Pulsar (EM1000)

Ethernet Modem

User’s Manual

001025 - C

Pulsar (EM1000) User's Manual

Part Number 019-0066 • 001025 - C • Printed in U.S.A.

Copyright

Z-World reserves the right to make changes and improvements to its products without providing notice.

Trademarks

- Dynamic C and Dynamic Cx86™ are trademarks of Z-World, Inc.
- Windows® is a registered trademark of Microsoft Corporation
- PLCBus™ is a trademark of Z-World
- Hayes Smart Modem® is a registered trademark of Hayes Microcomputer Products, Inc.

Notice to Users

When a system failure may cause serious consequences, protecting life and property against such consequences with a backup system or safety device is essential. The buyer agrees that protection against consequences resulting from system failure is the buyer's responsibility.

This device is not approved for life-support or medical systems.

All Z-World products are 100 percent functionally tested. Additional testing may include visual quality control inspections or mechanical defects analyzer inspections. Specifications are based on characterization of tested sample units rather than testing over temperature and voltage of each unit. Z-World may qualify components to operate within a range of parameters that is different from the manufacturer's recommended range. This strategy is believed to be more economical and effective. Additional testing or burn-in of an individual unit is available by special arrangement.

Company Address

Z-World, Inc.
2900 Spafford Street
Davis, California 95616-6800
USA

Telephone: (530) 757-3737
Facsimile: (530) 753-5141
Web Site: http://www.zworld.com
E-Mail: zworld@zworld.com

TABLE OF CONTENTS

Blank

ABOUT THIS MANUAL

This manual provides instructions for installing, testing, configuring, and interconnecting the Z-World EM1000 Ethernet modem using DOS commands.

Assumptions

Assumptions are made regarding the user's knowledge and experience in the following areas:

- Ability to design and engineer the target system that the controller used with the EM1000.
- Understanding of the basics of operating a software program and editing files under Windows on a PC.
- Knowledge of basic Intel assembly language and architecture for controllers with an Intel™386 EX processor.

For documentation from Intel, refer to the following texts.

Intel™386 EX Embedded Microprocessor User’s Manual
Intel™386 SX Microprocessor Programmer’s Reference Manual

Acronyms

Table 1 lists and defines the acronyms that may be used in this manual.

Table 1. Acronyms

Acronym	Meaning
EPROM	Erasable Programmable Read-Only Memory
EEPROM	Electronically Erasable Programmable Read-Only Memory
LED	Light-Emitting Diode
NMI	Nonmaskable Interrupt
RAM	Random Access Memory
SRAM	Static Random Access Memory
UART	Universal Asynchronous Receiver Transmitter

Icons

Table 2 displays and defines icons that may be used in this manual.

Table 2. Icons

Icon	Meaning	Icon	Meaning
👓	Refer to or see	✎	Note
☎	Please contact	Tip	Tip
⚠	Caution	⚡	High Voltage
FD	Factory Default		

Conventions

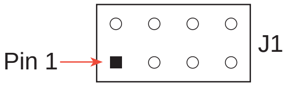
Table 3 lists and defines the typographic conventions that may be used in this manual.

Table 3. Typographic Conventions

Example	Description
`while`	Courier font (bold) indicates a program, a fragment of a program, or a Dynamic C keyword or phrase.
`// IN-01…`	Program comments are written in Courier font, plain face.
Italics	Indicates that something should be typed instead of the italicized words (e.g., in place of *filename*, type a file's name).
Edit	Sans serif font (bold) signifies a menu or menu selection.
`. . .`	An ellipsis indicates that (1) irrelevant program text is omitted for brevity or that (2) preceding program text may be repeated indefinitely.
[ ]	Brackets in a C function's definition or program segment indicate that the enclosed directive is optional.
< >	Angle brackets occasionally enclose classes of terms.
a | b | c	A vertical bar indicates that a choice should be made from among the items listed.

Pin Number 1

A black square indicates pin 1 of all headers.



Measurements

All diagram and graphic measurements are in inches followed by millimeters enclosed in parenthesis.

Blank

CHAPTER 1: OVERVIEW

Chapter 1 provides a comprehensive overview and description of the EM1000. The following sections are included.

- Features
- Options and Accessories
- Software

Overview

Z-World's EM1000 is an industrial-grade Ethernet modem that interfaces RS-232 devices to Ethernet networks. The EM1000 allows a point-to-point logical link between two RS-232 devices across an Ethernet network using any software protocol that the RS-232 devices have in common. The EM1000 can act either as a "listener," or it can establish links automatically or on demand. The EM1000 can be reconfigured during operation without manual intervention to target different remote devices sequentially. The RS-232 devices will operate through the EM1000 without interfering with other devices sharing the Ethernet network even if the other devices are using a different serial software protocol.

Figure 1-1 shows a front view of the EM1000.

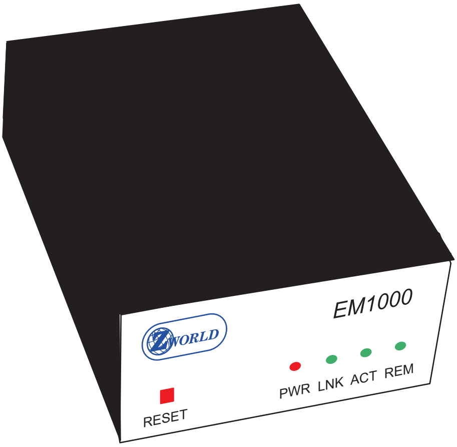


Figure 1-1. EM1000 Front View

The EM1000 has two DE9 RS-232 connectors, one male, one female, and a 10BaseT jack for connection to the Ethernet/Internet.

Ethernet networks have some major advantages over RS-232 or RS-485 networks.

- Distances between units can be greatly extended. Standard Ethernet networks support distances of up to 100 m between hubs, and fiber optic implementations support several kilometers between devices.
- Many facilities already have an Ethernet network in place, thereby saving the costs involved in wiring additional networks.

- Ethernet supports a layered protocol, which allows the communication between devices to be independent of other traffic on the Ethernet cable. Existing standard protocols such as TCP/IP, FTP, SMTP, Telnet, and SYSLOG can be adapted to service the needs of virtually any new device.
- Media-independent versions of Ethernet are available for specialty environments and requirements, for example, fiber optics for long distances and harsh electromagnetic environments.
- The Ethernet is the entry point into the Internet, where connected devices can be accessed and monitored from virtually anywhere.
- Ethernet cabling is low cost and off-the-shelf.
- Network installation procedures are well known and are readily available through established providers.

Features

The EM1000 offers the following features.

- 386EX microprocessor operating at 20 MHz.
- One RS-232 or TTL level data port (supports 3-wire to 7-wire operation).
- One RS-232 configuration port (supports 3-wire operation).
- One RJ-45 Ethernet port compliant with the IEEE 802.3 standard for 10 BaseT Ethernet protocol.
- Four status LEDs—power, bus active, local active, remote connection established.
- Built-in **`ping`** utility to check the availability of connections to remote network targets
- Security feature can be set up to disallow communications except from one designated IP address.
- Flexible software configuration options, including remote configuration via Telnet, and reconfiguration during operation (allows the modem to establish sequential links with multiple destinations) without manual intervention.
- Aluminum base plate or rubber feet for enclosure.

Options and Accessories

The EM1000 is designed to work with Z-World controllers and other devices that have an RS-232 serial communication capability. Z-World provides several accessories and options to assist with the evaluation and use of the EM1000.

Versions

The EM1000 is available in two versions. Table 1-1 lists their features.

Table 1-1. EM1000 Series Features

Model	Features
EM1000	Standard full-featured modem with enclosure, aluminum base plate and rubber feet, mating power connector, and operating software.
EM1010	EM1000 without enclosure or aluminum base plate.

The EM1000 is also available in an OEM version that has no brand markings on the covers.

Development Kit

The Development Kit enables customers to evaluate the EM1000 and to develop applications. The Development Kit includes the following items.

- One external 24 V DC, 400 mA power supply.
- Two cables to connect the EM1000 to a PC serial port (one cable has a male DE9 end for the **CONSOLE** port, one cable has a female DE9 end for the **COM1/RS-232** port).
- One PC to 10-pin header connecting cable to configure the EM1010.
- One Ethernet crossover patch cable (orange).
- User’s manual with schematics.

Power Supply

The 24 V DC, 400 mA power supply is available separately.

Ethernet Hub

A four-port Ethernet hub with its own power supply is available. The Ethernet hub allows EM1000s and other Ethernet devices to be connected together to simulate a network, and can also be used in a real network.

Cable Kit

A cable kit with four standard 3 ft (0.91 m) Ethernet patch cables (blue) is available.

For ordering information, or for more details about the various options and prices, call your Z-World Sales Representative at (530) 757-3737.

Software

The EM1000 uses a DOS-based operating system. All the necessary software is supplied already loaded.

Refer to Chapter 3, “Configuration Reference,” and to Chapter 4, “Commands,” for complete details on commands and how to develop and use applications with the EM1000.

Blank

CHAPTER 2: GETTING STARTED

Chapter 2 provides instructions for connecting the EM1000 to a host PC, and demonstrating that the unit is working. The following sections are included.

- Initial EM1000 Setup
- Connecting the Modem to a Host PC

Initial EM1000 Setup

Parts Required

- 9 V–32 V DC, 2.0 W unregulated DC power supply
- DE9 male to DE9 female programming cable (EM1000) OR
  10-pin header connector to DE9 female programming cable (EM1010)

Power Supply Connections

The EM1000 is configured directly through its RS-232 **CONSOLE** port. Before proceeding further, first connect the bare wires from the AC adapter to the +DCIN and GND terminals of the power connector as shown in Figure 2-1. Do not plug the AC adapter in until you are instructed to do so.

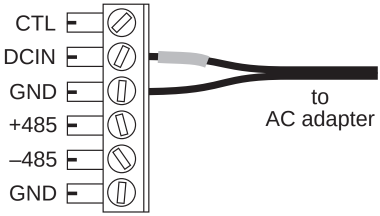


Figure 2-1. Power Supply Connections

If a power supply other than the AC adapter supplied with the Developer's Kit is used, ensure that the input voltage specifications (9 V to 32 V DC) are not exceeded. Appendix A contains complete specifications for the EM1000.

Connecting the Modem to a Host PC

1. Call up a terminal emulation program such as Windows HyperTerminal on your PC. Select an available COM port on the PC and set the terminal emulation program to the following parameters.

   Baud rate = 9600

   Data bits = 8

   Parity = none

   Stop bits = 1

   These parameters are used only for the setup and programming connection through the **CONSOLE** port, and do not affect the baud rate through the **COM1/RS-232** data port or other **COM1** parameters.

   See Chapter 3, "Configuration Reference," for information on how to set the **COM1/RS-232** parameters..

2. Click **OK** in the HyperTerminal application once the above parameters have been set.

3. Connect the female DE9 connector of the programming cable to the selected PC COM port. Connect the other end to the modem as explained below.

   EM1000—Connect the male DE9 connector of the programming cable to the **CONSOLE** connector on the back of the EM1000 as shown in Figure 2-2.

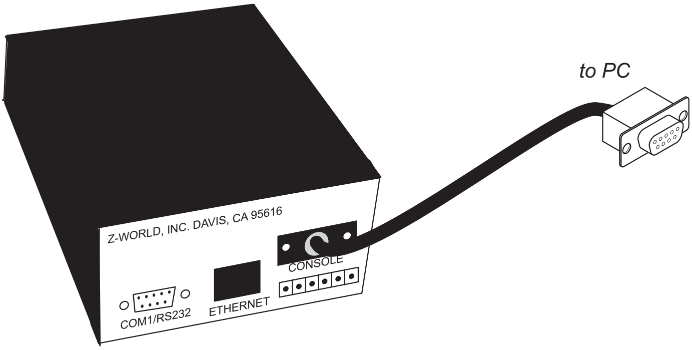


Figure 2-2. EM1000 Programming and Power Connections

EM1010—Connect the 10-pin serial header connector of the programming cable to header J2 as shown in Figure 2-3. Be sure to have the marked side of the programming cable in line with pin 1 of header J2.

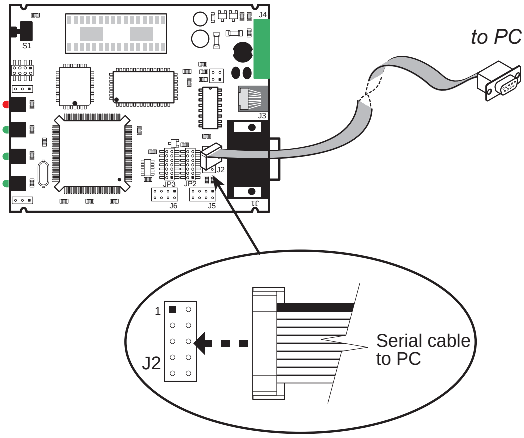
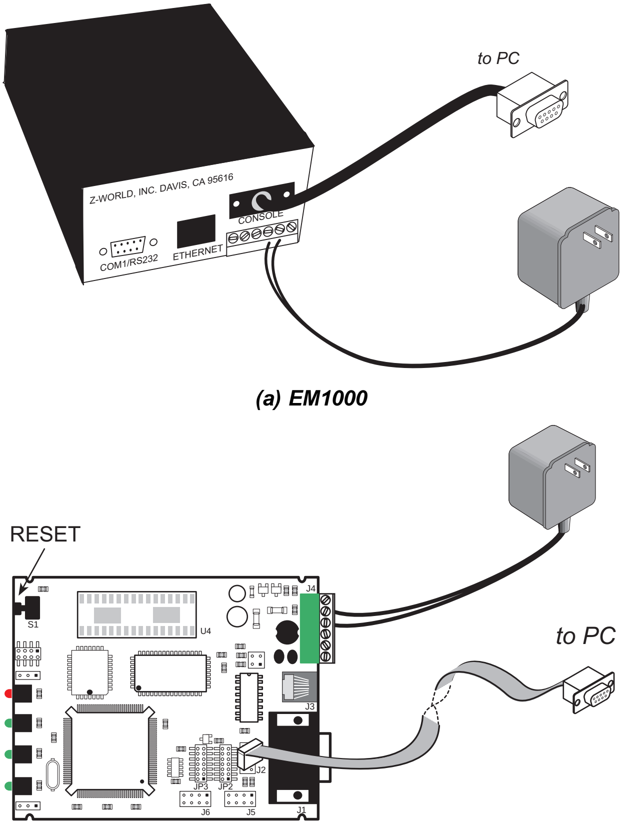


Figure 2-3. EM1010 Programming and Power Connections

> ⚠ The 10-pin serial header connector used for the EM1010 is not keyed and can be placed offset . Be careful to place the connector completely on header J2 as shown in Figure 2-3.

4. Connect the power connector to the modem as shown in Figure 2-4. Plug in the AC adapter.



(a) EM1000

(b) EM1010

Figure 2-4. EM1000 Power Connections

5. With power applied to the EM1000 (the red PWR LED should be lit), hold down the **RESET** button (shown in Figure 2-5 for the EM1000, and in Figure 2-4(b) for the EM1010), for at least 5 seconds, then release it.

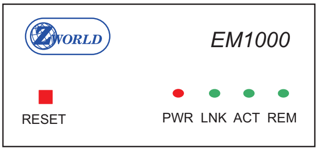


Figure 2-5. EM1000 Front Panel

Tip Check the baud rate setting for the terminal emulation program (step 1) if gibberish is displayed. You may also have to select a font such as Terminal by clicking on **Font** in the **View** menu of the terminal emulation program.

6. The following message is displayed via the terminal emulation program the first time the EM1000 is used.

```
WATTCP.CFG FILE NEEDS TO BE CONFIGURED.
REFER TO USER MANUAL.
->
```

Refer to the section on “Configuring COM1/RS-232 Port” in Chapter 3, “Configuration Reference,” for an example of a default configuration. Appendix C, “Handling DOS Files,” provides further details.

See Appendix B, “Two-Modem Test Setup,” for a sample test network that illustrates how the EM1000 works.

Tip

The **RESET** button is flush with the front panel, and it may be necessary to use the insulated tip of a pencil or similar tool to push the button. When resetting the EM1000, hold the **RESET** button down for at least 5 seconds to initiate the reset.

Tip

Use the programming cable supplied with the Developer’s Kit to avoid problems with mismatched plugs.

Tip

Connect a ground strap from the enclosure to a solid earth ground when installing the EM1000 in a network.

Tip

Mount the EM1000 away from sources of high voltages, strong magnetic fields, or in areas where the temperature rating of the unit may be exceeded. Be sure to string the Ethernet cables away from those same areas.

Blank

CHAPTER 3:

CONFIGURATION REFERENCE

Chapter 3 describes the built-in flexibility of the EM1000 and describes how to configure the available inputs/outputs. The following sections are included.

- Inputs and Outputs
- Operation
- Configuring the **COM1/RS-232** Port
- Specific Configurations for a Real Network
- Software Description

Inputs and Outputs

The EM1000 has three ports on its back end as shown in Figure 3-1.

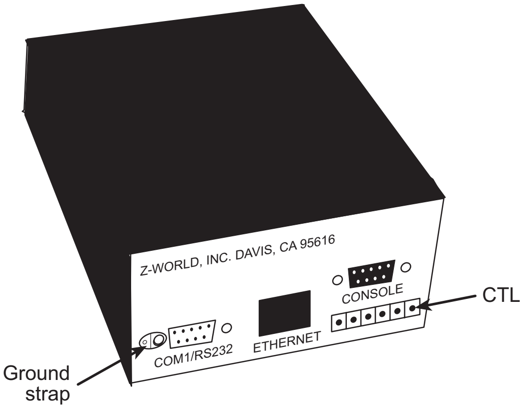


Figure 3-1. EM1000 Ports

The three ports are described below.

- DE9 **CONSOLE** port used to configure EM1000. This port appears as 10-pin header J2 on the EM1010.
- DE9 **COM1/RS-232** port to connect serial device.
- RJ-45 Ethernet port.

Figures 3-2, 3-3, and 3-4 show the pinouts for these three ports.

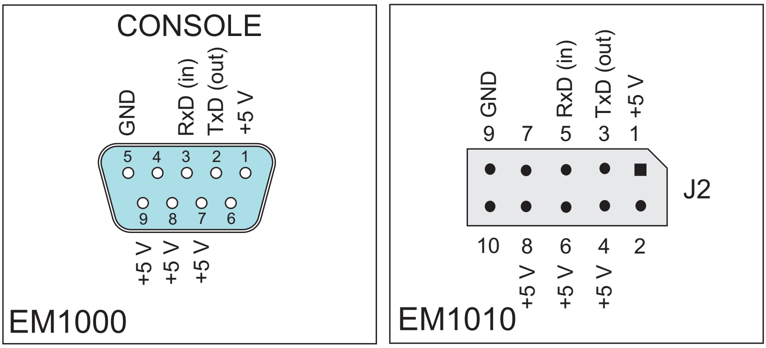


Figure 3-2. ***CONSOLE/Header J2 Pinout***

The **CONSOLE** port is used exclusively to configure the EM1000. The **CONSOLE** port operates at 9600 bps, 8 data bits, no parity, and 1 stop bit.

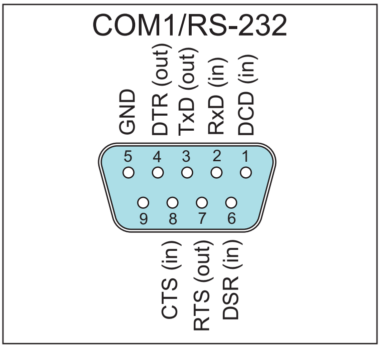


Figure 3-3. **COM1/RS-232** ***Pinout***

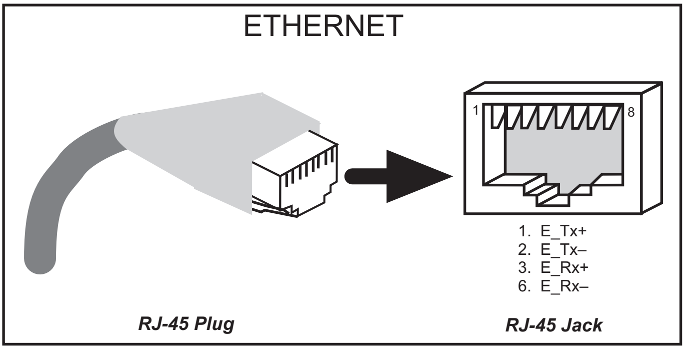


Figure 3-4. RJ-45 Ethernet Port Pinout

The **COM1/RS-232** port is normally used for data throughput to a remote device, but can also be used to configure the EM1000 under certain conditions. The **COM1/RS-232** port operates over a wide range of baud rates and other parameters.

There is a **CTL** input, accessible from the rear panel, that can be interfaced to a logic signal or open-collector output from a control device. This input can switch the **COM1/RS-232** port from the data throughput mode to the configuration mode and back again. This is useful when a link has been established to a remote device, and there is a need to establish a connection to a different remote device. The DSR signal on the **COM1/RS-232** port can also be used to switch the **COM1/RS-232** port back and forth between the data throughput mode and the configuration mode.

See the “Changing Remote EM1000s While Running” section under “Special Configurations” later in this chapter for more information on using the **COM1/RS-232** port to reconfigure the EM1000.

Operation

Configuration Options

In the simplest scenario, the EM1000 is configured via a terminal temporarily connected to the **CONSOLE** port. This allows the modem to be set up to send data to one other specified target location on the Ethernet or to receive data from any remote location.

In a more complicated scenario, the RS-232 device (such as a Z-World controller or a serial interface device) may need to send data first to one remote device, then to another. Devices such as Z-World controllers or PC-based products are programmable. This makes it possible for these devices to change the modem's configuration parameters, including a remote TCP/IP destination, during operation. To be able to do this, an RS-232 device needs either two RS-232 serial ports, an unused digital output, or the ability to manipulate the DSR input on the **COM1/RS-232** data port. When two RS-232 serial ports are available, one can be connected to the **COM1/RS-232** data port and the other can be connected to the **CONSOLE** configuration port. Alternatively, the unused digital output would be connected to the EM1000's **CTL** input shown in Figure 3-1 instead of to the **CONSOLE** port. In both cases, the RS-232 devices also need to be able to send a series of reconfiguration command strings instead of the data they would normally be sending.

Even when an RS-232 device lacks the ability to reconfigure the target address, the ability to send its data to a fixed destination anywhere on the network is still extremely useful.

The design provides for configuring the **COM1/RS-232** data port to handle logic-level signals to allow for compatibility with future Z-World products.

Hardware Flow Control

The EM1000 is designed so that flow control (RTS/CTS) does not have to be used. This allows the EM1000 to be used with simple "3-wire" devices—those that have transmit, receive, and ground. When the attached device has hardware flow control signals such as RTS/CTS or DSR/DTR available, other options provide even more flexibility.

End-to-End Data Integrity

The EM1000 uses TCP/IP to transmit data through the Ethernet port. TCP/IP provides internal mechanisms to ensure that data are received successfully by the remote end of the link, and the EM1000 relies on this for its most basic operation. The linked RS-232 devices can use other higher level protocols such as Xmodem, FTP, or Modbus, which use the underlying TCP/IP protocol. These optional higher level protocols allow for an additional level of data integrity, and also provide for timeouts and other high-level "end-to-end" protocol functions.

Data Transparency

During normal operation, most of the data sent to the EM1000 via the **COM1/RS-232** port are sent transparently to the remote node on the network. The EM1000 basically takes what comes in the serial port, and transmits it without regard to the data format, or conversion of the data. To lessen the amount of time the EM1000 is actually transmitting on the Ethernet, the incoming serial data are packetized. The packetization is controlled by setup parameters relating to the idle time between characters on the serial port, the receipt of some predefined “sentinel characters,” or a fixed number of bytes being received by the serial port (or a combination of all three techniques).

Packetization is discussed later in this chapter.

The modem handles most higher level protocols, such as Xmodem, in a transparent fashion if the packetization parameters are set up correctly for that protocol. Serial data are not sent to the remote node on the network while the **COM1/RS-232** port has been placed in a configuration mode with the DSR or CTL signals.

Some protocols, such as Zmodem, might not work over a TCP/IP network because of latency/timeout issues.

Open Connections

The EM1000 can establish a connection only to one other EM1000 (or other Ethernet device) at one time. For example, if a third EM1000 on a network tries to establish a connection with two EM1000s that are already connected, then the third EM1000 could time out.

LEDs

The EM1000 has four LEDs on its front panel, shown in Figure 3-5. There is a logical “hierarchy” to the operation of the LEDs. When power is applied, the PWR LED comes on. If there is activity on the Ethernet link, and the Ethernet cable is plugged in, then the LNK LED comes on. The ACT LED will flash when there is traffic on the Ethernet network, and the REM LED comes on when communication is finally established to the other end of the link.

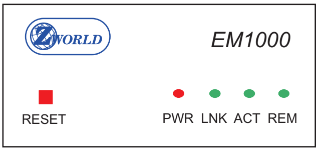


Figure 3-5. EM1000 Front Panel

The REM LED may stay on for a short period of time after the link is no longer established.

Configuring the COM1/RS-232 Port

The terminal emulation program will display a **->** prompt as shown below when the EM1000 has been connected to a terminal or host PC as described in Chapter 2, “Getting Started.” This prompt is from the software that is loaded in the EM1000. Type **`EXIT <return>`** to get the DOS prompt.

`->` (EM1000 software prompt)

`B:\>` (DOS prompt)

Press **RESET** to switch back to the EM1000 software prompt. This is usually done when all programming changes have been finished.

The **RESET** button is flush with the front panel, and it may be necessary to use the insulated tip of a pencil or similar tool to push the button. When resetting the EM1000, hold the **RESET** button down for at least 5 seconds to initiate the reset. Alternatively, the power may be cycled off and back on.

The EM1000 has many operating parameters, and they must be set before the modem will operate correctly. Most of the parameters default to states that will work in a typical situation.

Setting Modem Operating Parameters

Before proceeding further, it is essential to get the necessary information about your IP parameters from your network administrator. Your setup should be verified by the network administrator before any configuring is done or before the EM1000 is connected to an existing operational network.

Test and debug your EM1000 configuration off-line (unconnected to an existing network) to insure that your configuration works correctly. This will instill confidence that your configuration will work when connected to a network, and will reduce the possibility of a network interruption.

Appendix B, “Two-Modem Test Setup,” provides examples of a test network to check your configuration.

Be sure that all EM1000s are connected on the same side of any “firewall,” or ensure that the firewall will allow your particular combinations of IP and port addresses to pass through.

Configuration Settings

There are two configuration files that store configuration settings for the EM1000—**WATTCP.CFG** and **DWN.EM**. **WATTCP.CFG** should contain three lines only, one each for **my_ip**, **gateway**, and **netmask**. **DWN.EM** contains all the other configuration settings. **DWN.EM** can also override the settings in the **WATTCP.CFG** file.

Table 3-1 lists the default settings for the parameters. These are overwritten by the contents of the **DWN.EM** file on power-up or when the **RESET** button is pressed.

Table 3-1. Default Parameters

allowedname=any	hostdataport=8888
allowedport=0	hoststateport=8889
baud=19200	maxpacket=1024
bufferlength=200	my_ip= [from WATTCP.CFG file]
buffertimeout=2	netmask= [from WATTCP.CFG file]
configmode=data	parity=none
connect=request	password=none
databits=8	remotedataport=8888
domain=(null)	remotestateport=8889
dropsentinels=disabled	rtsmode=disabled
dsrmode=disabled	rts=on
dtr=on	spy=disabled
gateway= [from WATTCP.CFG file]	stopbits=1
hostname=(null)	verbose=disabled

Refer to Chapter 4, “Commands,” for further details on these parameters.

Configuring IP Parameters

Edit the **WATTCP.CFG** file using the built-in editor. The file is displayed with line numbers while you edit the file. Insert the changed line after the original line, then delete the old line.

1. With the terminal/PC connected to the **CONSOLE** port as described in Chapter 2, “Getting Started,” you will get the **->** prompt. Type

   ```
   exit <return>
   ```

2. At the DOS prompt **B:\>** type

   ```
   listen <return>
   ```

3. At the DOS prompt **B:\>** type

   ```
   edit wattcp.cfg <return>
   ```

3. The following lines should appear.

   ```
   0: my_ip=10.10.10.10
   1: gateway=10.10.10.1
   2: netmask=255.255.255.0
   ```

   Even though **hostip** is a synonym for **my_ip**, the **WATTCP.CFG** file recognizes only **my_ip**.

   If you are using a DNS (nameserver), then there will be up to three additional lines as follows. Line 3, modified for your network, is required; lines 4 and 5 are optional.

   ```
   3: nameserver=10.10.10.254
   4: domainslist=mycompany.com
   5: hostname=test1
   ```

4. Change the parameters as advised by your network administrator.
5. Save the file.
6. Push the **RESET** button for 5 seconds, then release the button to get the modem command prompt **->**.

See Appendix C, “Handling DOS Files,” for more information on editing a file.

Tip Write down the IP address of the modem on a label that is attached to the modem.

A Telnet session can be used to check the operation of the EM1000 in a network using the IP parameters set according to the above steps.

See Appendix E, “Telnet,” for a sample Telnet session.

Packetization Options

In most cases, the default values for packetization options

```
bufferlength=200
buffertimeout=2
```

should work well. The default values allow for data to be sent from the EM1000 when data from the serial port stop for a short period of time, which applies to most situations. The packetization options need to be considered only if the default **`bufferlength`** or **`timeout`** need to be changed.

The attached RS-232 device sends in a stream of data to the **COM1/ RS-232** port on the EM1000. The user must select how to break up the stream into packets for transport over the Ethernet. This selection depends on the format and timing of the data coming from the attached RS-232 device. There are three strategies to choose from.

1. Time-Based—The EM1000 measures the delay between characters received on the **COM1/RS-232** port. If the delay exceeds a certain (configurable) period, the data already received and placed in the buffer are then transmitted. This is a good fallback strategy if the data from the remote device are sent in bursts followed by a time gap.
2. Sentinel Characters—The modem watches the serial data stream for special characters (for example, **<CR>** or **<LF>**) and, if detected, the current packet is ended and transmitted. A further configuration item is whether the sentinel characters are actually transmitted in the packet or whether they are discarded prior to transmission.
3. Fixed-Size Packets—The EM1000 transmits the packet after receiving a fixed number of characters.

These strategies are not mutually exclusive. That is, the user may elect to use all the strategies at the same time. For example, the user may plan to send packets after a carriage return is detected in the serial stream, but would also like to send a packet after a 10-second timeout or 255 characters.

The most general solution is a packet timeout, either alone, or in conjunction with sentinel characters or the packet length. If your transmitting device sends data of nonuniform length, and does not use sentinel characters, or if you wish to send pure binary data, then you ***must*** use the time-based option (the other options rarely work in this case). The time-based option also provides a fallback in other cases where data will *eventually* get sent, even if the expected “fixed” number of characters is not sent to the EM1000 or if the expected sentinel characters are missing or get trashed.

If your transmitting device sends sentinel characters (such as **<CR>**), then you could use a **<CR>** as a sentinel character to trigger the packet transmission. Even if this is done, you may also elect to send the accumulated data after a timeout value in the event that a noise burst trashes the **<CR>**, or the incoming data are interrupted prior to the **<CR>** being sent. Sentinel characters cannot generally be used when “binary” data are being transmitted.

If your transmitting device sends a data packet with a fixed number of characters, then you have the *option* of using the **`bufferlength`** parameter. This option should be used in conjunction with the **`timeout`** option, so that the data packet does eventually get sent in its entirety even though a character is somehow “missed” the first time.

Where pure binary data are being sent (no sentinel characters and the data length is not uniform), then the only viable option is **`timeout`**. If the transmitting device can send the data in a packet, with very little time between characters followed by a time gap, then this would actually be the preferred mechanism. If the **`timeout`** value is set to a value longer than the intercharacter time, the data will be transmitted by the soon after the last character in the packet.

The **`timeout`** option should also be tried if the other mechanisms prove not to be reliable.

If more than one of these options is enabled, they will work “in parallel,” and fairly complicated situations can occur. For example, if the timeout is set to 10 seconds, the **`bufferlength`** parameter is set to 100 characters, and sentinel characters is set to **<CR>**, then the following sample scenarios could happen.

1. Four characters are received by the EM1000 in 0.1 seconds, followed by nothing for 9.9 seconds. The four characters will be transmitted by the EM1000 after 10 seconds.
2. Four characters, followed by a **<CR>**, are received by the EM1000 in 0.1 seconds The four characters will be transmitted immediately after the **<CR>**.
3. One hundred characters (not including a **<CR>**) are received by the EM1000 in 3 seconds. The one hundred characters will be transmitted by the EM1000 immediately after the last one arrives.
4. 145 characters (with a **<CR>** as the 23rd character) are received by the EM1000 in 4 seconds. The EM1000 sends characters 1 through 23 as one packet because of the sentinel character, then sends characters 24 through 124 as another packet because of the **`bufferlength`** parameter setting. The EM1000 then waits another 10 seconds and sends characters 125 through 145 because of the **`timeout`** value.

Specific Configurations for a Real Network

You must first decide what your connection strategy will be. Use

`connect=demand`

when the EM1000 does not initiate the opening of a link, but listens for another EM1000 or other Ethernet device trying to open the link to it.

`connect=active`

to initiate the opening of a link automatically to the remote address when the EM1000 is powered up or reset. The link will stay open. If the link cannot be established immediately, then the EM1000 will retry every 3 seconds.

`connect=request`

when this EM1000 will initiate a link request, using the **`open`** command, or via the CTS line (**`rtsmode`**) or DSR line (**`dsrmode`**) to the remote address. The link is dropped manually with a **`close`** command, or when CTS or DSR go false (if they were used to open the link). If an attempted link fails, there are no automatic retries by the EM1000, although the RS-232 device may be set up to initiate an automatic retry.

Refer to the description for **`dsrmode`** and **`rtsmode`** in Chapter 4, “Commands,” for further details.

Some sample configurations that may meet your needs are provided below.

RS-232 (3-wire, listen mode)

The 3-wire RS-232 device waits for another remote device to communicate with it. The remote EM1000 will activate the link. Use these settings.

```
connect=demand
configmode=data
dsrmode=disabled
rtsmode=disabled
```

RS-232 (3-wire)

The RS-232 device is either a 3-wire device (RxD, TxD, and GND) or has other signals, but the other signals are not used. On power-up or reset, the EM1000 will establish communication with a fixed destination. The link will remain up at all times. Data will flow freely in both directions without flow control. Use these settings.

```
connect=active
configmode=data
dsrmode=disabled
rtsmode=disabled
```

RS-232 (5-wire, RTS/CTS flow control)

The RS-232 device is a 5-wire device (RxD, TxD, RTS, CTS, and GND), and the RTS/CTS lines are used for flow control (to slow down or stop data from the RS-232 device if the link is not responding). On power-up or reset, the EM1000 will establish communication with a fixed destination. The link will remain up at all times. Use these settings.

```
connect=active
configmode=data
dsrmode=disabled
rtsmode=flow
```

RS-232 (5-wire, CTS initiates link)

The RS-232 device is a 5-wire device (RxD, TxD, RTS, CTS, and GND), and the CTS line is used to initiate a link to the remote location. On power-up or reset, and after CTS is true, the EM1000 will establish communication with a fixed destination. The link will remain up until CTS is false. RTS will indicate the status of the link. Use these settings.

```
connect=request
configmode=data
dsrmode=disabled
rtsmode=connect
```

RS-232 (5-wire, activate remote DTR)

The RS-232 device needs to send data to a remote location, and also manipulate the DTR line at the remote end. Note that the remote end must be another EM1000. On power-up or reset, the EM1000 will establish a link with a remote modem at a fixed location. Use these settings.

```
connect=active
configmode=data
dsrmode=passthrough
rtsmode=disabled
```

Changing Remote EM1000s While Running

There are four ways available for the local EM1000 to change which remote EM1000 it is communicating with without initiating a reset or a power cycle.

1. Connect a totally independent RS-232 device to the **CONSOLE** port. This separate device and the data source connected to the **COM1/ RS-232** port need to coordinate their activities when the destination links change. Set

   ```
   configmode=data
   ```

   for the **COM1/RS-232** port. Configuration commands and the address of the remote target will be sent only to the **CONSOLE** port. This method is especially useful during manual debugging.

2. If the RS-232 device has a second serial port, it can be hooked to the **CONSOLE** port. The RS-232 device will need to send data using the **COM1/RS-232** port and to send commands via the **CONSOLE** port. Set

   ```
   configmode=data
   ```

   for the **COM1/RS-232** port. Configuration commands and the address of the remote target are sent only to the **CONSOLE** port.

3. If the RS-232 device has an independently controlled RS-232 output signal (such as DTR or RTS), the DTR or RTS signal can be connected to the EM1000's DSR input to toggle the **COM1/RS-232** port between configuration mode and data throughput mode. Set

   ```
   configmode=dsr
   dsrmode=configselect
   ```

   for the **COM1/RS-232** port.

   The DTR output signal acts as a toggle to change the **COM1/RS-232** port from data throughput mode to configuration mode. If the DTR signal is high, the **COM1/RS-232** port is in a data throughput mode. If the DTR signal is low, the **COM1/RS-232** port is in a configuration mode.

   The RS-232 device connected to the **COM1/RS-232** port's DSR line must be able to control its DTR or RTS output independently from the normal data flow.

   The DTR/DSR signal cannot be used in throughput or connect modes when it used to toggle between modes.

   Instead of using the DTR signal from the attached RS-232 device, the RTS signal may be used as long as the RTS signal can be controlled independently from the data flow.

4. If the RS-232 device has an "open-collector" digital output, that output can be connected to the CTL line on the EM1000. The open-collector output signal has to be coordinated with the data output to the **COM1/RS-232** port. When the CTL line is low, the **COM1/RS-232** port is in a configuration mode. When CTL goes high, the **COM1/RS-232** port returns to a data throughput mode. Set

   ```
   configmode=gpio
   ```

   for the **COM1/RS-232** port.

   The Z-World Web site at http://www.zworld.com provides some demonstration Dynamic C programs to illustrate the use of this method with a PK2200 controller.

Example

The following sample scenario illustrates “on the fly” configuration changes that allow multiple links to be sequentially established. This scenario is based on connecting the open-collector output from the RS-232 device to the CTL line on the EM1000. A single RS-232 serial port will suffice for the RS-232 device in this example.

1. Set the local EM1000 for

   ```
   configmode=gpio
   connect=demand
   ```

   The local EM1000 is configured to allow configuration changes in response to the CTL input while the local EM1000 running. The EM1000 is in a listening mode in case another modem requests a link.

2. Set the remote EM1000s for

   ```
   connect=demand
   ```

   This means they, too, are in a listening mode.

3. The RS-232 device connected to the local EM1000 decides to send data to test3.mycompany.com, test4.mycompany.com, and test5.mycompany.com.

4. The RS-232 device starts the reconfiguration by pulling `gpio` to ground.

5. The RS-232 device sends the destination address

   ```
   remotename=test3.mycompany.com
   ```

6. The RS-232 device unit sets

   ```
   connect=request
   ```

   This takes the EM1000 out of listening mode.

7. The RS-232 device sends an `open` command to open a connection.

8. The RS-232 device returns the EM1000 to the data throughput mode by pulling `gpio` up.

9. The RS-232 device sends the desired data to test3.mycompany.com via the **COM1/RS-232** port.

10. The RS-232 device reconfigures the **COM1/RS-232** port to the configuration mode by pulling `gpio` to ground.

11. The RS-232 device sends a `close` command to close the connection.

Refer to the Z-World Web site at http://www.zworld.com for sample Dynamic C programs that provide more details for specific Z-World controllers.

Repeat steps 4–11, changing the destination address in Step 5 as needed.

Software Description

The software is based on a DOS operating system, with internal flash memory used for drives A and B. The data on drive A are read-only, and consist of system files. The B drive contains the EM1000 operating software, startup files, and configuration files shown in Table 3-2. The only files that can/should be modified by the user are **DWN.EM** and **WATTCP.CFG**.

Table 3-2. EM1000 Configuration and Operating Files

File	Description
User-Modifiable Files	
DWN.EM	Working configuration file.
WATTCP.CFG	Working IP settings file.
Fixed Files	
ACTIVE.BAT	When executed, makes **ACTIVE.CFG** and **ACTIVE.EM** the working **WATTCP.CFG** and **DWN.EM** files, used only during initial checkout.
ACTIVE.CFG	Default IP settings for **active** mode, used only during initial checkout.
ACTIVE.EM	Default configuration settings for **active** mode, used during initial checkout.
EDIT.COM	Used to edit **DWN.EM** and **WATTCP.CFG**.
ETHERCOM.EXE	Main executable file, uses **DWN.EM** to configure EM1000.
FTPD.EXE	Web update that allows future firmware updates to be done via FTP
LISTEN.BAT	When executed, makes **LISTEN.CFG** and **LISTEN.EM** the working **WATTCP.CFG** and **DWN.EM** files, used only during initial checkout.
LISTEN.CFG	Default IP settings for **listen** mode, used only during initial checkout.
LISTEN.EM	Default configuration settings for **listen** mode, used during initial checkout.
NE2000.COM	Ethernet packet driver.
STARTUP.BAT	Sets initial operating conditions for the EM1000 and invokes **ETHERCOM.EXE** on power-up or reset.
UP.COM	Used to transfer configuration files or future upgrades of other files from a PC to the EM1000 using the Xmodem protocol.

Blank

Chapter 4: Commands

Chapter 4 describes the commands and their parameters used to program the EM1000. The following sections are included.

- Categories
- Command Context
- Descriptions

Categories

The commands associated with the EM1000 fall into five categories, and are listed below.

1. Interactive Commands

   `close`
   `dtr`
   `echo`
   `established`
   `exit`
   `help`
   `location`
   `open`
   `password`
   `ping`
   `resolve`
   `rts`
   `show`
   `source`
   `spy`
   `statistics`
   `verbose`
   `version`

2. Hardware Setup Commands

   `baud`
   `configmode`
   `databits`
   `parity`
   `stopbits`

3. Flow Control Commands

   `dsrmode`
   `rtsmode`

4. Packetization Commands

   `bufferlength` or `length`
   `buffertimeout` or `timeout`
   `dropsentinels` or `drop`
   `maxpacket`
   `sentinels`

5. TCP/IP Commands

   `allowedname`
   `allowedport`
   `connect`
   `domain` or `domainslist`
   `gateway`
   `hostip` or `my_ip`
   `hostname`
   `hostdataport` or `local_dataport`
   `hoststateport`
   `nameserver`
   `netmask`
   `remotename` or `target_name`
   `remotedataport` or `remoteport` or `target_dataport`
   `remoteretry`
   `remotestateport` or `target_stateport`
   `remotetimeout`

See the section on “Setting Modem Operating Parameters” and Table 3-1 in Chapter 3, “Programming Reference,” for the default parameter values and for information on how to change them.

Command Context

Most commands can be used regardless of whether the EM1000 is in a configuration mode or a data throughput mode. However, some commands are restricted. For example, the `close` command does not make sense unless the `open` command has already been issued after `connect=request`. In another example, `allowedname` is meaningless when `connect=request` or `connect=active`.

The list below identifies the context in which certain commands are used.

```
connect=request
   open
     close
   remotedataport
   remotename
   remotestateport
   dsrmode=connect
   rtsmode=connect
```

```
connect=demand
   allowedname
   allowedport
   hostdataport
   hoststateport
   remotetimeout
```

```
connect=active
   remotedataport
   remotename
   remotestateport
```

```
dsrmode=configselect
configmode=dsr
```

The DTR signal to the DSR input on the **COM1/RS-232** port determines whether the EM1000 is in command or data throughput mode.

```
dsrmode=disabled
   dtr on
   dtr off
```

```
rtsmode=disabled
   rts on
   rts off
```

```
domain
remotename
```

The DNS server name does not have to be specified when an IP address is used, but must be specified when a “text” address is used.

Descriptions

`allowedname [dnsname | any]`

The **`allowedname`** command is used to restrict access to the EM1000. If **`allowedname`** is configured, the EM1000 will only accept incoming connections from the specified remote address. The **`dnsname`** parameter can be a fully qualified domain name, a partially qualified domain name, or an IP address. The **`dnsname`** parameter is resolved to an IP address before being used. Changing **`allowedname`** causes any established data and state connections to be closed.

The default **`allowedname`** is **`any`**, and allows connections from any IP address.

`allowedport [port]`

The **`allowedport`** command is used to restrict access to the EM1000. If the **`allowedport`** is configured, the EM1000 will only accept incoming connections on the specified remote port. The **`port`** number can be any valid number between 0 and 65535. Changing the **`allowedport`** causes any established data and state connections to be closed.

The default **`port`** is 0 and allows connections from any remote port.

`baud [rate]`

The **`baud`** command establishes the baud rate for the **COM1/RS-232** port. The **`rate`** can be any valid baud rate. Changing **`rate`** causes the **COM1/RS-232** port to be reconfigured, which may cause the loss of characters being sent or received at that time. Exercise care when setting **`rate`** since it is possible to enter values other than the standard 1200 bps, 2400 bps, …, 19,200 bps. The EM1000 will try to use the rate that was set.

The default **`rate`** is 19,200 bps.

`bufferlength [count]`
`length [count]`

The **`bufferlength`** command specifies the minimum number of data bytes to accumulate before a packet transmission is scheduled. The actual packets have a length between **`bufferlength`** and **`maxpacket`** inclusive. A higher data throughput results in larger packets.

If **`bufferlength`** is set higher than **`maxpacket`**, the effect will be the same as if **`bufferlength`** was equal to **`maxpacket`**.

This buffering strategy attempts to reduce overall network utilization by buffering data until sufficient bytes are received to warrant a network transmission. The **`count`** parameter specifies the number of bytes buffered before a network transmission occurs. Data received on the **COM1/RS-232** port are buffered until **`count`** bytes are received, then a network packet is transmitted. **`count`** is any number between 0 and 1024.

length is a synonym for **bufferlength**.

The default **count** is 0 and causes each byte received to be transmitted in its own network packet.

The buffering strategy is also affected by the **buffertimeout** and **sentinels** commands. Each buffering strategy command operates independently of and in parallel with the others.

buffertimeout [ticks]
timeout [ticks]

The **buffertimeout** command changes the buffering strategy for the **COM1/RS-232** port. The buffering strategy attempts to reduce overall network utilization by buffering data until sufficient bytes are received to warrant a network transmission. The **ticks** parameter specifies the number of clock ticks (55 ms) that must elapse between successive bytes before a network transmission occurs. Data received on the **COM1/RS-232** port is buffered until the **ticks** number of ticks elapsed between the reception of any two bytes, then a network packet is transmitted.

timeout is a synonym for **buffertimeout**.

The default **ticks** is 0, which causes no timeout criteria to be enforced.

The buffering strategy is also affected by the **bufferlength**, **maxpacket**, and **sentinels** commands. Each of the buffering strategy commands operates independently of and in parallel with the others.

Tip

Erratic operation, including lockups, may occur when a setting of **buffertimeout = 1** is used during periods of high Ethernet traffic and/or noisy links. A minimum value of 2 is recommended when **buffertimeout** is not 0.

close

The **close** command causes any established data and state connections to be closed. Any buffered data are transmitted before the connection is closed. The following action occurs after the **close** command in each connection configured via the **connect** command.

1. **Listen/Demand**—The EM1000 returns to listening for incoming connections.
2. **Active**—The EM1000 begins to attempt to establish a connection.
3. **Request**—The EM1000 waits until requested to attempt to establish a connection.

`configmode [gpio | dsr | config | data]`

The **`configmode`** command sets the configuration for the **COM1/RS-232** port. The **COM1/RS-232** port can operate in two possible modes: configuration mode and data throughput mode. When the **COM1/RS-232** port is in the configuration mode, a command prompt appears and most of the EM1000 commands can be used. The configuration mode allows a device connected to the **COM1/RS-232** port to control the configuration of the EM1000 by issuing the same commands that are used for the **CONSOLE** port. When the **COM1/RS-232** port is in the configuration mode, data received from the **COM1/RS-232** port are not transmitted across the network. When the **COM1/RS-232** port is in the data throughput mode, data received from the **COM1/RS-232** port are sent across the network, and data received from the network are sent to the **COM1/RS-232** port. The command arguments are listed below.

`gpio`—The general-purpose input pin (CTL) is used to determine whether the **COM1/RS-232** port is operating in a configuration mode or a throughput mode. If the CTL signal is low, the **COM1/RS-232** port is in a configuration mode; if the CTL signal is high (or open), the **COM1/RS-232** port is in a data throughput mode.

`dsr`—The DSR signal at the **COM1/RS-232** port is used to determine of the **COM1/RS-232** port is operating in a configuration mode or a throughput mode. If the DSR signal is low, the **COM1/RS-232** port is in a configuration mode; if the DSR signal is high, the **COM1/RS-232** port is in a data throughput mode. The **`dsrmode`** command must also be set to **`configselect`** to condition the DSR to operate in this way.

`config`—The **COM1/RS-232** port is forced into a configuration mode.

`data` –The **COM1/RS-232** port is forced into a data throughput mode.

`connect [demand | listen | active | request]`

The **`connect`** command determines how the EM1000 operates relative to the network. The EM1000 either listens for incoming connections or attempts to actively attempt to connect to a remote device. The command arguments for each mode are described below.

`demand`—Synonym for **`listen`**.

`listen`—The EM1000 listens to the network for incoming connection requests. The EM1000 listens on the **`hostdataport`** for incoming data connections, and it listens on the **`hoststateport`** for incoming state connections. The incoming connections may be restricted to a particular IP address by the **`allowedremotename`**

command. The incoming connections may be restricted to a particular port by the **allowedremoteport** command. Only one connection can be established at a time.

active—The EM1000 attempts to establish a remote connection. If the attempt fails, the EM1000 will continue to try until a successful connection is established. If a connection is established and then is closed sometime in the future, the EM1000 will make an immediate attempt to re-establish the connection. The **remotename** command is used to configure the IP address of the remote connection. The **remotedataport** command is used to configure the port for the remote data connection. The **remotestateport** command is used to configure the port for the remote state connection. When the EM1000 is in the active mode, it does not listen for outside connection requests.

request—The EM1000 waits until a request to establish a connection is made. There are three methods of requesting a connection. Regardless of the method used to attempt to establish a connection, if the attempt to open a connection fails, the EM1000 returns to waiting for another request; no automatic retries are attempted. The **remotename** command is used to configure the IP address of the remote connection. The **remotedataport** command is used to configure the port for the remote data connection. The **remotestateport** command is used to configure the port for the remote state connection. The three methods of requesting a connection are described below.

> **open**—The **open** command can be used to request that the EM1000 attempt to establish a data and state connection. The **close** command is used to request that any open data or state connection be closed.
>
> **dsrmode**—The **dsrmode** command can be used to configure the EM1000 to monitor the **COM1** DSR signal for transitions. When the DSR signal transitions from 0 to 1, the EM1000 will attempt to open a connection. When the DSR signal transitions from 1 to 0, the EM1000 will close any open data or state connection.
>
> **rtsmode**—The **rtsmode** command can be used to configure the EM1000 to monitor the **COM1** CTS signal for transitions. When the CTS signal transitions from 0 to 1, the EM1000 will attempt to open a connection. When the CTS signal transitions from 1 to 0, the EM1000 will close any open data or state connection.

The default **connect** is **request**.

databits [bits]

The **databits** command establishes the number of data bits in each byte for the **COM1/RS-232** port. The **bits** parameter can be either 7 or 8. Changing **bits** causes the **COM1/RS-232** port to be reconfigured, which may cause the loss of characters being sent or received at that time.

The default **bits** is 8.

domain [names]
domainslist [names]

The **domain** command establishes the local DNS domain that the EM1000 is a member of. The **names** parameter is a comma separated list of DNS names. The **names** parameter is used by various other commands when attempting to resolve a DNS name to an IP address. When resolving DNS names, each of the DNS names in the **names** parameter is appended to the specified **remotename** in an attempt to produce a fully qualified DNS name that can be resolved to an IP address. For example, if the domain is

mycompany.com

and **remotename** is

unit3,

the EM1000 will attempt to establish a connection to **unit3.mycompany.com**, which corresponds to an IP address such as

201.202.123.101

The **domain** command is optional—**domain** and **hostname** can always be replaced with the real IP address of the remote device.

domainslist is a synonym for **domain**.

The default **names** is an empty string, which causes no domain name to be appended to DNS names while attempting to resolve the name to an IP address.

drop [enabled | on | true | yes | disabled | off | false | no]
dropsentinels [enabled | on | true | yes | disabled | off | false | no]

The **drop** command changes the way sentinel characters are treated when received on the **COM1/RS-232** port. Sentinel characters trigger a flush of buffered data, and the sentinel characters themselves can either be forwarded onto the network just like other characters in the data stream, or they can be discarded. The **enabled**, **on**, **true**, and **yes** parameters are synonyms for dropping the sentinels. The **disabled**, **off**, **false**, and **no** parameters are synonyms for forwarding the characters.

dropsentinels is a synonym for **drop**.

The default for **drop** is **disabled**, which causes the sentinel characters to be forwarded.

dsrmode [connect | passthrough | configselect | disabled | off | no | false]

The **dsrmode** command changes how the **COM1/RS-232** DSR/DTR signals are used. Each parameter is described below.

connect—The DSR signal is used to **open** a connection or to **close** a connection. In the **connect** mode, the transition of DSR from 0 to 1 is a request to **open** a data and state connection. Depending on the **connect** command setting, network connection activity might be initiated. The transition of DSR from 1 to 0 is a request to **close** any open data or state connections. In the **connect** mode, the DTR signal is used to indicate that a connection has been successfully established for the data connection. The state connection does not activate the DTR signal. The DTR signal is high to indicate that the connection is established, and is low when the connection is closed.

passthrough—The **passthrough** mode is intended to make a transparent connection for the DSR/DTR signal pair. Both the local and the remote EM1000s must have this set for this mode to work. The DSR signal is passed through the network to the remote unit via the state connection. The remote unit sets its own DTR output signal to the same state as the local EM1000 DSR input signal. Likewise, the remote unit will send the EM1000 state information via the state connection whenever it detects a change on its DSR input signal. The local EM1000 will then set its DTR output signal to the same state as the remote unit's DSR input signal.

configselect—The DSR signal is used to determine the **configmode** for the **COM1/RS-232** port. The **COM1/RS-232** port can operate either in a command mode or a data throughput mode. When **dsrmode** is set to **configselect**, the DSR signal is used to determine if the **COM1/RS-232** port is in a command mode or a data throughput mode. The **configmode** command must also be set to **dsr** to operate with the **configselect** parameter.

disabled —The DSR signal is not used and is ignored. The DTR signal is initially high, but can be changed via the **dtr** command. **off**, **no**, and **false** are synonyms for **disabled**.

The default **dsrmode** is **disabled**, which causes the **COM1/RS-232** DSR signal to be ignored.

dtr [enabled | on | true | yes | disabled | off | false | no]

The **dtr** command is used to change the state of the **COM1** DTR output signal. The DTR signal can be set high or low. The DSR/DTR pair of signals is used for various purposes, and is configured via the **dsrmode** command.

The EM1000 will respond correctly to changes in the DTR signal only when **dsrmode** is configured to be **disabled**. Any other parameter configurations can cause improper operation of the EM1000.

enabled, **on**, **true**, and **yes** are synonyms for setting DTR high. **disabled**, **off**, **false**, and **no** are synonyms for setting DTR low.

The default for **dtr** is **enabled**, which causes the DTR signal to be high.

echo [arg1 arg2 …]

The **echo** command displays each of its arguments on a separate line. It can be useful for debugging command scripts. It is not normally used outside of a source file.

Refer to the description of the **source** command later in this chapter.

established

The **established** command displays the connection state for the data, state, and Telnet connections.

exit

The **exit** command terminates the application.

The **exit** command is not available from a Telnet session.

gateway [ipaddress]

The **gateway** command is used to change the default gateway for network packet routing. The **ipaddress** parameter can be supplied, but the normal way for configuring **gateway** is with the **WATTCP.CFG** file.

The default **gateway** is configured via the **WATTCP.CFG** file.

help [regexp]

The **help** command is used to display a short on-line description of each command. The **regexp** argument is a regular expression that can be used to restrict the help text to matching commands. The regular expression syntax consists of asterisk (*), question mark (?), open square bracket ([), closed square bracket (]), and carrot (^) characters. The syntax for regular expression matching follows.

*—Matches any sequence of characters.

?— Matches any single character.

[]—Matches any character in the group.

[^]—Matches any character not in the group.

hostip [ipaddress]
my_ip [ipaddress]

The **hostip** command is used to change the IP address of the EM1000. The **ipaddress** parameter can be changed from the command line or it can be included in the **DWN.EM** file. The normal way for configuring **hostip** is with the **WATTCP.CFG** file.

my_ip is a synonym for **hostip**.

The default **my_ip** is configured via the **WATTCP.CFG** file.

hostip is not recognized in the **WATTCP.CFG** file—use **my_ip** in the **WATTCP.CFG** file.

hostname [dnsname]

The **hostname** command is used to change the DNS name of the EM1000. The **dnsname** parameter can be changed from the command line or it can be included in the **DWN.EM** file. The normal way for configuring **hostname** is with the **WATTCP.CFG** file.

The default **hostname** is configured via the **WATTCP.CFG** file.

hostdataport [port]
local_dataport [port]

The **hostdataport** command is used configure the **port** number for incoming data connections. When the **connect** command is configured for a **listen** or a **demand** mode, the **hostdataport** is the internal **port** number to which the EM1000 **COM1/RS-232** port is connected.

local_dataport is a synonym for **hostdataport**.

The **port** number can be any valid number between 0 and 65535. Changing the **hostdataport** command causes any established data and state connections to be closed.

The default **hostdataport** is 8888.

hoststateport [port]
local_stateport [port]

The **hoststateport** command is used to configure the **port** number for incoming state connections. When the **connect** command is configured for a **listen** or a **demand** mode, the **hoststateport** is the **port** number the EM1000 listens on for connection requests.

local_stateport is a synonym for **hoststateport**.

The **port** number can be any valid number between 0 and 65535. Changing the **hoststateport** command causes any established data and state connections to be closed.

The default **hoststateport** is 8889.

location

The **location** command contains a string that may be included in the **DWN.EM** file, for example,

```
location = "near the entry door at the north end"
```

This string is returned when **location** is typed at a Telnet command prompt.

maxpacket [count]

The **maxpacket** parameter limits the length of any data packet transmitted by the EM1000. The **bufferlength** parameter specifies the minimum number of data bytes to accumulate before a packet transmission is scheduled. If **bufferlength** is set higher than **maxpacket**, the effect will be the same as if **bufferlength** was equal to maxpacket.

The default **maxpacket** setting is 1024, which is also the maximum value.

Avoid setting **maxpacket** to 1, which will cause each data byte received to be sent in its own packet, impacting the efficiency of the Ethernet. For most efficient use, the packet length should contain at least 60 data bytes.

nameserver [ipaddress]

The **nameserver** command is used to change the DNS server IP address of the EM1000. The **ipaddress** parameter can be changed from the command line or it can be included in the **DWN.EM** file. The normal way for configuring **nameserver** is with the **WATTCP.CFG** file.

The default is for **nameserver** to be configured via the **WATTCP.CFG** file.

netmask [ipmask]

The **netmask** command is used to change the subnet mask of the EM1000. The **ipmask** parameter can be changed from the command line or it can be included in the **DWN.EM** file. The normal way for configuring **netmask** is with the **WATTCP.CFG** file.

The default is for **netmask** to be configured via the **WATTCP.CFG** file.

open

The **open** command attempts to established data and state connections. The following actions occur for the connections configured via the **connect** command.

> **listen/demand**—Nothing. The **open** command has no meaning in this context.
>
> **active**—Nothing. The **open** command has no meaning in this context.
>
> **request**—The EM1000 will attempt to establish a connection. The **remotename** command is used to configure the IP address of the remote connection. The **remotedataport** command is used to configure the port for the remote data connection. The **remotestateport** command is used to configure the port for the remote state connection.

parity [none | even | odd]

The **parity** command establishes the **parity** in each byte for the **COM1/RS-232** port. Changing the **parity** causes the **COM1/RS-232** port to be reconfigured, which may cause the loss of characters being sent or received at that time.

The default **parity** is **none**.

password [text]

The **password** command establishes a password for Telnet sessions. The **text** argument is the new password. Once a Telnet session is established and the EM1000 is configured for a password, the Telnet session will prompt for the password before allowing any commands to be executed.

ping [dnsname]

The **ping** command is used to verify network connectivity to a remote unit. The **dnsname** parameter can be a fully qualified domain name, a partially qualified domain name, or an IP address. The **dnsname** parameter is resolved to an IP address before being used.

An ICMP Ping packet is sent, and the EM1000 then waits up to 10 seconds for a response.

The **ping** command might not work if your EM1000 is behind a firewall.

remotename [dnsname]
target_name [dnsname]

The **remotename** command is used to configure the name and IP address of the remote unit. The **dnsname** parameter can be a fully qualified domain name, a partially qualified domain name, or an IP address. The **dnsname** parameter is resolved to an IP address before being used. The following actions occur for the connections configured via the **connect** command.

> **listen/demand**—Nothing. The **remotename** command has no meaning in this context.
>
> **active**—The EM1000 will use the **dnsname** parameter as the remote address.
>
> **request**—The EM1000 will use the **dnsname** parameter as the remote address.

Changing the **remotename** command causes any established data and state connections to be closed.

target_name is a synonym for **remotename**.

The default **remotename** is an empty string, which causes no remote connections to be attempted.

Refer to the discussion of the **domain** command for more information.

remotedataport [port]
target_dataport [port]
remoteport [port]

The **remotedataport** command is used to configure the **port** number of the remote EM1000. The **port** parameter can be any valid **port** between 0 and 65535. The following actions occur for the connections configured via the **connect** command.

> **listen/demand**—Nothing. The **remotedataport** command has no meaning in this context.
>
> **active**—The EM1000 will use the **port** parameter as the remote port for data connections.
>
> **request**—The EM1000 will use the **port** parameter as the remote port for data connections.

Changing the **remotedataport** causes any established data and state connections to be closed.

target_dataport and **remoteport** are synonyms for **remotedataport**.

The default **remotedataport** is 8888.

remoteretry [x]

If an active EM1000 cannot connect with a remote device, it will try to re-establish the link. If the remote device was actually taken out of service for some reason, then the attempted retries will result in unnecessary link activity. To get around this, there are two modes of operation that can be established:

> The option exists to program the retry time value in the **DWN.EM** file. The value is programmed using the command **REMOTERETRY=x**. **x** can be a value from 0 to 3600. When **x** is 0, a new algorithm is used—the link is attempted again after 1 second, then 2 seconds, then 4 seconds, …, up to 64 seconds (further attempts are made at 64-second intervals). If **x** is non-zero, the attempts at a link are made at **x**-second intervals.
>
> If this command line is not included in the **DWN.EM** file, the retries will occur at 1-second intervals.

remotestateport [port]
target_stateport [port]

The **remotestateport** command is used to configure the **port** number of the remote unit. The **port** parameter can be any valid **port** between 0 and 65535. The following actions occur for the connections configured via the **connect** command.

> **listen/demand**—Nothing. The **remotestateport** command has no meaning in this context.
>
> **active**—The EM1000 will use the **port** parameter as the remote port for state connections.
>
> **request**—The EM1000 will use the **port** parameter as the remote port for state connections.

Changing **remotestateport** causes any established data and state connections to be closed.

target_stateport is a synonym for **remoteStatePort**.

The default **remoteStatePort** is 8889.

remotetimeout [x]

A programmable inactivity timeout allows the link to close down automatically (in the **connect=demand** mode) after a programmed timeout in the serial data stream. This command can be included in the **DWN.EM** file in this form:

```
REMOTETIMEOUT=x
```

x can be any number between 1 and 3600, and is in seconds.

resolve [dnsname]

The **resolve** command is used to verify that a DNS name can be resolved to an IP address. The **dnsname** parameter can be a fully qualified domain name, a partially qualified domain name or an IP address.

rts [enabled | on | true | yes | disabled | off | false | no]

The **rts** command is used to change the state of the **COM1/RS-232** RTS signal. The RTS signal can be set high or low. The RTS/CTS pair of signals is used for various purposes, and is configured via the **rtsmode** command.

> The EM1000 will respond correctly to changes in the RTS signal only when **rtsmode** is configured to be **disabled**.

enabled, **on**, **true**, and **yes** are synonyms for setting RTS high. **disabled**, **off**, **false**, and **no** are synonyms for setting RTS low.

The default for **`rts`** is **`enabled`**, which causes the RTS signal to be high.

`rtsmode [connect | flow | passthrough | flowcontrol | disabled | off | no | false]`

The **`rtsmode`** command changes how the **COM1** RTS/CTS signals are used. Each parameter is described below.

`connect`—The CTS signal is used to **`open`** a connection or to **`close`** a connection. In the **`connect`** mode, the transition of CTS from 0 to 1 is a request to **`open`** a data and state connection. Depending on the **`connect`** command setting, network connection activity might be initiated. The transition of CTS from 1 to 0 is a request to **`close`** any open data or state connections. In the **`connect`** mode, the RTS signal is used to indicate that a connection has been successfully established for the data connection. The state connection does not activate the RTS signal. The RTS signal is high to indicate that the connection is established, and is low when the connection is closed.

`flow`—The RTS/CTS signals are used to control flow between the EM1000 and the RS-232 device attached to the **COM1/RS-232** port. **`flowcontrol`** is a synonym for **`flow`**.

`passthrough`—The **`passthrough`** mode is intended to make a transparent connection for the RTS/CTS signal pair. The CTS signal is passed through the network to the remote unit via the state connection. The remote unit sets its own RTS signal to the same state as the local EM1000 CTS. Likewise, the remote unit will send the EM1000 state information via the state connection whenever it detects a change on its CTS signal. The local EM1000 will then use the remote's CTS state to configure its own RTS signal.

`disabled` —The CTS signal is not used and is ignored. The RTS signal is initially high, but can be changed via the **`rts`** command. **`off`**, **`no`**, and **`false`** are synonyms for **`disabled`**.

The default **`dsrmode`** is **`disabled`**, which causes the **COM1/RS-232** CTS signal to be ignored.

`sentinels [decimal1 decimal2 …]`

The **`sentinels`** command changes the buffering strategy for the **COM1/RS-232** port. The buffering strategy attempts to reduce overall network utilization by buffering data until sufficient bytes are received to warrant a network transmission. The **`sentinels`** command causes the EM1000 to monitor the data and search for characters that will trigger a flush of the buffered data. The **decimal** parameters specify characters that trigger buffer flushes. Additionally, the **dropsentinels** command provides an option for the **sentinels** characters to be removed or passed through to the remote unit.

The default is for **sentinels** to not be configured, which causes no data to trigger buffer flushes. To turn the **sentinels** command off, type **sentinels disabled**, or **sentinels off**.

The buffering strategy is also affected by the **buffertimeout**, **bufferlength**, and **maxpacket** commands. Each of the buffering strategy commands operates independently of the others.

show [regexp]
all [regexp]

The **show** command is used to display the current settings for each of the configuration parameters. The **regexp** argument is a regular expression that can be used to restrict the text to matching commands. The regular expression syntax consists of asterisk (*****), question mark (**?**), open square bracket (**[**), closed square bracket (**]**), and carrot (**^**) characters. The syntax for regular expression matching follows.

*****—Matches any sequence of characters.

?— Matches any single character.

[]—Matches any character in the group.

[^]—Matches any character not in the group.

all is a synonym for **show**.

source [filename]

The **source** command is invoked from the **CONSOLE** port or from Telnet to read a file of EM1000 commands from the disk and execute each command. Note that the restricted Telnet commands described in Appendix E, "Telnet," will not function. The source file is created using the edit command at the DOS (**B:\>**) prompt.

spy [enabled | on | true | yes | disabled | off | false | no]

The **spy** command is used to monitor data traffic. When **spy** is enabled, the Line Status and Modem Status on the **COM1/RS-232** port are also made visible on the **CONSOLE** port. The **COM1/RS-232** data are bracketed by the following special characters.

{}—Received Line Status and Modem Status information is enclosed in braces.

() - Transmitted Line Status and Modem Status information is enclosed in parenthesis.

`{}`—Received Line Status and Modem Status information is enclosed in braces.

`()` - Transmitted Line Status and Modem Status information is enclosed in parenthesis.

The Line Status and Modem Status information consists of three bytes each, a letter and two hex digits. The Line Status information is encoded as LXX, where the “L” indicates Line Status, and the “XX” are the actual hex digits. The “XX” digits represent the contents of the **COM1/RS-232** UART Line Status Register.

The Modem Status information is encoded as “MXX,” where the “M” indicates Modem Status, and the “XX” are the actual hex digits. The “XX” digits represent the contents of the **COM1/RS-232** UART Modem Status Register.

The **`spy`** command shows the status of the above parameters for the **CONSOLE** port, the **COM1/RS-232** port (if the EM1000 is in configuration mode), and the Telnet port. To set **`spy`** on these ports, it is necessary to actually set the value from that port. For example, if you are in a Telnet session, and you type **`spy=on`**, then the **`spy`** mode is activated for the Telnet port, but has no effect on the **`spy`** mode on the other ports.

Generally, **`spy`** is used for debugging, and should be turned **`off`** for normal operation. Running the EM1000 with **`spy`** enabled causes some degradation in reliability and throughput.

If an EM1000 is passing data at a baud rate higher than 9600 bps, and **`spy`** is enabled for that EM1000, the overall data rate through the EM1000 may be decreased to allow the RS-232 device connected to the **CONSOLE** port to **`spy`** at the baud rate (9600 bps) fixed for the **CONSOLE** port. The baud rate does not actually change on the **COM1/RS-232** port, but the average data rate is reduced to an effective 9600 bps.

Refer to Appendix G, “Data and Signal Flow,” for more information on monitoring data and signal flow in **`spy`** mode.

`statistics [clear]`

The **`statistics`** command displays the transmitted, received and connection statistics. The statistics are gathered continuously as the EM1000 executes. The **`clear`** parameter can be used to reset the statistics counters to zero.

`stopbits [bits]`

The **`stopbits`** command establishes the number of stop bits in each byte passing through the **COM1/RS-232** port. The **`bits`** parameter can be either 1 or 2. Changing **`bits`** causes the **COM1/RS-232** port to be reconfigured, which may cause the loss of characters being sent or received at that time.

The default for **`bits`** is 1.

`verbose [enabled | on | true | yes | disabled | off | false | no]`

The **`verbose`** command enables the display of additional informative messages through the **CONSOLE** port, the **COM1/RS-232** port when it is used as a command port, or through the Telnet port.

The three ports are configured separately. To set **`verbose`** on these ports, it is necessary to actually set the value from that port. For example, if you are in a Telnet session, and you type **`verbose=on`**, then the **`verbose`** mode is activated for the Telnet port. This has no effect on the other ports.

The **CONSOLE** port is configured either from the **`DWM.EM`** file or by an interactive command. The **COM1/RS-232** port can be configured only when it is used as a command port; it is not possible to use the **`DWN.EM`** file to configure **verbose** to be enabled on the **COM1/RS-232** port. The Telnet is configured exclusively through a Telnet session.

The **`enabled`**, **`on`**, **`true`**, and **`yes`** parameters are synonyms for enabling the display. The **`disabled`**, **`off`**, **`false`**, and **`no`** parameters are synonyms for disabling the display.

The default for **`verbose`** is **`disabled`**, which causes the additional informative information not to be displayed. Turning **`verbose=on`** may slow down the EM1000 performance.

`version`

The **`version`** command displays the version number of the application.

APPENDIX A: SPECIFICATIONS

Appendix A provides comprehensive EM1000 physical, electronic and environmental specifications. The following sections are included.

- Electrical and Mechanical Specifications
- Headers and Jumpers

Electrical and Mechanical Specifications

Table A-1 lists electrical, mechanical, and environmental specifications for the EM1000.

Table A-1. EM1000 Specifications

Parameter	Specification
Enclosure Size	3.558" × 4.28" × 1.718" (90 mm × 109 mm × 44 mm)
Board Size	3.418" × 4.16" × 0.65" (87 mm × 106 mm × 17 mm)
Operating Temperature	-40 °C to 70° C
Humidity	5% to 95%, noncondensing
Input Power	9 V to 32 V DC, 2 W
Processor	Intel 386EX
SRAM	512K standard
Flash EPROM	512K standard
Serial Ports	One 3-wire RS-232 at 9600 bps One 7-wire RS-232 at up to 115,200 bps One 10BaseT Ethernet

EM1000 External Dimensions

Figure A-1 provides the external dimensions for the EM1000 board.

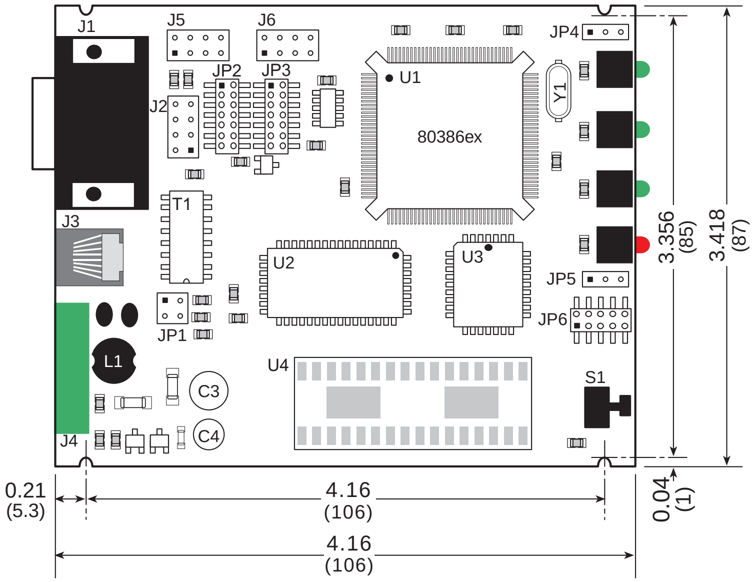


Figure A-1. EM1000 External Dimensions

When you mount the EM1000/EM1010 board in your own enclosure, Z-World recommends a clearance of 0.75" (19 mm) above the printed circuit board and 0.25" (6 mm) below the printed circuit board.

Figure A-2 provides the external dimensions for the EM1000 enclosure.

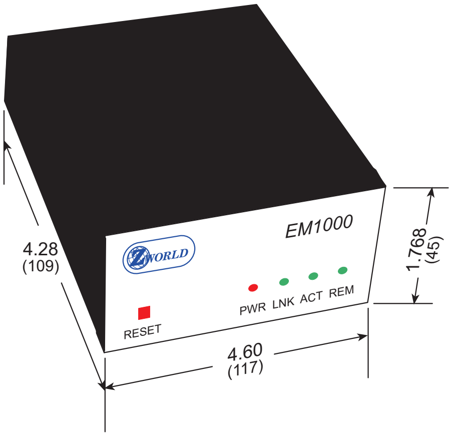


Figure A-2. EM1000 Enclosure External Dimensions

Base Plate

Figure A-3 shows the dimensions and the locations of the mounting holes of the EM1000 base plate. The base plate is made from 0.050" aluminum.

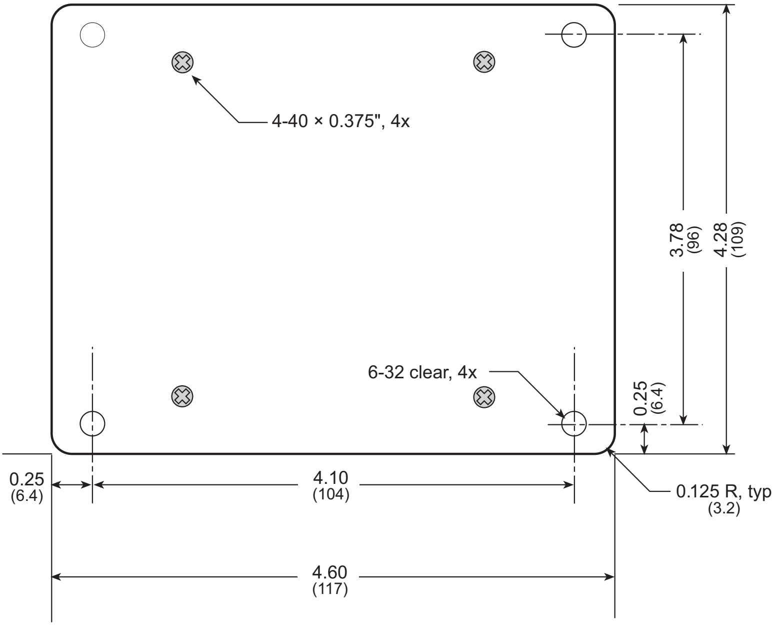


Figure A-3. EM1000 Base Plate Dimensions

The base plate is attached to the EM1000 enclosure with the four 4-40 × 0.375" screws shown in Figure A-3. The base plate facilitates mounting the EM1000. If the base plate is not needed, it may be removed and replaced with the four rubber feet supplied.

Headers and Jumpers

Figure A-4 illustrates the locations of headers on the EM1000 board.

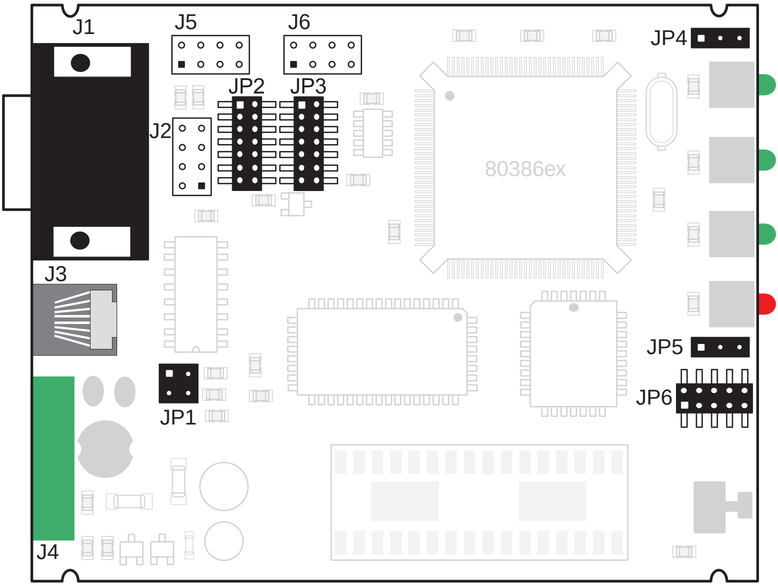


Figure A-4. Locations of EM1000 Headers

Table A-2 lists the header functions and Table A-3 lists the jumper configurations.

Table A-2. EM1000 Headers

Header	Description
J1	Configurable **COM1/RS-232** serial port.
J2	Configuration **CONSOLE** port 10-pin header.
J3	RJ-45 Ethernet port.
J4	Power supply connector, **CTL** input.
J5	Reserved for future use.
J6	Reserved for future use.

Table A-3. EM1000 Jumper Connections

Header	Pins	Description	Factory Default
JP1	1–2 3–4	Connect to enable RS-485 termination resistors.	Connected
JP2	1–2 3–4 5–6 7–8 9–10 11–12 13–14	Connect to enable TTL-level signals on **COM1/RS-232** port.	Not connected
JP3	1–2 3–4 5–6 7–8 9–10 11–12 13–14	Connect to enable RS-232 level signals on **COM1/RS-232** port.	Connected
JP4	1–2	Nonmaskable interrupt connected to watchdog output.	
	2–3	Nonmaskable interrupt connected to ground.	Connected
JP5	1–2	RS-485 driver enable connected to P3.5.	Connected
	2–3	RS-485 driver connected to RTS1.	
JP6	1–2 3–4	SRAM installed at U4.	
	1–3 2–4	Flash EPROM installed at U4.	Connected
	5–7 6–8	Boot from flash EPROM (U3).	Connected
	7–9 8–10	Boot from chip installed at U4 (no chip presently installed).	

Do ***not*** change the factory-default jumper settings. The alternative jumper settings have been incorporated into the EM1000 design to allow for future enhancements.

Appendix B:
Two-Modem Test Setup

Appendix B shows how to set up a test network using two EM1000s.

Installation in Test Network

When you are first learning how to use the EM1000, Z-World recommends that you install the EM1000 in a small test network instead of an existing network. Two EM1000s are needed. Their Ethernet ports can be connected using the Ethernet crossover cable supplied with the Developer's Kit, or they may be connected via an Ethernet hub using standard Ethernet cables.

☎ Z-World has the Ethernet hub and standard Ethernet cables for sale. For ordering information, or for more details about the various options and prices, call your Z-World Sales Representative at (530) 757-3737.

Connect via Crossover Cable

1. Connect the power supplies to both EM1000s according to the instructions in Chapter 2, "Getting Started."
2. Connect the **CONSOLE** port on one EM1000 to the serial port of a terminal or PC running a terminal emulation program using the DE9 to DE9 programming cable supplied with the Developer's Kit.
3. Connect the RJ-45 Ethernet ports on the EM1000s using the Ethernet crossover cable supplied with the Developer's Kit. Figure B-1 illustrates the setup.

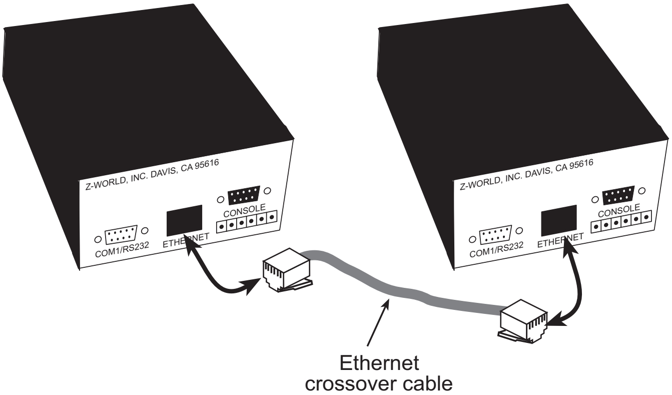


Figure B-1. Test Setup Using Crossover Cable

4. Plug in the AC adapters to supply power to the EM1000s.
6. At the **`B:\>`** prompt, type **`listen.bat <return>`**.

Tip Type **`exit <return>`** at the **->** prompt to get the **`B:\>`** prompt.

6. Press the **RESET** button for at least 5 seconds, then release it. The following shows an example of the display parameters that should appear.

```
my_ip=10.10.10.10
gateway=10.10.10.1
netmask=255.255.255.0

baud 19200
verbose on
configmode data
rtsmode off
dsrmode off
bufferlength 200
buffertimeout 1
connect listen
```

7. At the **->** prompt, type **`ESTABLISHED`**. The following message should be displayed.

```
established data=disconnected
established state=disconnected
established telnet=disconnected
```

8. Connect the **CONSOLE** port on the other EM1000 to the serial port of a terminal or PC running a terminal emulation program using the DE9 to DE9 programming cable supplied with the Developer's Kit.

9. At the **`B:\>`** prompt, type **`active.bat <return>`**.

10. Press the **RESET** button for at least 5 seconds, then release it. The following shows an example of the display parameters that should appear.

```
my_ip=10.10.10.11
gateway=10.10.10.1
netmask=255.255.255.0

baud 19200
verbose on
configmode data
rtsmode off
dsrmode off
bufferlength 200
buffertimeout 1
connect active
```

11. At the **->** prompt, type **`ESTABLISHED`**. The following message should be displayed.

```
established data=connected
established state=connected
established telnet=disconnected
```

The message in step 11 indicate that both EM1000s and their associated software are fully functional so that data can be exchanged between their **COM1/RS-232** serial ports.

Connect via Ethernet Hub

1. Connect the power supplies to both EM1000s according to the instructions in Chapter 2, "Getting Started."
2. Connect the **CONSOLE** port on one EM1000 to the **COM2** port of a terminal or PC using the DE9 to DE9 programming cable supplied with the Developer's Kit.
3. Connect the RJ-45 Ethernet ports on both EM1000s to a port on the Ethernet hub using the Ethernet cables supplied with the Evaluation Kit. Figure B-2 illustrates the setup.

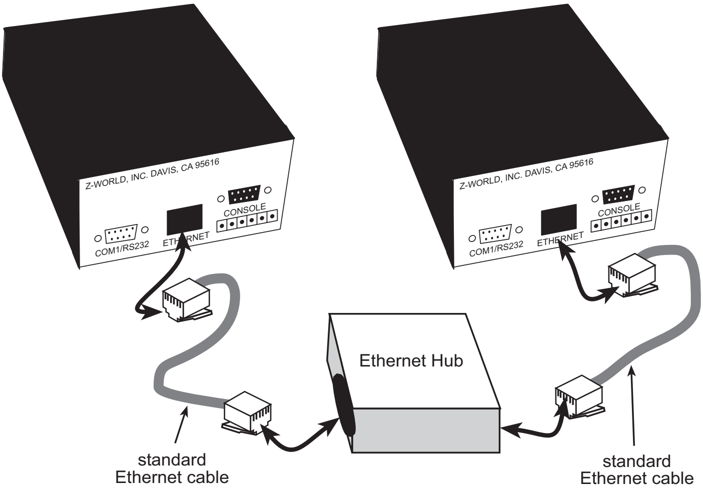


Figure B-2. Test Setup Using Ethernet Hub

4. Set the **Normal/Uplink** switch on the Ethernet hub to **Normal**.
5. Plug in the AC adapters for the EM1000s and the 5 V AC adapter for the Ethernet hub.
6. At the **`B:\>`** prompt, type **`listen.bat <return>`**.

Tip Type **`EXIT <return>`** at the **`->`** prompt to get the **`B:\>`** prompt.

7. Press the **RESET** button for at least 5 seconds, then release it. The following shows an example of the display parameters that should appear.

```
my_ip=10.10.10.10
gateway=10.10.10.1
netmask=255.255.255.0

baud 19200
verbose on
configmode data
rtsmode off
dsrmode off
bufferlength 200
buffertimeout 1
connect listen
```

8. At the **->** prompt, type **`ESTABLISHED`**. The following message should be displayed.

```
established data=disconnected
established state=disconnected
established telnet=disconnected
```

9. Connect the **CONSOLE** port on one EM1000 to the serial port of a terminal or PC running a terminal emulation program using the DE9 to DE9 programming cable supplied with the Developer's Kit.

10. At the **`B:\>`** prompt, type **`active.bat <return>`**.

11. Press the **RESET** button for at least 5 seconds, then release it. The following shows an example of the display parameters that should appear.

```
my_ip=10.10.10.11
gateway=10.10.10.1
netmask=255.255.255.0

baud 19200
verbose on
configmode data
rtsmode off
dsrmode off
bufferlength 200
buffertimeout 1
connect active
```

12. At the **->** prompt on the other EM1000, type **`ESTABLISHED`**. The following message should be displayed.

```
established data=connected
established state=connected
established telnet=disconnected
```

The message in step 12 indicate that both EM1000s and their associated software are fully functional so that data can be exchanged between their **COM1/RS-232** serial ports.

Blank

Appendix C: Handling DOS Files

Appendix C shows how to edit and install new DOS files when the EM1000 is connected to a terminal or to a PC running a terminal emulation program.

Editing Files

The EM1000 has a DOS environment, and many standard DOS commands (**`dir`**, **`copy`**, **`type`**, etc.) will work from the terminal or the PC running a terminal emulation program.

The built-in editor is similar to the DOS EDLIN editor. Invoke the editor by typing

```
edit filename <return>
```

The first 20 lines of the file will be displayed with line numbers. To delete line 3, type **`d3`**. To insert a line after line 4, type **`i4`**. To quit inserting, press **<ctrl Z>**. To redisplay the file starting at line 23, type **`L23`**. Generally, to make changes to a line it is best to insert the changed line after the line that is being replaced, then delete the old line.

Tip Remember to press **<ctrl Z>** when the insertion is finished before trying to delete the old line.

It is also possible to replace a single line by typing the line number, **`<return>`**, then entering the new line. The new line will overwrite the existing line.

When you are done, type **`s`** to save and exit, or **`q`** to quit (no changes).

Editing WATTCP.CFG

Tip Before editing the **`WATTCP.CFG`** file, which is used to configure the operating software, you may wish to try out the editor on your own test file.

Once the EM1000 is connected and powered up as explained in Chapter 2, "Getting Started," you will get a **`->`** prompt from the EM1000 software, which has started running. Type

```
exit <return>
```

to get the DOS **`B:\>`** prompt.

1. Type

   ```
   edit wattcp.cfg
   ```

   at the **`B:\>`** prompt. The display will show

```
Flashlite Line Editor v1.0


Enter h for help


->   0: my_ip=10.10.2.195
     1: gateway=10.10.2.137
     2: netmask=255.255.255.0
     3:

>>
```

2. Type

   ```
   i2 <return>
   ```

   at the **>>** prompt to insert a line after line 2.

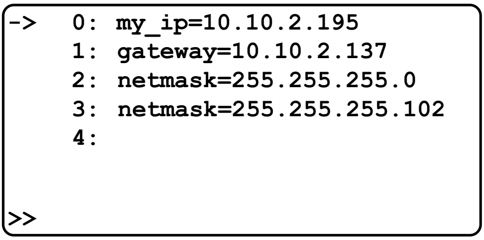
3. Type the new line, for example,

   ```
   netmask=255.255.255.102 <return>
   ```

4. Type

   <ctrl Z>

5. Type `L0` to display the file starting at line 0. The display will show



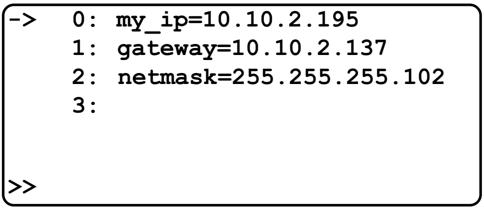
```
->   0: my_ip=10.10.2.195
     1: gateway=10.10.2.137
     2: netmask=255.255.255.0
     3: netmask=255.255.255.102
     4:


>>
```

6. Type

   ```
   d2 <return>
   ```

   at the **>>** prompt to delete line 2.

7. Type `L0` to display the file starting at line 0. The display will show

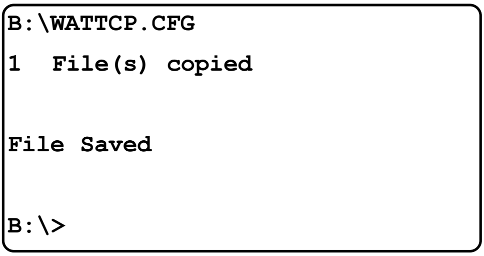


```
->   0: my_ip=10.10.2.195
     1: gateway=10.10.2.137
     2: netmask=255.255.255.102
     3:


>>
```

8. Type

   ```
   s <return>
   ```

   at the **>>** prompt to save the revised file. The display will show



```
B:\WATTCP.CFG
1  File(s) copied


File Saved


B:\>
```

Press the **RESET** button for 5 seconds, then release it, to return to the EM1000 **–>** prompt, if needed.

General Considerations

When editing a file, it is possible to put comments in the file by putting a **`#`** or **`;`** at the beginning of the line.

It is a good idea to archive the setup files (**`DWN.EM`** and **`WATTCP.CFG`**).

1. Type

   ```
   down
   ```

   at the **`B:\>`** prompt.

2. Send the files via Xmodem on the terminal emulation program to the attached PC. Since each EM1000 will have different configuration files, it is a good idea to archive the files separately in a secure location.

XDOS Command Reference

When the EM1000 application is not being run, the EM1000 is in DOS mode. The EM1000 uses XDOS, a compact operating system for embedded applications. The XDOS command structure is nearly identical to MS/ PC DOS version 3.3. The switches for the DIR command have been changed and expanded. XDOS does not support redirected input or output with the use of "<" and ">", but does support pipes (|). None of the external DOS commands are provided due to storage constraints. XDOS does not support installable file system functions.

XDOS commands are followed by a function description and their syntax, including available parameters and switches. Items in boldface type must be entered. Capitals or lowercase letters may be used. Items in italics are parameters. Those in boldface italics must be entered, those in [ ] are optional. All switches are optional. They are shown as `[`**`/x`**`]`. Spaces and punctuation are to be included. An ellipsis ... following items means that you may repeat the items as often as needed. Do not enter the ellipsis or the square brackets. Most XDOS commands allow the use of wildcards in filenames and extensions. When wildcards (? = one character, * = any character or characters) are used, the command is executed once for each matching file.

Nomenclature

`[`*`D:`*`]`—drive specification: a letter followed by a colon (:), e.g. , A:, if no drive is specified, the default drive is used.

`[`*`path`*`]`—the path DOS must take in traveling from one directory to another; directory names are separated by a backslash (\).

`[`*`filename`*`]`—up to 8 characters used to name a file.

`[.`*`ext`*`]`—a three-character extension may be added to a filename. An extension is separated from a filename by a period.

Commands

`cd`

`chdir`

Changes the current directory

Syntax: **`cd`** `[[`*`D:`*`]`*`path`*`]` or **`chdir`** `[[`*`D:`*`]`*`path`*`]`

copy

Copies a file, combines two or more files into one file, or transfers data between files and DOS devices.

Syntax: **copy** [*D:*][*path*]*filename*[*.ext*][*switches*]
[+[*D:*][*path*]*filename*[*.ext*][*switches*]
[*D:*][*path*][*filename*[*.ext*]][*switches*]

Switches: **/v**—verify the contents of new file
/a—copy file in ASCII format
/b—copy file in binary format

date

Displays or changes the current DOS date.

Syntax: **DATE [*mm-dd-yyyy*]**

del
erase

Deletes (erases) one or more files from a disk.

Syntax: **del** [*D:*][*path*][*filename*[*.ext*]] or
erase [*D:*][*path*][*filename*[*.ext*]]

dir

Lists directory entries.

Syntax: **dir** [*D:*][*path*][*filename*[*.ext*]][*switches*]

Switches: **/a**—display file attributes

/b—sort by file size (in bytes)

/d—sort entries by date and time

/f—display entries by alphabetic file name order

/n—display entries in directory order (do not sort)

/s—include system and hidden files in output

/h—display this Help screen (any invalid key)

md
mkdir

Creates a subdirectory.

Syntax: **md** [*D:*]*path* or **mkdir** [*D:*]*path*

path

Specifies directories that DOS is to search when trying to locate executable files.

Syntax: **path** [[*D:*]*path*[*;*[*D:*]*path ...*]]

prompt

Sets the DOS system prompt.

Synatx: **PROMPT** [*text*]

Text:	Resulting Character(s):
$t	The current time stored by DOS
$d	The current date stored by DOS
$p	The current directory
$v	The version of DOS being used
$n	The default drive
$g	The character >
$l	The character <
$b	The character |
$q	The character =
$$	The character $
$_	Carriage return plus line feed

ren

Renames a file.

Synatx: **ren** [*D:*][*path*]*filename*[*.ext*] *filename*[*.ext*]

rd
rmdir

Deletes a subdirectory.

Synatx: **rd** [*D:*]*path* or **rmdir** [*D:*]*path*

time

Displays or changes the current DOS time

Synatx: **time** [*hh:mm:ss.xx*]

type

Displays the contents of a file.

Syntax: **type** [*D:*][*path*]*filename*[*.ext*]

ver

Displays the DOS version number.

Syntax: **ver**

vol

Displays the volume label of specified drive.

Syntax: **vol [*D:*]**

Blank

APPENDIX D: INSTALLING NEW FIRMWARE

Two options are now available to install new firmware on the EM1000:

- via the XMODEM protocol when the EM1000 is connected to a terminal or to a PC running a terminal emulation program.
- via FTP over the existing Ethernet connection to the EM1000.

Via XMODEM Protocol

To install new software, have the terminal emulation program running on your PC, with the PC connected to the **CONSOLE** port of the EM1000 as explained in Chapter 2, “Getting Started.”

Before proceeding with these steps, you must have the new software on the PC that is being connected to the EM1000.

Tip Select a font such as Terminal by clicking on **Font** in the **View** menu of the PC terminal emulation program.

1. Apply power to the EM1000. At the **->** prompt type

   ```
   exit <return>.
   ```

2. The EM1000 will display the DOS prompt **B:\>**.
3. Save the bootstrap files by typing

   ```
   copy up.com b:\bin\up.com <return>.
   ```

4. If you want to save your current configuration files, type

   ```
   copy dwn.em b:\bin\dwn.em <return>
   copy wattcp.cfg b:\bin\wattcp.cfg <return>.
   ```

5. Type

   ```
   del *.*
   ```

   and answer **Y <return>**.
6. Type

   ```
   up filename.exe <return>.
   ```

   and move on to start Step 7 within 10 to 15 seconds to avoid a timeout.

Tip If you obtained the upgrade file from Z-World’s Web site (www.zworld.com), check the documentation file and use the file name and extension in the Web documentation.

7. Click on **TRANSFER** and **SEND FILE** in the terminal emulation program. Select the file name from the directory where you stored the upgrade when you downloaded it. Also, select the XMODEM protocol. Then click on **SEND**. There will be a **B:>** prompt after the upload is finished.

If you .get a timeout message, repeat Steps 6 and 7.

8. Type

   ```
   filename <return>
   ```

   Files will unzip into the **`B:`** directory.

9. Type

   ```
   dir
   ```

   at the **`B:\>`** prompt. If you see a **`README.TXT`** file, read it before proceeding.

10. If Step 8 was completed as described above, type

    ```
    del filename.exe <return>
    ```

11. To recover the files saved in Step 3, type

    ```
    copy b:\bin\up.com b:\up.com <return>.
    ```

12. To recover the files saved in Step 4, type

    ```
    copy b:\bin\dwn.em b:\dwn.em <return>
    copy b:\bin\wattcp.cfg b:\wattcp.cfg <return>..
    ```

If directed by the **`README.TXT`** file, edit the **`WATTCP.CFG`** file and the **`DWN.EM`** files.

The new version of the software is now installed. Press the **RESET** button for 5 seconds and then release it to return to the EM1000 **`->`** prompt.

Via FTP

Future firmware updates are possible via FTP over the existing Ethernet connection to the EM1000 once the FTPD command and firmware dated on or after August 28, 2000, has been installed.

Tip

To check the version of firmware on your EM1000, connect a PC or terminal to the **CONSOLE** port of the EM1000 as explained in Chapter 2, "Getting Started." Type **`version`** at the EM1000 **`->`** prompt. The PC will display the firmware version.

Install FTPD

The **`FTPD.EXE`** file is available on the Z-World Web site (www.zworld.com) along with the EM1000 documentation. Install the **`FTPD.EXE`** file using the XMODEM protocol described above. **`FTPD.EXE`** and the firmware dated on or after August 28, 2000, have to be installed initially using the XMODEM protocol.

Install Future Firmware Releases

Follow these steps to upload a future firmware release using FTP.

1. Connect a PC or terminal to the **CONSOLE** port of the EM1000 as explained in Chapter 2, "Getting Started." Have the terminal emulation program running on the PC.

Tip Select a font such as Terminal by clicking on **Font** in the **View** menu of the PC terminal emulation program.

2. If you get the **->** prompt, type

   ```
   exit <return>
   ```

   The EM1000 will display the DOS prompt **B:\>**.

3. Save the bootstrap files by typing

   ```
   copy up.com b:\bin\up.com <return>.
   ```

4. If you want to save your current configuration files, type

   ```
   copy dwn.em b:\bin\dwn.em <return>

   copy wattcp.cfg b:\bin\wattcp.cfg <return>.
   ```

5. At the DOS command prompt, type

   ```
   FTPD <return>
   ```

   to start the FTP server. The EM1000 will ***not*** provide a visible response to the command.

6. Start an FTP client on the host computer that has the new firmware to be transferred to the EM1000.

   If you are using a Unix machine, enter

   ```
   ftp my_ip <return>
   ```

   If you are using a Web browser, enter the IP address in the location/ address box as specified by the **my_ip** parameter in the target EM1000. If prompted for a userid and password, just press enter for both (i.e., there is no userid or password).

7. Send the firmware self-extracting archive to the EM1000, naming it ***filename*.EXE**.

   For example, the appropriate Unix commands would look like this:

   ```
   % ftp hostip
   > bin  (set binary mode transmission—most important!)
   > put new_release.exe filename.EXE
          (send, giving remote file name)
   ```

8. Quit the FTP client once the transfer is complete. The transfer should take less than 20 seconds.

> At this point, **FTPD** is still running on the EM1000, and another transfer session could be established if desired.

9. Reset the EM1000 by presssing the **RESET** button for 5 seconds once all the new firmware is uploaded.

10. At the **->** prompt, type

    ```
    exit <return>.
    ```

    The EM1000 will display the DOS prompt **B:\>**.

11. Type

    ```
    filename <return>
    ```

    where ***filename*** is the name given to the self-extracting archive in Step 7. You will be prompted for a **Y/N** response for each file to be overwritten from the self-extracting archive.

Tip If you are sure that you wish all the files to be overwritten, type ***filename*** **-o <return>**.

12. To recover the files saved in Step 3, type

    ```
    copy b:\bin\up.com b:\up.com <return>.
    ```

    To recover the files saved in Step 4, type

    ```
    copy b:\bin\dwn.em b:\dwn.em <return>
    copy b:\bin\wattcp.cfg b:\wattcp.cfg <return>.
    ```

If directed by the **README.TXT** file, edit the **WATTCP.CFG** file and the **DWN.EM** files.

The new version of the firmware is now installed. Press the **RESET** button for 5 seconds and then release it to return to the EM1000 –> prompt.

The **FTPD** firmware can also be used to list directory contents and delete files. Note that not all the FTP functions are implemented in the **FTPD** firmware in order to minimize flash memory use.

Blank

APPENDIX E: TELNET

Windows is shipped with a Telnet package that allows you to access and reconfigure the EM1000 from a remote location. Telnet packages are also available with other operating systems.

1. From the Telnet window, click on **OPEN**, then select **REMOTE SYSTEM**.
2. Type the IP address of the destination EM1000 in the **HOSTNAME** box. Click on **TELNET** port and **terminal type = VT100**. Click on the **CONNECT** button.
3. You will see a short message and a request to enter your password if the EM1000 was configured to require a password. Enter your password. Telnet should display a version number message followed by the EM1000 **–>** prompt.

The EM1000 may now be reconfigured in the same way as when a terminal or PC is connected directly to the **CONSOLE** port.

The **`exit`** command is not available in a Telnet session. Although Telnet sessions with an EM1000 using firmware dated before March 27, 2000 were not able to handle **`gateway`**, **`my_ip`**, and **`netmask`**, care must still be exercised when using these commands with later versions of the firmware since the Telnet session will be lost when changes are made. It will be necessary to re-establish a new Telnet session to the new IP address. If you need to power up the EM1000 with the new values, re-establish a new Telnet session, then type

```
show > DWN.EM <return>
```

at the EM1000 -> prompt. This allows the current configuration settings to override the prior **`DWN.EM`** file contents.

Tip

A Telnet password is recommended to prevent unauthorized reconfiguration of the EM1000.

Sample Telnet Session

Tip

Before starting, you may need to get the IP address from your network administrator. The **`WATTCP.CFG`** file will have to be reconfigured as described in Appendix C, “Handling DOS Files,” to establish the EM1000’s IP address.

1. Connect the EM1000 to a terminal or a PC running a terminal emulation program as explained in Chapter 2, “Getting Started,” and apply power.
2. Type

   ```
   exit <return>
   ```

   to get the **`B:\>`** prompt.
3. Type

   ```
   copy listen.em dwn.em <return>
   ```
4. Push the **RESET** button for 5 seconds, then release it to return to the EM1000 **->** prompt.
5. Connect the EM1000 to an existing Ethernet network using a standard Ethernet patch cable connected to the RJ-45 **ETHERNET** port.
6. Open a Telnet connection on a PC that has Ethernet access. Use the IP address for the Telnet **HOSTNAME**. Select the TELNET port, and VT100 emulation. Click on the **CONNECT** button
7. Telnet should display a version number message followed by the EM1000 **->** prompt. Type **`help`** to see a list of the available commands. This confirms that the EM1000 is connected to the Ethernet, and is at the correct IP address. Exit the Telnet session.

Blank

APPENDIX F: SENDING E-MAIL

The EM1000 can be used to send outgoing e-mail messages when a properly formatted message is sent to it from the attached RS-232 device.

In order to do this, the attached RS-232 device must be able to reconfigure the EM1000 during operation. Generally this is done using a second serial port, by manipulating the DSR signal on the **COM1/RS-232** port, or by manipulating the CTL line.

Refer to Chapter 3, “Configuration Reference,” for more details on reconfiguring the EM1000 during its operation.

The general procedure for sending an e-mail message is as follows. Specific details may be different for your network and your mail server.

1. With the EM1000 in configuration mode, set the **`remotename`** parameter set to that of the e-mail server.
2. Set the **`remotedataport`** parameter to 25.
3. Type **`open`** to open a connection to the e-mail (SMTP) server.
4. A sequence of properly formatted strings must be sent to the server, for example, from an attached Z-World controller.

Tip Use **`spy=enabled`** to watch the data flow to and from the Ethernet modem.

5. Close the connection to the e-mail server.

Refer to a text on TCP/IP or SMTP mail servers for more details on mail servers.

Some e-mail servers do not allow e-mail forwarding.

Tip Be sure to work with your local network systems administrator when developing and testing any e-mail or network application.

The Z-World Web site at http://www.zworld.com provides a sample e-mail program.

APPENDIX G: DATA AND SIGNAL FLOW

The data and signal flow described in this appendix can only be monitored when the **spy** mode on the EM1000 is enabled.

Figure G-1 shows the flow of data and control signals through the EM1000.

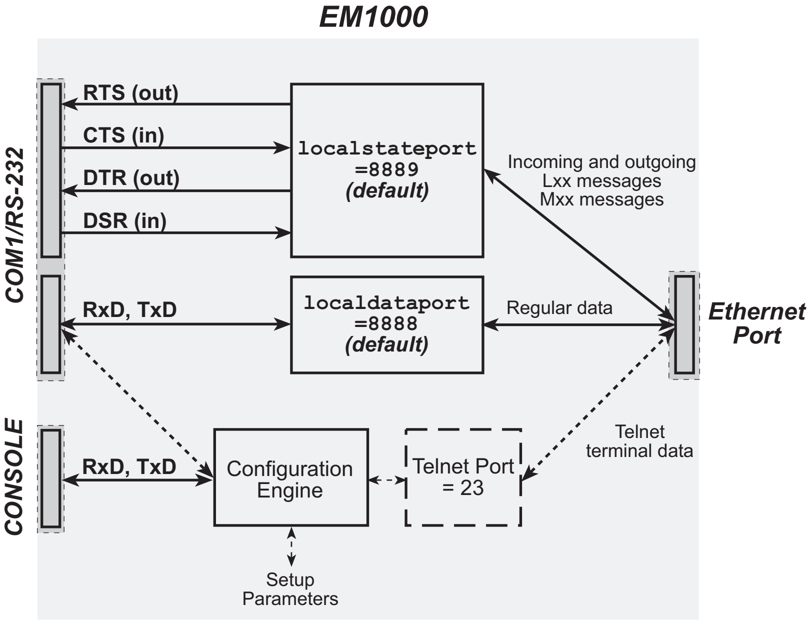
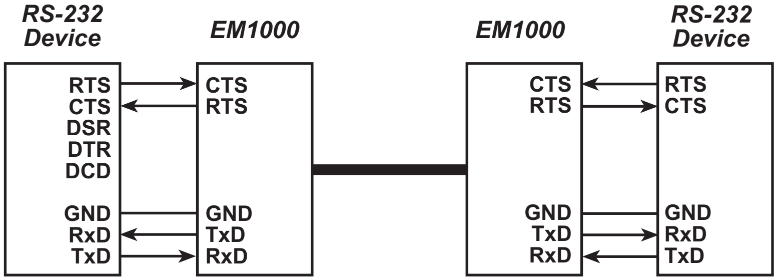


Figure G-1. Schematic Representation of Data and Control Signal Flow (`connect` command configured for `demand` mode)

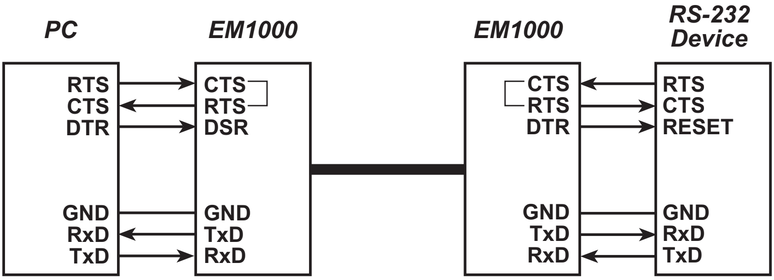
The **`ETHERCOM.EXE`** application transfers raw binary ASCII data over the (RxD, TxD) data connection through the **COM1/RS-232** serial port and the data port, port 8888 (default), inside the EM1000. The data connection can be used to connect an EM1000 to other devices, such as SMTP and Telnet devices. There is no structure to the data transferred through the data connection. Data arriving at the input of the **COM1/RS-232** serial port are transferred through the data port connection to the Ethernet port, while data from the Ethernet port are transferred to the output of the **COM1/RS-232** serial port. The fact that the data transferred via the data port are unstructured is what allows the device connected to the **COM1/RS-232** serial port of the EM1000 to communicate with virtually any TCP/IP device. The **`ETHERCOM.EXE`** application transfers the data to/from the **COM1/RS-232** serial port without translation (with the single exception of sentinel character settings).

Control signals, which affect whether the EM1000 is in the configuration mode or the data throughput mode, pass through the **COM1/RS-232** port.

Figure G-2 shows the external connections with the EM1000.



(a) `rtsmode=flow` *(traditional RTS/CTS configuration)*



(b) `rtsmode=flow`*,* `dsrmode=passthrough`

Figure G-2. External Data and Control Signal Connections

There is additional information, which cannot be conveyed via the raw data connection through the data port, that an EM1000 can share and coordinate with another EM1000. The information is conveyed to the remote EM1000 via the state connection (port 8889 by default). Unlike the data connection, the state connection transfers structured information in the form of three-byte packets. The information the state connection carries between two EM1000s pertains to the state of the UART's line status register and modem status registers. The line status register contains information about break conditions, and the modem status register contains information about CTS and DSR signals.

The **`ETHERCOM.EXE`** application detects a break condition on the input of its local **COM1/RS-232** serial port when the UART sets the break interrupt bit (bit 4) in the line status register (register 5). The **`ETHERCOM.EXE`** application can create a break condition on the output of its local **COM1/ RS-232** serial port by setting the set break bit (bit 6) in the UART's line control register (register 3).

A break condition detected on the input of **COM1/RS-232** serial port will always cause the **`ETHERCOM.EXE`** application to send a notification to the remote EM1000. Consequently, the EM1000 that receives a notification that a break condition has occurred on the remote link will always generate a break condition on its local **COM1/RS-232** serial port.

Line status changes are transferred via the state connection with the syntax L##, where the L is the literal character "L" and the ## is the hex ASCII representation of the new line status register. The only bit that is important is the break interrupt bit (bit 4). If the break interrupt bit is set, the **`ETHERCOM.EXE`** application that originated the state connection message detected a break interrupt on its local **COM1/RS-232** serial port. The **`ETHERCOM.EXE`** application that receives the L## message will generate a break condition on the output of its local **COM1/RS-232** serial port.

The modem status changes require proper configuration of the **`dsrmode`** and **`rtsmode`**. If the EM1000 is configured for **`dsrmode=passthrough`**, the DSR/DTR information is conveyed through the state connection. If the EM1000 is configured for **`rtsmode=passthrough`**, the RTS/CTS information is conveyed through the state connection. The **`dsrmode`** and the **`rtsmode`** operate independently, and either one independently or both can be in the data throughput mode.

When the EM1000 is configured for **`dsrmode=passthrough`**, then any change in the DTR input to the EM1000's local **COM1/RS-232** serial port will cause a modem status message to be sent to the remote EM1000 via the state connection. Consequently, any modem status message received via the state connection that indicates that the DTR signal at the remote EM1000 has changed state will cause a corresponding change at the DSR output of the local **COM1/RS-232** serial port. For end-to-end operation, both EM1000s must be in the **`dsrmode=passthrough`** mode.

When the EM1000 is configured for **`rtsmode=passthrough`**, then any change in the CTS input to the EM1000's local **COM1/RS-232** serial port will cause a modem status message to be sent to the remote EM1000 via the state connection. Consequently, any modem status message received via the state connection that indicates that the CTS signal at the remote EM1000 has changed state will cause a corresponding change at the output of the local **COM1/RS-232** serial port. For end-to-end operation, both EM1000s must be in the **`rtsmode=passthrough`** mode.

Modem status changes are transferred via the state connection with the syntax M##, where the M is the literal character "M" and the ## is the hex ASCII representation of the new modem status register.

Table G-1 provides an overview of the control registers associated with the Line Status Register and the Modem Status Register.

Table G-1. Control Register Overview

Register	Register Address (A2–A0) & DLAB	D0	D1	D2	D3	D4	D5	D6	D7
Line Status Register (LSR)	5	Data Ready (DR)	Overrun Error (OE)	Parity Error (PE)	Framing Error (FE)	Break Interrupt (BI)	Transmitter Holding Register (THRE)	Transmitter Empty (TEMT)	Error in Receiver FIFO
Modem Status Register (MSR)	6	Delta Clear to Send (DCTS)	Delta Data Set Ready (DDSR)	Trailing Edge Ring Indicator (TERI)	Delta Data Carrier Detect (DDCD)	Clear to Send (CTS)	Data Set Ready (DSR)	Ring Indicator (RI)	Data Carrier Detect (DCD)

RTS/CTS information is conveyed by two bits. The delta clear to send bit (bit 0) indicates that the modem status message contains information about the remote EM1000's CTS signal, and the new state is indicated by the clear to send bit (bit 4).

DSR/DTR information is conveyed by another two bits. The delta data set ready bit (bit 1) indicates that the modem status message contains information about the remote EM1000's DSR signal, and the new state is indicated by the data set ready bit (bit 5).

Appendix H: Ethernet Networks

Wiring an Ethernet Network

One of the easiest to use and least expensive Ethernet wiring technologies today is 10BaseT. 10BaseT hubs and interconnect wiring are readily available, reliable, and inexpensive compared to most other networks. The EM1000 uses 10BaseT technology and interconnects.

If possible, the "star" wiring configuration shown in Figure F-1 is preferred. In this configuration, devices are all tied to hubs, and the hubs are then tied together. The connected devices can be up to 100 m from the hubs, and the hubs can be up to 100 m apart.

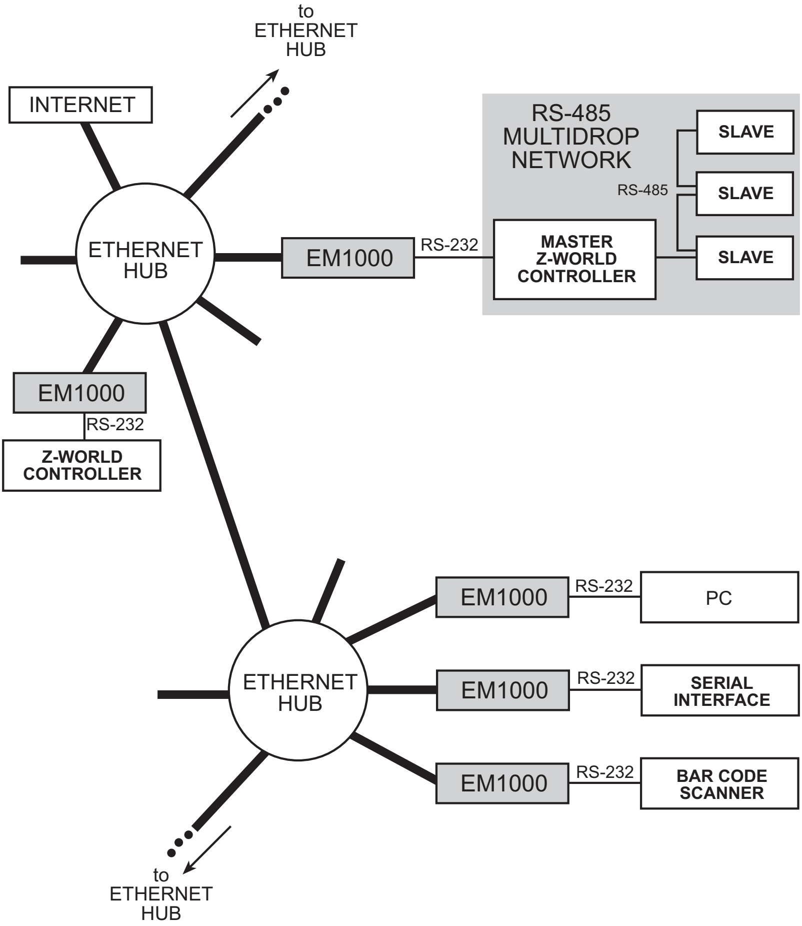


Figure H-1. Star Ethernet Configuration

It is possible to "daisy chain" connections to Ethernet devices, but the hardware will not operate as reliably as with the hub method, and so the "daisy chain" is not recommended. Also, the time for a transmitted packet to be received is not predictable on heavily loaded daisy chained buses.

Although Category 3 cable can be used for 10BaseT, the "Category 5" cable is recommended for a new installation.

TCP/IP Ports

Port numbers are divided into two categories—"well-known ports" and "not-so-well-known ports." Well-known ports are used to implement well-known protocols such as Telnet, SMTP, HTTP, SNMP, and SYSLOG. Most of the well-known ports are assigned a number less that 1024. For general-purpose "end-to-end" data transmission, the EM1000 should use ports 1024 and above that do not conflict with the ports listed in Table F-1. The only restriction on the ports is that the connecting parties must know what port to use for the connection, and two applications cannot use the same port at the same time. The address of any particular modem consists of the TCP/IP address plus a local port number. To reach that modem, you need to send to set your **`target_name`** and **`target_dataport`** to match the remote EM1000's **`my_ip`** and **`local_dataport`**.

Table H-1. Reserved TCP Port Numbers

Port Number	Keyword	Description
0		Reserved
1–4		Unassigned
5	RJE	Remote job entry
7	ECHO	Echo
9	DISCARD	Discard
11	USERS	Active users
13	DAYTIME	Daytime
15	NETSTAT	Who is up or NETSTAT
17	QUOTE	Quote of the day
19	CHARGEN	Character generator
20	FTP-DATA	File Transfer Protocol (data)
21	FTP	File Transfer Protocol
23	TELNET	Terminal connection
25	SMTP	Simple Mail Transfer Protocol

continued…

Table H-1. Reserved TCP Port Numbers (continued)

Port Number	Keyword	Description
37	TIME	Time of day
39	RLP	Resource Location Protocol
42	NAMESERVER	Host name server
43	NICNAME	Who is?
53	DOMAIN	Domain name server
57	MTP	Deprecated
67	BOOTPS	Bootstrap protocol server
68	BOOTPC	Bootstrap protocol client
69	TFTP	Trivial File Transfer Protocol
75		Any private dial-out service
77		Any private RJE service
79	FINGER	Finger
80	HTTP	Hypertext Transfer Protocol
87	LINK	TTY link
95	SUPDUP	SUPDUP Protocol
101	HOSTNAME	NIC host name server
102	ISO-TSAP	ISO-TSAP
103	X400	ISO Mail
105	CSNET-NS	CSNET-NS
108	POP	Post office
109	POP2	Post office
110	POP3	Post office
111	PORTMAP	Port map
113	AUTH	Authentication service
115	SFTP	Secure File Transfer Protocol
117	UUCP-PATH	UUCP path service
119	NNTP	Network News Transfer Protocol
123	NTP	Network Time Protocol
139	NetBios	Session service
144	NEWS	News

continued…

Table H-1. Reserved TCP Port Numbers (continued)

Port Number	Keyword	Description
153	SGMP	Secure Internet Group Management Protocol
158	TCPREPO	PC Mail
160–223		Reserved
170	PRINT-SRV	Network postscript
175	VMNET	VM network
400	VMNET0	VM network
512	EXEC	
513	LOGIN	
514	SHELL	
515	PRINTER	Line printer spooler
520	EFS	LucasFilm
526	TEMPO	New date
530	COURIER	RPC
531	CONFERENCE	Chat
532	NETNEWS	Read news
540	UUCP	UUCP daemon
543	KLOGIN	Kerberos authenticated rlogin
544	KSHELL	Kerberos remote shell
556	REMOTEFS	RFS server—Brunhoff remote file system
600	GARCON	
601	MAITRD	
602	BUSBOY	
750	KERBEROS	Kerberos authentication—TCP
751	KERBEROS_MASTER	Kerberos authentication
754	KRB_PROP	Kerberos slave propagation
888	ERLOGIN	Login and environment passing
1109	KPOP	Pop with Kerberos
1524	INGRELOCK	

continued…

Table H-1. Reserved TCP Port Numbers (concluded)

Port Number	Keyword	Description
2053	KNETD	Kerberos demultiplexor
2105	EKLOGIN	Kerberos encrypted rlogin
5555	RMT	
5556	MTB	MTB backup
9535	MAN	Remote manual server
9536	W	
9537	MANTST	Remote manual server, testing
10000	BNEWS	
10001	QUEUE	
10002	POKER	
10003	GATEWAY	
10004	REMP	
10012	QMASTER	

Industrial Network Concerns

Questions sometimes arise about the use of networks in industrial applications. There are at least two main factors to be considered.

1. Network reliability and the effect on a “remote” process if the network cable is damaged (or the stops working for some other reason).
2. “Time-critical” data.

Because of the network cable could be damaged catastrophically, many companies elect to have local control, or at a minimum local “backup” control at the remote location. In this situation, a local controller is in charge of the process, but it reports its status to the master, and receives high-level directives from the master.   The local controller makes decisions such as “if input57 is on, and analoginput12 voltage is between 3 V and 4 V, then activate output39”. Then as a background task, it reports to the master that so far today, 3425 bottles have been produced.   And perhaps a few times per day, the master tells the controller “Stop making bottles, and start making jars,” or vice versa. The essence of the local control concept is that if the network fails, the local controller will allow production to continue for some period of time while the network is down.

The second issue, time-critical data, arises when the network is used to "close the control loop." For example, if the network master device was connected directly to inputs, an A/D converter, and to an output through the network, then it would be responsible for making the decision "if input57 is on, and analoginput12 voltage is between 3 V and 4 V, then activate output39." Not only is this scenario subject to network failure, but there is the concern about the master being busy handling other demands, and not getting around to looking at input57 until it has gone on (but then gone back off?) Answer: input39 does not get turned on, and something bad happens (or something doesn't happen which should have happened). With the control loop closed locally with an embedded controller, time-critical data are handled in a time-critical fashion that is guaranteed not to miss any key events (in this case, input57 closing).

Again, because of the concern about missing key data to make control decisions, many companies elect to have local embedded control, which can be time responsive to the local process, and passes only higher level commands and status through the network. A Z-World controller connected to the Ethernet with the EM1000 is ideal for this purpose.

In the past, the issue of "determinism" has been raised when discussing the Ethernet as compared to some other buses. The use of star configurations based on switched hubs instead of daisy-chained configurations leads to deterministic operation, and fewer headaches, although at minor additional expense.

Blank

APPENDIX I: OPERATING TIPS

Lock-Up with High Data Rates or Noisy RS-232 Lines

The original firmware release was prone to locking up, especially at 19,200 bps. This was caused by internal buffer overflows. Although this problem has been rectified, the following precautions should be taken:

1. Use hardware flow control (CTS/RTS) if possible.

   Since there is a finite amount of buffer space in the EM1000, it cannot be used as a speed-matching buffer indefinitely without some form of pacing or flow control.

2. If 38,400 bps is used, then the client devices should implement end-to-end error detection since the occasional character (0.05%) may be dropped.

   Using hardware flow control can alleviate this problem, but not eliminate it entirely. The client devices work best with moderately short messages (up to about 200 bytes) before requiring acknowledgment from the peer device. At 19,200 bps, characters will not be dropped unless internal buffering is exhausted because of the peer not accepting the data fast enough.

3. If hardware flow control is not available, ensure that there is some form of end-to-end flow control between the two client devices, for example, XON/XOFF software flow control.

   This is especially important if the two devices have different data rates, but also applies if the same nominal speed setting is used since there may be slight differences in crystal frequency between the originating device and the receiving EM1000. For example, if the originating device actually runs at 19,200.1 bps, and the receiving EM1000 is running at exactly 19,200 bps, then the net accumulation is 0.1 bps. Extended over one day (86,400 seconds), this would amount to a total of 864 bytes, which would have to be buffered by the EM1000s. In the longer timeframe, this would overflow the internal buffering.

4. The modem control lines (CTS, DSR, and DCD) that are inputs to the EM1000 should be properly controlled, even if not in use. They should be tied to fixed RS-232 levels, or to other lines that do not vary rapidly.

5. Use the correct “baud” setting. Incompatible settings (such as feeding 2400 bps to a port configured for 9600 bps) may result in what appears to be a large number of line breaks. This may overload the EM1000’s processor, which will result in dropped data.

If these guidelines are not followed, the EM1000 will still function, but with reduced performance and increased probability of lock-up. If the modem control lines pick up electrical noise, then it is possible for the EM1000 to spend excessive time trying to follow the variations, which may cause it to lose data or lock up.

Similarly, if the receiving device is accepting data at a rate that is slower than the sending device (and no flow control is used), both EM1000s will eventually run out of buffer space and start dropping data.

As a guide, two EM1000s can buffer approximately 4000 data bytes between them, in each direction, before data starts being discarded. This is because the EM1000 is designed for data transport not data buffering.

If both data transfer directions are being used simultaneously (full duplex), then the sustained data throughput rate is approximately 22,000 bps in each direction. This limits the practical serial port speed to 19,200 bps for this sort of application. If the data arrive in bursts, then 38,400 bps can be used, if the slight data loss can be tolerated. The maximum throughput can only be obtained when the EM1000s are not otherwise loaded by spy operations, heavy Telnet use, rapidly varying DTR or RTS pass-through, or verbose mode.

In summary, the EM1000 will drop characters under stress conditions rather than lock up; however lock-up is still a possibility under extreme conditions.

If flow control cannot be used, and data are streamed primarily in one direction, then the following techniques may alleviate the problem of gradual buffer overflow. It is preferable to use stop bits, but the second option below may be tried as a last resort.

1. Set the originating device so that it sends two stop bits. The receiving EM1000 (and client device) should be configured to send and expect one stop bit.
2. Set the receiving EM1000 to have a data rate slightly higher than nominal. A rate of 1–2% higher than nominal should not cause any problems for asynchronous RS-232, but will allow data to “sink” at a rate that exceeds the rate at which it is “sourced.” This may not be possible at the higher speeds because of hardware limitations. For example, the next higher speed above 19,200 bps is 23,040 bps, which would be too much of a mismatch. This technique is only applicable to speeds below 4800 bps. When determining a higher speed, remember that the speed must equate to 115,200 bps divided by an integer: 4800 bps is 115,200 bps/24, therefore the next higher speed above 4800 bps is 115,200 bps/23, or about 5009 bps.

RTS/DTR Pass-Through Mode and Line Breaks

The latest firmware sends modem and line status signals at a lower priority than the data. Not all transitions will be sent. The original firmware would attempt to send all transitions. This was not satisfactory since the modem control signals could toggle faster than the nominal data rate, which would cause data to be lost.

Now, transitions are only sent when sufficient "bandwidth" is available. Not all transitions are sent. For example, if DSR is dropped then raised within the space of a few data bytes, the corresponding transitions may not be communicated to the peer.

Notwithstanding this, it is still important to prevent the modem control lines from varying rapidly (certainly not more than once per data byte) to avoid overloading the EM1000's processor.

Line breaks, if detected, are transmitted to the peer device at a maximum rate of once every 600 ms. This is because the generated break condition lasts for 500 ms. The 100 ms difference is provided for latency and other factors. The device that originates the break condition may not follow the RS-232 standard and generate a short break (less than 250 ms). In this case, if the device starts transmitting data immediately after the break, then some of this data may be dropped since the peer EM1000 always reproduces the break condition as a 500 ms break. Depending on the buffer capacity, the data following the break may or may not get transmitted.

Troubleshooting Hints

- If you are not using the latest firmware, then upgrade. The software is available on the Z-World web site. You can determine the version of your software in the EM1000 by typing **`version`** at the -> prompt. The EM1000 firmware can be easily updated by downloading the new firmware via the **CONSOLE** port. Refer to Appendix D, “Installing New Firmware,” for more instructions.
- A “continuous” flow of data from the serial device into the EM1000 will not appear at the other end of the link as a continuous flow, especially on a heavily loaded bus. This is due to the way data are packetized for transmission over the Ethernet.
- When using Hyperterm, if you change the baud rate or any other settings, be sure to **Save** the settings, **Exit** Hyperterm, and then start Hyperterm up again using the new settings. Otherwise, you may send data at the wrong baud rate to the EM1000 and lock up the unit.
- The easiest “packetization” scheme to understand is the model where the serial device sends a packet of X characters, followed by a time delay. To use this mechanism, be sure that **`bufferlength`** is set higher than the length of the packet your device is sending, and then be sure that your serial device has a timing gap greater than the value in **`buffertimeout`** (where 1 = 0–55 ms, 2 = 55–110 ms, etc.).
- Use hardware flow control if your serial device supports it. Activating hardware flow control allows the EM1000 to halt data flow from the RS-232 device temporarily (perhaps because the EM1000 has not been able to establish a link to the remote device, or because of heavy Ethernet traffic). Ethernet networks can be heavily loaded, which means that data that your serial devices sends through the EM1000 may not get to the other end immediately. If your serial device continues to send data to the EM1000 even though the last packet has not been sent yet from the EM1000 over the Ethernet, then it may overflow the serial receive buffer, and you will lose data or perhaps cause a lock-up. The serial RS-232 buffer is 512 bytes long. In addition to the hardware signals being hooked up, To use flow control, you need to hook up the hardware RTS/CTS signals and remember to set **`rtsmode=flow`** in the **`DWN.EM`** file.
- Depending on the type of flow control you are using, be sure that you do not accidentally have RTS or DTR (both are outputs) on the EM1000 tied to an RS-232 output on the attached device. Be sure that you do not have CTS, DSR, or DCD (all inputs) tied to an RS-232 input on the attached device. Failure to do so will probably result in EM1000 lock-up when the attached device is power cycled.

- If you are not using flow control, be sure that the CTS is tied to RTS (both on the EM1000 **COM1/RS-232** port), and be sure that DTR is tied to both DSR and DCD (all three signals on the EM1000 **COM1/ RS-232** port). Failure to do so may result in EM1000 lock-up over some period of time. If you suspect that this is happening, you can put the **CONSOLE** port or the Telnet port into **`spy`** mode. If you see a bunch of M## characters intermixed with the data, or after the data have stopped, then you may have crosstalk, a short circuit between connector pins, or a misconnected wire.
- If you are not using hardware flow control, then make sure you don't send too much data, or send data so fast that you overflow the **COM1/ RS-232** receive input buffer. Failure to do this will result in lost characters, and possibly lock-up. If you still encounter problems, try dropping the baud rates of both the attached device and the EM1000.
- If you cannot use flow control, then consider having an end-to-end acknowledgment of some sort. Your serial device would send a packet of information to the EM1000, followed by a time gap. When **`buffertimeout`** expires, the EM1000 tries to send the data. When the data eventually reach the other end ("eventually" because Ethernet timing is somewhat unpredictable), then the receiving device sends a short message back to the transmitting serial device saying that it got the last packet, and it's OK to for the serial device to send the next packet.
- If you cannot use flow control or have end-to-end acknowledgments, and you have a heavily loaded Ethernet network, then you may need to lower the baud rates, and/or increase the dead time between your serial device transmissions.
- Make sure the baud rates, parity, and stop bit settings of the attached device and the EM1000 are the same. Failure to do so will result in short-term lock-up of the unit.
- Do ***not*** twist the wire pair used to connect the RS-232 signals to the EM1000. This leads to crosstalk and to possible lock-ups. The Ethernet category 5 cable is stranded and twisted, and although the RS-232 wire may be stranded, the wire pair must ***not*** be twisted.
- If the unit locks up, try resetting the unit by Telnetting to it and "reassigning" it the same address it already has. If this doesn't work, you may need to reset the unit manually.
- If your unit locks up occasionally, try tweaking the baud rates—e.g., if your attached device is set to 9600, then try **`baud=9605`** or **`baud=9595`** and see if this fixes the problem. This should be a last resort.

- When **`buffertimeout=0`** or **`buffertimeout`** is a large value, and you are using **`bufferlength`** to initiate packetization and sending of the data, be aware that the EM1000 may "end up" with some data stored in its internal buffer, waiting for additional data to come into its serial port (or waiting for the long timeout to occur). The data are not lost, they are just waiting for the EM1000 buffer to be triggered to send it.
- If you occasionally get lock-ups, and none of the above remedies work, try to correlate the lockup with another event. For example, if the Ethernet cable is routed near an arc welder, which is only used occasionally, this may cause difficulties occasionally. Or perhaps the power supply to the EM1000 is shared with another device, and when that device glitches, it transmits the glitch to the EM1000.
- Ground the EM1000 enclosure to a reliable earth ground. This will prevent noise from nearby devices from radiating into the EM1000 and potentially causing problems.
- When you are done debugging, turn off the **`spy`** and also the **`verbose`** modes if you are not using the responses in an automated system. Note that there are three **`spy`** settings and three **`verbose`** settings, one each for the **COM1/RS-232** port, the **CONSOLE** port, and the Telnet port. The settings for each port can only be changed ***from that port***.

Blank

APPENDIX J:
FREQUENTLY ASKED QUESTIONS

Q. Is any programming required to use the Ethernet Modem?

A. No programming is required. The EM1000 is normally configured using a terminal connected temporarily to the **CONSOLE** port. In those cases where the EM1000 is a “master” in a point-to-multipoint arrangement, then the attached device will need to send some simple text-based commands to the EM1000. Demo programs are available to demonstrate this. Although a wide variety of configuration options is available to handle unusual cases, in most situations, simple modifications to the default parameters will suffice. This procedure takes only a minute or two.

Q. What baud rates can the EM1000 handle on the data port?

A. This depends on many factors. Generally, operation to 19,200 bps is no problem, and if flow control is used (in conjunction with the **`rtsmode=flow`** configuration string), operation at baud rates to 38,400 bps is possible.

Q. How does the EM1000 handle 10 Mbps on one side, and lower baud rates on the other side?

A. The TCP/IP protocol stack works via a handshake mechanism, where there is an acknowledgement process that prevents the transmitter from sending too much data for the receiver to process. The internal buffer is set to 2K. If more that 2K of data are received by the TCP/IP software layer, and the EtherCom application has not read the data, then no further incoming data will be acknowledged, and then the sender of the data will not be allowed to send any more. The depth of the buffering really doesn’t matter because the TCP/IP layer of software will throttle the sender of the information so that no data are lost.

Q. What is the temperature rating of the EM1000?

A. The EM1000 operating and storage temperature range is –40°C to +70°C. Some early brochures showed a more restricted temperature range, which was incorrect.

Q. Can the EM1000 be used with daisy-chain hookups (e.g., 10Base2) rather than with hubs?

A. The EM1000 is optimized for use with standard 10BaseT cables and hubs. Z-World recommends this because the resulting network is more reliable than with a daisy-chain network. If a connection is broken in a hub-based system, only one unit goes down, as opposed to the entire network going down if a daisy-chain wire is broken.

Q. Can the EM1000 be used with RS-485 devices?

A. If you have a need for the EM1000 to interface to a number of RS-485 devices, the best approach is to use a Z-World controller as a programmable RS-485 link master. The controller must have an available RS-232 port and an RS-485 port. Then program the Z-World controller to act as a master on the RS-485 network, and transfer data from the multiple RS-485 devices through the RS-232 port to the EM1000 (and onwards to its final destination). The EM1000 does not directly support RS-485.

Q. I can't use some commands from Telnet, or via the COM1/RS-232 port, or when using the SOURCE command invoked from Telnet or COM1.

A. Some of the command options available directly at the **CONSOLE** port are related only to initial setup and configuration (for example, the **`exit`** command). If you are using Telnet to a very remote location, and you were to accidentally type **`exit`**, then you lose the TCP/IP connection, the Telnet connection, and you have no way to restart the modem (except to actually GO there, and hook up to the **CONSOLE** port). In the initial release of the software, the following commands could not be executed via Telnet or the **COM1/RS-232** port: **`exit`**, **`gateway`**, **`my_ip`**, and **`netmask`**. In the 01/13/2000 version of the software, the **`hostip`** command could also be executed from the **COM1/RS-232** port. The 03/27/2000 version of the software also allows **`my_ip`**, **`netmask`**, and **`gateway`** to be changed via the Telnet port. This feature should be exercised with caution since changing these paramaters will cause the Telnet session to be lost. It will be necessary to re-establish the Telnet session to the new IP address. If you want the new parameters to be used after an EM1000 reset or power cycle, you need to issue this command after the new Telnet session is established:

```
SHOW > DWN.EM
```

Q. Why is the built-in editor so primitive? Why can't we provide a full-screen editor?

A. This is due to the fact that the only way to display information in the EM1000 is to send it to a terminal device through the serial port. There would be no efficient way to refresh the screen after each keystroke. The editor is line-based, so you basically type a command on a line, or insert a line, then the modem responds with a full line, or bunch of lines. Since the editing requirements for the product are quite slim, this approach is not hard to deal with.

Q. Can the EM1000 be used in a point-to-multipoint application?

A. Yes. The EM1000 can be made to first establish a connection to a remote address, communicate data, drop that connection, then establish a connection to a totally different remote address. To reprogram the modem on the fly for this type of operation, it is necessary that the attached device either have a second serial port, or a controllable **DTR** line from serial port connected to the **COM1/RS-232** port of the modem, or an available digital output that can connect to the **CTL** line of the modem.

Q. Can I use logic-level signals with the modem instead of RS-232 level signals?

A. Yes. Onboard jumpers allow the **COM1/RS-232** port to be set for either logic levels or RS-232 levels.

Q. I need an "IP Addressable Controller Card." Does Z-World have such a thing?

A. Yes. The combination of a Z-World controller plus the EM1000 provides this capability.

Q. Does the EM1000 support standard TCP/IP ports?

A. Yes. The EM1000 can be set up to transmit to, and receive from any TCP/IP port.

Q. I'm getting gibberish on my Hyperterminal display, and the baud rate and other settings are correct. What do I need to do?

A. Try selecting a different font in Hyperterm. On initial use, Hyperterm will sometimes not use the font that it says is actually in use, and you need to change the font in order to get it to work properly. When using Hyperterm, it may also be necessary to save baud rate changes (or even exit the program, and re-enter) after making setup changes.

Q. How many EM1000s do I need in order to play with in a realistic situation?

A. If you have an existing Ethernet installation, and your system administrator provides authorization and support, you can purchase a single unit and do some useful things with it. If you are starting a network from scratch, or if you want to check the units out prior to installing on an operating network, you need at least 2 EM1000s.

Q. Does the EM1000 have battery-backed memory?

A. No. A battery is not needed since the executable program is located in flash memory. There is no battery to become depleted ever need replacement.

Q. Can I upgrade the EM1000 software in the future?

A. Yes. As updates become available, you can download the file(s) from the Z-World web site, and then reload your EM1000 by connecting the **CONSOLE** port on the EM1000 to your PC. See Appendix D, "Installing New Firmware," for the complete details.

Q. What types of data transfers are efficient over Ethernet?

A. The Ethernet is a high-speed bus, and efficiency is not normally an issue. However, in terms of raw data transfer capability, the best scenario is to open a link, keep it open, and send large blocks of data. There is some overhead both in the modem and on TCP/IP when you open and close links, so opening and closing links a lot tends to reduce the overall throughput. Also, to send even a single byte from one end of the link to the other requires additional overhead bytes, so sending large blocks of data is more efficient than sending more numerous small blocks.

Q. Does the EM1000 have any built-in remote network management tools?

A. Yes, there are some simple tools built-into the EM1000. Using Telnet, you can determine the status of open and closed connections (IP addresses and port numbers), determine the number of packets and bytes transmitted and received, and watch the actual data flowing through the EM1000. Also, it is possible to issue most of the configuration commands that can be issued locally via Telnet.

Q. How does IP and port addressing work on the EM1000?

A. Think of the EM1000 as having two modes — a **`listen`** (open on demand) mode where the device at the other end of the link opens the link, and an **`active`** mode where the EM1000 opens the link. Quite often, the listening device is called a server, and the active device is called a client.

`my_ip` and **`hostdataport`** are EM1000 commands used when the EM1000 is in listen/demand/server mode. They establish the EM1000 network address where the EM1000s COM1 port resides. ***These commands set up a logical connection inside the EM1000 from the Ethernet to the* COM1/RS-232 *port.*** You will need to ensure that the client's "remote IP address and remote port number" correspond to the **`hostip`** and **`hostdataport`** that were programmed in the EM1000. The client device will be using its own IP address and port number, which are independent of the EM1000's **`my_ip`** or **`hostdataport`**. When a connection is initially made from a client to an EM1000 in **`listen`** mode, the EM1000 will report "Data connection established to [client's IP address and port number]." The client may display a similar message, such as "connection established to EM1000's my_ip and hostdataport."

For the EM1000, `my_ip` is set in the `WATTCP.CFG` file, and `hostdataport` is set in the `DWN.EM` file.

`remotename` and `remoteport` are used when the EM1000 is in active (client) mode, and will be opening the link. `hostdataport` has no meaning for an EM1000 in active mode. The EM1000 will open the link and attempt to exchange its COM1 data with the remote server at the address defined by `remotename + remoteport`. `remotename` must be set to the IP address of the remote server, and remoteport must be set to a server port number at which the server is listening. If successful, the EM1000 will report "Data connection established to [remotename + remoteport]."

If the remote device is an EM1000 in listen mode, then this active unit's `remotename` and `remoteport` need to correspond to the remote unit's `my_ip` and `hostdataport`.

The EM1000 has two more command parameters that are related to addressing. `localstateport` and `remotestateport` work identically to `hostdataport` and `remoteport`, except they refer to the EM1000 modem control signals (RTS, CTS, DTR, DSR) instead of the data transmit and receive signals.

When using only EM1000s in a network, you can generally leave the port addresses at the default values and not worry about it. If your EM1000s are in active mode, be sure that their `remotename` and `remoteport` parmeters correspond to the server's. If your EM1000s are in listen mode, be sure that your `my_ip` and `hostdataport` are set the same as the active (client) unit's `remotename` and `remoteport`.

Q. How do I feed data directly from the EM1000 into an application program running on the PC (such as a spreadsheet or database)?

A. There are third-party software packages, such as TCPWedge from Taltech (www.taltech.com), which provide a DDE connection between the TCP/IP port in the computer to the application program. These programs can generally be configured to get data into and out of a database or spreadsheet with no programming involved. This allows a remote EM1000 to send data to the PCs TCP/IP address and port number, and then have that data automatically imported to the application. It is even possible to have multiple instances of TCPWedge running, and have multiple EM1000s talking to the same application in your PC!

When TCPWedge is in client mode, the EM1000 must be in demand mode. If the EM1000 is in either request or active mode, `hostdataport` is already in use, and TCPWedge cannot communicate with it.

Q. I have an EM1000 connected to only 1 other PC. I can't make it communicate over the Ethernet.

A. Be sure your PC has an IP address assigned. If this is a virgin PC, try setting the IP address to 10.10.10.20, and `netmask` to 255.255.255.0. This is done in the control panel / network / protocols / TCPIP screens. Then reboot your computer. If you have pulled the PC off a running Ethernet network, or intend to put it back on an existing network, check with your system administrator before changing the IP address and `netmask`.

Q. How can I remotely close an open link to the EM1000?

A. Situations may occur where a data link has been established to the EM1000, and you wish to break the link manually. For example, a device establishes a connection to an EM1000 that is in the listen mode. Then the device is supposed to do its business and drop the link, but it actually fails to drop the link. You can drop the link manually by Telnetting into the EM1000, setting `connect=request`, then `close`. Then set the connect mode back to `connect=demand`.

Q. I can't seem to Telnet to the host data port of my EM1000.

A. During debugging, it is sometimes convenient to Telnet not to the actual Telnet port in the EM1000, but to the data port (default 8888) of the EM1000. This way, your Telnet session will show directly what the device attached to the remote EM1000 is sending into the serial port. However, for this to work, the EM1000 must be set to `connect=demand`.

Q. When I power cycle my RS-232 device, the EM1000 locks up.

A. Your **COM1/RS-232** "handshake" signals are probably not hooked up properly. If you were communicating data successfully before the lock-up, then your transmit and receive lines are probably OK (pins 2 and 3 on **COM1/RS-232**). However, your other signals may not be hooked up correctly. If you attach an in-line LED mini tester, all the signal lines should be ON or OFF (red LED or green LED). If you have any LEDs that are off, or very weakly red or green, you may have an incorrect hookup.

Q. **I want to use a protocol such as XMODEM or ZMODEM to transmit data from one EM1000 to the other.**

A. You may need to look at the timing details of the protocol you want to use, and tweak the packetization parameters in both EM1000s. Generally, from testing done at Z-World, it appears that the ZMODEM protocol is the most efficient when using the default paramaters (**`bufferlength=200`**, **`buffertimeout=2`**), and that 9600 bps is the best baud rate to use. Depending on the type of equipment you are transmitting and receiving from, your results may be better at different baud rates or using different protocols. If you do not have ZMODEM available, then try using XMODEM (CRC). The throughput seems to be about half that of ZMODEM, and there is a significant lag on startup.

Blank

Z-World, Inc.
2900 Spafford Street
Davis, California 95616-6800 USA

Telephone:	(530) 757-3737
Facsimile:	(530) 753-5141
Web Site:	http://www.zworld.com
E-Mail:	zworld@zworld.com

Part No. 019-0066
001025 - C

Printed in U.S.A.