-server topology. In this layout, one Mercury is connected to a server and multiple Mercury's are connected to clients. The server Mercury is configured for Broadcast (allowing communication to all of the clients), but each of the client Mercuries are configured for Point-to-Point (allowing communication only to the server: the clients don't hear each other's messages).

CHAPTER 3 - SERIAL PORT PROTOCOLS

The Mercury's serial interface can interpret and process serial data in one of five ways: passthrough, passthrough2, ProxLink, Telnet, and LPD. The format of the information presented to the unit's serial port is different depending on which of these five modes is selected.

Note: Not all protocols are available on all units. Procotols that require the use of the TCP/IP network protocol (passthrough with TCP, passthrough2, Telnet, and LPD) require that the TCP/IP option be purchased separately on Mercury-RF1 units. This restriction does not apply to Mercury-EN units.

PASSTHROUGH

For applications where an RS-232 cable is to be replaced by a pair of Mercury units without changes to the existing network protocol, or when Mercury units are being used to form a broadcast or client-server topology, the passthrough protocol should be used. With the passthrough protocol, the unit accepts a stream of serial data at its serial (RS-232) port and passes it over the radio network (via either the RMP or TCP/IP network protocols) to one or more receiving units. The data arrives at the receiving unit(s) which then transmits the data out of its serial port(s).

With this protocol, data is sent transparently to the receiving unit, as though the units were directly connected via serial cables. The data is not filtered or interpreted by either of the Mercury units. This protocol is most useful if the Mercury is to be used as a drop-in replacement for serial cable; that is, that the device the Mercury is connected to does not know that its data is being transmitted by radio.

The passthrough protocol has several configuration options, some designed to optimize data throughput. These options tell the Mercury when to transmit the data. They are necessary because there is communication overhead in transmitting the data over the radio; sending one hundred (100) bytes of data takes roughly half the amount of time as it takes to send one thousand (1000) bytes. Through configuration, the user can help the Mercury decide how to minimize this overhead. Configuration details are covered in Chapter 6.

PASSTHROUGH2

The passthrough2 protocol, like the passthrough protocol, makes no changes to the serial data stream as it is transmitted or received. It differs in the fact that it creates two independent network connections for transmitted and received data, allowing transmitted data to be sent to a different location than received data originates from.

The passthrough2 protocol is only available using the TCP/IP network protocol (it isn't needed for RMP due to the fact that RMP does not restrict the source and destination locations of data as the TCP/IP protocol does). Normally with the passthrough protocol, one bi-directional TCP/IP connection is made to a remote computer (or another Mercury). Since there is only one connection, data must travel in both directions on it. In order for the destination of data to differ from the source of data, two connections must be made. The passthrough2 protocol allows this to happen. Please see Chapter 6 for more details on configuration of this protocol.

PROXLINK

The ProxLink protocol is designed to allow better control of the Mercury's data stream by wrapping the data to be sent or received in a package that designates the source or desired destination of the data. This allows the user to write application software that needs to communicate to multiple remote sites simultaneously without having to broadcast all data to all locations.

This protocol also allows control over hardware settings of the Mercury such as baud rate, radio domain and channel, low power states, etc. Details on the format of this procotol are available in Chapter 7.

TELNET

Telnet is a protocol that uses the TCP/IP networking protocol. Originally designed for UNIX-based operating systems, it allows a person on one computer to run applications or otherwise control a remote computer. The Mercury provides a "telnet prompt" much like that used on UNIX systems. From this prompt, a user can open connections to any computer that is connected to the same network as

the Mercury unit and is running Telnet daemon software.

At this time, there are only two commands that the telnet prompt recognizes. The first is "**help**". This gives a short on-screen instruction about the commands. The other command is "**open**". "**open**" takes one mandatory argument, the IP address of the machine to connect to. It also takes one optional argument, the TCP port number to open the connection on. If the port number is omitted, the default value of 23 is used.

There are some configuration options for Telnet, such as forcing a destination IP address and offering a connect prompt. These features are designed for extending dumb terminals into Internet terminals while setting restrictions on the destination of the connection. Please refer to Chapter 6 for more information.

LPD

LPD is a printer driver protocol used mainly by UNIX based systems. This is uni-directional protocol: data is received on the network connection and sent out the serial port. Data received on the serial port with this protocol is ignored. Mercury's configured to use this protocol can buffer up to approximately 32K of data from the network. The document is printed while it is being received on the network.

To use this protocol, you must have a UNIX, Windows NT Server, or AS/400 system (or other operating system that supports LPD) with a printer configuration set up to use the Mercury as a remote host. Configuration of the printer varies from one brand to the next; we recommend that you consult documentation that came with your system to properly configure the printer and Mercury for serial settings and flow control. Your operating system must put the data to be printed into a format that the printer will recognize. The Mercury does not perform data translation or formatting.

There is no configuration for LPD on the Mercury. When configuring your operating system, use the network hostname or IP address of the Mercury as the remote printer host (UNIX systems should be set up using the Mercury as a remote LPD capable workstation), and any name for the printer. See Application Note #1 for specific instructions on configuring a Windows NT Server platform for LPD printing.

CHAPTER 4 - ETHERNET BRIDGE NETWORK TOPOLOGIES

Note: This chapter pertains to the Mercury-EN only. If you do not have a Mercury-EN, please skip to Chapter 5, Installation.

This chapter describes the bridging capabilities of the Mercury-EN.

Mercury Ethernet bridging is a plug-and-play operation. No configuration is needed to set up any of the bridge topologies described below.

The only configuration that may be required would be to set different domains and/or channels to prevent interference from other Mercury networks. Other optional configuration, such as designating a Master, may optimize network performance. Please see Chapter 6, Configuration, and Appendix F, Performance Hints, for more details.

BASIC (SINGLE BRIDGE NODE) ACCESS POINT

The simplest use of a Mercury-EN as a bridge involves connecting a single Mercury-EN to a device[1] or network with a 10Base-T interface. Any other Mercury station that is within radio range and is set to the same domain[2] can then be accessed by any device connected to the Mercury-EN.

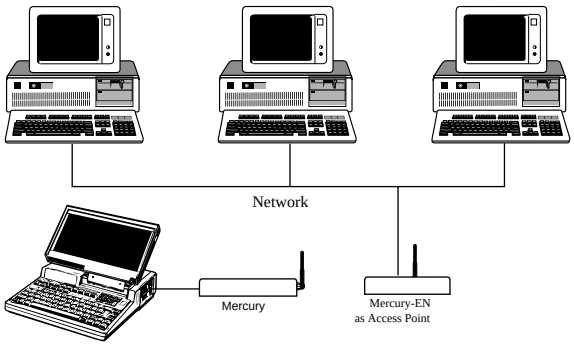


Figure 4-1 Mercury used as Access Point (and devices on the network)

[1] When connecting a Mercury-EN 10Base-T port directly to another device (other than a network hub or switch), you will need an "inter-connect" cable (sometimes known as a "crossover" or "null-Ethernet" cable). Many network supply dealers can provide you with one.

[2] For information about the domain setting, see "Using Domains to Separate Networks", below, or Chapter 6, Configuration.

CELLULAR (MULTIPLE BRIDGE NODES) ACCESS POINTS

To use the basic access point topology, it is necessary for all stations to be within radio range of the access point. This is not always possible if the range of the Mercury is insufficient to cover the area required by all stations. The solution is to add additional access points to the wired network. Each access point will cover a certain amount of the required area, allowing stations in range of any of the access points to communicate with the network and each other. Additionally, as stations move around, they may leave the range of one access point and enter another's range. This process is called roaming (much like roaming of cellular telephones: when one cell site becomes out-of-range, the phone will find another to continue the call with).

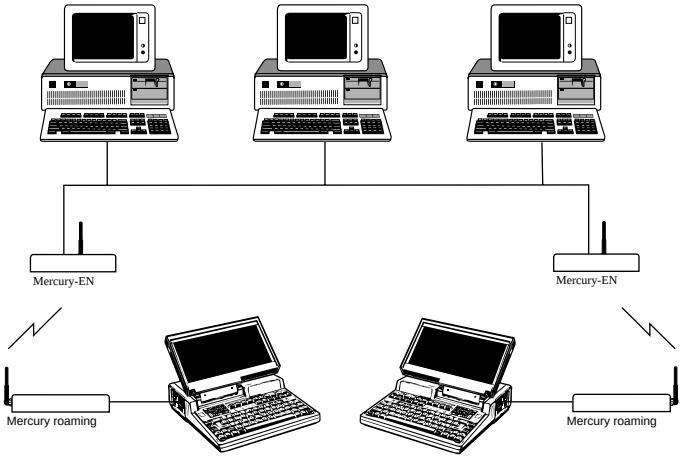


Figure 4-2 Multiple Mercury's used as Access Points (with network devices roaming)

This topology is used in the case where there are two independent wired networks that need to be joined. For example, in Figure 4-3, each building has its own network. Since the two buildings are used by the same company, computer resources between the buildings need to be shared. Using two Mercury-EN to connect the networks eliminates the need to run wire between buildings (potentially down the street from each other) or involve the local phone company.

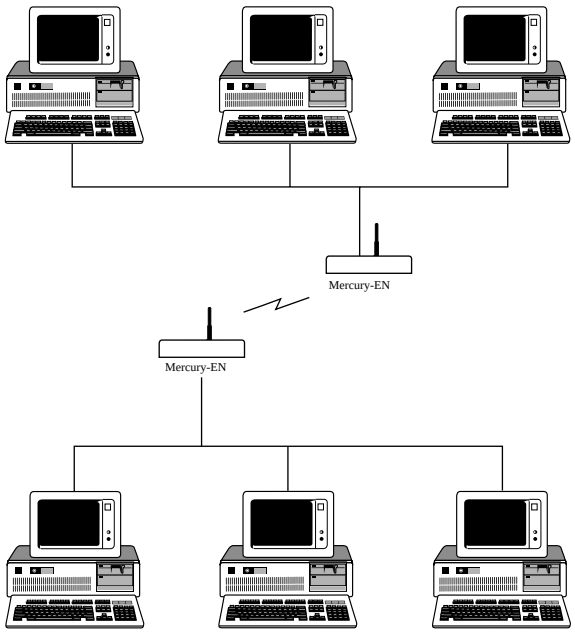


Figure 4-3 Two buildings' networks connected together by Mercury-EN's

USING DOMAINS TO SEPARATE NETWORKS

Frequently there can more than one network in a given physical location. If so, there needs to be some way of differentiating which network the Mercury is communicating to. This is done with domains. A domain is a number between 0 and 15 that uniquely identifies a wireless network. Only Mercury units with the same domain number will communicate with each other.

In Figure 4-4, the Mercury connected to network "A" has a domain number of 5. The Mercury connected to network "B" has a domain number of 3. The mobile Mercury unit that also has a domain number of 5 can only communicate on network "A". The Mercury on network "B" will not hear any of domain 5's messages.

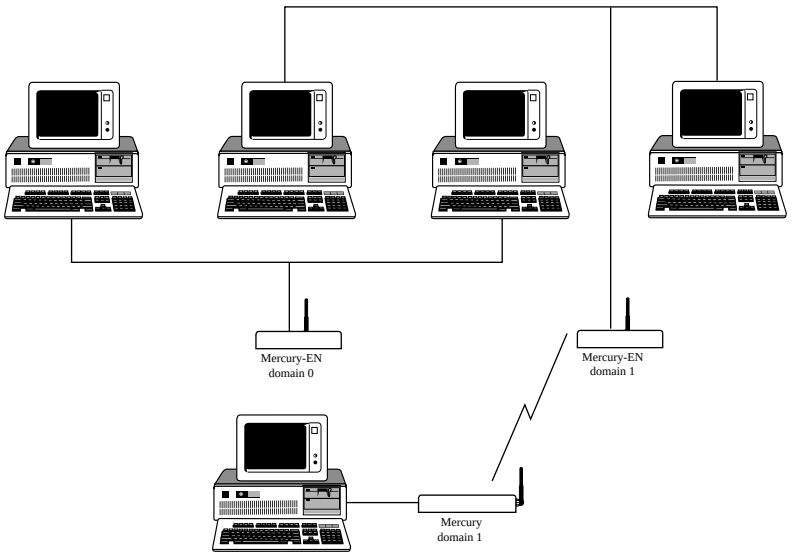


Figure 4-4 Two overlapping networks, separated by domain

COMBINING BRIDGE AND SERIAL TOPOLOGIES

The Mercury-EN contains both bridging and serial interface functionality. It is possible to use both features at the same time. Figure 4-5 shows Point-of-Sale (POS) terminals connecting to a server using the 10Base-T connection while simultaneously connecting printers to the serial port. With this configuration, the server can access not only every POS terminal, but also all of the printers (even if the terminals are powered off).

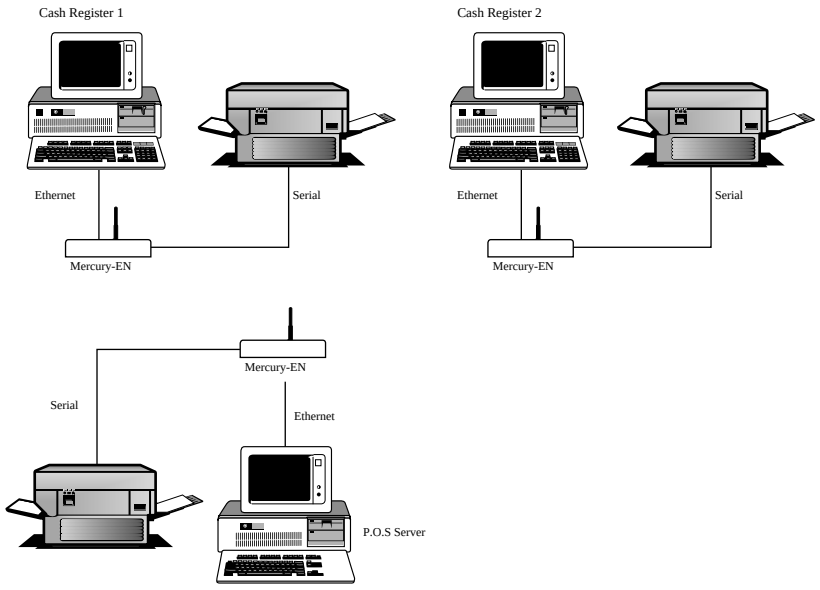


Figure 4-5 Point of Sale terminals with external printers

CHAPTER 5 - INSTALLATION

The Mercury package includes the following items:

- ❑ Mercury-RF1 or Mercury-EN unit
- ❑ 2.4 GHz omnidirectional antenna
- ❑ 7.5V DC power supply
- ❑ RS-232 serial cable

Additional Hardware Requirements:

- ❑ Any device with an RS-232 port (Terminal, PC, Instrument, etc.)

Configuration Requirements

- ❑ RS-232 Serial Terminal or
- ❑ PC with terminal emulation software

INSTALLATION PROCEDURE

Follow the steps below to install the Mercury.

1. Attach the antenna using the custom connector on the front of the unit.

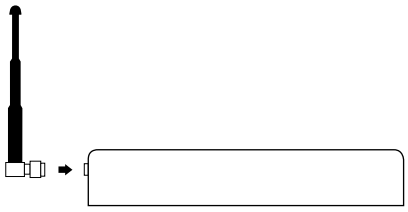


Figure 5-1 Attachment of the Mercury antenna

Note: *Take care not to over tighten the antenna attached to the connector. This will void the warranty.*

2. Attach the RS-232 Cable to the Mercury and then to your RS-232 serial and/or 10Base-T Ethernet device.
3. Plug the power supply into the Mercury power jack and plug the power supply into an AC outlet.

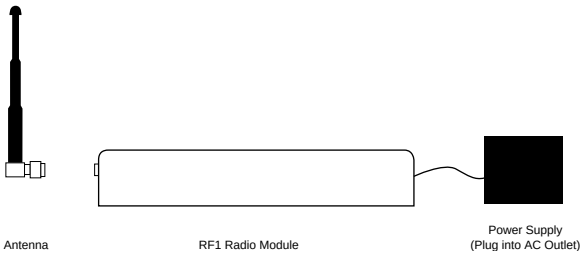


Figure 5-2 Mercury radio module hardware installation

After you have completed step 3, the LED marked “Power” should light continuously, and the LED marked "Error" should be off. If the error LED lights up, then an error has occurred. See Appendix G to troubleshoot this problem.

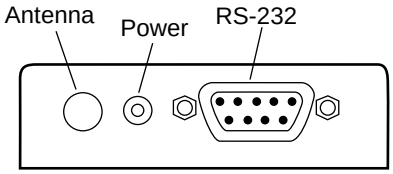


Figure 5-3 Front view of Mercury-RF1 radio unit

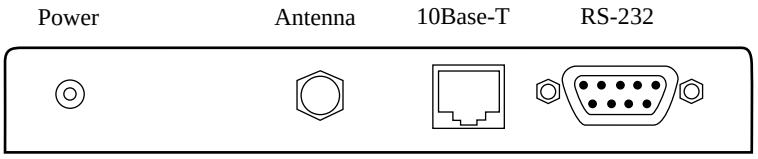


Figure 5-4 Front view of Mercury-EN radio unit

CHAPTER 6 - CONFIGURATION

REQUIRED EQUIPMENT

- ❑ Mercury unit.
- ❑ RS-232 serial terminal or PC with RS-232 port running a terminal emulation software package. (e.g. HyperTerminal, ProComm, or Telix)

Note: *The Mercury comes with a DB-9 male to DB-9 female RS-232 cable. Attach the DB-9 male side (the one with the pins) to the Mercury and attach the DB-9 female connector (the one with the holes) to the terminal or PC. Gender changers or DB-9 to DB-25 converters may be required depending on your terminal's serial port connector type.*

THINKING AHEAD

Now it's time to plan how your Mercury units will communicate with each other and your network. If you are only planning on using a Mercury-EN for bridging in a Basic Access Point topology or two Mercury units as a replacement for RS-232 cables, you may skip this entire chapter and read Appendix A, which gives instructions for these specific configuration types. However, if you need to make changes to baud rate, radio domains, etc., you will want to at least skim this chapter.

1. Select one of 16 domains. The domain number separates Mercury units into separate logical radio "networks" so that units in different domains cannot interfere with each other over the air.
2. Select one of 15 radio channels. Only a unit acting as the Master for its configured domain will use this setting. All other units will adapt to the Master's setting.
3. Select the node type this unit will be. Choices are Master, Alternate Master, and Station. There must be exactly one Master at a time in any given domain. As many units as necessary can be configured as Alternate Masters if no single unit is guaranteed to always be present, or if there needs to be a backup in case the designated Master malfunctions. See Appendix F for for information about this selection process.

4. Choose an IP address. If the Mercury is joining an existing network, be sure to ask your network administrator for an available address. If the Mercury is operating standalone (not within a network), the IP address can be anything that is allowed within TCP/IP addressing rules. If you are planning on only using the RMP protocol, you can skip this step.
5. Select the serial and network communication protocols. See Chapter 3 for your options. Also, if you are using the passthrough protocol with TCP, it would be a good idea to have an understanding of how TCP/IP network configuration works. If you are at a site that has UNIX workstations or Internet access, there may be a person there who can help you.
6. Choose the appropriate serial port parameters (baud, bit count, parity) to match your application.

Bringing Up the Main Menu

There are two ways to bring up the Main Menu. The first is to open a Telnet connection to the Mercury on port 23 (23 is the default for most Telnet programs). However, this only works after the Mercury has been assigned a TCP/IP address. The second method is as follows:

1. Attach the terminal or PC to the Mercury using the included RS-232 cable. (If you are using your own cable, make sure that pins 1 and 9 are not connected; otherwise, a malfunction may occur.)
2. If you are using a PC, start your terminal emulation software.
3. The PC or terminal must be set to the following to configure the Mercury: 9600 baud, 8 data bits, no parity, and 1 stop bit.
4. Connect power to the Mercury.
5. Put the Mercury into configuration mode by pressing the "Configure" button on the top of the unit. The button is located in a hole next to the word CONFIGURE on the unit's label. You must use a paper clip or other thin metal object to press the button. (Do not use a plastic stick to press the button; plastic may get jammed in the hole or break off.) The Mercury should respond by displaying the MAIN MENU screen as in Figure 6-1.

If the Mercury does not respond within a few seconds, disconnect power for a few seconds, then reconnect power and press the configuration button again. If the terminal displays random characters, check the baud rate and bit settings in your terminal emulation software to insure 9600 baud, 8 data bits, no parity, and 1 stop bit.

If the Mercury still does not respond with the configuration mode main menu, verify that there is not a cable problem by observing the green “Serial TX” LED when pressing the ENTER key. Each time the key is pressed, the "Serial TX" light should blink faintly and quickly. If is not blinking, there may be a problem in the cable connection. See Appendix C for more information on the serial cable pinout.

If the "Serial TX" light blinks when the ENTER key is press and the unit does not respond, check to see if the serial configuration is set to 8 data bits, no parity, 1 stop bit.

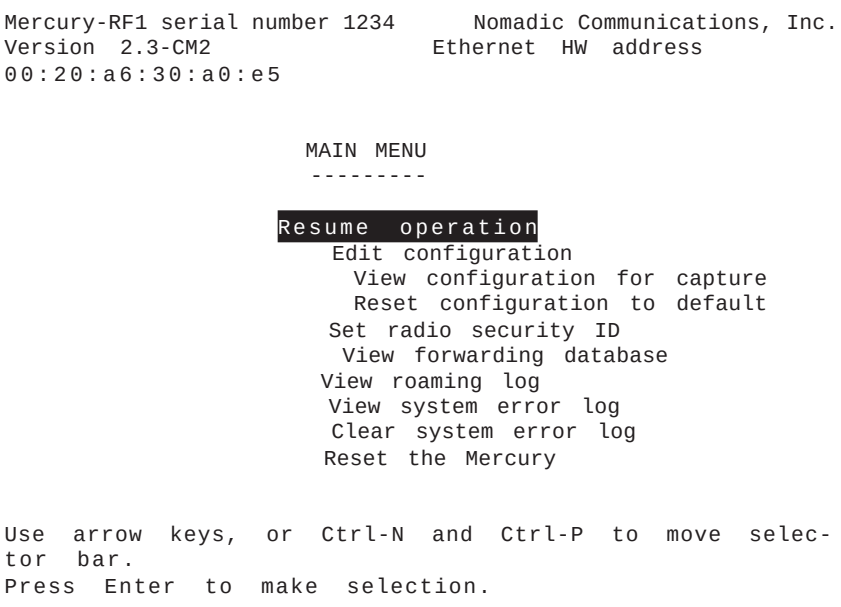


Figure 6-1 Mercury unit configuration mode main menu

6. Once the main menu is displayed, use arrow keys to move the highlighted bar. If the arrow keys don’t work, you can move the bar by holding down the Control key while pressing N (for Next) and

P (for Previous) to move the bar. To select an entry, press the Enter key.

Note: *For X Window System users. If you are using configuration by telnetting from an xterm, you can select menu items by clicking on them with your mouse.*

7. To modify the configuration (as described below), select the menu item "Edit configuration". Another menu, listing available files to edit, will then be displayed. Selecting one of them will bring up an editor that you can use to modify the file. File selection and editor operation are described below.

8. After you have finished configuring the Mercury, select the "Reset the Mercury" menu item, and then answer "Yes" to the confirmation. This will reset the device, allowing the new configuration to take effect as well as place it into operating mode. Now you are ready to use your new configuration..

MAIN MENU OVERVIEW

Here is a list of the menu selections and what they do:

RESUME OPERATION

This option exits Configuration. It returns the Mercury to the state it was in before the Configure button was pressed.

EDIT CONFIGURATION

Brings up a list of files to edit. Descriptions of the files and their contents are below.

VIEW CONFIGURATION FOR CAPTURE

Displays all configuration settings in one ASCII dump. If you select this option, it will give you an opportunity to enable capture mode in your terminal software. It will then display all configuration settings and give the option to disable capture mode. This option can be used to keep a record of what settings you have made for a particular Mercury unit, or to generate a file for Nomadic Communications Technical Support if you have any difficulties.

RESET CONFIGURATION TO DEFAULT

Sets all configuration files to their factory default. It will prompt you to ask if you are sure you want to make the changes.

SET RADIO SECURITY ID

Changes the security value, which is a 20-character ASCII string. This values prevents units with different settings from communicating with each other. Also, once a value is set, it cannot be retrieved.

VIEW FORWARDING DATABASE

Lists the MAC addresses of all network nodes detected, and which network interface they were last seen on.

VIEW ROAMING LOG

Lists the MAC addresses of the access points the Mercury has associated/disassociated with.

VIEW SYSTEM ERROR LOG

Shows a list of errors if any have occurred. Use this option if the Mercury's Error LED is lit to see what kind of error the Mercury is generating. See Appendix D for a list of the error messages.

CLEAR SYSTEM ERROR LOG

Removes all messages from the error log described above.

RESET THE MERCURY

Performs a hardware reset. Use this after making configuration changes to allow the changes to take effect.

EDIT CONFIGURATION MENU

The Edit Configuration menu contains three selections.

RETURN TO MAIN MENU

Goes back to the previous menu selections.

SYSTEM

Brings up the editor screen with the configuration file for options that are not communication dependent.

RS-232 PORT (UART 0)

Brings up the editor screen with the configuration file for the serial port and per-connection network settings.

RANGELAN2 ETHERNET / BRIDGED ETHERNET (LAN 0)

Brings up the editor screen with the configuration file for the radio parameters and IP network interface settings.

ENTER TCP/IP LICENSE CODE (may not display on some units)

Use this menu option if you purchased a Mercury-RF1 without TCP/IP support, and have since acquired an upgrade option from Nomadic Communications.

THE EDITOR

Selecting one of the configuration files above will bring that file into the editor. Once inside the editor, you may use arrow keys to move the cursor around. If the arrow keys don't work with your terminal emulator, you can use Ctrl-P for up [previous], Ctrl-N for down [next], Ctrl-B for left [back], and Ctrl-F for right [forward] for cursor motion.

For faster motion, you can use Ctrl-A to jump to the beginning of the line, and Ctrl-E to jump to the end. (People familiar with the Emacs editor should feel at home with these keystrokes.)

To make changes, simply move the cursor to the point you want to change and type. To delete text behind the cursor, move the cursor to the position immediately following the character to remove and press either the Backspace or Delete keys, or type Ctrl-H. To delete text in front of the cursor, press Ctrl-D. To delete text from the cursor to the end of the line, press Ctrl-K.

To save changes, press Ctrl-W. After the changes are saved (it will take less than one second), you will be returned to the Edit Configuration menu. Although changes will be saved, they will not take

effect until you power the Mercury off and back on. If you decide that you don't want to save the changes you have made, press Ctrl-X. It will ask for confirmation and then return you to the Edit Configuration menu.

If while typing the screen display becomes corrupted or confused (as frequently happens with many terminal emulation software packages that don't emulate VT100 correctly), you can press Ctrl-L to force a screen redraw.

Configuration File Format

People familiar with the Windows **WIN.INI** file format will recognize the format of the file. It is broken down into sections; sections define a particular grouping of options. Each section contains at the top a section header which is a string of text surrounded by square brackets: [ ]. This is the "section title". After each section header, there is a list of entries containing equal signs, the text before the equal sign is a "key" and the text after the equal sign is the value. Changing the value of different keys is how configuration changes are performed. For example, here are the first two lines of the uart0 file:

```
[hardware]
baud = 9600
```

In this example, "hardware" is the name of a section. Until the next section name in the file, all entries must be either key/value pairs (such as the "baud = 9600" entry) or comments. Key/value pairs listed before a section name are invalid.

Comments may be stored in the configuration file by inserting a pound sign ('#') before the text you would like to add. This allows you to leave an explanation as to why certain settings have been made, who made the changes, etc. You can write anything you want in a comment, but the comment ends at the end of the line. If you want muti-line comments, insert the # at the beginning of each line. For example:

```
# This is a comment.
# This is line #2 of the comment.
this = no comment
```

```
# But this is one.
```

FILE CONTENTS

SYSTEM

[configure]

This contains settings that pertain to the operation of the Configuration menus. Currently, there is only one: password.

password

This allows the setting of a password that will be asked for upon entry to the Configuration screen. Up to 12 alphanumeric characters will be accepted. Do NOT use any characters other than numbers and letters in this password. Although the password is not hidden from the screen while editing, it will be hidden when entering configuration.

[cpu sleep]

This contains settings that control the low power modes of the microprocessor on the Mercury after all other hardware within the Mercury is sleeping.

state

This is the low power mode to put the processor into once all communications hardware is sleeping. The value of this entry can be `standby`, `doze`, or `stop`. In standby, the processor consumes 7mA; in doze it consumes about 500µA; and in stop, it consumes less than 100µA. However, wakeup occurs whenever any hardware needs servicing; standby wakes up immediately, doze takes about a tenth of a second, and stop takes over 2 seconds.

shut off led power

When the Mercury's processor enters low power mode, this entry determines whether or not to shut off power to the status LEDs

on the top of the Mercury unit. Possible values are `yes` and `no`.

`negate sleep output`

Flow control pins on the RS-232 connector can be configured to output a status when the Mercury goes into low power mode. Setting this value to yes and enabling a sleep output in the flow control section of uart0 will cause flow control lines to become asserted (logic 1, the same value that flow control normally uses to tell the other device on the serial cable to stop sending data) during the low power state.

RS-232 PORT (UART0)

`[hardware]`

These are the conventional RS-232 serial UART configurations found on all UARTS.

`baud`

This selects the data transfer rate of the RS-232 serial port. The baud rate can be anywhere between 112.5 and 115,200 bps. All standard rates (300, 1200, 2400, 9600, 19200, 38400, 57600, and 115200) are clocked precisely, but values in between will be rounded to the closest possible hardware setting.

Note: *Closest possible hardware setting does not mean that the value will be rounded to one of the standard rates. The Mercury's UART is fully capable of operating at non-standard speeds. The resolution of possible baud rates is smaller at lower baud rates than at higher baud rates. If you have unusual baud rate needs, please contact Nomadic Communications technical support to discuss what rates are actually possible.*

`data bits`

This selects the number of bits in a UART character frame that are used to transmit data. Possible values are 7 and 8.

`parity`

This sets the parity used in the UART character frame to check for correct data transmission. Options are none, odd, and even.

`stop bits`

This selects the number of bits used to represent an end of

character bit in the UART frame. The value can be 1, 1.5, or 2.

[software]

The [software] section is used to control the receiving and sending of bytes over the serial port. Proper setting of these values can significantly enhance the efficiency of data transmission, because the radio is "packet based" and the UART is "stream based". Stream based means that the data is transmitted and received one byte at a time, without any mechanism to separate chunks of data from other chunks of data; what the data means and how the data is to be delimited is purely up to the devices generating and using it. Packet based means that the data is grouped into chunks; wrapper information is added to this data, specifying where the data is destined to go. Computer networks are packet based by design. The radio in the Mercury is really a computer network interface. So the Mercury's radio is packet based.

This becomes important as the data is moved between the UART and the radio. When data is received over the radio, sending it to the UART is simple: the wrapper information is removed and the data inside the wrapper is sent out the serial port as fast as it can go. However, data being received over the serial port must be converted into packets, and this process is not quite so simple.

In order to perform this conversion, the Mercury must be given a set of rules that tell it that it's okay to start transmitting the data already received. The data can't be transmitted as soon as it's received, because as explained earlier there is overhead in the radio transmitting process. The values in this section define these conversion rules.

Note: *These rules don't ever change the data in any way. They simply specify when data can be transmitted over the radio.*

line length

This is the maximum number of characters to wait for before transmitting the data over the radio. The default is 1408, so that once 1408 characters have been received by the UART, they will be transmitted regardless of the content of the data. This value can range from 1 to 1408, but be careful of using values too small: they could result in data loss.

`input timeout`

This is the amount of time, calculated by the number of characters that would be received at the current baud rate, to wait after some data has been received on the UART before giving up waiting for more. For example, with the default value of 10 and at 9600 baud, 960 characters can be received in one second. Thus the amount of time to wait would be 10 ¸ 960 or 1/96th of a second. Now, once a character has been received the timer starts with this timeout. If the timer expires, the data is considered to be complete and gets transmitted immediately. But if instead another character is received before this timeout, the timer is restarted from the beginning.

The range for this value is from 1 to 65536.

`delimiters`

Delimiters are special characters that specify the end of the data. Once any of the characters listed in this option are received, the data is transmitted immediately. In serial communications, there is frequently a character reserved to mean "end of transmission". In human-interface applications, this character is the "newline" or Enter key. For computer to computer communication, this value may be different. If one exists, adding it to this list will greatly improve communication efficiency. Up to four delimiters can be listed here. They are specified as a space separated list of ASCII values. The values can be written as decimal or hexadecimal numbers.

`rxbd count`

`txbd count`

`buffer count`

These three values are used mainly to control the buffer size internal to the software. Changing of these values is not recommended except with the assistance of Nomadic Communications Technical Support. rxbd and txbd count default to 8 and buffer count defaults to 12.

`protocol`

This specifies the data format of the communication between the UART and radio network. Possible values are `passthrough`,

`passthrough2`, `telnet`, and `lpd`. Passthrough forwards any data received over the UART or radio interfaces to the other interface with the data unchanged. Passthrough2 operates much like passthrough, but opens a socket in each direction so that if a Mercury loses power, it can re-establish communications much faster when power returns. Telnet gives a command prompt similar to that of the telnet application found on UNIX systems. Telnet also interprets the data received over the radio, removing special character sequences known as "DO" and "DONT" requests. LPD is a UNIX print serving protocol. LPD received data in a particular format over the radio, and converts it into the data stream that should be sent to the printer. Data sent by the printer is ignored.

Note: When setting up to use LPD, you should set the printing parameters so that the file is sent off using the Mercury as the remote host. Any remote printer name can be used, so long as it fits within the guidelines for naming a printer.

`[flow control]`

The Mercury supports the following six flow control options: Recogize RTS, Generate CTS, Recognize DTR, Generate DSR, Recognize XON/XOFF, and Generate XON/XOFF. It also supports the original RS-232 specification for flow control where CTS is generated only when RTS is asserted. These options are explained below.

The incoming flow control options specify what method of communication is used between the Mercury and the device it is communicating with over the RS-232 cable to instruct the Mercury when to start and stop sending data. Since the Mercury is a DCE device (data communications equipment), the three methods of incoming flow control are software (also known as XON/XOFF), RTS ("request to send"), and DTR ("data terminal ready"). These three methods are all signals that the host computer sends to the Mercury when it either has too much data to work with and wants the Mercury to stop sending, or when it doesn't have enough data and wants the Mercury to start sending what it has.

Software flow control is implemented as two specific characters that are sent on the wire, embedded with the data. RTS and DTR are signals that each have their own wires, independent of the data wires.

The outgoing flow control options specify what method of communica-

tion is used between the Mercury and the host computer to instruct the computer to start and stop sending data. The host computer is a DTE device (data terminal equipment) and thus uses software, CTS ("clear to send"), and DSR ("data set ready") as its flow contol signals. These signals are sent to the host computer by the Mercury as its buffers fill and empty.

Software flow control works the same outgoing as it does incoming, as described above. CTS and DSR work the same way RTS and DTR do as well.

When one side wants the other to stop sending data, it can use as many of these flow control options as it wants to communicate its request to the other side. You must select in the configuration which you want the Mercury to use.

For both incoming and outgoing flow control, use the words "yes" and "no" to enable or disable recognition.

There is one additional allowed value for the CTS entry. This value is "rts". This may seem confusing, but originally the UART protocol was only defined to restrict data flow in one direction. What would happen is the modem would be allowed to send data to the host computer at any time (because modems were not capable of buffering data; if the data was not sent to the host immediately, the next piece of data would cause it to be lost) but the host computer would have to ask permission from the modem to send data. The host would do this by asserting the RTS line ("requesting" to send data). The modem would see this, and check to see if it were ready to accept data. If so, it would then assert the CTS line ("clearing" the computer to send), giving it permission. Once the host computer finished sending the data, it would drop the RTS line, and the modem would subsequently drop the CTS line.

This is the behavior that is emulated when "rts" is used as the value for the CTS entry. Note that this is obsolete behavior, and most devices are designed for RTS to really mean flow control in the computer's direction. But the ability is there if you have older equipment. Also note, if you use this setting that you should set the RTS entry to "no" since that line will not have the meaning of incoming flow control.

```
incoming software
```

`rts`

`dtr`

`outgoing software`

`dsr`

Allowed values are `yes` and `no`.

`cts`

Allowed values are `yes`, `no`, and `rts`.

`[i/o control]`

I/O control defines control over digital inputs and outputs of the Mercury, separately from the data lines. Digital input and output are shared with the flow control lines (RTS, DTR, CTS, DSR), but they are not flow control. They give the ability to send digital data from one Mercury to another without interpretation by the Mercury's themselves. They can also provide information to the device they are connected to about the status of low-power modes.

`rts`

`dtr`

These are input lines. When set to "passthrough", the status of these lines will be forwarded to the Mercury unit specified by the "socket" option, described below. When set to "none", they do not function as digital inputs.

`cts`

`dsr`

These are output lines. When set to "negate" or "assert", they will output a continuous digital value. "negate" outputs a logic 1, or "mark" which is electrically negative. This is also the state seen on an RS-232 connector when the cable is unplugged. "assert" outputs a logic 0, or "space" which is electrically positive. When set to "sleep", these lines will be negated when the unit is conserving power (subject to other sleep configuration) and

asserted otherwise. When set to "passthrough", CTS will output the value that the remote Mercury is receiving on its RTS line, and DSR will output the remote DTR value. The source of these remote signals depends on the remote Mercury's "socket" setting in this section.

`resend interval`

Ordinarily, the digital input lines are transmitted whenever they change. In some cases (such as when the remote unit is turned off and back on so that it forgets the previous output states), this is insufficient. Setting this value to a non-zero number causes the state of the input lines to be re-transmitted in a regular interval. The value of this setting is in seconds.

`socket`

This entry specifies a network connection to use to send the RTS and DTR input states to a remote Mercury. The value of this setting must be the name of a section that describes the network connection to use. Also, the connection must use the RMP protocol. See the section on "rmpbind" under "Network Bindings", below.

`[sleep]`

This section governs the low power state of the UART.

`rts causes sleep`

This is a yes/no entry that dictates whether the RTS line can cause the UART to go into low power mode. If this is yes, setting the RTS line to a logic 1 (the same state as if you wanted the Mercury to "stop transmitting data") will cause the UART to shutdown.

`rts timer`

Once the RTS line is set to logic 1, this is the amount of time in seconds to wait before actually shutting down the UART.

`inactivity causes sleep`

This is a yes/no entry that states whether the UART will be shut

down due to an amount of time passing without data transfer.

`inactivity timer`

This is the amount of time, in seconds, to pass without activity before the UART is put to sleep.

`shut off rs232 driver power`

This yes/no entry says whether to save additional power by turning off the RS-232 driver chip. When this chip is off, none of the output pins on the RS-232 port will be driven.

`shut off led power`

If this is set to yes, the LEDs on the top of the Mercury will be shut off to save power when the UART goes into sleep.

`negate sleep output`

If this is set to yes and sleep is set as an output for one of the flow control lines, the state of the UART will be shown on the corresponding flow control line. Note that this flow control line will not be readable if the RS-232 driver is shut off.

`[passthrough]`

This section specifies the operating parameters for the passthrough protocol. Currently, there is only one entry: socket.

`socket`

The value of this entry represents the name of another section. The section it specifies contains the values necessary to "bind a socket"; in other words, what needs to be known to create a network connection. For example, this value by default is `rmpbind`. The connection used by passthrough mode will create a socket that uses the values in the [rmpbind] section further down in the file to create the network connection.

`[passthrough2]`

This section specifies the operating parameters for the passthrough2

protocol. With passthrough2, there are two sockets: one for incoming data (listen), and one for outgoing data (connect). Note: both of these sockets must use the TCP/IP protocol.

`listen`

As in the socket key for the [passthrough] section above, the value of this entry represents the name of another section that defines a network connection. This value by default is `tcpbind1a`. A later section in this file is also named `[tcpbind1a]`. Thus the connection used by passthrough mode will create a socket that uses the values in the [tcpbind1a] section to create the network connection.

`connect`

The value of this entry points to a network connection definition that initiates the connection instead of listening for it. The default for this setting is `tcpbind1b`.

`[telnet]`

This section specifies the operating parameters for the Telnet protocol.

`connect`

This entry dictates what kind of prompt is seen on the serial port output. Two choices are available: `wait for keystroke` and `command prompt`. The default, command prompt, will give a prompt "`telnet>`" much like Telnet software on a UNIX system gives. From this prompt, you can use the "open" command to connect to a particular machine. For example, typing "`open 10.10.10.129`" will open a Telnet session to the machine with the IP address 10.10.10.129. The second choice, wait for keystroke, allows for a configuration that only connects to one machine, requiring only a keypress to initiate the connection.

`wait data 1`

This entry states what text to use as a prompt in wait for keystroke mode. The value is specified as a combination of a string and ASCII values. The default value is `"Press any key to connect to host..."`. (The double-quote marks are a part of

this value, unlike in previous examples.) Binary data can be added by writing data as hex or decimal, outside of quote marks. For example, the ANSI clear-screen command string (escape, left-bracket, H, escape, left-bracket, J) can be added to the front of this string like this: `0x1b "[H" 0x1b "[J" "Press any key to connect to host..."`.

`wait data 2`

This entry states what text to display after the keystroke is received. The format of the data is the same as in "wait data 1". The default is "`0xd 0xa`" (without the quotes), which will move the cursor to the next line.

`ip address`

When in wait for keystroke mode, this specifies the IP address of the machine to automatically connect to. 10.10.10.129 is the default.

`tcp port`

When in wait for keystroke mode, this specifies the TCP port number to connect to automatically. The default is `23`.

`reopen after shutdown`

After one connection has been completed, should the prompt to connect again reappear? The default is `on`.

Network Bindings:

This section describes the meaning behind each of the entries that describe network binding. Five prewritten bindings are provided as examples.

`[rmpbind]`

This binding can be used for both the passthrough protocol and for the i/o control socket setting.

`protocol`

This example uses the RMP protocol, so the value of this entry is

"rmp". RMP binding options are described below. If you wish to use the TCP/IP protocol, skip to the next example.

source address

This is the value that this Mercury will use to identify its serial port when sending serial data to and receiving serial data from other Mercury units (in short, this is its "port address"). The default value is "default" which will cause the Mercury to use its serial number as the address.

source address filter

Setting this value will tell the Mercury to only accept data coming from the address specified. For example, if this entry is set to "1234", only data originating from a Mercury with the port address of 1234 will be accepted; all other data will be ignored. The default value is "none".

destination address

This value tells the Mercury what port address to send data received on the serial port to. It can be the port address of another Mercury's serial port, or it can be "broadcast" or "dynamic". Broadcast means to send the data to all Mercury's. Dynamic means to send the data to the Mercury it last received data from. Dynamic has the effect of causing two Mercury units that are by themselves to communicate to each other. The default value is "dynamic".

transmit try count

For non-broadcast data, this specifies the number of attempts that the Mercury should make in transmitting each piece of data to a remote Mercury. A transmission may fail if the destination Mercury is out of range or turned off; when this happens, the data would be lost if additional attempts are not made. This value gives the user the ability to tell the Mercury how diligently to attempt transmission. The maximum value is 65000. The default value is "infinite".

`transmit retry interval`

When attempting addition transmit attempts (specified with "`transmit try count`", above), it can be useful to additionally specify how long to wait between attempts. This setting gives that ability. The value is specified in 1/100ths of a seconds, so that 100 means 1 second. The maximum value is 65000. The default value is 100.

`[tcpbind1a]`, etc.

Four prewritten bindings, tcpbind1a, tcpbind1b, tcpbind2, and tcpbind3 are provided for TCP/IP. When protocol is set to passthrough, only one binding is used because only one network socket is created (tcpbind2 and tcpbind3 are passthrough binds). When protocol is set to passthrough2, two complimentary bindings (one connect and one listen) are used (tcpbind1a and tcpbind1b are complimentary) because a socket is created in each direction.

`protocol`

These examples use the TCP protocol, so the value of this entry is "`tcp`".

`type`

Can be either `listen` or `connect`. This specifies whether the Mercury will wait for a connection from another computer, or attempt to initiate the connection itself.

`ip address`

Used only if the type is connect. This specifies the IP address of the computer to connect to.

`local tcp port`

If the type is listen, this will specify the TCP port number that the Mercury will wait for connections on.

`remote tcp port`

If the type is connect, this specifies the TCP port number to

connect to on the remote computer.

`reopen after shutdown`

If the type is listen, this states whether a second connection will be accepted after the first connection terminates. If the type is connect, this states whether another connection attempt will be made if the first connection is closed by the remote computer.

`socket connect data`

This specifies data to be written to the connection once it is successful. The value is specified as a combination of a string and ASCII values. For example, the default value is `"Hello!" 0xd 0xa`. (The double-quote marks are a part of this value, unlike in some previous examples.) The item in quotes and the ASCII values together make up the data to be written. This example would cause 8 bytes to be written to the socket; these are, in hexadecimal, 0x48 0x65 0x6c 0x6c 0x6f 0x21 0x0d 0x0a. In fact, the data could have been specified on this line by using exactly those eight hexadecimal values instead of as a combination of string and hex.

`serial connect data`

This causes data to be written to the UART once a connection is successful. The format is the same as in `socket connect data`.

`serial disconnect data`

This causes data to be written to the UART when the network connection has been closed. This data is appended to any data already in progress: the closing of the socket will not cancel UART transmission of any data already received over the socket. The format of this data is the same as in `socket connect data`.

`serial fail data`

This causes data to be written to the UART if the socket connection were to fail. This should only happen in the case of a connect type where the remote machine was unavailable or refused the connection. However, it is valid to use it to catch listen problems should any software problem occur. The format is the same as in socket connect data.

RANGELAN2 ETHERNET / BRIDGED ETHERNET (LAN0)

`[hardware]`

`station type`

In order for a frequency hopping radio to work, in each network there must be one unit that coordinates the hops. This unit is called the Master. It might help to think of the Master as the conductor of a frequency hopping orchestra. The Master keeps time so all units know when to hop and what frequency to hop to.

Units classified as Stations synchronize to the Master and follow its signal to learn what frequency in the pattern the Master is currently using.

An acting "Master" can be configured either as a Master or Alternate Master. Alternate Masters act either as a Master or a Station. If an Alternate Master unit is unable to locate any other Master within range, it acts as a Master. If a Master station is already present, then the Alternate Master acts as a Station. When there are multiple Alternate Masters, they coordinate amongst themselves to determine who will become the Master.

There must be at least one station on the network designated the Master station. For most operating environments, there will be at least one unit always present. This should be set as the Master. Other units may be then configured as Alternate Masters, in case failure of the Master occurs.

In environments in which no unit will always be present, you may set several units to be Alternate Masters, with no unit

configured as "the" Master. In these cases, the Alternate Masters will negotiate amonst themselves to determine which will act as the Master for the time being. If that unit leaves, the negotiation will be performed again. This process takes a few seconds (time to realize that the current Master is no longer present, and then time to negotiate the new Master). For performance considerations regarding the setting of this parameter, refer to Appendix E, Performance Hints.

The Station Type choices are as follows:

- `master`
- `alternate master`
- `station`

`domain`

In order to establish communications, all Station Types require the same Domain number. Radios on different Domains cannot communicate with each other. The Domain is a software filter which does not affect the actual radio frequency or the frequency hop sequence, but specifies one unit to control the timing of the hop sequence.

The Domain is a number between 0 and 15 with 0 being the default setting.

`channel`

Each Master can select one of 15 Channels to establish communications with Stations. Each Channel number sets a unique frequency hopping sequence allowing for multiple subnetworks with higher data rate transmission capability in the same air space. You may think of the Channel as a pipe. In order to communicate, radios must be on the same Channel and there must be one (and only one) Master that provides the timing for that Channel.

There are 15 independent Channels designated 1 through 15 with 1 being the default setting. This means that there are 15 different sequences of frequency hops. Each Channel is at a different frequency at a different time. For networks with multiple Masters (like in a roaming environment), set each Master to a differ-

ent channel for optimum performance.

You need only set the Channel on a Master or Alternate Master. Stations will ignore the parameter if it is set, because they will synchronize to the setting the Master is using.

`subchannel`

The Subchannel is a software code that is appended to each radio packet. It does not affect the frequency hopping sequence like a Channel does. Use a Subchannel if you need more than 15 Masters in the same area and, therefore, all the Channels are used.

For example, you can use Channel 1, Subchannel 1 for Network A and Channel 1, Subchannel 2 for Network B. The two networks will not communicate with one another. They are, however, still sharing the 1.6 Mbps pipe since they are both using Channel 1.

The Subchannels are designated 1 through 15 with 1 being the default setting.

You need only set the Subchannel on a Master or Alternate Master. Stations will ignore the parameter if it is set, because they will synchronize to the setting the Master is using.

`peer to peer`

This value affects the way the RangeLAN2 radio routes through the Access Point. Changing this value is not recommended except with the assistance of Nomadic Communications Technical Support.

`[sleep]`

This section governs the low power state of the radio.

`state`

This is the type of sleep mode to go into. Possible selections are `broadcast`, `directed`, and `standby`. In broadcast and directed, network traffic may wake the device up; the type of traffic you want to cause wakeup, broadcast (sent to everyone) or directed (sent only to this radio) specifies which value to use. If you do not want radio traffic to wake the unit up, additional power savings can be found by using standby. This mode will cause the radio to wake up (taking about two seconds to do so) when data is ready to

be transmitted.

rts causes sleep

This is a yes/no entry that dictates whether the RTS line can cause the radio to go into low power mode. If this is yes, setting the RTS line to a logic 1 (the same state as if you wanted the Mercury to "stop transmitting data") will cause the radio to go into the desired state.

rts timer

Once the RTS line is set to logic 1, this is the amount to time in seconds to wait before actually shutting down the radio.

inactivity causes sleep

This is a yes/no entry that states whether the radio will be shut down due to an amount of time passing without data transfer.

inactivity timer

This is the amount of time, in seconds, to pass without activity before the radio is put to sleep.

shut off led power

If this is set to yes, the LEDs on the top of the Mercury will be shut off to save power when the radio goes into sleep.

negate sleep output

If this is set to yes and sleep is set as an output for one of the flow control lines, the state of the radio will be shown on the corresponding flow control line. Note that this flow control line will not be readable if the RS-232 driver is shut off due to the UART also being in low power mode.

[rmp]

This section sets the configuration of the RMP protocol.

`ethertype`

Changing this value is not recommended except with the assistance of Nomadic Communications Technical Support

`[ip]`

This section sets the configuration of the IP protocol.

`ip address`

This is the IP address that will be used by other computers to communicate with this Mercury.

`netmask`

This is a value that, when logical ANDed with the IP address, specifies the range of IP addresses within the local network.

`broadcast`

This is the IP address that is used in the local network to refer to all computers simultaneously. The default of "automatic" will work for almost all configurations. There should be no need to change this value.

`route`

This value references section names that specify the routing options for this network interface card. The default of "automatic" will work for most configurations. Please consult Nomadic Communications technical support before changing this value.

`gateway`

This value specifies the IP address of your Internet router or firewall, if present. By default, this value is set to 'none'. Change this to the IP address of your gateway if you intend to use one.

CHAPTER 7 - PROXLINK PROTOCOL
PPX-1 / MCP PACKETIZED MODE SPECIFICATION

This chapter describes the format that data passed to a Mercury's serial port must be in when the Mercury is configured for the ProxLink protocol.

PPX-1

Proxim Packet Exchange protocol version 1 is an error checking protocol that is an envelope around an MCP data packet. This envelope is as follows:

BYTE	SOP	Start of header (0x01)
BYTE	LEN_H	High byte of length
BYTE	LEN_L	Low byte of length
BYTE	LEN_C	Checksum: $\sim LEN_L + \sim LEN_H$ (sum of inverse of length bytes)
BYTES	DATA	MCP protocol data (see below)
BYTE	$CSUM_H$	High byte of checksum
BYTE	$CSUM_L$	Low byte of checksum

Table 7-1

The DATA entry is typically an MCP command (see MCP below).

MERCURY COMMAND PACKETIZED PROTOCOL (MCP)

The Mercury Command Packetized Protocol is an easy to use, full-featured protocol designed to allow the programmer complete flexibility in controlling the Mercury-RF1. Mercury-RF1's MCP protocol comes in two versions: one is backward compatible with Proxim's ProxLink packetized protocol format, the other offers more advanced features that were not available in the ProxLink.

The MCP packet is unwrapped from a PPX-1 packet after receipt over the serial port.

Packet Types

Most of the text in this section is from the ProxLink Packetized Mode Specification manual. There are a few commands added, such as Domain value, Master value/name control, and Security ID; and a few have been changed: the Channel and Subchannel arguments have a different range than in the Proxim modem.

INITIALIZE MODEM

Resets modem. Any pending commands are cancelled and modem reverts to its default power-up state.

ProxLink compatible mode

BYTE	COMMAND	ASCII 'I', Hex 0x49
BYTE	SEQNO	Sequence number

Table 7-2

Mercury-RF1 mode

BYTE	COMMAND	ASCII 'I', Hex 0x49
BYTE	SEQNO	Sequence number
BYTE	MCP VERSION	Version of MCP to use

Table 7-3

VERSION specifies which version of MCP packet format the user's software expects to use. VERSION starts at 1, and will increase as new commands and response formats change (so old software will never be broken, regardless of what changes are made). Version 1 is ProxLink compatible, and version 2 contains advanced Mercury-RF1 commands. The current version is 2, so any values greater than this will fail.

INITIALIZATION CONFIRMATION

Response that the initialization is complete. This packet is almost identical to the Modem Version Report, except that the response is 'i' instead of 'v'.

ProxLink compatible mode

BYTE	RESPONSE	ASCII 'i', Hex 0x69
BYTE	SEQNO	From original command
BYTE	$PORTADDR_H$	Bits 24-31 of local address
BYTE	$PORTADDR_3$	Bits 16-23 of local address
BYTE	$PORTADDR_2$	Bits 8-15 of local address
BYTE	$PORTADDR_L$	Bits 0-7 of local address
BYTE-(7)	STRING	Version of Mercury firmware
BYTE	RADIOTYPE	Type of radio being used

Table 7-4

Mercury-RF1 mode

BYTE	RESPONSE	ASCII ‘i’, Hex 0x69
BYTE	SEQNO	From original command
BYTE	$SERNUM_H$	Bits 24-31 of serial number
BYTE	$SERNUM_3$	Bits 16-23 of serial number
BYTE	$SERNUM_2$	Bits 8-15 of serial number
BYTE	$SERNUM_L$	Bits 0-8 of serial number
BYTE	$PORTADDR_H$	Bits 24-31 of local address
BYTE	$PORTADDR_3$	Bits 16-23 of local address
BYTE	$PORTADDR_2$	Bits 8-15 of local address
BYTE	$PORTADDR_L$	Bits 0-7 of local address
BYTE	MCP VERSION	Version of MCP in use
BYTES	RADIO VERSION	NULL-terminated radio version identifier
BYTE	RADIOTYPE	0x10 for RangeLAN2
BYTES	INTERFACE VERSION	NULL-terminated Mercury version identifier
BYTE	INTERFACE TYPE	0x00 for RS-232

Table 7-5

MCP VERSION indicates which MCP protocol version is being used. It will use the highest version it is capable of, but not higher than the version requested. For example, if the request is for version 3, and the software only knows up to version 1, it will return 1; however, if the request is for version 2 and the software knows version 5, it will use version 2.

DATA PACKET RECEIVED

This message is generated spontaneously when a packet is received over the radio.

BYTE	RESPONSE	ASCII 'd', Hex 0x64
BYTE	RESERVED	0x00
BYTE	LEN_H	Bits 8-15 of data length
BYTE	LEN_L	Bits 0-7 of data length
BYTE	$SRCADDR_H$	Bits 24-31 of source address
BYTE	$SRCADDR_3$	Bits 16-23 of source address
BYTE	$SRCADDR_2$	Bits 8-15 of source address
BYTE	$SRCADDR_L$	Bits 0-7 of source address
BYTES	DATA	Serial packet data

Table 7-6

SRCADDR is the port address of the sending modem.

TRANSMIT DATA PACKET

Tells the radio to send the data to the specified radio modem.

BYTE	COMMAND	ASCII 'T', Hex 0x54
BYTE	SEQNO	Sequence number
BYTE	LEN_H	Bits 8-15 of data length
BYTE	LEN_L	Bits 0-7 of data length
BYTE	$DESTADDR_H$	Bits 24-31 of destination address
BYTE	$DESTADDR_3$	Bits 16-23 of destination address
BYTE	$DESTADDR_2$	Bits 8-15 of destination address
BYTE	$DESTADDR_L$	Bits 0-7 of destination address
BYTES	DATA	Serial packet data

Table 7-7

DESTADDR is either the port address of the receiving radio modem or the broadcast address of the group of radio modems that are to receive the packet.

STATUS INDICATION OF TRANSMITTED PACKET

Tells the user that the data has or has not been successfully sent and the reason why.

BYTE	RESPONSE	ASCII 't', Hex 0x74
BYTE	SEQNO	From original command
BYTE	LEN_H	From original command
BYTE	LEN_L	From original command
BYTE	$DESTADDR_H$	From original command
BYTE	$DESTADDR_3$	From original command
BYTE	$DESTADDR_2$	From original command
BYTE	$DESTADDR_L$	From original command
BYTE	STATUS	See table below

Table 7-8

Status can be one of the following:

0x00	Successful Transmission	For point-to-point, packet was received by destination modem.
0x01	Retries Exhausted	Destination modem in range, but packet could not be sent.
0x02	Internal Timeout	Software timeout in the radio driver occurred. Probably indicates a hardware problem with the radio.
0x03	No Response from Destination	For point-to-point messages only means that it was not possible to detect remote modem. Could be off or out of range.
0x80	Buffers Full	Out of buffer memory. If any data is available to read, do so otherwise wait a moment and try again.
0xa0	Duplicate Packet	Packet was received already. Probably means that the response from the first packet was lost.
0xff	No Master	Timeout looking for a master. Communication could not be made. Perhaps master is off/out of range or domain number/security ID is wrong. In old protocol mode, this error will be returned as 0x03.

Table 7-9

REQUEST RADIO SIGNAL STRENGTHS

Asks for the instantaneous radio signal strength, regardless of whether it is from a transmission or from extraneous noise. This function can be used to check noise levels to determine the best channel to be transmitting on if this modem is a master.

BYTE	COMMAND	ASCII 'R', Hex 0x52
BYTE	SEQNO	Sequence number

Table 7-10

SIGNAL STRENGTH REPORT

Returns the strength values for each of the channels in the range 0x00 to 0xff. 0xff is maximum strength. In old protocol mode, values will be returned for channels 1 - 7 (other channels are not available in this mode). In new protocol mode, channel is the last channel number sampled, and value is the strength measured.

ProxLink compatible mode

BYTE	RESPONSE	ASCII 'r', Hex 0x72
BYTE	SEQNO	From original command
BYTE	SSCHAN1	Radio power received on channel 1
BYTE	SSCHAN2	Radio power received on channel 2
BYTE	SSCHAN3	Radio power received on channel 3
BYTE	SSCHAN4	Radio power received on channel 4
BYTE	SSCHAN5	Radio power received on channel 5
BYTE	SSCHAN6	Radio power received on channel 6
BYTE	SSCHAN7	Radio power received on channel 7

Table 7-11

Mercury-RF1 mode

BYTE	RESPONSE	ASCII 'r', Hex 0x72
BYTE	SEQNO	From original command
BYTE	CHANNEL	Last channel radio signal received on
BYTE	VALUE	Signal strength received

Table 7-12

REQUEST MODEM VERSION

Asks for the serial number of the radio modem, version of the software, and hardware type.

BYTE	RESPONSE	ASCII 'V', Hex 0x56
BYTE	SEQNO	Sequence number

Table 7-13

MODEM VERSION REPORT

Returns the requested version data.

ProxLink compatible mode

BYTE	RESPONSE	ASCII 'v', Hex 0x76
BYTE	SEQNO	From original command
BYTE	$PORTADDR_H$	Bits 24-31 of local address
BYTE	$PORTADDR_3$	Bits 16-23 of local address
BYTE	$PORTADDR_2$	Bits 8-15 of local address
BYTE	$PORTADDR_L$	Bits 0-7 of local address
BYTE-(7)	STRING	Version of Mercury firmware
BYTE	RADIOTYPE	Type of radio being used

Table 7-14

Mercury-RF1 mode

BYTE	RESPONSE	ASCII ‘i’, Hex 0x69
BYTE	SEQNO	From original command
BYTE	$SERNUM_H$	Bits 24-31 of serial number
BYTE	$SERNUM_3$	Bits 16-23 of serial number
BYTE	$SERNUM_2$	Bits 8-15 of serial number
BYTE	$SERNUM_L$	Bits 0-8 of serial number
BYTE	$PORTADDR_H$	Bits 24-31 of local address
BYTE	$PORTADDR_3$	Bits 16-23 of local address
BYTE	$PORTADDR_2$	Bits 8-15 of local address
BYTE	$PORTADDR_L$	Bits 0-7 of local address
BYTE	MCP VERSION	Version of MCP in use
BYTES	RADIO VERSION	NULL-terminated radio version identifier
BYTE	RADIOTYPE	0x10 for RangeLAN2
BYTES	INTERFACE VERSION	NULL-terminated Mercury version identifier
BYTE	INTERFACE TYPE	0x00 for RS-232

Table 7-15

GO TO STANDBY

Puts the modem into low-power mode. The radio is shut off and no packets can be transmitted or received until woken up by asserting RTS.

BYTE	COMMAND	ASCII 'G', Hex 0x47

Table 7-16

Note: There is no seqno for this command.

STANDBY CONFIRMATION

Acknowledges that standby is about to go into effect. Any commands still pending at this time can be assumed to be lost. When modem is woken up, it will be in a reset (default) state.

BYTE	RESPONSE	ASCII 'g', Hex 0x67

Table 7-17

SET MASTER MODE

Puts the RangeLan2 into Master, Alternate Master, or Station mode. This command is only available in new protocol mode.

BYTE	COMMAND	ASCII 'M', Hex 0x4d
BYTE	SEQNO	Sequence number
BYTE	MODE	New master/station setting
12 BYTES	STRING	Master name

Table 7-18

Mode is 0x00 for station, 0x01 for alternate master, and 0x02 for master. String is the name the master should broadcast. If this is being set to a master but the name is zeroed, the existing name is unchanged. This call can be used to retrieve information about the current master.

MASTER MODE CONFIRMATION

Acknowledges the successful setting of the mode.

BYTE	RESPONSE	ASCII 'm', Hex 0x6d
BYTE	SEQNO	From original command
BYTE	MODE	From original command
12 BYTES	STRING	Possibly from original command

Table 7-19

Mode is 0x00 for station, 0x01 for alternate master, and 0x02 for master. String is the name of the master that the radio is synchonized to. If this unit is acting as a master, this will be the same as the name passed to it.

SET DOMAIN

Sets the domain that masters will broadcast and stations will lock on to. This command is only available in new protocol mode.

BYTE	COMMAND	ASCII 'N', Hex 0x4e
BYTE	SEQNO	Sequence number
BYTE	DOMAIN	New domain value

Table 7-20

Domain is the domain number in the range 0x00-0x0f to identify unique networks within the roaming system.

DOMAIN CONFIRMATION

Acknowledges the successful setting of the domain number.

BYTE	RESPONSE	ASCII 'n', Hex 0x6e
BYTE	SEQNO	From original command
BYTE	DOMAIN	From original command

Table 7-21

SET RADIO CHANNEL

Sets the channel (hopping pattern) the radio should use to transmit. If in old protocol mode, this will set the domain; it will also set the channel but only if it is master. With the new protocol, it will set the channel, but again, only if it is master. If it is not a master, the channel is determined by the current master for the currently selected domain.

BYTE	COMMAND	ASCII 'C', Hex 0x43
BYTE	SEQNO	Sequence number
BYTE	RADIOCHAN	New channel value

Table 7-22

In old protocol mode, radiochan is a value ASCII "1" to "7" (hex 0x31 to 0x37). In new protocol mode, use 0x01 to 0x0f to get the full range of available channels.

RADIO CHANNEL CONFIRMATION

Returns the value that the radio is currently on. This will be the same as the value set if this is currently acting as a master, else it will be the actual value the station is synchronized to. The value returned depends on the protocol in use. In the old protocol, the domain is returned; in the new one, the current channel is.

BYTE	RESPONSE	ASCII 'c', Hex 0x63
BYTE	SEQNO	From original command
BYTE	RADIOCHAN	Possibly from original command

Table 7-23

SET NETWORK SUBCHANNEL

This is the value used to separate different networks using the same channel. The subchannel does not actually change any frequencies used, it just sends an additional value in each packet sent to filter packets for different networks.

ProxLink compatible mode

BYTE	COMMAND	ASCII 'S', Hex 0x53
BYTE	SEQNO	Sequence number
BYTE	$SUBCHAN_L$	Bits 8-15 of subchannel
BYTE	$SUBCHAN_H$	Bits 0-7 of subchannel

Table 7-23

Mercury-RF1 mode

BYTE	COMMAND	ASCII 'S', Hex 0x53
BYTE	SEQNO	Sequence number
BYTE	SUBCHAN	Subchannel value (1-15)

Table 7-24

The subchannel for the ProxLink was a 16-bit number. The subchannel for the RangeLAN2 is a 4-bit number, with 0 being an invalid value. So the high byte and low byte are xor'd together and and'd with 0x0f. Then if the number is 0, it is changed to 8. In new protocol mode, only 0x01 through 0x0f are valid.

SUBCHANNEL CONFIRMATION

Returns the value that the radio is currently on. This will be the same as the value set if this is currently acting as a master, else it will be the actual value the station is synchronized to.

ProxLink compatible mode

BYTE	RESPONSE	ASCII 's', Hex 0x73
BYTE	SEQNO	From original command
BYTE	SUBCHAN$_L$	From original command (bits 0-7)

Table 7-25

Mercury-RF1 mode

BYTE	RESPONSE	ASCII 's', Hex 0x73
BYTE	SEQNO	From original command
BYTE	SUBCHAN	From original command

Table 7-26

If the value set in Set Network Subchannel was successfully changed, then this is the value that will be returned. Else, the previously set value will. If there was no previously set value, then the real value the radio is set at is returned.

SET BAUD RATE

This function is used to change the baud rate the serial interface communicates at.

ProxLink compatible mode

BYTE	COMMAND	ASCII 'B', Hex 0x42
BYTE	SEQNO	Sequence number
BYTE	BAUDRATE	Baud rate (from table)

Table 7-27

Mercury-RF1 mode

BYTE	COMMAND	ASCII 'B', Hex 0x42
BYTE	SEQNO	Sequence number
BYTE	$BAUDRATE_H$	Bits 16-23 of baud rate
BYTE	$BAUDRATE_2$	Bits 8-15 of baud rate
BYTE	$BAUDRATE_L$	Bits 0-7 of baud rate

Table 7-28

In old protocol mode, baudrate is set by following this table:

3	1200
4	2400
5	4800
6	9600
7	19200

Table 7-29

In the new protocol mode, baudrate is a 24 bit number with the baud desired. Note, however, that not all rates are possible. The real rate selected will be as close to the requested rate as the hardware can

manage. For example, 9600 will be exact (system clock speed was selected intentionally to cause this; normal RS-232 values are all exact), but 9500 baud will be rounded to 9501 because of clock divider roundoff (real physical rate would be 9501.0309). With a 14.7 MHz system clock, all standard baud rates (300, 1200, 2400, and other doubles) are precise; lowest rate is 112.5 baud, and highest is 307200 baud (although the RS-232 driver chip is not guaranteed above 240000 baud).

BAUD RATE CONFIRMATION

Acknowledges new baud rate. Message is sent prior to changing rate.

ProxLink compatible mode

BYTE	COMMAND	ASCII 'b', Hex 0x62
BYTE	SEQNO	From original command
BYTE	BAUDRATE	From original command

Table 7-30

Mercury-RF1 mode

BYTE	COMMAND	ASCII 'b', Hex 0x62
BYTE	SEQNO	From original command
BYTE	$BAUDRATE_H$	From original command
BYTE	$BAUDRATE_2$	From original command
BYTE	$BAUDRATE_L$	From original command

Table 7-31

In old protocol mode, baudrate returned is baudrate sent. In new protocol mode, baudrate is a 24 bit number with the actual baud possible; this will be the closest physically possible setting to the value requested.

APPENDIX A - SIMPLE POINT-TO-POINT CONFIGURATION

If you are using Mercury-EN's for Ethernet bridging or you are using two Mercury-RF1's to connunicate to each other, and the RMP protocol is sufficient for your needs, then you won't need to perform any configurations to make this happen. You still may need to perform configuration if you need specific hardware parameters to be set (such as baud rate or radio domain settings). The default configuration will allow two Mercury units to communicate both via Ethernet bridging, as well as for Point-to-Point RMP serial communication. For these, the Mercury is plug-and-play.

Configuring two Mercury-RF1 units to communicate to each other via TCP/IP is a fairly straight-forward process. There is no need to read Chapter 6 (Configuration) if this is all the configuring your units will need. However, we strongly recommend that you read that chapter anyway, because most applications will require at least some hardware configuration.

Here is a 21 step process to configure your Mercury units for Point-to-Point TCP/IP operation. These instructions assume that you have a terminal or computer with terminal emulation software in VT100 or ANSI emulation mode, configured to 9600 baud, 8 data bits, no parity, and 1 stop bit, and that you have both Mercury units powered up and ready to work with.

1. Connect the first Mercury to the computer via the supplied serial cable, and apply power.
2. Press the configure button on top of the Mercury. A menu should appear.
3. Using the arrow keys, select "Reset configuration to default" from the menu. Confirm the selection on the next screen.
4. Select "Edit configuration".
5. From the edit menu, select "RS-232 port (uart0)". A screen of configuration data will appear.
6. Scroll down to the [software] section, "protocol" option. Change the line to read "protocol = passthrough2".
7. Press Control-W to save the file. You will be returned to the edit menu.

8. Remove power from the first Mercury, and disconnect it from the computer.
9. Connect the second Mercury to the computer and apply power.
10. Press the configure button on this second unit.
11. Again, select "Reset configuration to default" and confirm.
12. Select "Edit configuration".
13. From the edit menu, select "RS-232 port (uart0)". A full screen of configuration information will appear.
14. Scroll down to the [software] section, protocol option. Change the value to "protocol = passthrough2".
15. Scroll down to an area that looks like this (near line 90; line numbers are shown at the bottom of the screen):

```
[tcpbind1b]
protocol = tcp
ip address = 10.10.10.129
remote tcp port = 4000
```

16. On the line that says "ip address = 10.10.10.129", change it to read "ip address = 10.10.10.128".
17. Press Control-W to save the file. You will be returned to the edit menu.
18. Select "RangeLAN2 ethernet (lan0)"
19. Scroll down to the line that says "ip address = 10.10.10.128" (near line 22). Change it to read "10.10.10.129".
20. Press Control-W to save the file.
21. Disconnect power from this Mercury.

That's it. Once you power the two units back on, they will communicate with each other.

If you have any problems with these instructions, you can read more about the configuration system in Chapter 6, or you may call Nomadic Communications technical support for assistance.

APPENDIX B - MERCURY SPECIFICATIONS

- Radio
  - Frequency: 2.4 - 2.4835 GHz
  - Compatible with the Proxim RangeLAN2
  - Technology: Frequency hopping, spread spectrum
  - Channels: 15 independent
  - Output power: 100 mW
  - Data rate: 1.6 Mbits/sec
  - Modulation: 4fsk (bsk in back off mode)
- Serial Interface
  - Data rates: 112.5 to 230,400 baud
  - Data format: 7 or 8 data bits; 1, 1.5, or 2 stop bits; even, odd, or no parity
  - Control lines: TxD, RxD, RTS, CTS, DTS, DSR @ RS-232 levels
  - Connector: Female DB-9
  - Configure and Reset control lines
  - Pinout (wired to the DCE [Data Communication Equiment] specification), except for pins 1 and 9

    1 - Configure, 2 - Tx, 3 - Rx, 4 - DTR, 5 - GND, 6 - DSR, 7 - RTS, 8 - CTS, 9 - Reset
- Ethernet Interface
  - Data rate: 10Mbit
  - Connector: RJ-45 (10Base-T)
  - Pinout

    1 - Tx+, 2 - Tx- 3 - Rx+, 4 - NC, 5 - NC, 6 - Rx-, 7 - NC, 8 - NC

❑ Software

— Pass-through mode

> transparent to application

> off-line configuration

> line length, delimiters, input timeout as message format options

❑ Power

— Input voltage: 7.5 - 15 VDC

— Current consumption:

	CPU	Radio	Total
Sleep	12mA	2mA	14mA
Receive	125mA	175mA	300mA
Transmit	125mA	350mA	475mA

— Power connector: 2.5 mm DC power jack

❑ Physical

— Imperial Dimensions: 5.5 x 2.77 x .88 inches, weight: 10 Ounces

— Metric Dimensions: 14.0 x 7.0 x 2.2 centimeters, weight: 200 grams

— Temperature: 20° to 60°C

— Humidity: 20% to 90% (non-condensing)

Nomadic Communications' Mercury radios are certified under the FCC's (Part 15) spread spectrum regulations allowing for their unlicensed use in the United States.

Range will vary depending on a number of factors including the physical environment between Mercury units.

Power supplies other than the one provided may cause damage and will void the warranty.

Swivel antenna attached with a non-standard connector.

APPENDIX C - SERIAL PORT SPECIFICATIONS

This appendix describes the Mercury's serial port specification. The Mercury unit uses the RS-232 serial port standard for communicating between a DTE (Data Terminal Equipment) device and a DCE (Data Communications Equipment) device, where the Mercury unit conforms to the DCE side of the connection and the user's device or terminal must conform to the DTE side. Note that the serial cable Nomadic Communications provides with the Mercury unit complies with this standard.

The necessary serial cable pinout is shown in Figure B-1. The cable is wired to be a standard 9-pin straight-thru from a computer or terminal. The computer or terminal transmits data on pin 3 and receives data on pin 2. Similarly, the Mercury unit receives data on pin 3 and transmits data on pin 2.

Category	Pin	Signal Name	Abbr.
Data	3	Transmitted Data	TD
	2	Received Data	RD
Control	7	Request To Send	RTS
	8	Clear To Send	CTS
	6	Data Set Ready	DSR
	4	Data Terminal Ready	DTR
Electrical	5	Signal Ground	
System	1	Configure	
	9	Reset	

Figure B-1 Cable Pinout Description

The Mercury is configured to ultimately support two sets of handshake lines for flow control: RTS/CTS (Request to Send/Clear to Send), and DTR/DSR (Data Terminal Ready/Data Set Ready).

If you intend to use a cable other than the one that came in the package, there are two pins you should be aware of and make sure are not connected on the Mercury side.

Pin 1 on the DB-9 side of the cable is the Configure pin. Grounding and then bringing high this pin on the Mercury will cause the Setup Mode main menu to come up in the same way that pressing the Configure button will. If the pin is left grounded, the Mercury will be in a constant state of trying to bring up the menu. You should not have this pin connected.

Pin 9 on the DB-9 side of the cable is the Reset pin. Forcing this pin to ground and back to 5V will reset the Mercury in the same way it would if power were turned off and back on. If this pin left grounded, the Mercury will be in a constant reset state. You should not have this pin connected.

APPENDIX D - LED STATUS INDICATORS AND ERRORS

MERCURY-RF1 AND MERCURY-EN LEDS

There are eight LEDs on the Mercury-RF1 and Mercury-EN:

- Power – Lights when power is applied.
- Error – Lights when the software detects an error condition.
- Serial Tx – Lights when the Mercury is transmitting data through the serial port.
- Serial Rx – Lights when the Mercury is receiving data from the serial port.
- Master – Lights when the Mercury is acting as domain Master.
- Station – Lights when the Mercury is synchronized to a Master.
- CD – Lights when the radio detects a carrier.
- Radio Tx/Rx – Lights with the radio is transmitting or receiving a data packet.

MERCURY-EN LEDS

There are two additional LEDs on the Mercury-EN:

- Link – Lights when a network is detected to be connected to the other side of the 10Base-T cable.
- Lan Tx/Rx – Lights when network activity is present on the 10Base-T cable.

ERROR LED

Errors are indicated by the unit's Error LED. Upon power-up, the Error LED will blink very briefly. If, however, the error light stays on, this indicates either a hardware or configuration error. In order to determine which one, connect the Mercury to a PC or terminal using the supplied serial cable (not one of your own making) and

press the Configure button (see Chapter 5, Configuration). If the menu appears, it is a configuration error, and you can view the Event Log to determine the nature of the error. If the menu does not appear, try disconnecting the serial cable from the Mercury and turning the power off and back on; if the Error LED remains lit, call Nomadic Communications Customer Service (at 650/988-7200) for assistance. If the Error LED does not remain on in this case, check the serial cable you are using to insure that pins 1 and 9 are disconnected. If you are sure they are (i.e., the pins are physically missing from the connector), call Customer Service for assistance; otherwise, change the serial cable you are using to a properly modified one.

When the Mercury unit is placed into sleep mode, the Power LED will be dimmed and the rest of the LEDs will be turned off for the duration of the standby operation. When the unit leaves the standby mode, the LEDs will come back on if they are configured to do so. In any case, if a serious error condition occurs (i.e., hardware failure or configuration error), the LEDs will always come back on, regardless of their configured state. The LEDs will not come back on for errors such as UART framing and parity errors although these errors will still be shown in the Event Log.

Error Codes

Errors displayed in the Event Log are as follows:

Hardware Errors

FLASH failure; unable to read or write configuration.

FLASH may be damaged. Configuration cannot be accessed or saved.

Initialization of interface "lanX" failed.

Radio could not be initialized. Possibly a hardware or software failure.

Software Errors

Flow control overflow.

This indicates a software bug.

Flow control underflow.

This indicates a software bug.

xxxx: file does not exist.

Configuration file could not be found. This indicates a software bug.

CONFIGURATION ERRORS

xxxx: [yyyy]: section does not exist.

Section named yyyy in configuration file named xxxx was missing.

xxxx: [ip]: "zzzz": entry refers to non-existant section.

Entry zzzz refers to a section that is not located in file xxxx.

xxxx: [yyyy]: "zzzz": entry does not exist.

Entry zzzz in section yyyy of file xxxx was missing.

xxxx: [ip]: "zzzz": entry is invalid.

Entry zzzz in section yyyy of file xxxx contains an invalid value.

Unable to bring up interface "lan*X*".

Configuration values for the radio may be incorrect.

xxxx: <[yyyy]zzzz>: Unable to add route.

Route values are out of range compared to the interface values.

SCCUART_Restart: Out of buffer memory; efficiency reduced.

Buffer allocation values are too high. Values were automatically reduced.

APPENDIX E - SPREAD SPECTRUM TECHNOLOGY

In 1985, the FCC (Federal Communications Commission) allocated three frequency bands for a radio transmission technique know as Spread Spectrum communications, originally developed by the military. This transmission technique has much greater immunity to interference and noise compared to traditional radio transmission techniques. In addition an increasing number of users can use the same frequency (similar to cellular). In addition, these rules of designed to drive usage towards local data communications. Under the regulations, users of FCC certified Spread Spectrum products do not require a license from the FCC. The only requirement is that the manufacturers of Spread Spectrum products must meet FCC spread spectrum regulations.

INTRODUCTION

FCC rule changes in 1985, combined with the continuing evolution of digital technology has catalyzed the development of spread spectrum data communication radios. These radios offer significant performance and operation benefits to end-users.

In this section, the operation of spread spectrum radio technology is presented, performance differences between spread spectrum and conventional radios, regulatory implications, and applications are discussed.

CONVENTIONAL RADIO OPERATION

For the purposes of this discussion, the term "conventional" will refer to all non-spread spectrum radios. Figure D-1 depicts the radio signal from a conventional radio.

This signal is referred to as narrow-band, which means that it contains all of its power in a very narrow portion of the radio frequency bandwidth. Due to the relatively small portion of the radio band that an individual radio transmission occupies, the FCC has traditionally favored these conventional radios. However, as a result of the very narrow frequency, these radios are prone to interference (a single interfering signal at or near their frequency can easily render the radio inoperable).

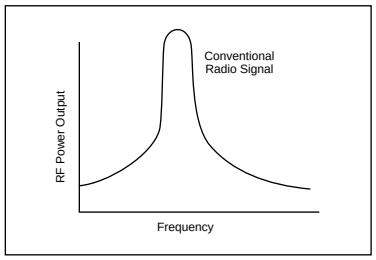


Figure D-1 Narrow Band Radio Signals

SPREAD SPECTRUM RADIO OPERATION

Spread spectrum is a technique that takes a narrow band signal and spreads it over a broader portion of the radio frequency band. This has the operational advantage of being resistant it interference, however, due to unfounded concerns over the increased frequency space it occupies, the FCC until recently, did not permit commercial use of the technology.

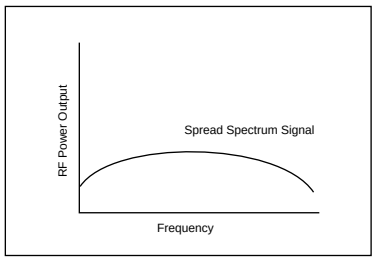


Figure D-2 Spread Spectrum Radio Signal

In performing Spread Spectrum, the transmitter takes the input data and spreads it in a predefined method. Each receiver must understand this predefined method and spread the signal before the data can be interpreted.

There are two basic methods to performing the spreading: (1) Frequency Hopping, and (2) Direct Sequencing. Frequency hopping spreads its signals by "hopping" the narrow band signal as a function of time.

Due to its noise immunity and its superior interception avoidance, Spread Spectrum radio technology has been used by the military for more than 40 years.

Spread Spectrum and the FCC

The FCC allows the use of Spread Spectrum technology in three radio bands, 902-928 MHz, 2400-2483.5 MHz and 5725-5850 MHz for transmission under 1 Watt of power. This power limit prevents interference within the band over long distances.

Background Summary

The following observations can be made from the background information:

* Spread Spectrum technology can provide an inherent better information link than conventional radio technology
* Spread Spectrum is a proven technology.
* The FCC has approved Spread Spectrum for commercial applications and continues to support its use.
* Due to the power limitations, spread spectrum may not be direct substitute for wired area conventional technology.

Advantages to Spread Spectrum

No FCC Site License The FCC will grand a one time license on the radio product. After that license is granted, the product can be sold anywhere in the U.S.

Limited Site Survey Required

When an FCC site license is granted for conventional radios, the complete RF system configuration must be specified. As a result, conventional radio installations require expensive and time consuming site surveys. Spread spectrum installations circumvent the need for any site surveys.

INTERFERENCE IMMUNITY

Spread Spectrum radios are inherently more noise immune than conventional radios. Thus they will operate with higher efficiency than conventional technology.

MULTI-CHANNEL

Conventional radios operate on a specific frequency controlled by a matched crystal oscillator. The specific frequency is allocated as a part of the FCC site license, and the equipment must remain on that frequency (except for very low power devices such as cordless phones).

Spread Spectrum data radios offer the opportunity to have multiple channels which can be dynamically changed through software. This allows for many applications such as repeaters, redundant base station and overlapping antenna cells.

APPENDIX F - PERFORMANCE HINTS

This section gives the user some ideas as to how to increase performance and network satisfaction on a wireless network.

DETERMINING MASTER, ALTERNATE MASTER, AND STATIONS

Mercury uses a spread spectrum frequency hopping technique. This means that the radio signal is constantly moving from one frequency to another in a predefined sequence. In order for several Mercury radios to communicate, they must be at the same frequency at the same time.

Proxim devised a method whereby one unit, called the Master station, sets the pace for the other radios. All stations look to the Master station to determine where and when to hop. If there is no Master station present, a station configured as an Alternate Master station will decide to become the Master for that session.

This configuration leaves the system administrator for the network with the task of configuring each wireless station on the network as Master station, Alternate Master station or just a Station. In most cases, using the default configurations for each of the drivers will work fine. But there may be times when the administrator wants to change the configuration for performance, or other issues. Here are several factors to consider:

1. In every wireless Mercury network, at most one station must act as the Master Station. If you need to set up additional Master stations, they should be configured as Alternate Master stations so that there is only one true Master station on the network.
2. The Master station must be within range of the other wireless stations on the network.
3. The Master station should not be a station which will be moved or turned off like a notebook computer or a user's personal machine.
4. By default, the Mercury is configured as an Alternate Master. This configuration allows for easy plug-and-play operation. It is not, however, the most efficient.
5. On a network where none of the units are guaranteed to be present, it may be advisable for all units to be Alternate Masters if

there is no Mercury configured as the Master.

6. You will achieve better performance by configuring the fewest number of machines possible as Alternate Masters.

MICROWAVE OVENS

Microwave ovens operate in the same frequency band as the Mercury. Therefore, if you use a microwave within range of Mercury you may notice network performance degradation. However, both your microwave and your Mercury network will continue to function.

RANGE

Every environment is unique with different obstacles, barriers, materials, etc. and therefore, it is difficult to determine the exact range that will be achieved with testing. The site survey tool was developed to aid in this process. Additionally, Proxim has developed some guidelines to estimate the range that users will see when the product is installed in their facility, but there are no hard and fast specifications.

Radio signals may reflect off obstacles or be absorbed by others depending on their construction. For example, with two Mercury radios you may achieve up to 1000' in open space outdoors where the two antennas are line of sight, meaning they see each other with no obstacles. However, the same two units will only achieve up to 500' of range when they have to travel through the cubicles usually used in modern offices. If there are office walls to penetrate, the signal range may decrease even further to up to 300'.

APPENDIX G - TROUBLESHOOTING

The Mercury is designed to be very easy to install and operate. If you do experience difficulties, however, use the information in this chapter to help diagnose and solve the problem. If you cannot resolve the problem, contact Nomadic Communications, Inc. at 650/ 988-7200.

HOW TO OBTAIN HELP WITH YOUR LAN INSTALLATION

If you require assistance to install your LAN, Nomadic Communications will be able to examine your needs and recommend the most cost-effective solution for your LAN whether you are installing a new LAN or adding on to an existing one.

GENERAL PROBLEMS

This section discusses some of the most common problems encountered when using Mercury-RF1 and the possible solutions.

1. **Out of range problems.** If the two Mercury units are not within range you will be unable to establish reliable communication.
2. **Mismatched Domains and Security IDs.** If you are able to load the software driver successfully, but cannot communicate with another machine on the network, it is possible that you do not have the same Domain and Security ID as on the other unit.
3. **Sleep mode.** The Mercury unit that you are attempting to connect to may be in sleep mode. In this mode, it will not receive any radio signals. If the Mercury is connected to a server or is configured as a Master, disable sleep mode.
4. **Multiple Masters or no Master.** There must always be exactly one acting Master on a subnetwork.
5. **No antenna attached.** The Mercury antenna unit must be attached to the unit in order for the Mercury to work properly.